VIETNAM NATIONAL UNIVERSITY HO CHI MINH CITY
HO CHI MINH CITY UNIVERSITY OF TECHNOLOGY
FACULTY OF COMPUTER SCIENCE AND ENGINEERING

REPORT
CAPSTONE PROJECT

DETECTION, RECOGNITION, AND EXTRACTION OF TABLE STRUCTURE DATA

Major: Computer Science

THESIS COMMITTEE : COMPUTER SCIENCE 2
SUPERVISOR(s) : Assoc. Prof. Quan Thanh Tho, PhD
Mr. Bang Ngoc Bao Tam, MEng
REVIEWER : Dr. Tran Tuan Anh

---o0o---

STUDENT 1 : Nguyen Luat Gia Khoi (1952079)
STUDENT 2 : Pham Bui Minh Huan (1952056)

HO CHI MINH CITY, 06/2023

_____ Date: _____

Assoc. Prof. Quan Thanh Tho, PhD (Project Instructor)
Associate Professor
Faculty of Computer Science and Engineering

_____ Date: _____

Mr. Bang Ngoc Bao Tam, MEng (Project Instructor)
Master of Engineering
Faculty of Computer Science and Engineering

Declaration of Authenticity

We declare that this research is our own work, conducted under the supervision and guidance of Assoc. Prof. Dr. Quan Thanh Tho and Mr. Bang Ngoc Bao Tam. The result of which is legitimate and has not been published in any form prior to this. All materials used within this research are collected by us through various sources and are appropriately listed in the references section.

In addition, within this research, we also utilized the results of several other authors and organizations who have all been aptly referenced.

In any case of plagiarism, we stand by our actions and will be held responsible. University of Technology - Vietnam National University HCMC, therefore, is not responsible for any copyright infringements conducted within our research.

Ho Chi Minh City, November 2022

Authors

Pham Bui Minh Huan

Nguyen Luat Gia Khoi

Acknowledgements

We would like to express our deep and sincere gratitude to my supervisor, Dr. Quan Thanh Tho, for his patience, guidance, and support throughout my student years. We have benefited greatly from your wealth of knowledge and passion. We are extremely grateful that you took us under your guidance and continued to support us over the years.

Our families and friends also deserve endless gratitude for their everlasting love, sacrifice, and support. Our accomplishments and success are in great part because of their faith. There is no word that can describe my love for our families.

Members

Name	Student ID	Workload
Pham Bui Minh Huan	1952056	Data preparation, Table classification, Table recognition, Pipeline development
Nguyen Luat Gia Khoi	1952079	Table detection, Model pruning, Pipeline development

Abstract

It is a well-known fact that we are in the age of booming technology, especially in the field of digitalization. The need for optical character recognition technology is now higher than ever as companies and institutions shift from paper-based processes to virtual workspaces. Despite the multitude of available methods for localizing tables in document images and decomposing them into their structural building blocks, these tasks still prove to be difficult even with all the recent advances in modern document processing systems. Due to the high degree of intra-class variability, this proves to be a challenging problem as there is no formal definition of what a table looks like. There are also very few off-the-shelf systems that can handle table structure data as most systems utilize divide-and-conquer technology, specializing in dealing with a single specific type of tables such as full-border or borderless ones. Realizing the application potential of table structure recognition models in many different existent problems, especially when every company is on its way to digitalizing their documents. We develop an end-to-end pipeline that takes inputs as images and transforms any detected table in them into a tabular format. Furthermore, we also implement a parameter pruning method for Transformer-based models that can significantly optimize the number of parameters in a Transformer-based model without compromising so much accuracy.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This chapter presents the problem statement, research target, and scope of the research on a topic that aims to contribute to the field of computer science, as well as society. The research focuses on addressing a current challenge or issue in the area of computer science and aims to provide a better understanding of the problem and propose solutions to it. Through this research, we hope to contribute to the advancement of the field and make a positive impact on society.

1.1 Problem statement

In today's digital age, many companies have found themselves in great need of OCR technology. Tables are ubiquitous in almost every field, from medical results to financial statements to tax forms. As a result, there is a huge demand for OCR technology that can accurately recreate table structures from a wide variety of table formats in documents, whether they are in PDF or image format. However, there are a number of challenges to be overcome, such as different layouts, the inconsistent use of ruling lines, and diverse table contents. These factors contribute to a lack of a formal pattern for tables, making it difficult to design a system that can detect, recognize, and recreate table structures in CSV format. Despite these challenges, the real-world applications of OCR technology make it a necessary and worthwhile pursuit.

One of the most common modeling approaches for table structure recognition

task is to frame it as some form of object detection [1–3], which utilizes a number of general-purpose architectures, such as Faster R-CNN, for detection. However, the lack of complete, unambiguous ground truth leads to the underperformance of these models when applied to table structure recognition out-of-the-box. Therefore, some approaches decompose the problem into more specific subtasks, such as row and column detection in DeepDeSRT [4], which ignores spanning cells, or image-to-markup without text content as in models trained on TableBank [5]. Other approaches propose pipelines that branch into different cases separately, such as bordered and borderless tables [6, 7].

In this research, we conduct a survey and comparison between state-of-the-art models in the tasks of Table Detection (TD) and Table Structure Recognition (TSR). Additionally, originating from the idea of [6, 7], we develop an end-to-end Table OCR pipeline that deals with all kinds of tables, categorized into full-border or borderless, by combining multiple state-of-the-art models in the tasks of TD and TSR. The pipeline takes an input image containing some table(s) and generates output spreadsheet(s) corresponding to each detected table in the input image. We also work on the pre and post-processing modules between main stages throughout the pipeline to provide the best results. In this research, we also implement and analyze a Transformer-based pruning method originated from a study by Khetan et. al. [8]. The need for a Transformer-based pruning method is mentioned below.

Table OCR's adaptability to various digitalization applications is coupled with its ability to run on different platforms, including high-power servers and low-resource hardware such as mobile phones and embedded systems. However, compressing and optimizing large models to fit these low-power devices has become a pressing issue. Pruning deep learning models has become necessary in overcoming this challenge.

The Transformer has been a dominant force in many deep learning fields, from natural language processing to image recognition and audio generation. Given the optimal performance of the Transformer architecture, there has been a limited amount of research on pruning this particular model. However, potential avenues for exploration do exist and are further elaborated on in the 'Related Works' section. For instance, researchers may consider investigating how pruning can be used to reduce the computational complexity of the Transformer [9–14] or how the method can be adapted to

improve its performance in low-resource settings [15–18]. Ultimately, while pruning the Transformer architecture presents a unique set of challenges, it also presents a number of potential benefits that are worth investigating further.

Our research implements a pruning method specifically designed for transformer-based models, which originated from the work of Khetan et al. [19]. This pruning method focuses on transformer-based models due to their potential and power in the aforementioned fields. The models pruned by this method are shown to achieve a drop of only 2% in average precision (AP) compared to the original model while reducing 66% of parameters. On GriTS metrics, which are introduced in section 2.3.2, it achieves a drop of only 1% in GriTS_Top and 3% in GriTS_Loc while reducing 66% of parameters. By implementing this method, we aim to improve the efficiency of these models, making them more accessible to low-resource devices.

1.2 Rationale

Academically speaking, in completing this research we had provided a thorough look at the topic of OCR of tables in digital and scanned documents from the many definitions and problems with table structure data, what input is to be expected and what is the desired output that can be of practical use. We also tested and compared available approaches to create a fully detailed picture of what has been done, what should be done, and what can be improved. These can serve as the basis for many future works.

Our research also consists of building a system that combines the best approaches and models to handle all different kind of tables in one system. There are many significant real-life applications and demand for such a system across the fields. Hospitals, companies, government departments, and any institution that need to deal with a lot of documents can all make use of this system, especially in a country where digitalizing process of documents is still in the early steps like Vietnam.

Pruning deep learning models has become necessary in order to compress and optimize large models for low-power devices. Transformer-based models are powerful and have potential in various deep learning fields, such as natural language processing, image recognition, and audio generation. However, pruning these models presents a

unique set of challenges due to their optimal performance. Nonetheless, pruning the Transformer architecture presents a number of potential benefits, such as improving the efficiency of these models and making them more accessible to low-resource devices. To achieve this goal, we propose a pruning method specifically designed for transformer-based models. Ultimately, our research proposes a valuable contribution to the field of table structure recognition technology by implementing a pruning method that can improve the efficiency of transformer-based models.

1.3 Goals

This research aims to provide a clear picture and summary of the topic of table structure OCR by testing and comparing available solutions. We also build a deep learning system that given a scanned or pdf document with tables can detect, recreate and output a uniform output in CSV or excel format. Additionally, we implement a parameter pruning method for Transformer-based models based on previous research conducted by [19]

1.4 Scope

In this thesis, we aim to deal with the following problems:

- **Summary research**: Presentation on the topic of OCR of tables in documents. We also perform testing and make comparisons to determine the strengths and weaknesses of available approaches and ways they can be improved.
- **Deep learning system**: A deep learning system that is expected to detect and recognize tables in documents then proceed to OCR the contents and reconstruct the tables in excel files.
- **Parameter pruning method for Transformer-based models**: A learning-based method for pruning Transformer-based models that substantially reduces the number of parameters without sacrificing much of model accuracy.

1.5 Contribution

We provide a summary of our key contributions in this study below:

- Conducted a comprehensive evaluation and comparison of the latest models for TD and TSR tasks.
- Acquired insightful conclusions regarding the respective strengths and weaknesses of the models. These conclusions can aid researchers in choosing the most suitable TD or TSR model.
- Proposed a pipeline that effectively combines the strengths of multiple solutions for improved accuracy and lays a solid foundation for future research and development in the table OCR field.
- Reimplemented Khetan et al.'s work [19] and conducted an evaluation on multiple metrics for both TD and TSR tasks to prove whether the approach can be generalized for transformer-based models and different domains. The models pruned by this method are shown to achieve a drop of only 2% in average precision (AP) compared to the original model while reducing 66% of parameters. On GriTS metrics, which are introduced in section 2.3.2, it achieves a drop of only 1% in GriTS_Top and 3% in GriTS_Loc while reducing 66% of parameters. By implementing this method, we aim to improve the efficiency of these models, making them more accessible to low-resource devices.

1.6 Thesis structure

Chapter 1: Introduction

State the problem, the target of our research, and how such research will contribute toward the field of computer science as well as society. We also further define the scope of the research.

Chapter 2: Overview analysis and related works

Present our research on prior achievements and solutions of the topic of Table Detection (TD) and Table Structure Recognition (TSR), compare and analyze the

strengths and weakness of these available systems. For the parameter pruning problem, we present different pruning methods ranging from naive ones for machine learning algorithms to modern methods for deep learning models, especially Transformer-based models. From such insights, we go on to define our chosen techniques and technology as well as the functional and non-functional requirements of the project. Diagrams are also being presented to provide more details.

Chapter 3: Theoretical background

Present our knowledge and practical experience on the topics. Define concepts that are needed in the system and the theory behind the proposed technology and model.

Chapter 4: Proposed solution

Present in detail the architecture, implementation, and workflow of our solutions. A detailed description of the components is also provided for the table OCR pipeline. For the parameter pruning method, the detailed algorithm and implementation are also carefully presented.

Chapter 5: Experimental Results and Discussion

Experimental results for various TD and TSR models are provided, using multiple defined metrics, accompanied by discussions on the advantages and disadvantages of each candidate model. Conclusions regarding pipelines, TD/TSR models, and the efficiency of the pruning methods are also presented.

Chapter 6: Summary and future development

Summary of what we have achieved so far, the challenges, and lessons learned along the way. Plans to further enhance and develop the project in the future.

Reference

Reference on previous works that have been cited as an inspiration or base for our research

Appendix

Work assignment and activity log

Chapter 2

Related works

In this chapter, we delve into the rich history of Table Detection (TD) and Table Structure Recognition (TSR), examining the accomplishments and limitations of previous research to better understand the landscape. By comparing and analyzing the various strengths and weaknesses of existing systems, we aim to identify opportunities for improvement and innovation. To address the issue of parameter pruning, we explore a range of methods, from simple machine learning algorithms to advanced deep learning techniques, with a particular focus on Transformer-based models. From such insights, we go on to define our chosen techniques and technology as well as the functional and non-functional requirements of the project. Diagrams are also being presented to provide more details.

2.1 Related works

A taxonomy of the topic of table extraction from image and scanned document has been provided at 2.1, in which we introduce the main subtasks of our problem and the approaches that have been put forth. Our research will follow this taxonomy and go into the details of each task as well as their respective current solutions.

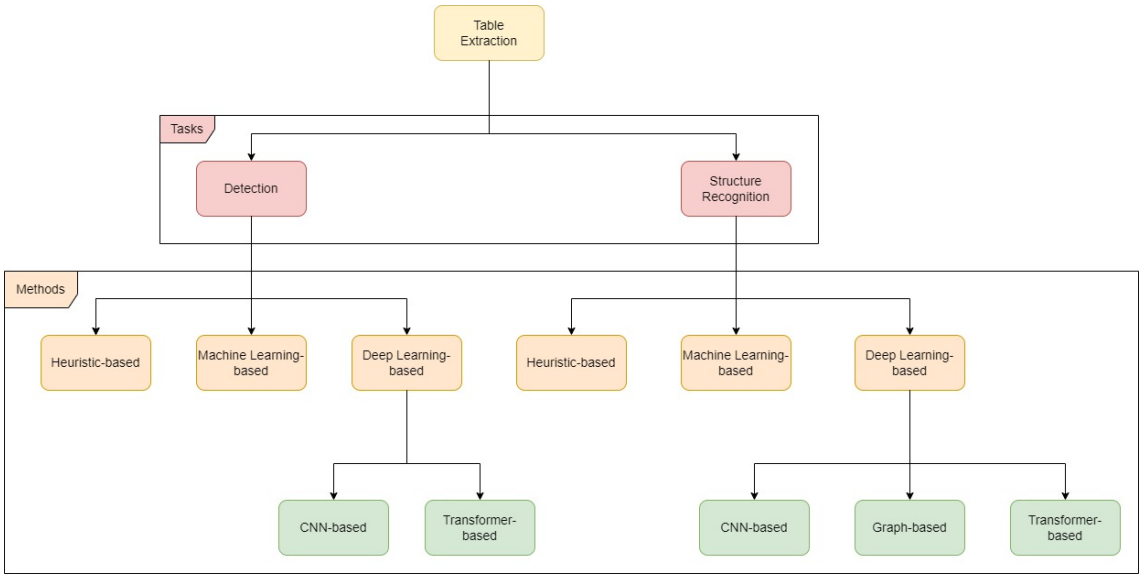


Figure 2.1: Taxonomy of research topic Table extraction from image and scanned document

2.1.1 Table Detection

The goal of table detection is to locate tables in a document using bounding boxes and the goal of table structure recognition is to determine a table's row and column layout information. Researchers employed many techniques that fall under the categories of heuristic-based techniques, machine learning techniques, or deep learning techniques. In the 1990s, 2000s, and the beginning of 2010, heuristic-based techniques predominated. Katsuhiko et. al. [20] explains a novel technique for extracting table structure from document photos. Each cell in a table is represented by a conventional two-dimensional row and column pair arrangement. There are two steps in this process: expanding the bounding boxes of the cells and assigning row and column numbers to each edge. Hassan et al. [21] locate and segment tables by analyzing spatial features of text blocks. By examining the spatial characteristics of text blocks, Hassan et al. [21] prosed a technique that locates and segments tables. In 2009, Ruffolo et al. [22] introduced PDF-TREX, a heuristic bottom-up method for recognizing tables in single-column PDF documents. It aligns and groups page elements into paragraphs and tables using the spatial characteristics of those elements. The table header served as a starting point for the method of Fang et. al. [23] to identify the

table region and decompose its elements. A method for table detection based on the recognition of unique table start and trailer patterns was then proposed by Harit et al. in their paper [24].

Data-driven image-based table detection techniques have been significantly impacted by the quick advancement of machine learning in computer vision. First unsupervised machine learning method for the task of table detection was proposed by Kieninger and Dengel [25] in 1998. A supervised machine learning algorithm based on hierarchical representation using the MXY tree was proposed by Cesarini Francesca et al. [26] in 2002. Additionally, machine learning algorithms are used for various tasks in table detection and structure detection, such as the feature extraction task proposed by Kasar et al. [27] and the task of sequence labeling by Silva et al. [**?**]. Silva proposed a hidden Markov model (HMM) for table location using probabilistic graphical models and interdependent classification.

The use of deep learning in computer vision is prominent nowdays. It has a significant impact on scanned image for table detection. Convolutional neural networks (CNNs) are the best candidate for deep learning in image processing methods for document analysis. In document analysis and image processing, CNNs for object detection have been widely used [28–31]. Faster-RCNN [32] demonstrated positive effects at table detection and attained cutting-edge performance on ICDAR 2013 [33]

Pascal Fischer et. al. [34] presented a multistage pipeline named Multi-Type-TD-TSR, which offers an end-to-end solution for the problem of table recognition. It utilizes the Faster RCNN together with FPN (feature pyramid networks) for table detection. The model was trained and evaluated on TableBank [5] dataset. Madhav Agarwal et. al. [35] proposed CDeC-Net consisting of a multistage extension of Mask R-CNN with a dual backbone having deformable convolution for detecting tables varying in scale with high detection accuracy at higher IoU threshold. The model is trained on all currently available dataset and evaluated on ICDAR 2013 [33], ICDAR 17 [36], ICDAR 19 [37], UNLV [**?**], Marmot [23], TableBank [5], PubLayNet [38]. Devashish Prasad [6] introduced CascadeTabNet, which a Cascade mask Region-based CNN High-Resolution Network (Cascade mask R-CNN HRNet) based model that detects the regions of tables and recognizes the structural body cells from the detected tables at the same time. The model is trained on ICDAR 19 Track A

(Modern) [37], Marmot, and an open-source dataset on github [39]. It is evaluated on ICDAR 19 Track A (Modern) [37], TableBank [5], ICDAR 2013 [33].

Abdelrahman Abdallah et. al. [40] has introduced TNCR, a new table dataset with varying image quality collected from free websites. The TNCR dataset contains 9428 high-quality labeled images, which can be used for table detection in scanned document images and their classification into five different classes. In this paper, the authors implemented state-of-the-art deep learning-based methods for table detection to create several strong baselines including Cascade R-CNN, Cascade Mask R-CNN, YOLO, Cascade RPN, Deformable DERT, Hybrid Task Cascade. These models are trained and evaluated on TNCR dataset.

Junlong Li et. al. [41] proposed DiT, a self-supervised pre-trained Document Image Transformer model using large-scale unlabeled text images for Document AI tasks, which is essential since no supervised counterparts ever exist due to the lack of human-labeled document images. They leverage DiT as the backbone network in a variety of vision-based Document AI tasks, including document image classification, document layout analysis, table detection as well as text detection for OCR. The model is trained on three datasets including RVL-CDIP [42], ICDAR 19 Track A [37] (Modern) and PubLayNet [38] and evaluted on ICDAR 2019 [37] (Modern, Archival, both Modern and Archival).

Brandon Smock et. al. [43] from Microsoft first introduced their PubTable-1M dataset for table detection (TD) and table structure reconstruction (TSR) task and proposed a transformer-based architecture to detect table. The model was trained and evaluated on PubTable-1M dataset.

2.1.2 Table Structure Recognition

One of the most common modeling approaches for table structure recognition (TSR) is to frame the task as some form of object detection [4, 6, 7]. Other methods include those that use image-to-text conversion [5] and graph-based methods [3, 44]. While a number of general-purpose architectures, such as Faster R-CNN [32], exist for these model patterns, their underperformance when applied to TSR out-of-the-box is frequently observed due to the particularities of tables and the relative lack of training data.

Some techniques, such as row and column detection in DeepDeSRT [4], which ignores spanning cells, or image-to-markup without cell text content, as in models trained on TableBank [5], model TSR in ways that are only partial solutions to the task. Other methods [6,7] employ unique pipelines that branch to take different cases into account separately, such as training different models to distinguish between tables with and without clearly visible cell borders.

A pair of deep learning models were proposed by Chris Tensmeyer et al. [45] split a table image into its basic grid of cells and then combine those cells to recover cells that span multiple rows and columns. Key insight is to pool information over large regions of the table image such as entire rows/columns of pixels or previously predicted cell regions . Besides, heuristic techniques can outperform the merge model, which is only effective in certain situations.

Using a novel graph neural network model, Zewen Chi et al. [3] reformulate the task as an edge prediction problem on graphs. It uses a stack of graph attention blocks to encode a table and then predicts the relationships between cells to decode the table's structure. For the first time, Brandon Smock et al. [43] demonstrated that it is possible to achieve state-of-the-art performance within a standard object detection framework without the need for any special customization for these tasks. They did this by adopting DETR [1] for all three table extraction tasks (table detection, table structure recognition, and table functional analysis).

In addition to using engineered model elements or tailored training procedures, many of the approaches previously mentioned also incorporate rules or other unlearned processing stages that are specific to the TSR task. By incorporating this prior knowledge, the burden of learning the task from data is reduced. There is currently no solution that uses a straightforward supervised learning approach with off-the-shelf architecture to completely solves the TSR task, achieves state-of-the-art performance.

2.1.3 Parameter Pruning Methods

There is a vast amount of literature on the subject of pruning trained neural networks. Researchers have been investigating this topic for decades. Classical works, such as those by LeCun et al. [46] and Hassibi and Stork [47], date back to the

early 90s. More recent works by Han et al. [18] have also gained a lot of attention. There have been two main approaches to pruning networks: structured pruning and unstructured pruning. Structured pruning, as explored by Li et al. [9] and Molchanov et al. [10], involves removing entire groups of neurons from the network in order to create a smaller architecture. Unstructured pruning presented in [11], on the other hand, involves removing individual parameters from the network, resulting in a sparse model.

In the field of natural language processing, researchers have also explored structured pruning in feed-forward language models. For example, Murray and Chiang [48] investigated the use of structured pruning in this context. See et al. [49] and Kim and Rush [50] have also proposed pruning approaches for machine translation. These techniques have the potential to reduce the size and complexity of neural networks, making them more efficient and easier to train. A closely related line of work is Neural Architecture Search (NAS). It aims to efficiently search the space of architectures [12–14]. Quantization is another technique to reduce the model size. This is done by quantizing the model parameters to binary [15, 16], ternary [17], or 4 or 8 bits per parameter [18]. Khetan et. al. [19] revisited the BERT model and proposed a learning-based parameter pruning method to create a lighter model, namely schuBERT. Their focus was on reducing the number of parameters, but their methods can be applied to other objectives such as FLOPs or latency. The authors demonstrated that algorithmically reducing certain correct architecture design dimensions is more effective in creating efficient, lighter BERT models than simply reducing the number of Transformer encoder layers. Specifically, schuBERT achieves 6.6% higher average accuracy on GLUE and SQuAD datasets than BERT with three encoder layers, despite having the same number of parameters. As the authors do not offer a detailed implementation of their method, we attempted to implement it in this study and apply it to a state-of-the-art model on the PubTables-1M dataset - the Table Transformer model [43]. This model is one of the models used in our pipeline that can be adapted for both TD and TSR tasks.

Figure 2.2 shows the taxonomy of the topic of parameter pruning methods.

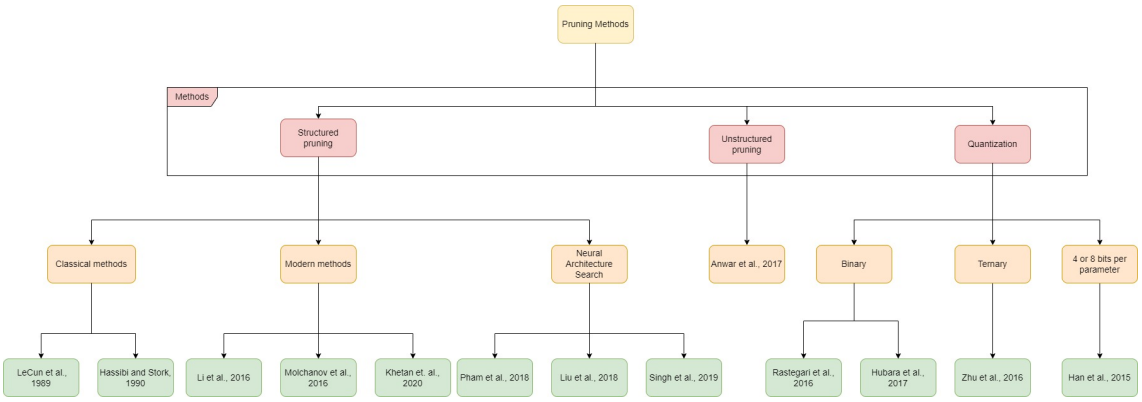


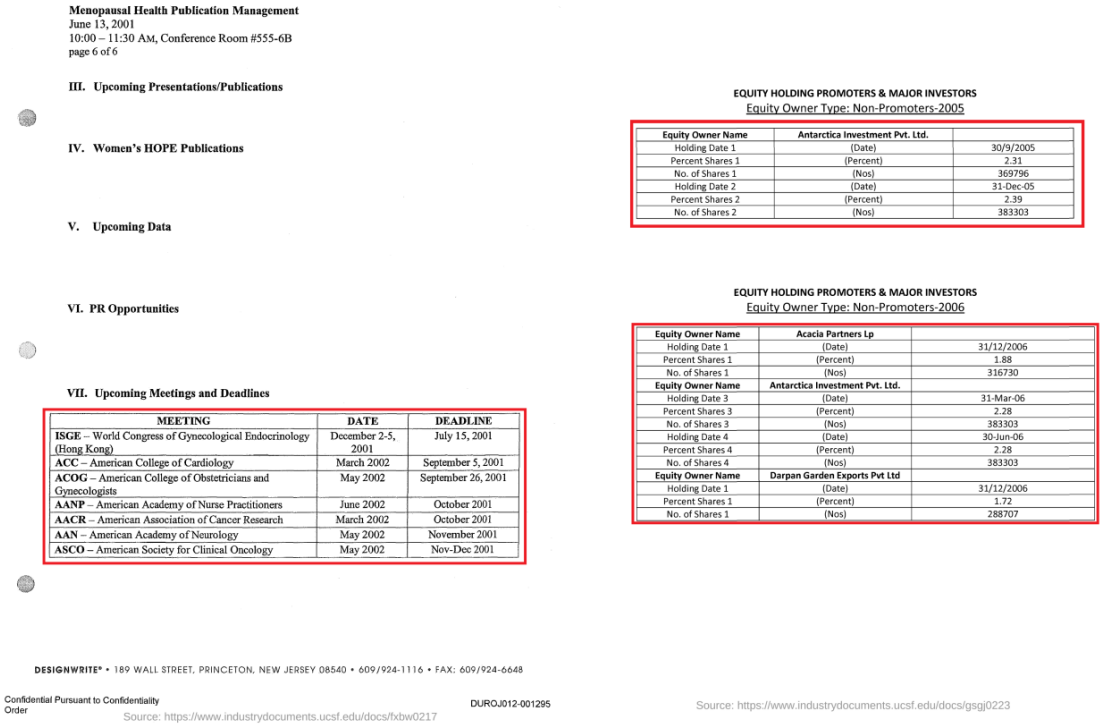
Figure 2.2: Taxonomy of research topic Parameter pruning methods

2.2 Dataset

In this section, we describe datasets that are used during the course of our research. We only mention currently available datasets since some of them have been retracted from the internet.

2.2.1 ICDAR 2013

The referenced ICDAR 2013 [33] dataset served as the competition's official practice dataset. Authors have always intended to evaluate systems broadly rather than focusing on a specific subset of documents, and this dataset, as well as the actual competition dataset, were generated by methodically collecting PDFs from a Google search in order to make the selection as impartial as possible. To obtain documents in the public domain, they are restricted to two governmental sources with the additional search terms site:europa.eu and site:*.gov. The ICDAR2013 dataset contains 150 tables, 75 of which are in 27 EU excerpts and 75 of which are in 40 US Government excerpts. Table regions are the rectangular areas of a page that are identified by their coordinates. A table can span multiple pages, so multiple regions can be included in a single table. The two sub-tasks of ICDAR2013 are table detection or location and table structure recognition. A few examples from this dataset are shown in figure 2.3.



Menopausal Health Publication Management
June 13, 2001
10:00 – 11:30 AM, Conference Room #555-6B
page 6 of 6

III. Upcoming Presentations/Publications

IV. Women's HOPE Publications

V. Upcoming Data

VI. PR Opportunities

VII. Upcoming Meetings and Deadlines

MEETING	DATE	DEADLINE
ISGE – World Congress of Gynecological Endocrinology (Hong Kong)	December 2-5, 2001	July 15, 2001
ACC – American College of Cardiology	March 2002	September 5, 2001
ACOG – American College of Obstetricians and Gynecologists	May 2002	September 26, 2001
AANP – American Academy of Nurse Practitioners	June 2002	October 2001
AACR – American Association of Cancer Research	March 2002	October 2001
AAN – American Academy of Neurology	May 2002	November 2001
ASCO – American Society for Clinical Oncology	May 2002	Nov-Dec 2001

DESIGNWRITE® • 189 WALL STREET, PRINCETON, NEW JERSEY 08540 • 609/924-1116 • FAX: 609/924-6648

Confidential Pursuant to Confidentiality Order

DUROJ012-001295

Source: https://www.industrydocuments.ucsf.edu/docs/fxbw0217

EQUITY HOLDING PROMOTERS & MAJOR INVESTORS
Equity Owner Type: Non-Promoters-2005

Equity Owner Name	Antarctica Investment Pvt. Ltd.	
Holding Date 1	(Date)	30/9/2005
Percent Shares 1	(Percent)	2.31
No. of Shares 1	(Nos)	369796
Holding Date 2	(Date)	31-Dec-05
Percent Shares 2	(Percent)	2.39
No. of Shares 2	(Nos)	383303

EQUITY HOLDING PROMOTERS & MAJOR INVESTORS
Equity Owner Type: Non-Promoters-2006

Equity Owner Name	Acacia Partners Lp	
Holding Date 1	(Date)	31/12/2006
Percent Shares 1	(Percent)	1.88
No. of Shares 1	(Nos)	316730
Equity Owner Name	**Antarctica Investment Pvt. Ltd.**	
Holding Date 3	(Date)	31-Mar-06
Percent Shares 3	(Percent)	2.28
No. of Shares 3	(Nos)	383303
Holding Date 4	(Date)	30-Jun-06
Percent Shares 4	(Percent)	2.28
No. of Shares 4	(Nos)	383303
Equity Owner Name	**Darpan Garden Exports Pvt Ltd**	
Holding Date 1	(Date)	31/12/2006
Percent Shares 1	(Percent)	1.72
No. of Shares 1	(Nos)	288707

Source: https://www.industrydocuments.ucsf.edu/docs/gsgj0223

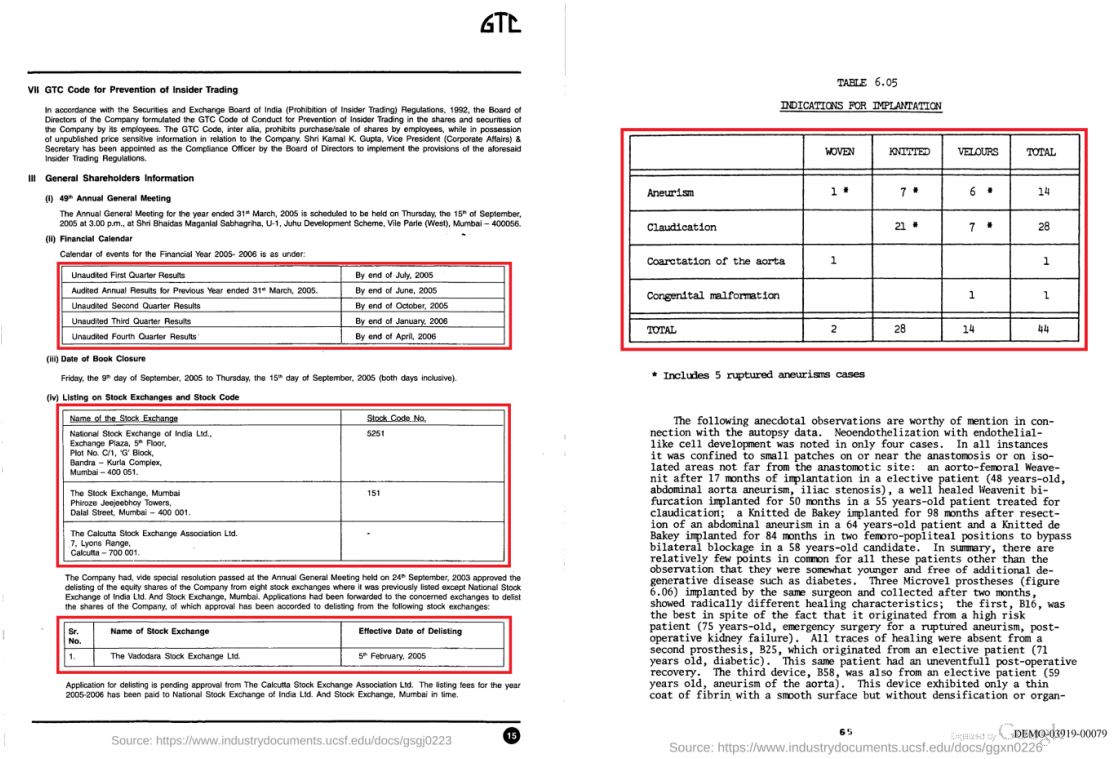
Figure 2.3: Examples of images in ICDAR 2013

2.2.2 ICDAR 2017

For the ICDAR2017 Page Object Detection (POD) competition, dataset ICDAR2017 [36] was released. This dataset is used to test various table detection algorithms. It is considerably bigger than the ICDAR2013 table dataset. It contains figures, tables, and formulae, totaling 2417 images. The dataset is divided into 817 images with 350 tabular regions for testing and 1600 images with 731 tabular areas for training. Two examples of this dataset are shown in Figure 2.4.

2.2.3 ICDAR 2019

ICDAR2019 [37] proposed a dataset for table detection (TRACK A) and table recognition (TRACK B). The dataset is divided into two types, historical and modern datasets. The dataset is separated into historical and contemporary datasets. It includes 839 images for testing and 1600 images for training. Tracks A and B of



GTL

VII GTC Code for Prevention of Insider Trading

In accordance with the Securities and Exchange Board of India (Prohibition of Insider Trading) Regulations, 1992, the Board of Directors of the Company formulated the GTC Code of Conduct for Prevention of Insider Trading in the shares and securities of the Company by its employees. The GTC Code, inter alia, prohibits purchase/sale of shares by employees, while in possession of unpublished price sensitive information in relation to the Company. Shri Kamal K. Gupta, Vice President (Corporate Affairs) & Secretary has been appointed as the Compliance Officer by the Board of Directors to implement the provisions of the aforesaid Insider Trading Regulations.

III General Shareholders Information

(i) 49th Annual General Meeting

The Annual General Meeting for the year ended 31st March, 2005 is scheduled to be held on Thursday, the 15th of September, 2005 at 3.00 p.m., at Shri Bhaidas Maganlal Sabhagriha, U-1, Juhu Development Scheme, Vile Parle (West), Mumbai – 400056.

(ii) Financial Calendar

Calendar of events for the Financial Year 2005- 2006 is as under:

Unaudited First Quarter Results	By end of July, 2005
Audited Annual Results for Previous Year ended 31st March, 2005.	By end of June, 2005
Unaudited Second Quarter Results	By end of October, 2005
Unaudited Third Quarter Results	By end of January, 2006
Unaudited Fourth Quarter Results	By end of April, 2006

(iii) Date of Book Closure

Friday, the 9th day of September, 2005 to Thursday, the 15th day of September, 2005 (both days inclusive).

(iv) Listing on Stock Exchanges and Stock Code

Name of the Stock Exchange	Stock Code No.
National Stock Exchange of India Ltd., Exchange Plaza, 5th Floor, Plot No. C/1, 'G' Block, Bandra – Kurla Complex, Mumbai – 400 051.	5251
The Stock Exchange, Mumbai Phiroze Jeejeebhoy Towers, Dalal Street, Mumbai – 400 001.	151
The Calcutta Stock Exchange Association Ltd. 7, Lyons Range, Calcutta – 700 001.	-

The Company had, vide special resolution passed at the Annual General Meeting held on 24th September, 2003 approved the delisting of the equity shares of the Company from eight stock exchanges where it was previously listed except National Stock Exchange of India Ltd. And Stock Exchange, Mumbai. Applications had been forwarded to the concerned exchanges to delist the shares of the Company, of which approval has been accorded to delisting from the following stock exchanges:

Sr. No.	Name of Stock Exchange	Effective Date of Delisting
1.	The Vadodara Stock Exchange Ltd.	5th February, 2005

Application for delisting is pending approval from The Calcutta Stock Exchange Association Ltd. The listing fees for the year 2005-2006 has been paid to National Stock Exchange of India Ltd. And Stock Exchange, Mumbai in time.

15

TABLE 6.05

INDICATIONS FOR IMPLANTATION

	WOVEN	KNITTED	VELOURS	TOTAL
Aneurism	1 *	7 *	6 *	14
Claudication		21 *	7 *	28
Coarctation of the aorta	1			1
Congenital malformation			1	1
TOTAL	2	28	14	44

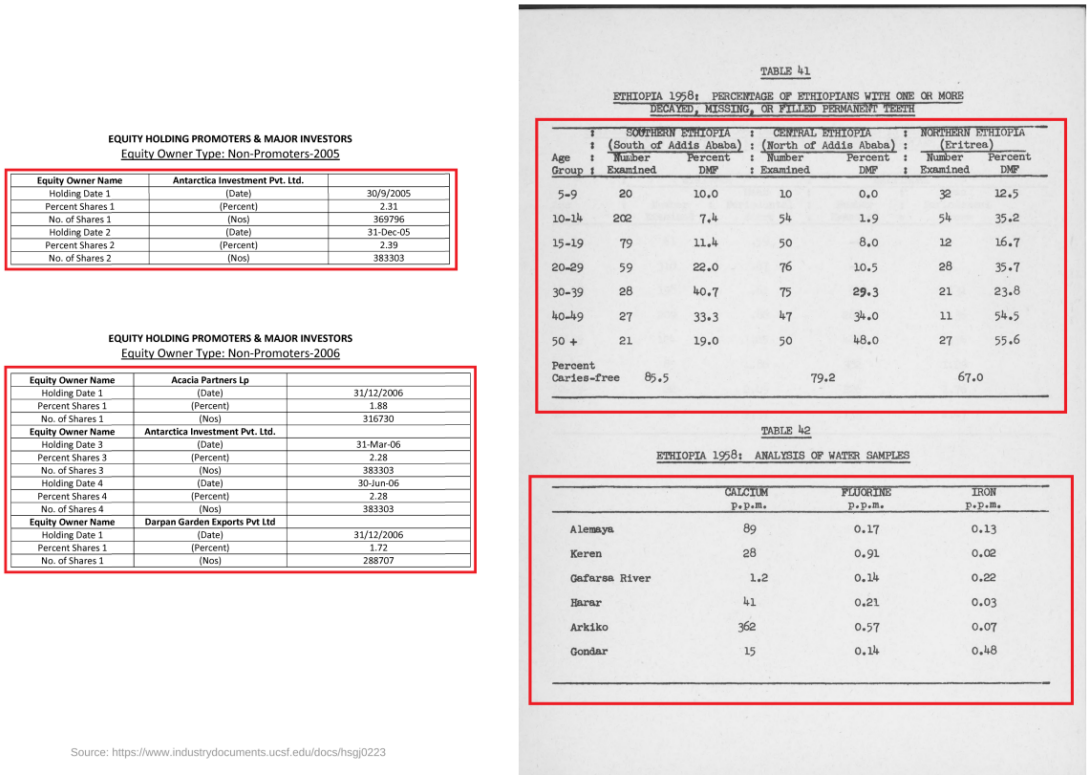
* Includes 5 ruptured aneurisms cases

The following anecdotal observations are worthy of mention in connection with the autopsy data. Neoendothelization with endothelial-like cell development was noted in only four cases. In all instances it was confined to small patches on or near the anastomosis or on isolated areas not far from the anastomotic site: an aorto-femoral Weavenit after 17 months of implantation in a elective patient (48 years-old, abdominal aorta aneurism, iliac stenosis), a well healed Weavenit bifurcation implanted for 50 months in a 55 years-old patient treated for claudication; a Knitted de Bakey implanted for 98 months after resection of an abdominal aneurism in a 64 years-old patient and a Knitted de Bakey implanted for 84 months in two femoro-popliteal positions to bypass bilateral blockage in a 58 years-old candidate. In summary, there are relatively few points in common for all these patients other than the observation that they were somewhat younger and free of additional degenerative disease such as diabetes. Three Microvel prostheses (figure 6.06) implanted by the same surgeon and collected after two months, showed radically different healing characteristics; the first, B16, was the best in spite of the fact that it originated from a high risk patient (75 years-old, emergency surgery for a ruptured aneurism, post-operative kidney failure). All traces of healing were absent from a second prosthesis, B25, which originated from an elective patient (71 years old, diabetic). This same patient had an uneventfull post-operative recovery. The third device, B58, was also from an elective patient (59 years old, aneurism of the aorta). This device exhibited only a thin coat of fibrin with a smooth surface but without densification or organ-

65

Figure 2.4: Examples of images in ICDAR 2017

the historical type contain 1200 images for training and 499 images for testing. The contemporary type contains 600 images in tracks A and B for training and 340 images for testing. For TRACK A, document images with one or more tables are provided. TRACK B has two sub-tracks: the first (B.1) provides the table region, and only the table structure recognition is required. The second sub-track (B.2) contains no prior knowledge. That is, both table region detection and table structure recognition must be performed. For the annotation of the dataset, a similar notation was derived from the ICDAR 2013 [33] Table Competition format, and the structures were stored in a single XML file. Each table element corresponds to a table with a single *Coords* element with a *points* attribute indicating the coordinates of the bounding polygon with N vertices. The positions of each cell element in the table are indicated by the attributes *start-row*, *start-col*, *end-row*, and *end-col*. The bounding polygon coordinates for this cell's box are indicated by the cell element's Coords, and the text within this cell is its content. A few examples from this dataset are shown in figure 2.5.



EQUITY HOLDING PROMOTERS & MAJOR INVESTORS

Equity Owner Type: Non-Promoters-2005

Equity Owner Name	Antarctica Investment Pvt. Ltd.	
Holding Date 1	(Date)	30/9/2005
Percent Shares 1	(Percent)	2.31
No. of Shares 1	(Nos)	369796
Holding Date 2	(Date)	31-Dec-05
Percent Shares 2	(Percent)	2.39
No. of Shares 2	(Nos)	383303

EQUITY HOLDING PROMOTERS & MAJOR INVESTORS

Equity Owner Type: Non-Promoters-2006

Equity Owner Name	Acacia Partners Lp	
Holding Date 1	(Date)	31/12/2006
Percent Shares 1	(Percent)	1.88
No. of Shares 1	(Nos)	316730
Equity Owner Name	Antarctica Investment Pvt. Ltd.	
Holding Date 3	(Date)	31-Mar-06
Percent Shares 3	(Percent)	2.28
No. of Shares 3	(Nos)	383303
Holding Date 4	(Date)	30-Jun-06
Percent Shares 4	(Percent)	2.28
No. of Shares 4	(Nos)	383303
Equity Owner Name	Darpan Garden Exports Pvt Ltd	
Holding Date 1	(Date)	31/12/2006
Percent Shares 1	(Percent)	1.72
No. of Shares 1	(Nos)	288707

Source: https://www.industrydocuments.ucsf.edu/docs/hsgj0223

TABLE 41

ETHIOPIA 1958: PERCENTAGE OF ETHIOPIANS WITH ONE OR MORE DECAYED, MISSING, OR FILLED PERMANENT TEETH

Age Group	SOUTHERN ETHIOPIA (South of Addis Ababa) Number Examined	SOUTHERN ETHIOPIA (South of Addis Ababa) Percent DMF	CENTRAL ETHIOPIA (North of Addis Ababa) Number Examined	CENTRAL ETHIOPIA (North of Addis Ababa) Percent DMF	NORTHERN ETHIOPIA (Eritrea) Number Examined	NORTHERN ETHIOPIA (Eritrea) Percent DMF
5-9	20	10.0	10	0.0	32	12.5
10-14	202	7.4	54	1.9	54	35.2
15-19	79	11.4	50	8.0	12	16.7
20-29	59	22.0	76	10.5	28	35.7
30-39	28	40.7	75	29.3	21	23.8
40-49	27	33.3	47	34.0	11	54.5
50 +	21	19.0	50	48.0	27	55.6
Percent Caries-free	85.5		79.2		67.0	

TABLE 42

ETHIOPIA 1958: ANALYSIS OF WATER SAMPLES

	CALCIUM P.P.M.	FLUORINE P.P.M.	IRON P.P.M.
Alemaya	89	0.17	0.13
Keren	28	0.91	0.02
Gafarsa River	1.2	0.14	0.22
Harar	41	0.21	0.03
Arkiko	362	0.57	0.07
Gondar	15	0.14	0.48

Figure 2.5: Examples of images in ICDAR 2019

2.2.4 TableBank

TableBank [5] proposed a novel weak supervision approach for automatically creating the dataset, which is orders of magnitude larger than existing human-labeled datasets for table analysis. It was created using a novel weak supervision approach. This method can produce large amounts of high-quality training data, unlike the conventional weakly supervised training set. Nowadays, a wide variety of electronic documents, including Latex (.tex) and Microsoft Word (.docx) files, are accessible online. By definition, the source code of these online documents includes mark-up tags for tables. It makes intuitive sense that these source codes manipulate each document by using the mark-up language to add bounding boxes. The borderline of each table can be specified in the Office XML code for Word documents. Table bounding boxes can be detected by modifying the code for Latex documents. For large-scale table analysis tasks, this method produces high-quality labeled data for

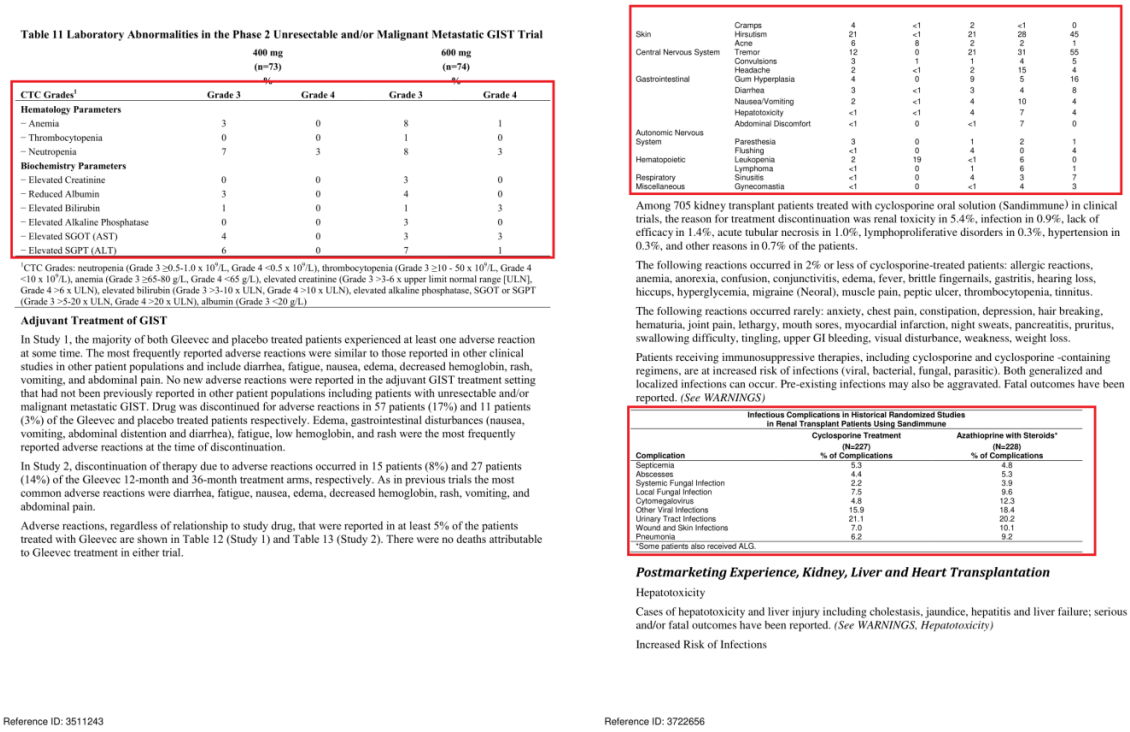


Table 11 Laboratory Abnormalities in the Phase 2 Unresectable and/or Malignant Metastatic GIST Trial

	400 mg (n=73) %		600 mg (n=74) %	
CTC Grades[1]	Grade 3	Grade 4	Grade 3	Grade 4
Hematology Parameters				
– Anemia	3	0	8	1
– Thrombocytopenia	0	0	1	0
– Neutropenia	7	3	8	3
Biochemistry Parameters				
– Elevated Creatinine	0	0	3	0
– Reduced Albumin	3	0	4	0
– Elevated Bilirubin	1	0	1	3
– Elevated Alkaline Phosphatase	0	0	3	0
– Elevated SGOT (AST)	4	0	3	3
– Elevated SGPT (ALT)	6	0	7	1

[1]CTC Grades: neutropenia (Grade 3 ≥0.5-1.0 x 10^9/L, Grade 4 <0.5 x 10^9/L), thrombocytopenia (Grade 3 ≥10 - 50 x 10^9/L, Grade 4 <10 x 10^9/L), anemia (Grade 3 ≥65-80 g/L, Grade 4 <65 g/L), elevated creatinine (Grade 3 >3-6 x upper limit normal range [ULN], Grade 4 >6 x ULN), elevated bilirubin (Grade 3 >3-10 x ULN, Grade 4 >10 x ULN), elevated alkaline phosphatase, SGOT or SGPT (Grade 3 >5-20 x ULN, Grade 4 >20 x ULN), albumin (Grade 3 <20 g/L)

Adjuvant Treatment of GIST

In Study 1, the majority of both Gleevec and placebo treated patients experienced at least one adverse reaction at some time. The most frequently reported adverse reactions were similar to those reported in other clinical studies in other patient populations and include diarrhea, fatigue, nausea, edema, decreased hemoglobin, rash, vomiting, and abdominal pain. No new adverse reactions were reported in the adjuvant GIST treatment setting that had not been previously reported in other patient populations including patients with unresectable and/or malignant metastatic GIST. Drug was discontinued for adverse reactions in 57 patients (17%) and 11 patients (3%) of the Gleevec and placebo treated patients respectively. Edema, gastrointestinal disturbances (nausea, vomiting, abdominal distention and diarrhea), fatigue, low hemoglobin, and rash were the most frequently reported adverse reactions at the time of discontinuation.

In Study 2, discontinuation of therapy due to adverse reactions occurred in 15 patients (8%) and 27 patients (14%) of the Gleevec 12-month and 36-month treatment arms, respectively. As in previous trials the most common adverse reactions were diarrhea, fatigue, nausea, edema, decreased hemoglobin, rash, vomiting, and abdominal pain.

Adverse reactions, regardless of relationship to study drug, that were reported in at least 5% of the patients treated with Gleevec are shown in Table 12 (Study 1) and Table 13 (Study 2). There were no deaths attributable to Gleevec treatment in either trial.

Reference ID: 3511243

	Cramps	4	<1	2	<1	0
Skin	Hirsutism	21	<1	21	28	45
	Acne	6	8	2	2	1
Central Nervous System	Tremor	12	0	21	31	55
	Convulsions	3	1	1	4	5
	Headache	2	<1	2	15	4
Gastrointestinal	Gum Hyperplasia	4	0	9	5	16
	Diarrhea	3	<1	3	4	8
	Nausea/Vomiting	2	<1	4	10	4
	Hepatotoxicity	<1	<1	4	7	4
	Abdominal Discomfort	<1	0	<1	7	0
Autonomic Nervous System	Paresthesia	3	0	1	2	1
	Flushing	<1	0	4	0	4
Hematopoietic	Leukopenia	2	19	<1	6	0
	Lymphoma	<1	0	1	6	1
Respiratory	Sinusitis	<1	0	4	3	7
Miscellaneous	Gynecomastia	<1	0	<1	4	3

Among 705 kidney transplant patients treated with cyclosporine oral solution (Sandimmune) in clinical trials, the reason for treatment discontinuation was renal toxicity in 5.4%, infection in 0.9%, lack of efficacy in 1.4%, acute tubular necrosis in 1.0%, lymphoproliferative disorders in 0.3%, hypertension in 0.3%, and other reasons in 0.7% of the patients.

The following reactions occurred in 2% or less of cyclosporine-treated patients: allergic reactions, anemia, anorexia, confusion, conjunctivitis, edema, fever, brittle fingernails, gastritis, hearing loss, hiccups, hyperglycemia, migraine (Neoral), muscle pain, peptic ulcer, thrombocytopenia, tinnitus.

The following reactions occurred rarely: anxiety, chest pain, constipation, depression, hair breaking, hematuria, joint pain, lethargy, mouth sores, myocardial infarction, night sweats, pancreatitis, pruritus, swallowing difficulty, tingling, upper GI bleeding, visual disturbance, weakness, weight loss.

Patients receiving immunosuppressive therapies, including cyclosporine and cyclosporine -containing regimens, are at increased risk of infections (viral, bacterial, fungal, parasitic). Both generalized and localized infections can occur. Pre-existing infections may also be aggravated. Fatal outcomes have been reported. *(See WARNINGS)*

Infectious Complications in Historical Randomized Studies in Renal Transplant Patients Using Sandimmune

Complication	Cyclosporine Treatment (N=227) % of Complications	Azathioprine with Steroids* (N=228) % of Complications
Septicemia	5.3	4.8
Abscesses	4.4	5.3
Systemic Fungal Infection	2.2	3.9
Local Fungal Infection	7.5	9.6
Cytomegalovirus	4.8	12.3
Other Viral Infections	15.9	18.4
Urinary Tract Infections	21.1	20.2
Wound and Skin Infections	7.0	10.1
Pneumonia	6.2	9.2

*Some patients also received ALG.

Postmarketing Experience, Kidney, Liver and Heart Transplantation

Hepatotoxicity

Cases of hepatotoxicity and liver injury including cholestasis, jaundice, hepatitis and liver failure; serious and/or fatal outcomes have been reported. *(See WARNINGS, Hepatotoxicity)*

Increased Risk of Infections

Reference ID: 3722656

Figure 2.6: Examples of images in TableBank

a variety of domains, including business documents, official filings, research papers, and more. The 417,234 high-quality labeled tables and their original documents from various domains make up the 417,234-table TableBank dataset. Figure 2.6 displays a few samples from this dataset.

2.2.5 PubTables-1M

A wide range of modeling techniques can benefit from using PubTables-1M [43], which contains almost one million tables from scientific articles, supports multiple input modalities, and contains comprehensive header and location information for table structures. It also uses a novel canonicalization procedure to address over-segmentation, a significant source of ground truth inconsistency seen in earlier datasets.

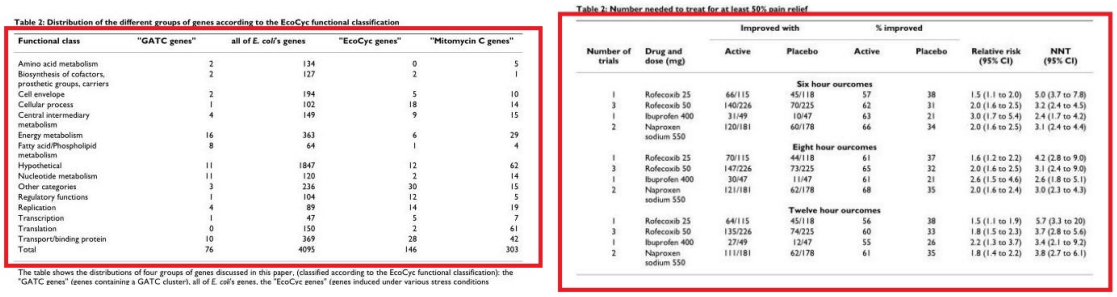


Table 2: Distribution of the different groups of genes according to the EcoCyc functional classification

Functional class	"GATC genes"	all of *E. coli*'s genes	"EcoCyc genes"	"Mitomycin C genes"
Amino acid metabolism	2	134	0	5
Biosynthesis of cofactors, prosthetic groups, carriers	2	127	2	1
Cell envelope	2	194	5	10
Cellular process	1	102	18	14
Central intermediary metabolism	4	149	9	15
Energy metabolism	16	363	6	29
Fatty acid/Phospholipid metabolism	8	64	1	4
Hypothetical	11	1847	12	62
Nucleotide metabolism	11	120	2	14
Other categories	3	236	30	15
Regulatory functions	1	104	12	5
Replication	4	89	14	19
Transcription	1	47	5	7
Translation	0	150	2	61
Transport/binding protein	10	369	28	42
Total	76	4095	146	303

The table shows the distributions of four groups of genes discussed in this paper, (classified according to the EcoCyc functional classification): the "GATC genes" (genes containing a GATC cluster), all of *E. coli*'s genes, the "EcoCyc genes" (genes induced under various stress conditions

Table 2: Number needed to treat for at least 50% pain relief

Number of trials	Drug and dose (mg)	Improved with		% improved		Relative risk (95% CI)	NNT (95% CI)
		Active	Placebo	Active	Placebo		
		Six hour ourcomes					
1	Rofecoxib 25	66/115	45/118	57	38	1.5 (1.1 to 2.0)	5.0 (3.7 to 7.8)
3	Rofecoxib 50	140/226	70/225	62	31	2.0 (1.6 to 2.5)	3.2 (2.4 to 4.5)
1	Ibuprofen 400	31/49	10/47	63	21	3.0 (1.7 to 5.4)	2.4 (1.7 to 4.2)
2	Naproxen sodium 550	120/181	60/178	66	34	2.0 (1.6 to 2.5)	3.1 (2.4 to 4.4)
		Eight hour ourcomes					
1	Rofecoxib 25	70/115	44/118	61	37	1.6 (1.2 to 2.2)	4.2 (2.8 to 9.0)
3	Rofecoxib 50	147/226	73/225	65	32	2.0 (1.6 to 2.5)	3.1 (2.4 to 9.0)
1	Ibuprofen 400	30/47	11/47	61	21	2.6 (1.5 to 4.6)	2.6 (1.8 to 5.1)
2	Naproxen sodium 550	121/181	62/178	68	35	2.0 (1.6 to 2.4)	3.0 (2.3 to 4.3)
		Twelve hour ourcomes					
1	Rofecoxib 25	64/115	45/118	56	38	1.5 (1.1 to 1.9)	5.7 (3.3 to 20)
3	Rofecoxib 50	135/226	74/225	60	33	1.8 (1.5 to 2.3)	3.7 (2.8 to 5.6)
1	Ibuprofen 400	27/49	12/47	55	26	2.2 (1.3 to 3.7)	3.4 (2.1 to 9.2)
2	Naproxen sodium 550	111/181	62/178	61	35	1.8 (1.4 to 2.2)	3.8 (2.7 to 6.1)

Figure 2.7: Examples of images in PubTables-1M

We show that these enhancements significantly boost training performance and provide a more accurate prediction of model performance during evaluation for recognizing table structures. The authors also demonstrate that all three tasks—detection, structure recognition, and functional analysis—are successfully accomplished by transformer-based object detection models trained on PubTables-1M without the need for any additional customization. Figure 2.7 shows two samples from this dataset.

2.2.6 PubLayNet

PubLayNet [38] is a dataset for document layout analysis. It contains images of research papers and articles and annotations for various elements on a page such as "text", "list", "figure" etc in these research paper images. The dataset was obtained by automatically matching the XML representations and the content of over 1 million PDF articles that are publicly available on PubMed Central. It contains 358,353 images divided into 335,703 training images, 11,245 validation images, and 11,405 test images. Figure 2.8 shows two examples of this dataset.

2.2.7 TNCR

TNCR [40] is a new table collection containing images of varied quality gathered from free access websites The TNCR dataset may be used to recognize tables in scanned document pictures and classify them into five categories. TNCR has roughly 6621 photos and 9428 captioned tables. To build numerous robust baselines, this work used state-of-the-art deep learning-based approaches for table detection. On

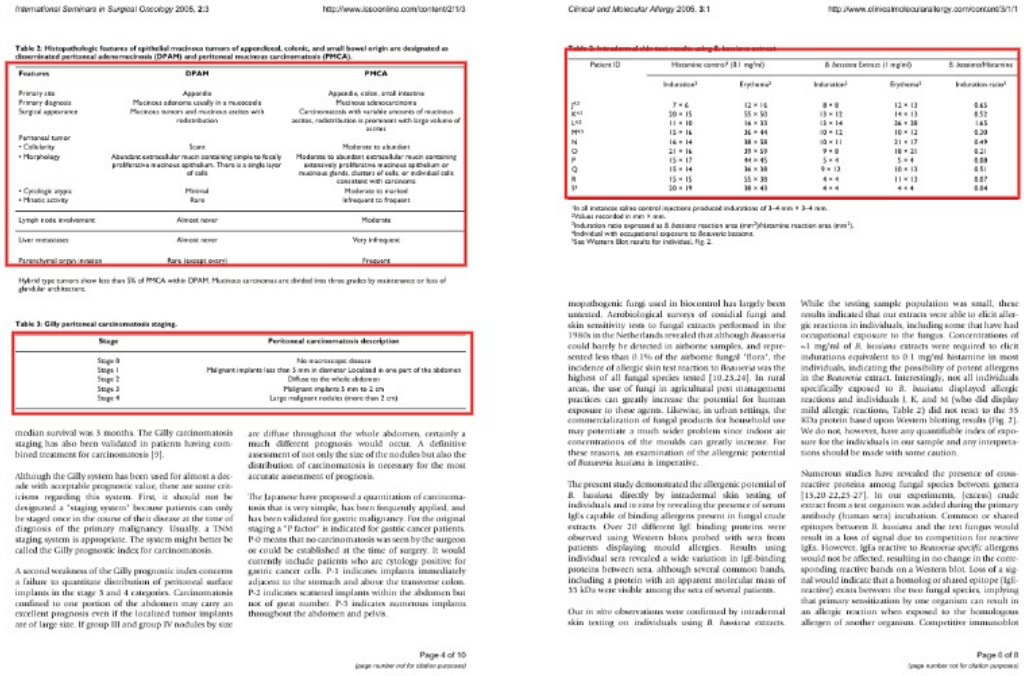


Figure 2.8: Examples of images in PubLayNet

the TNCR dataset, Deformable DERT with Resnet-50 Backbone Network delivers the best results compared to other methods, with an accuracy of 86.7%, recall of 89.6%, and F1-score of 88.1%. A few samples from this dataset are shown in figure 2.9.

2.2.8 Table classification dataset

As our pipeline is designed to be modular and divide and conquer different types of tables, apart from table detection and table structure recognition, there is also the additional need to evaluate the classification process. For this task, we constructed a small dataset by combining other listed public datasets and hand-labeled the data that was not labeled. In our implementation, the tables are classified into 2 types: "bordered" for tables with discernible and complete borders and "non-bordered" for ones with partial or no borders.

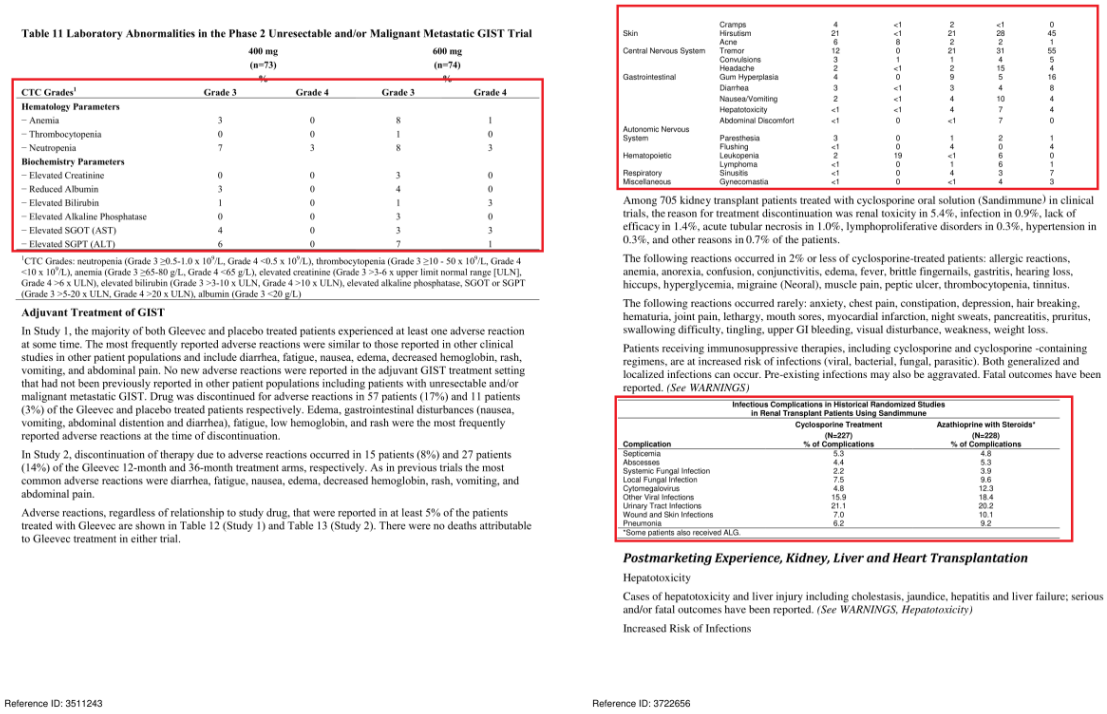


Table 11 Laboratory Abnormalities in the Phase 2 Unresectable and/or Malignant Metastatic GIST Trial

	400 mg (n=73) %		600 mg (n=74) %	
CTC Grades[1]	**Grade 3**	**Grade 4**	**Grade 3**	**Grade 4**
Hematology Parameters				
– Anemia	3	0	8	1
– Thrombocytopenia	0	0	1	0
– Neutropenia	7	3	8	3
Biochemistry Parameters				
– Elevated Creatinine	0	0	3	0
– Reduced Albumin	3	0	4	0
– Elevated Bilirubin	1	0	1	3
– Elevated Alkaline Phosphatase	0	0	3	0
– Elevated SGOT (AST)	4	0	3	3
– Elevated SGPT (ALT)	6	0	7	1

[1]CTC Grades: neutropenia (Grade 3 ≥0.5-1.0 x 10^9/L, Grade 4 <0.5 x 10^9/L), thrombocytopenia (Grade 3 ≥10 - 50 x 10^9/L, Grade 4 <10 x 10^9/L), anemia (Grade 3 ≥65-80 g/L, Grade 4 <65 g/L), elevated creatinine (Grade 3 >3-6 x upper limit normal range [ULN], Grade 4 >6 x ULN), elevated bilirubin (Grade 3 >3-10 x ULN, Grade 4 >10 x ULN), elevated alkaline phosphatase, SGOT or SGPT (Grade 3 >5-20 x ULN, Grade 4 >20 x ULN), albumin (Grade 3 <20 g/L)

Adjuvant Treatment of GIST

In Study 1, the majority of both Gleevec and placebo treated patients experienced at least one adverse reaction at some time. The most frequently reported adverse reactions were similar to those reported in other clinical studies in other patient populations and include diarrhea, fatigue, nausea, edema, decreased hemoglobin, rash, vomiting, and abdominal pain. No new adverse reactions were reported in the adjuvant GIST treatment setting that had not been previously reported in other patient populations including patients with unresectable and/or malignant metastatic GIST. Drug was discontinued for adverse reactions in 57 patients (17%) and 11 patients (3%) of the Gleevec and placebo treated patients respectively. Edema, gastrointestinal disturbances (nausea, vomiting, abdominal distention and diarrhea), fatigue, low hemoglobin, and rash were the most frequently reported adverse reactions at the time of discontinuation.

In Study 2, discontinuation of therapy due to adverse reactions occurred in 15 patients (8%) and 27 patients (14%) of the Gleevec 12-month and 36-month treatment arms, respectively. As in previous trials the most common adverse reactions were diarrhea, fatigue, nausea, edema, decreased hemoglobin, rash, vomiting, and abdominal pain.

Adverse reactions, regardless of relationship to study drug, that were reported in at least 5% of the patients treated with Gleevec are shown in Table 12 (Study 1) and Table 13 (Study 2). There were no deaths attributable to Gleevec treatment in either trial.

Reference ID: 3511243

	Cramps	4	<1	2	<1	0
Skin	Hirsutism	21	<1	21	28	45
	Acne	6	8	2	2	1
Central Nervous System	Tremor	12	0	21	31	55
	Convulsions	3	1	1	4	5
	Headache	2	<1	2	15	4
Gastrointestinal	Gum Hyperplasia	4	0	9	5	16
	Diarrhea	3	<1	3	4	8
	Nausea/Vomiting	2	<1	4	10	4
	Hepatotoxicity	<1	<1	4	7	4
	Abdominal Discomfort	<1	0	<1	7	0
Autonomic Nervous System	Paresthesia	3	0	1	2	1
	Flushing	<1	0	4	0	4
Hematopoietic	Leukopenia	2	19	<1	6	0
	Lymphoma	<1	0	1	6	1
Respiratory	Sinusitis	<1	0	4	3	7
Miscellaneous	Gynecomastia	<1	0	<1	4	3

Among 705 kidney transplant patients treated with cyclosporine oral solution (Sandimmune) in clinical trials, the reason for treatment discontinuation was renal toxicity in 5.4%, infection in 0.9%, lack of efficacy in 1.4%, acute tubular necrosis in 1.0%, lymphoproliferative disorders in 0.3%, hypertension in 0.3%, and other reasons in 0.7% of the patients.

The following reactions occurred in 2% or less of cyclosporine-treated patients: allergic reactions, anemia, anorexia, confusion, conjunctivitis, edema, fever, brittle fingernails, gastritis, hearing loss, hiccups, hyperglycemia, migraine (Neoral), muscle pain, peptic ulcer, thrombocytopenia, tinnitus.

The following reactions occurred rarely: anxiety, chest pain, constipation, depression, hair breaking, hematuria, joint pain, lethargy, mouth sores, myocardial infarction, night sweats, pancreatitis, pruritus, swallowing difficulty, tingling, upper GI bleeding, visual disturbance, weakness, weight loss.

Patients receiving immunosuppressive therapies, including cyclosporine and cyclosporine -containing regimens, are at increased risk of infections (viral, bacterial, fungal, parasitic). Both generalized and localized infections can occur. Pre-existing infections may also be aggravated. Fatal outcomes have been reported. *(See WARNINGS)*

Infectious Complications in Historical Randomized Studies in Renal Transplant Patients Using Sandimmune

Complication	Cyclosporine Treatment (N=227) % of Complications	Azathioprine with Steroids* (N=228) % of Complications
Septicemia	5.3	4.8
Abscesses	4.4	5.3
Systemic Fungal Infection	2.2	3.9
Local Fungal Infection	7.5	9.6
Cytomegalovirus	4.8	12.3
Other Viral Infections	15.9	18.4
Urinary Tract Infections	21.1	20.2
Wound and Skin Infections	7.0	10.1
Pneumonia	6.2	9.2

*Some patients also received ALG.

Postmarketing Experience, Kidney, Liver and Heart Transplantation

Hepatotoxicity

Cases of hepatotoxicity and liver injury including cholestasis, jaundice, hepatitis and liver failure; serious and/or fatal outcomes have been reported. *(See WARNINGS, Hepatotoxicity)*

Increased Risk of Infections

Reference ID: 3722656

Figure 2.9: Examples of images in TNCR

2.3 Evaluation metrics

2.3.1 Table detection

To choose the right metrics for the TD task, we carefully inspect the metrics that previous table detection models are evaluated on. Most researchers use the IoU-based method for their evaluation while others use the region-based one.

IoU-based method

Intersection of Union Figure 2.10 shows how to calculate IoU between two bounding boxes. Mathematically, IoU is calculated as:

$$IoU = \frac{|A \cap B|}{|A \cup B|}$$

where A and B are two bounding box areas.

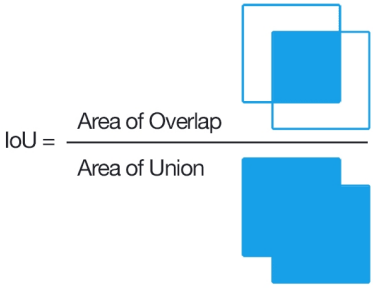


Figure 2.10: IoU formula

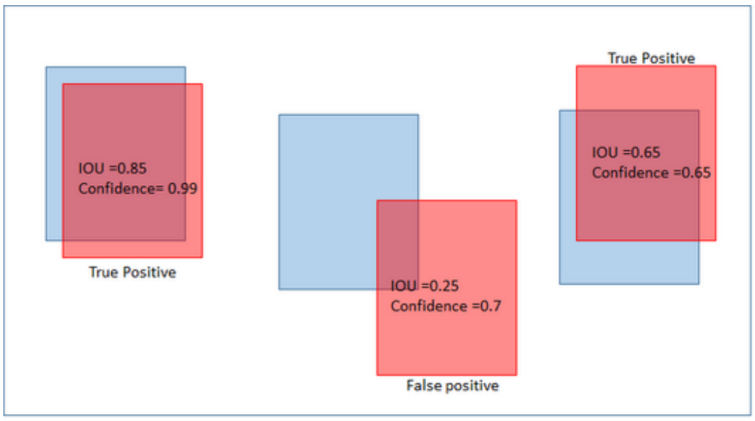


Figure 2.11: Examples of IoU values between bounding boxes

Examples of IoU values between bounding boxes are shown in figure 2.11 **Recall, precision, and F1-score** Recall, precision, and F1-score are calculated based on the number of true positives (TP) in our predictions. TP is determined by comparing the IoU between a predicted box and a ground truth with a chosen threshold, which ranges from 0.5 up to 0.9. After determining the number of TPs, we calculate recall, precision, and F1-score as below:

$$Precision = \frac{True\ Positive(TP)}{True\ Positive(TP) + False\ Positive(FP)}$$

$$Recall = \frac{True\ Positive(TP)}{True\ Positive(TP) + False\ Negative(FN)}$$

$$\text{F1-score} = \frac{2 * (Precision * Recall)}{Precision + Recall}$$

Region-based method

The region-based method also calculates recall, precision, and F1-score yet on the areas of bounding boxes rather than overlapping areas of pairs of bounding boxes. There is also no threshold required to calculate the metrics. Recall, precision, and F1-score in region-based method is calculated as below:

$$Precision = \frac{Area\ of\ Ground\ truth\ regions\ in\ Detected\ regions}{Area\ of\ all\ Detected\ table\ regions}$$

$$Recall = \frac{Area\ of\ Ground\ truth\ regions\ in\ Detected\ regions}{Area\ of\ all\ Ground\ truth\ table\ regions}$$

$$\text{F1-score} = \frac{2 * (Precision * Recall)}{Precision + Recall}$$

Average Precision

Average Precision (AP) is calculated based on the precision-recall curve. A precision-recall curve is a graph that illustrates the trade-off between precision and recall for a binary classification model. In the curve, the x-axis represents recall and the y-axis represents precision. The curve shows the relationship between precision and recall as the threshold for classifying positive examples is varied. Generally, a model with better classification performance will have a precision-recall curve that is closer to the upper right corner of the graph. Figure 2.12 shows an example of a precision-recall curve.

The general definition for the Average Precision (AP) is finding the area under the precision-recall curve:

$$AP = \int_1^0 p(r)dr$$

The following example demonstrates how to calculate precision and recall values for plotting the precision-recall curve. In this case, the dataset only includes 8 bicycles. We gather all predictions made for bicycles in all images and rank them

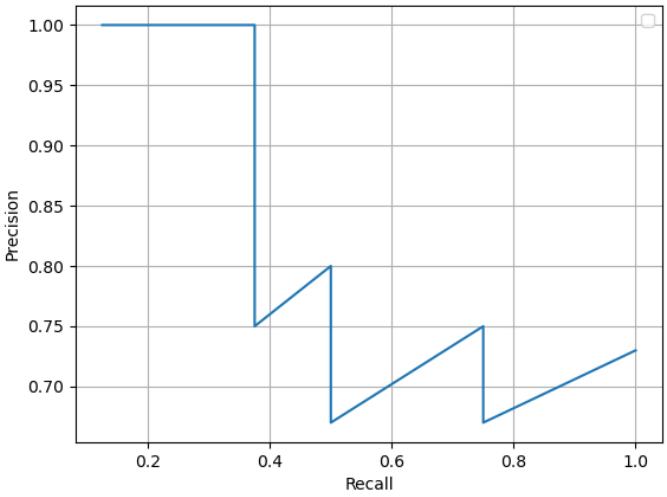


Figure 2.12: Example of a precision-recall curve

in descending order based on their predicted confidence level. Table 2.1 shows the precision and recall values at each step when a new prediction is examined. The second column indicates whether the prediction is correct or not. In this example, a prediction is correct if the Intersection over Union (IoU) is greater than or equal to 0.5.

This is how to calculate precision and recall using the row ranked #4. To calculate precision, we need to divide the number of true positives (TP) by the total predicted positives, which is 3/4 or 0.75 in this case. Recall is the ratio of true positives to the total actual positives, which is 3/8 or 0.375. Generally, recall values increase as we move down the prediction ranking. Precision, on the other hand, can fluctuate, decreasing with false positives and increasing again with true positives.

It's worth noting that both precision and recall are always between 0 and 1, and the same goes for AP. Before calculating AP for object detection, we often smooth out the zigzag pattern first. Figure 2.13 illustrates the precision-recall curve before and after being smoothed out. Graphically, at each recall level, we replace each precision value with the maximum precision value to the right of that recall level. This transforms the blue line into the orange lines, and the curve decreases monotonically instead of showing a zigzag pattern. Calculating AP value in this way makes it less susceptible

Rank	Correct?	Precision	Recall
1	True	1	0.125
2	True	1	0.25
3	True	1	0.375
4	False	0.75	0.375
5	False	0.8	0.5
6	True	0.67	0.5
7	False	0.71	0.625
8	True	0.75	0.75
9	True	0.67	0.75
10	False	0.7	0.875
11	True	0.73	1
12	True	0.67	1

Table 2.1: Table illustrating how to calculate precision and recall values for precision-recall curve

to small variations in the ranking. Mathematically, we replace the precision value for recall $\hat{r}$ with the maximum precision for any recall that is greater than or equal to $recall \geq \hat{r}$.

$$p_{interp} = \underset{\tilde{r} \geq r}{max}\ p(\tilde{r})$$

For evaluation, we use AP50, AP75, or AP95 with different values behind AP. They are specific instances of AP where the Intersection over Union (IoU) threshold used for evaluation is set to different values. IoU is a measure of the overlap between two bounding boxes. It is calculated as the ratio of the area of intersection between the two boxes to the area of their union. When evaluating an object detection model, a detection is considered a true positive if its IoU with the ground truth bounding box is above a certain threshold. By setting this threshold to different values, we can obtain different AP values. For AP[0.5:0.95], it is calculated by taking the average of AP values at different thresholds varying from 0.5 to 0.95.

Average Recall

Compared to AP, Average Recall (AR) is easier to calculate. This metric allows you to measure the proportion of true positive detections in your model while

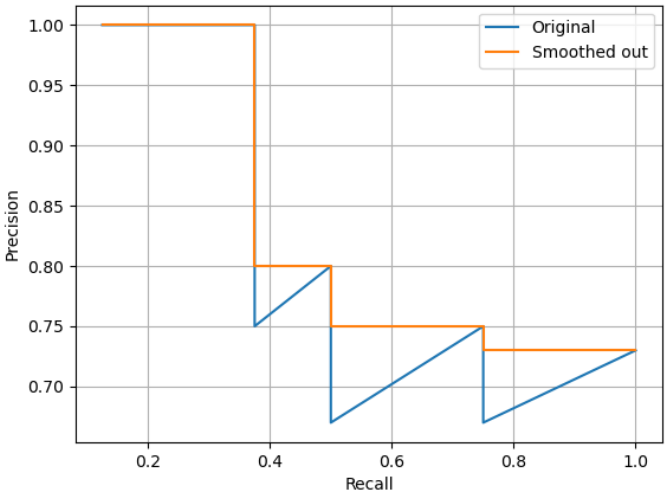


Figure 2.13: Precision-recall before and after being smoothed out

minimizing the number of false negatives.

To calculate AR, you first need to take a set of different IOU thresholds. These thresholds are used to determine the minimum IOU required between a predicted bounding box and a ground truth bounding box for it to be considered a true positive. Then, you must compute the recall of your model by detecting each category at these specific IOU thresholds.

After that, for each class, average the recall at each of the IOU thresholds. This gives you a sense of how well your model performs across different levels of IOU thresholds. Finally, take the average of this average for each class. This gives you the Average Recall (AR) for your model across all categories.

$$\frac{1}{n} \cdot \sum_{i=0}^{n-1} \cdot \frac{1}{10} \sum_{iou \in [0.5:0.05:0.95]} recall_{iou}^{i}$$

where n is the number of classes

2.3.2 Table structure recognition

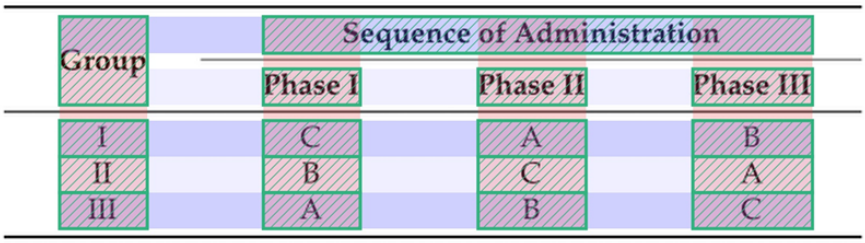
In this task, after careful consideration, we decided to choose Grid table similarity metric for table structure recognition(GriTS) published by Microsoft [51] as our metric because of its advantages. Firstly, it is a metric that encompass all subtasks needed to be addressed by any metric for table structure recognition that is: Cell topology recognition(layout of the cells, the rows and columns each cell occupies over a 2-D grid), Cell content recognition(text content), Cell location recognition(absolute coordinates). A sample is provided in figure 2.14.

In this metric, to define a similarity between tables, we use below equation with a particular choice of similarity function and a particular matrix of entries(property of each cell) to compare. This has the general form:

$$\mathrm{GriTS}_f(\mathbf{A}, \mathbf{B}) = \frac{2 \sum_{i,j} f\left(\tilde{\mathbf{A}}_{i,j}, \tilde{\mathbf{B}}_{i,j}\right)}{|\mathbf{A}| + |\mathbf{B}|},$$

where A and B represent tables-matrices of grid cells-and f is a similarity function between the grid cells' properties.

For each subtask, the grid cell property and similarity function is chosen differently. For cell topology, a custom set of information of each cell based on rowspan and columnspan is chosen as grid cell property and Intersection over Union is chosen as similarity function. For cell content, text content of each cell is chosen as grid cell property while a modified Normalized longest common subsequence algorithm is chosen as similarity function. For cell location, grid cell property is the absolute coordinates of the cell on the document with Intersection over Union is once again being used as similarity function. The authors of the original propose a heuristic approach to determine the final score by factoring the problem. Instead of determining the optimal similar subsequences of rows and columns jointly for each matrix, we determine the optimal subsequences of rows and the optimal subsequences of columns independently. This method uses dynamic programming in a nested manner, running twice: once to determine the optimal subsequences of rows and then to determine the optimal subsequences of columns. Further details can be read at [51].



(a) An example presentation table from the PubTables-1M dataset.

Group	Sequence of Administration	Sequence of Administration	Sequence of Administration
Group	Phase I	Phase II	Phase III
I	C	A	B
II	B	C	A
III	A	B	C

[0, 0, 1, 2]	[0, 0, 3, 1]	[-1, 0, 2, 1]	[-2, 0, 1, 1]
[0, -1, 1, 1]	[0, 0, 1, 1]	[0, 0, 1, 1]	[0, 0, 1, 1]
[0, 0, 1, 1]	[0, 0, 1, 1]	[0, 0, 1, 1]	[0, 0, 1, 1]
[0, 0, 1, 1]	[0, 0, 1, 1]	[0, 0, 1, 1]	[0, 0, 1, 1]
[0, 0, 1, 1]	[0, 0, 1, 1]	[0, 0, 1, 1]	[0, 0, 1, 1]

[136.42, 477.25, 160.62, 501.45]	[185, 477.25, 470.89, 487.22]	[185, 477.25, 470.89, 487.22]	[185, 477.25, 470.89, 487.22]
[136.42, 477.25, 160.62, 501.45]	[185, 491.48, 271.9, 501.45]	[284.5, 491.48, 371.39, 501.45]	[384, 491.48, 470.89, 501.45]
[136.42, 505.82, 160.62, 515.72]	[185, 505.82, 271.9, 515.72]	[284.5, 505.82, 371.39, 515.72]	[384, 505.82, 470.89, 515.72]
[136.42, 515.73, 160.62, 525.63]	[185, 515.73, 271.9, 525.63]	[284.5, 515.73, 371.39, 525.63]	[384, 515.73, 470.89, 525.63]
[136.42, 525.64, 160.62, 535.53]	[185, 525.64, 271.9, 535.53]	[284.5, 525.64, 371.39, 535.53]	[384, 525.64, 470.89, 535.53]

Figure 2.14: An example presentation table from the PubTables-1M dataset, along with corresponding ground truth grid cell matrices for different GriTS metrics. Each matrix entry corresponds to one grid cell. Entries that correspond to spanning cells are shaded darker for illustrative purposes.

Chapter 3

Theoretical Background

This chapter delves into the theoretical background of the proposed technology and model, presenting the necessary concepts and our knowledge and practical experience on the topics. This chapter provides a deeper understanding of the foundational theories that underpin the project, allowing readers to better appreciate the significance and potential of the proposed solution. Through a detailed exploration of various theoretical concepts, this chapter aims to establish a strong theoretical foundation for the project and provide readers with the necessary background knowledge to comprehend the subsequent chapters.

3.1 Object detection algorithms

The task of detecting and recognizing an unknown number of individual objects within an image,called object detection has always been a hot topic of interest in artificial intelligence field, especially in computer vision.

Early on, due to the lack of effective image representation, most of the early object detection algorithms were developed with a lot of handcrafted features [52–54]. These methods performance, however, reached a plateau as the hand-crafted features became saturated in the 2010s. Recent advances in deep learning have made models capable of learning robust and high-level feature representations of an image which in turn help the field of object detection move forward.In this deep learning era, object detection is grouped into two catergories: “one-stage” and ”two-stage” detection.

One of the new ideas that is of great influence is (Girshick 2013) and its subsequent improvement by multiple authors, such as Fast is RCNN (Girshick 2015) or Faster RCNN (Ren 2015). The main idea is to start off with the extraction of a set of object candidate boxes by selective search. Then each proposal is rescaled and fed into a pre-trained convolutional neural network model for feature extraction. The presence of an object within each region and its categories are finally predicted by some classifiers.

Another algorithm considered to be state-of-the-art in object detection is You Only Look Once (YOLO) [Redmon 2015]. The YOLO algorithm looks at parts of the image that have high probabilities of containing the object instead of looking at the complete image and uses regions to localize the object like previous approaches. This helps YOLO directly optimize detection performance. With YOLO, multiple bounding boxes and class probabilities are predicted at the same time, and the algorithm also predicts all bounding boxes across all classes for an image simultaneously. The input image is divided into an S $\times$ S grid where should the center of an object falls into a grid cell, that grid cell is responsible for detecting that object (3.1). Each grid cell has a number of confidence scores with respect to its predicted number of boxes. These scores indicate the confidence of the model in the fact that the box contain an object and also how accurate is the box predicted.

3.2 Graph-based Neural network

Recently, Graph Neural Network (GNNnetworks and)domains, has gained increasing popularity in various domains from social network, recommender system to natural science. In order to understand GNN, we must first understand the definition of graph. A graph G can be well described by V being a set of nodes and E being a set of edges

$$\mathbf{G} = (\mathbf{V}, \mathbf{E})$$

Edges can be either directed or undirected depending on whether there are directional dependencies between vertices; the vertices are also sometimes called nodes. The graph neural network was first proposed by Scarselli et al. in their paper titled "The Graph Neural Network Model." Since then, many variants of GNNs have

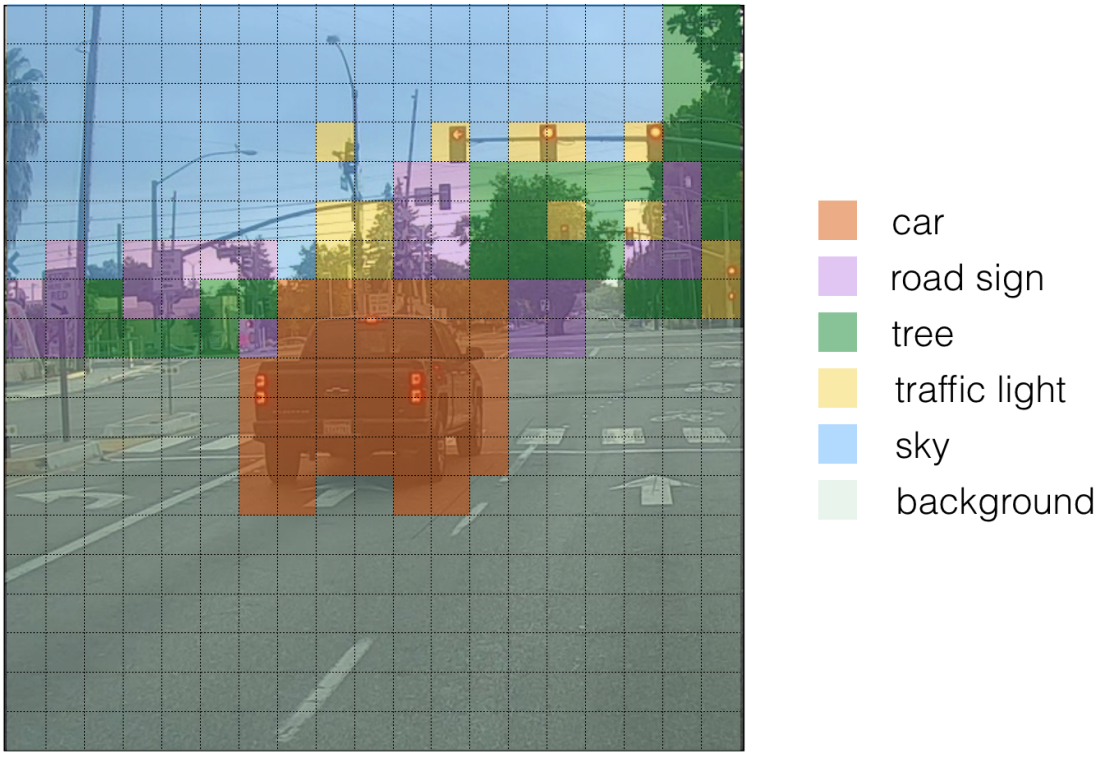


Figure 3.1: An example grid with coloured cells according to the class with highest predicted probability.

been developed and adapted to various domains. In general, there are three general types of tasks on graphs: graph-level, node-level, and edge-level. In a graph-level task, output is a single property for a whole graph. For a node-level task, we predict some properties for each individual node. Finally, we want to predict the property or presence of edges in an edge-level task.

In its simplest form, a GNN contains two main stages: learning new embeddings for all graph attributes (nodes, edges, and global) and making predictions by pooling. We could also make our learned embeddings aware of graph connectivity with message passing, therefore making more sophisticated predictions. Message passing is where neighboring nodes or edges exchange information and influence each other's updated embeddings. An example of a simple GNN is illustrated in 3.2

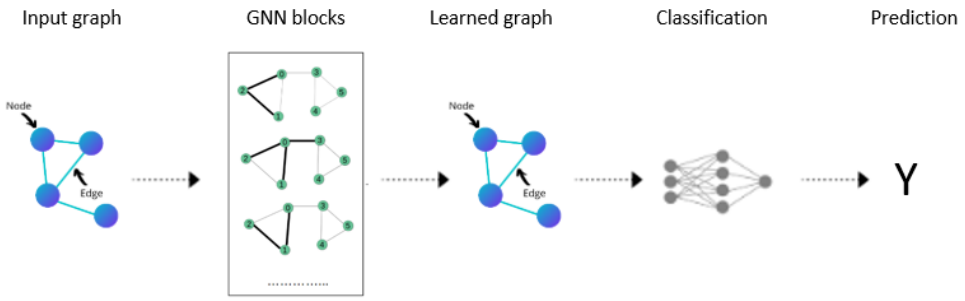


Figure 3.2: An simple end-to-end prediction with a GNN model.

3.3 Convolutional Neural Network

In recent years, convolution neural networks (CNN) have been the driving force behind many achievements in computer vision and artificial intelligence in general. first proposed by the famous scientist Yann LeCun in "Backpropagation Applied to Handwritten Zip Code Recognition" **??**. CNN takes advantage of the fact that the input consists of images by having neurons arranged in three dimensions: width, height, and depth. With this architecture, CNN is able to successfully capture the spatial and temporal dependencies in an image through the application of relevant filters. Thanks to the reduction in the number of parameters involved and the reusability of weights, the network can be trained to capture the sophistication of the image better.

In its simplest form, a CNN architecture consists of three main types of layers: Convolutional Layer, Pooling Layer, and Fully-Connected Layer. The main component is the Convolutional layers which are made up of a set of filters that are applied to an input image. The output of that operation is a feature map, which is a representation of the input image with the filters applied. These 3 types of layers are stacked to form a full CNN architecture. The way a simple CNN architecture works is depicted in 3.3.

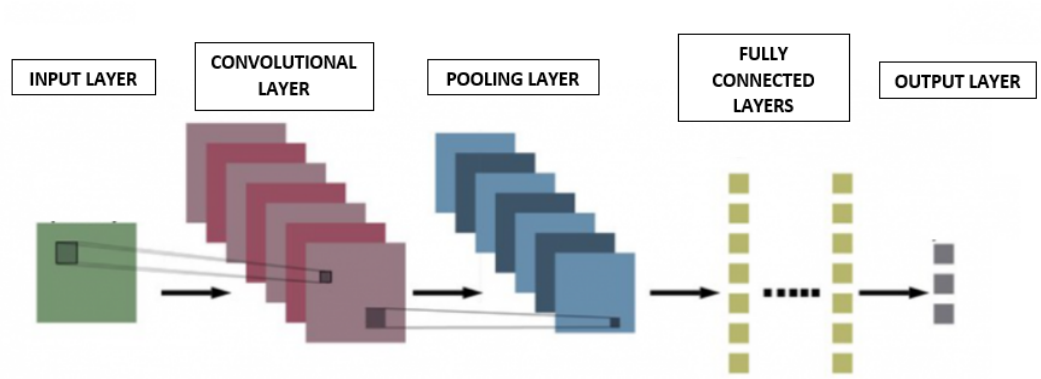


Figure 3.3: A simple CNN architecture.

3.4 Transfer Learning and Pretrained Models

Transfer learning is a technique in deep learning that involves taking a model trained for one task and fine-tuning it for a different task. This technique has become increasingly popular in recent years due to the difficulty of training large-scale deep learning models from scratch. Pretrained models serve as a starting point for transfer learning, as they allow researchers to leverage the knowledge learned by models trained on large datasets.

Transfer learning is a widely used technique in deep learning due to the significant resources needed to train these models on large and complex datasets. It involves training a base network on one task and dataset, then transferring the learned features to a second network for training on a different task and dataset. This approach is effective if the learned features are general and applicable to both tasks. This type of transfer learning, known as inductive transfer, narrows the range of possible models in a beneficial way by using a model trained on a related task (figure 3.4).

Transfer learning is an optimization technique that saves time or improves performance. In general, it may not be immediately apparent whether using transfer learning will benefit your domain until after you've developed and evaluated your model. Figure 3.5 shows that there are three possible benefits to look for when using transfer learning:

- **Higher starting point**: The initial skill level of the source model is higher

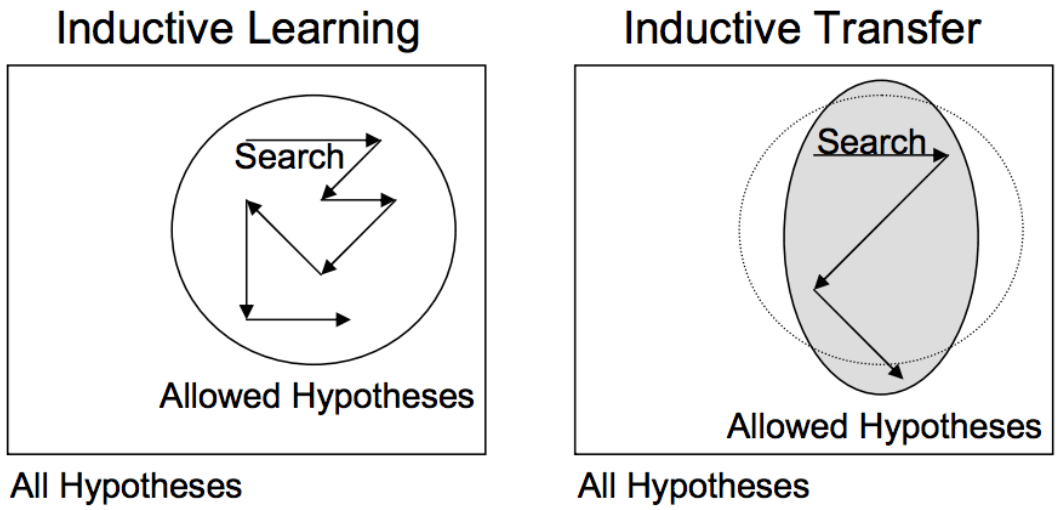


Figure 3.4: Inductive transfer

before refining the model.

- **Faster skill improvement**: The rate of skill improvement during training of the source model is steeper.
- **Better final performance**: The converged skill level of the trained model is better than it would be without refinement.

Ideally, all three benefits from a successful application of transfer learning would be accomplished. It's worth considering transfer learning if there is abundant data for a related task and the resources to develop or reuse a model for that task. Alternatively, a pre-trained model can be used as a starting point for one's own model. On some problems where there is limited data, transfer learning can enable the development of skillful models that could not be developed otherwise. Choosing the source data or source model may require domain expertise and/or intuition developed via experience.

Pretrained models are neural networks that have been trained on a large dataset, such as ImageNet, and are then made publicly available. These models have learned to recognize a wide range of features in images, which can be used as a starting point for training new models. By using a pretrained model, researchers can leverage the

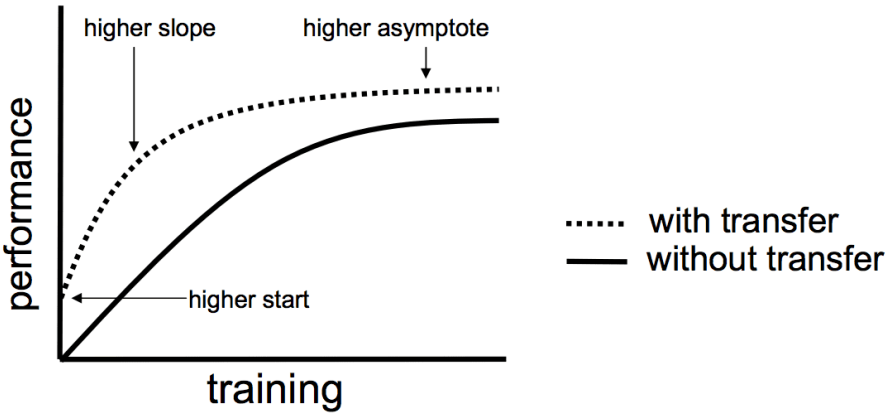


Figure 3.5: Three ways in which transfer learning might improve learning

knowledge learned by the model on a large dataset and build on top of it for their own task.

There are many benefits to using transfer learning and pretrained models. One of the main benefits is that it can significantly reduce the amount of time and resources needed to train a deep learning model. Training a large-scale model from scratch can take days or even weeks, while fine-tuning a pretrained model can take just a few hours or less. In addition, transfer learning can improve the performance of a model on a new task, as it allows the model to leverage the knowledge learned by the pretrained model on a large dataset.

There are several approaches to transfer learning, depending on the similarity of the tasks and the amount of labeled data available for the new task. One approach is to use the pretrained model as a feature extractor, and then train a new classifier on top of the extracted features. Another approach is to fine-tune the entire model, including the pretrained layers, on the new task. The choice of approach depends on the specific task and the amount of labeled data available.

One popular pretrained model is VGG16, which was trained on the ImageNet dataset for image classification. VGG16 has 16 layers and is known for its simplicity and high accuracy on the ImageNet dataset. Another popular pretrained model is ResNet, which was trained on the same dataset but has a much deeper architecture, with up to 152 layers. ResNet is known for its ability to train very deep models

without suffering from the vanishing gradient problem.

In addition to these models, there are many other pretrained models available for a wide range of tasks, including object detection, segmentation, and natural language processing. For example, BERT is a pretrained model for natural language processing that has achieved state-of-the-art performance on a wide range of tasks, including question answering and sentiment analysis.

While pretrained models can be a powerful tool for transfer learning, they are not always the best solution for every task. In some cases, fine-tuning a pretrained model may not improve performance, and training a model from scratch may be necessary. In addition, pretrained models may not always be available for a specific task or domain, and researchers may need to train their own models.

In conclusion, transfer learning and pretrained models are powerful tools in deep learning that can significantly reduce the amount of time and resources needed to train large-scale models. By leveraging the knowledge learned by models trained on large datasets, researchers can improve the performance of their models on new tasks. However, the choice of approach depends on the specific task and the amount of labeled data available, and pretrained models may not always be the best solution for every task.

3.5 Transformer

The Transformer architecture was first introduced in [55]. The architecture follows an encoder-decoder structure, where the encoder is responsible for transforming an input sequence into a sequence of continuous representations. These representations are then passed on to the decoder, which generates an output sequence. The output sequence is generated by utilizing both the encoder's output and its own output from the previous time step. This allows for the model to generate more accurate predictions and produce more natural outputs. It is important to note that the Transformer architecture has become a popular choice in natural language processing due to its ability to handle long-term dependencies and capture context effectively.

3.5.1 The Encoder

Figure 3.6 demonstrates the critical role of the encoder in the Transformer architecture. The encoder comprises six identical layers, each of which consists of two submodules.

The first submodule implements the multi-head self-attention technique, which employs several attention heads that process linearly projected queries, keys, and values in parallel, producing outputs. These outputs are then used to build the final result. Note that the multi-head mechanism used in this submodule is customizable and can be adjusted to match specific tasks.

The second submodule is a fully connected feed-forward network that uses two linear transformations and rectified linear unit (ReLU) activation. In simpler terms, the output of the feed-forward network is obtained by applying a non-linear function to a linear combination of the input and its weights and biases. Mathematically, the output of the feed-forward network is:

$$FF(x) = ReLU(W_1 x + b_1)W_2 + b_2$$

where W_1, W_2 are the weights and b_1 and b_2 are the biases.

The encoder conducts the same calculation in each of its six layers, but with different weights and biases. Additionally, a residual connection is applied for the two submodules, where the input of each is added to its output. This helps to preserve the original information while also allowing the model to learn new features.

After each module, a normalization layer is applied, where the sum of the module's output and its input is normalized. This helps to stabilize the training process and improve the accuracy of the model.

By definition, the Transformer architecture cannot record information about the relative positions of the words in the sequence. To address this, a positional encoding is added to the input embeddings. Specifically, sine and cosine functions of various frequencies are used to construct the positional encoding vectors, which have the same dimension as the input embeddings. The positional information is then injected by simply adding their sum to the input embeddings. This allows the model to capture the order of the sequence and process it effectively.

3.5.2 The Decoder

According to the figure 3.6 of the Transformer architecture, the decoder architecture is quite similar to that of the encoder. It is made up of six blocks stacked one on top of the other. Each block is composed of three submodules.

The first submodule takes the previous output of the decoder stack as its input. It then adds positional information to it and performs multi-head self-attention over it. The decoder is designed to focus exclusively on the words that come before them, whereas the encoder looks at every word in the input sequence, regardless of its position in the sequence. As a result, the prediction for a word at a position in the sequence can only be based on the known outputs for the words that come before it. To achieve this, the multi-head attention mechanism implements numerous, single attention functions simultaneously. This mechanism applies a mask to the results of the scaled multiplication of the matrices Q and K. By applying a mask, the mechanism suppresses matrix values to avoid illegal connections: :

$$mask(QK^T) = mask\left(\begin{bmatrix} e_{11} & e_{12} & \cdots & e_{1n} \\ e_{21} & e_{22} & \cdots & e_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ e_{m1} & e_{m2} & \cdots & e_{mn} \end{bmatrix}\right) = \begin{bmatrix} e_{11} & -\infty & \cdots & -\infty \\ e_{21} & e_{22} & \cdots & -\infty \\ \vdots & \vdots & \ddots & \vdots \\ e_{m1} & e_{m2} & \cdots & e_{mn} \end{bmatrix}$$

A multi-head self-attention mechanism, identical to the one used in the first submodule of the encoder, is implemented in the second decoder submodule. This mechanism receives the keys and values from the encoder's output, as well as the queries from the previous decoder submodule on the decoder side. As a result, the decoder can focus on every word in the input sequence.

A fully connected feed-forward network, which is identical to the one used in the encoder, is implemented in the third submodule.

The three submodules are followed by a normalization layer and have residual connections all around them. In the same way that was previously described for the encoder, positional encodings are likewise added to the input embeddings of the decoder. Overall, the decoder is constructed from six blocks stacked on each other, each of which is made up of three submodules, and works to generate output based

on the input it receives.

3.6 Parameter Pruning

Parameter pruning is a technique used in deep learning to reduce the size of a model by removing unnecessary weights. The goal is to reduce the memory footprint and computational requirements of the model while maintaining its accuracy. There are several reasons why pruning is necessary in deep learning models, including reducing model size, optimizing model performance, and increasing model interpretability.

Reducing model size is one of the most important reasons for pruning deep learning models. As models become larger, they require more memory and computational resources to run. This can be a problem, especially for models that are intended for deployment on mobile devices or other resource-constrained environments. By pruning unnecessary weights, we can reduce the size of the model without sacrificing its accuracy.

Another reason to prune deep learning models is to optimize model performance. Pruning can help to reduce overfitting and improve generalization performance. Overfitting occurs when a model becomes too complex and learns to memorize the training data instead of generalizing to new data. By pruning unnecessary weights, we can reduce the complexity of the model and improve its ability to generalize.

In addition to optimizing model performance, pruning can also help address hardware constraints. As deep learning models become more complex, they require more computational resources to train and run. By pruning unnecessary weights, we can reduce the memory footprint and computational requirements of the model, making it more efficient to train and run on resource-constrained hardware.

Finally, pruning can increase the interpretability of deep learning models. Deep learning models are often treated as black boxes, making it difficult to understand how they are making predictions. By pruning unnecessary weights, we can simplify the model and make it easier to interpret. This can be especially important in domains such as healthcare, where the decisions made by the model can have serious consequences.

There are several techniques for pruning deep learning models, including weight

pruning, neuron pruning, and filter pruning. Weight pruning involves removing small weights from the model, while neuron pruning involves removing entire neurons that are not contributing much to the model's output. Filter pruning involves removing entire filters from convolutional layers that are not contributing much to the model's output.

Weight pruning is one of the most commonly used pruning techniques. It involves setting small weights to zero, effectively removing them from the model. This can be done by setting a threshold value below which weights are considered unimportant and can be pruned. For example, if we set a threshold of 0.1, any weight with an absolute value less than 0.1 would be pruned.

Neuron pruning involves removing entire neurons from the model that are not contributing much to the model's output. This can be done by measuring the sensitivity of the output to changes in the input and removing neurons with low sensitivity. Neuron pruning can be more effective than weight pruning in some cases, as it can remove entire units that are not contributing much to the model's output.

Filter pruning involves removing entire filters from convolutional layers that are not contributing much to the model's output. This can be done by measuring the sensitivity of the output to changes in the filter and removing filters with low sensitivity. Filter pruning can be more effective than weight pruning in convolutional neural networks, as it can remove entire feature maps that are not contributing much to the model's output.

In conclusion, parameter pruning is a necessary technique in deep learning to reduce model size, optimize performance, and increase interpretability. There are several techniques for pruning deep learning models, each with its own strengths and weaknesses. By understanding these techniques and their applications, we can develop more efficient and interpretable deep learning models. Pruning is especially important in resource-constrained environments, where reducing model size and computational requirements is essential for efficient training and deployment.

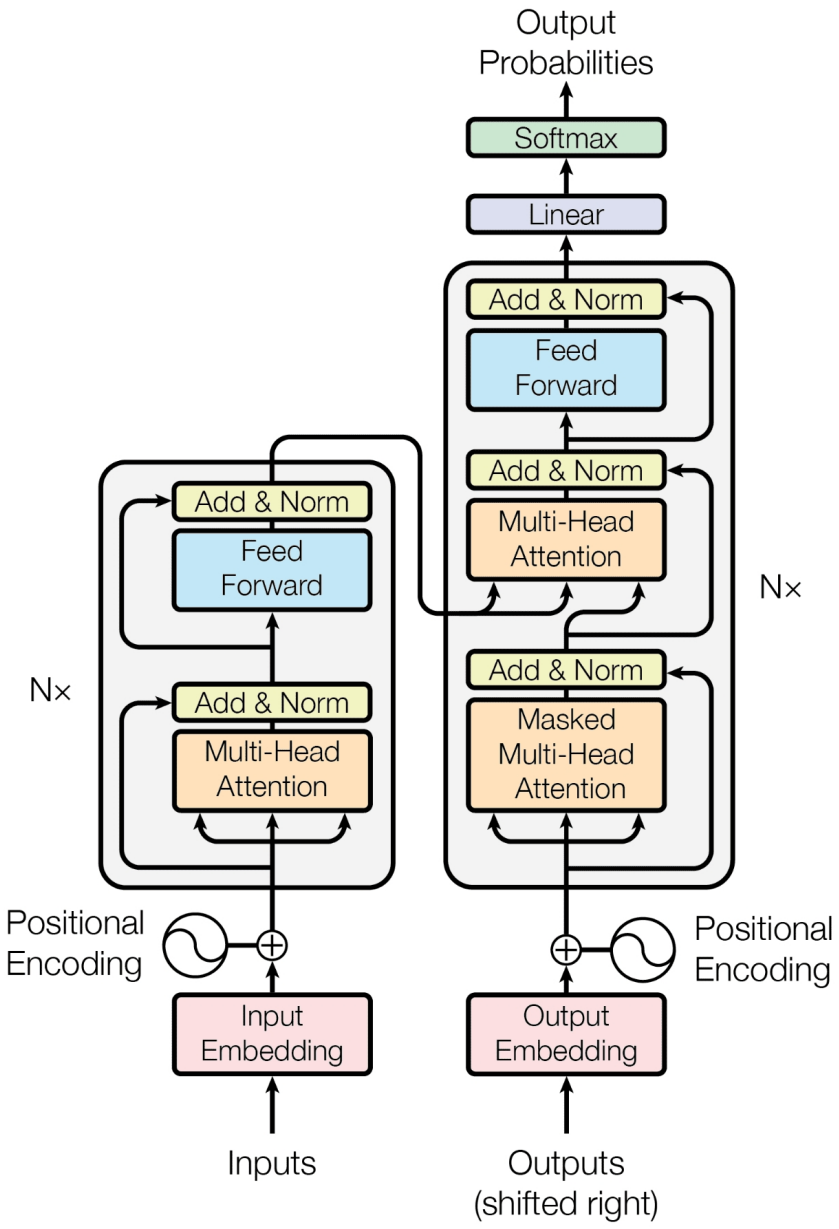


Figure 3.6: Transformer Architecture

Chapter 4

Proposed Solution

In this chapter, we present our proposed solution for the problem of table detection and structure recognition. We provide a detailed examination of the architecture, implementation, and workflow of our system, as well as a comprehensive description of the components that make up the table OCR pipeline. Additionally, we detail the algorithm and implementation of our parameter pruning method

4.1 Table OCR Pipeline

4.1.1 Overview

The lack of an off-the-shelf model that performs well on all types of tables is a significant challenge in the field of table recognition. To address this problem, we developed a novel end-to-end pipeline called Tabiness. This system is capable of detecting and recognizing table structures in both images and documents. Our pipeline is comprised of three main components: table detection, table structure recognition, and post-processing.

The table detection module employs deep learning models to locate the table regions in the input data and classify them into bordered or borderless tables. The table structure recognition module then processes the detected table regions and identifies the underlying structure of each table. This module utilizes different deep learning models to recognize different types of tables. Finally, the post-processing

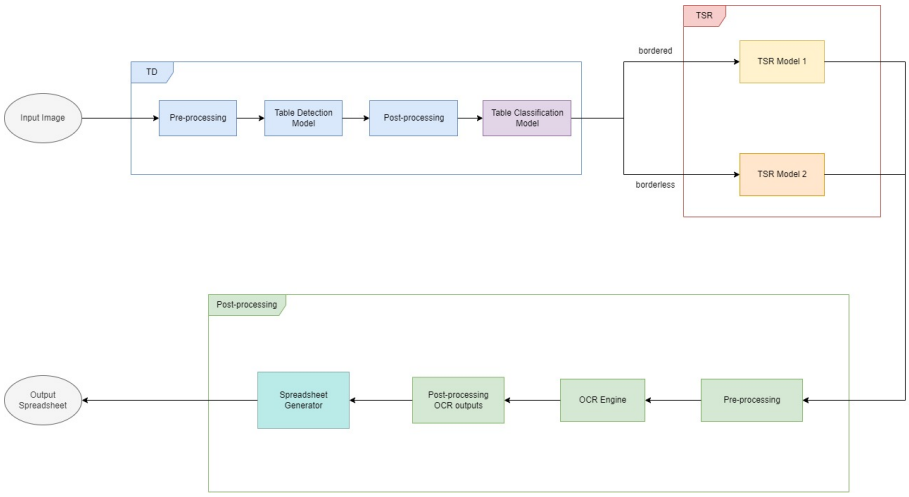


Figure 4.1: System Overall

module refines the recognition results, removes any errors or inconsistencies, does the OCR task, and transforms the final results into spreadsheets.

Our Tabiness pipeline provides a comprehensive solution to the challenging problem of table OCR. Each of its three main components employs deep learning models designed to optimize the recognition result. This pipeline offers a modular and systematic approach and has the potential to revolutionize the field of table OCR.

4.1.2 Details of components

Pipeline

Figure 4.1 provides an informative overview of the intricacies of our system. Our system is composed of three main components, each of which is essential to the overall functionality of the system: table detection (TD), table structure recognition (TSR), and postprocessing.

The TD component is composed of four modules: pre-processing, table detection, post-processing, and table classification. The input image is first pre-processed by being resized into an image shape that results in good detection results. The table detection model is responsible for detecting tables in the input image. The detected tables are then padded in four directions in the post-processing module before being passed into the table classification module. The table classification model classifies

the detected tables into two categories: bordered and borderless tables. With this categorization, we can better understand the structure of each table and more accurately identify the appropriate recognition models for each table type.

The TSR component is also composed of two modules, each with its own recognition model designed to handle different types of tables. TSR model 1 is specifically designed to recognize the structures of bordered tables. Meanwhile, TSR model 2 specializes in recognizing the structures of borderless tables. Each of these modules works in tandem to provide a complete understanding of the table structures within the system.

Finally, we have the postprocessing component. Composed of four modules, preprocessing, OCR engine, post-processing OCR results, and the spreadsheet generator, this component is responsible for reading and recognizing content in each cell of the tables. The pre-processing module refines the rows/columns recognition results as well as enhances the image quality of the detected cells. These refined TSR outputs are then passed into the OCR engine. The OCR engine reads the text in each cell and recognizes it. After that, the recognized text content is post-processed to enhance the OCR results. Finally, the spreadsheet generator converts the fully recognized table into spreadsheets in the form of comma-separated-value (CSV) or Microsoft Excel spreadsheets (XLSX/XLS). With these modules working together, we can provide a comprehensive and accurate understanding of the data contained within the tables.

Table Detection

As mentioned above, TD component includes four modules, one of which is the table detection model. The table detection model plays the role of a table detector. Tables in documents exist in many different forms including financial statements, receipts, repository management tables, etc., and they may be bordered and borderless. The quality of the pages also variates, from text to scanned images, which may be distorted by light, noises, etc. This leads to different qualities among input images. This diversity places a challenge for the generalization of the table detection model and the dataset. To choose the best baseline model for TD tasks, we perform a comparison among state-of-the-art deep learning table detection models including

DEtection TRansformer or DETR [56], CDeC-Net [35], CascadeTabNet [6], Multi-Type-TD-TSR [34], Cascade RCNNs trained on TNCR dataset [40], and Document Image Transformer or DiT [41].

DEtection TRansformer

The paper "End-to-End Object Detection with Transformers" [56] introduces the Detection Transformer (DETR) model, a new approach to object detection that uses a transformer-based architecture. The authors argue that traditional object detection methods, such as Faster R-CNN and SSD, suffer from a number of limitations, including the need for anchor boxes and non-maximum suppression, which can lead to inefficiencies and inaccuracies.

The DETR model addresses these limitations by using a transformer-based network to directly output a set of object queries and their corresponding class labels and bounding boxes. The model does not require anchor boxes or non-maximum suppression, making it more efficient and accurate than traditional object detection methods.

The authors describe the DETR model as a "plug-and-play" solution that can be easily applied to a wide range of object detection tasks. They demonstrate the effectiveness of the model on a number of benchmark datasets, including COCO and Pascal VOC, achieving state-of-the-art results on these datasets.

Figure 4.2 illustrates the high-level architecture of DETR. The DETR model consists of two main components: a backbone CNN and a transformer-based decoder. The backbone CNN is responsible for extracting feature maps from the input image, while the transformer-based decoder processes these feature maps and outputs the object queries and their corresponding class labels and bounding boxes.

The transformer-based decoder consists of a series of transformer blocks, each of which includes a multi-head self-attention mechanism and a feed-forward network. The multi-head self-attention mechanism allows the model to attend to different parts of the feature maps and learn spatial relationships between objects, while the feed-forward network processes this information to output the object queries and their corresponding class labels and bounding boxes.

The authors note that one of the key benefits of the DETR model is its ability to handle variable numbers of objects in an image. Traditional object detection

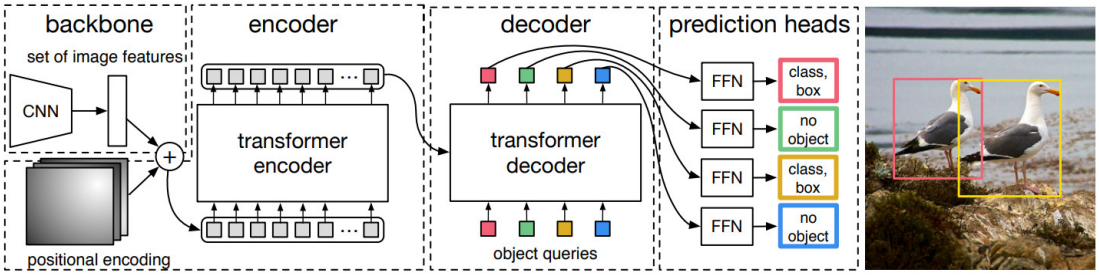


Figure 4.2: High-level architecture of DETR

methods require a predetermined number of anchor boxes, which can be inefficient and inaccurate for images with varying numbers of objects. The DETR model, on the other hand, can handle variable numbers of objects by outputting a fixed number of object queries and allowing the model to learn to attend to the most relevant objects in the image.

The authors also highlight the interpretability of the DETR model, noting that the model's attention maps can be used to visualize its decision-making process. This can be useful for understanding how the model is interpreting the input image and making predictions.

Overall, the DETR model represents a significant advance in the field of object detection. By using a transformer-based architecture, the model is able to achieve state-of-the-art results on benchmark datasets while avoiding the limitations of traditional object detection methods. The model's ability to handle variable numbers of objects and its interpretability make it a promising solution for a wide range of object detection tasks including table detection and table structure recognition.

This model is then applied into the task of table detection and table structure recognition in [57]. The target is to extract tables from unstructured documents, such as scientific articles. For the TD task, the DETR model is used to detect tables in input documents and classify them as either table or rotated table. For the TSR task, the DETR model is used to detect six object classes: table, table column, table row, table column header, table projected row header, and table spanning cell.

The authors evaluate their approach on the PubTables-1M dataset, which contains over 1 million tables from scientific articles. They compare their approach to several

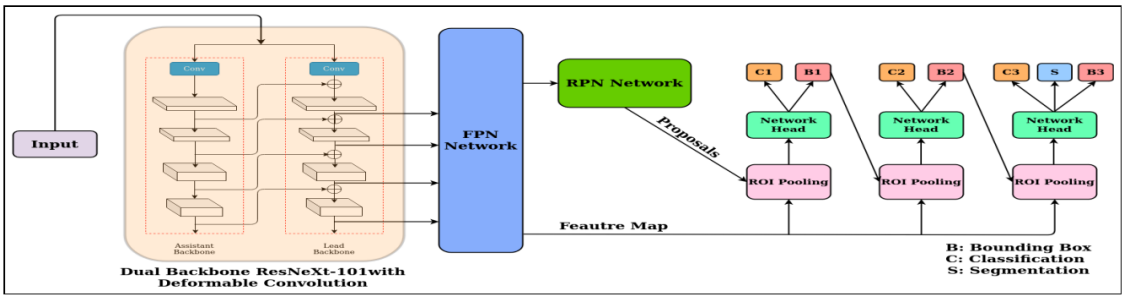


Figure 4.3: Illustration of CDeC-Net architecture

state-of-the-art methods and demonstrate that their pipeline achieves significantly better performance on both table detection and table structure recognition tasks.

CDeC-Net

The CDeC-Net model is a highly innovative deep learning network designed specifically to tackle the complex task of table detection in images of documents. What makes this network unique is its multi-stage extension of Mask R-CNN, complemented by a dual backbone that has deformable convolution. The network is end-to-end trainable, making it a highly flexible and adaptable solution.

Figure 4.3 illustrates the architecture of CDeC-Net. The backbone of the network is a composite of multiple identical backbones with composite connections between neighboring backbones. This unique design is intended to improve detection accuracy without adding too much computational cost. Additionally, the deformable convolution captures features using a variable receptive field and makes detection independent of fixed geometric transforms. These features make the network well-suited to address the problems of noisy detection at higher thresholds and the limited ability of CNNs to model large transformations due to the fixed geometric structures of the modules.

The effectiveness of the CDeC-Net model was evaluated by testing it on all publicly available benchmark datasets using extensive experiments. The results of these experiments demonstrated that the model has three key properties that make it an exceptional solution: (i) a single trained model performs well across all popular benchmark datasets; (ii) excellent performance is achieved across multiple, including higher, thresholds of IoU; and (iii) superior quantitative performance is consistently demonstrated by following the same protocol of recent papers for each of the benchmarks.

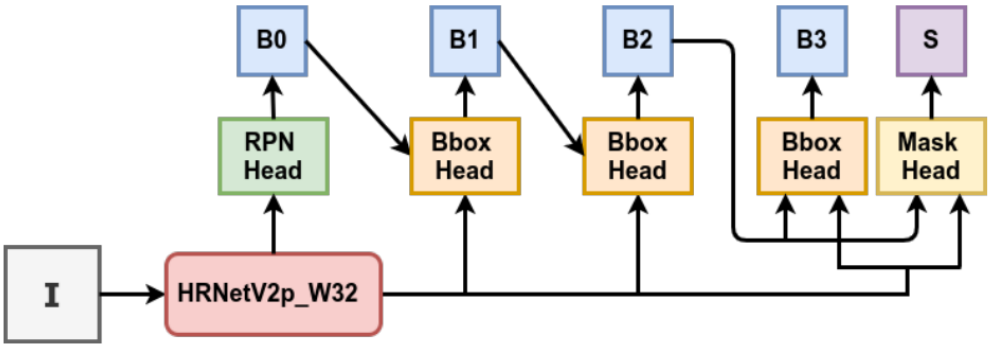


Figure 4.4: CascadeTabNet architecture

Overall, the proposed solution is a significant contribution to the field of table detection as it is generic and applicable across a wide variety of documents and use cases. Its innovative design and superior performance make it a valuable tool for researchers and professionals alike.

CascadeTabNet

CascadeTabNet is an improved deep learning-based end-to-end approach that uses a single Convolution Neural Network (CNN) model for both table detection and structure recognition. It is a Cascade mask Region-based CNN High-Resolution Network (Cascade mask R-CNN HRNet) based model that detects the regions of tables and recognizes the structural body cells from the detected tables simultaneously. The model classifies tables into two types as bordered (ruling-based) and borderless (no ruling-based) tables. It predicts the segmentation of cells only for the unbordered tables. For bordered tables, the model uses conventional text detection and line detection algorithms for extracting cells. Figure 4.4 illustrates the architecture of CascadeTabNet.

CascadeTabNet utilizes an iterative transfer learning approach that helps the model to learn from less amount of training data and perform well on multiple datasets. The model is trained gradually, starting from more general tasks and going towards more specific tasks. The authors demonstrate an effective way of image augmentation that enhances the accuracy of table detection and helps the model learn more effectively.

The model was evaluated on the ICDAR 2013, ICDAR 2019, and TableBank public datasets. The authors achieved 3rd rank in ICDAR 2019 post-competition results for table detection while attaining the best accuracy results for the ICDAR 2013 and TableBank dataset. They also attained the highest accuracy results on the ICDAR 2019 table structure recognition dataset.

CascadeTabNet is an effective and efficient model for table detection and structure recognition in document images, achieving state-of-the-art results on multiple datasets. Its iterative transfer learning approach and image augmentation techniques make it possible to achieve high accuracy results even with a small amount of data.

Multi-Type-TD-TSR

The Table Detection (TD) model used in this paper is a fully data-driven approach based on a Convolutional Neural Network (CNN) to localize tables inside images. Specifically, the authors utilized the ResNeXt-152 model proposed by Li et al. [5], which offers the state-of-the-art model for type-independent TD. The authors utilized a deterministic algorithm in order to address all three table types, i.e. unbordered, partially bordered, and fully bordered tables.

In terms of evaluation results, the authors achieved an F-score of 86.64% on the ICDAR 2019 dataset, which is a significant improvement over the previous state-of-the-art of 81.46%. The authors attribute the success of their model to its ability to differentiate between three different types of tables based on their borders, and to its use of state-of-the-art deep learning models for table detection. In addition, the authors highlight the importance of addressing real application conditions such as rotated images and noise artifacts inside the document image, which their model takes into account. Overall, the proposed Multi-Type-TD-TSR algorithm offers a promising solution for the automated extraction of tables from document images.

Cascade RCNNs trained on TNCR dataset

The paper [40] presents TNCR, a table dataset that can be used for table detection and classification into 5 different classes. The dataset contains 9428 high-quality labeled images collected from free websites. The authors implement state-of-the-art deep learning-based methods for table detection, including Cascade R-CNN with ResNeXt101-64x4d Backbone Network, which achieves the best performance compared to other methods with a precision of 79.7%, recall of 89.8%, and f1 score of

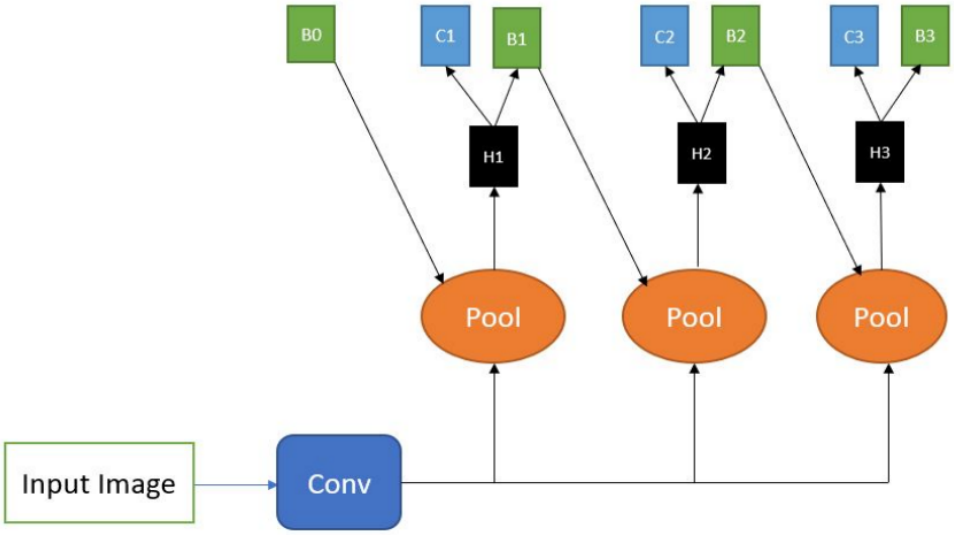


Figure 4.5: Cascade RCNNs architecture

84.4% on the TNCR dataset. Figure 4.5 illustrates the Cascade RCNN architecture. The authors hope that the dataset and trained model checkpoints will encourage more deep-learning approaches to table detection, classification, and structure recognition. The paper also provides an overview of related work on existing datasets and methods for table detection and structure recognition.

Document Image Transformer (DiT)

Li et. al. [41] introduced DiT, a self-supervised pre-trained Document Image Transformer model for vision-based Document AI tasks. Unlike other pre-trained models which rely on human-labeled data, DiT uses large-scale unlabeled text images to pre-train its backbone network. The proposed model is used in a variety of vision-based Document AI tasks, including document image classification, document layout analysis, table detection, and text detection for OCR. The pre-trained DiT model achieves state-of-the-art results on these downstream tasks, surpassing existing supervised and self-supervised pre-trained models. The authors achieve this by using Masked Image Modeling (MIM) as their pre-training objective. The model is trained to recover visual tokens from a discrete variational auto-encoder (dVAE) based on corrupted input document images. Figure 4.6 illustrates the pre-training

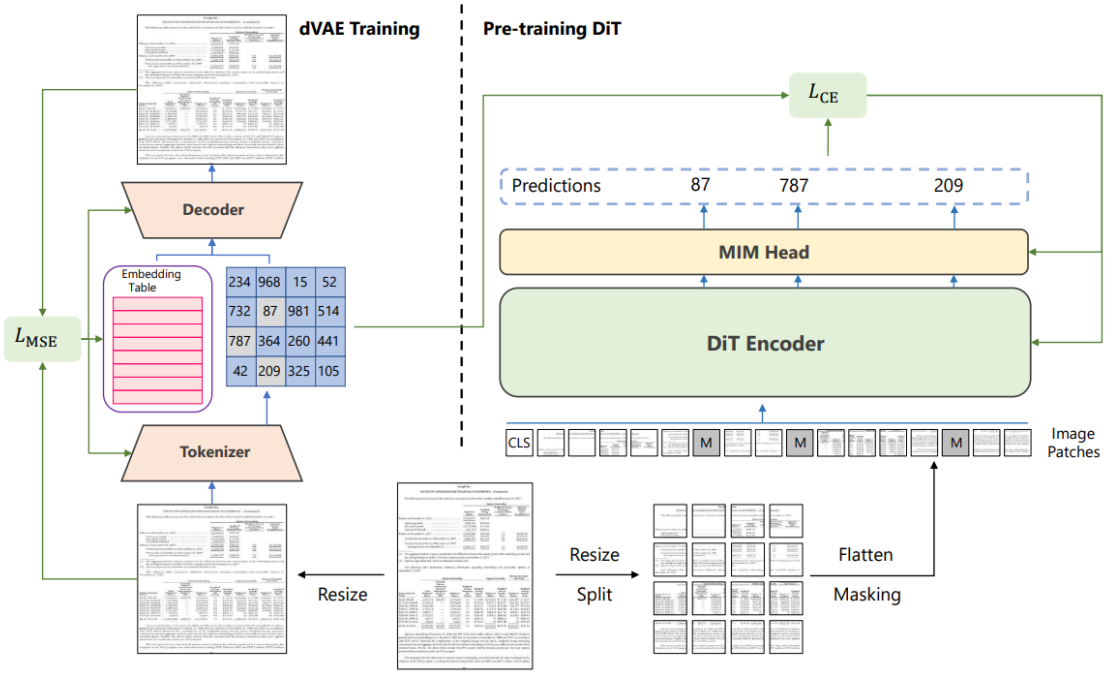


Figure 4.6: DiT pre-training mechanism with MIM pre-training

mechanism of DiT. The DiT model is the first large-scale self-supervised pre-trained model for vision-based Document AI tasks that do not rely on any human-labeled data. Fine-tuning experiments on four Document AI benchmarks, including the RVL-CDIP dataset for document image classification, the PubLayNet dataset for document layout analysis, the ICDAR 2019 cTDaR dataset for table detection, and the FUNSD dataset for text detection demonstrate the effectiveness of the proposed DiT model. Figure 4.7 shows how DiT can be applied as the backbone network in different detection frameworks. Specifically, the paper reports that DiT outperforms existing models on the table detection task, achieving a state-of-the-art F1 score of 96.55 on the ICDAR 2019 cTDaR dataset. The authors attribute the success of the DiT model to its ability to leverage large-scale unlabeled data to learn the global patch relationship within each document image, which can support a variety of Document AI tasks.

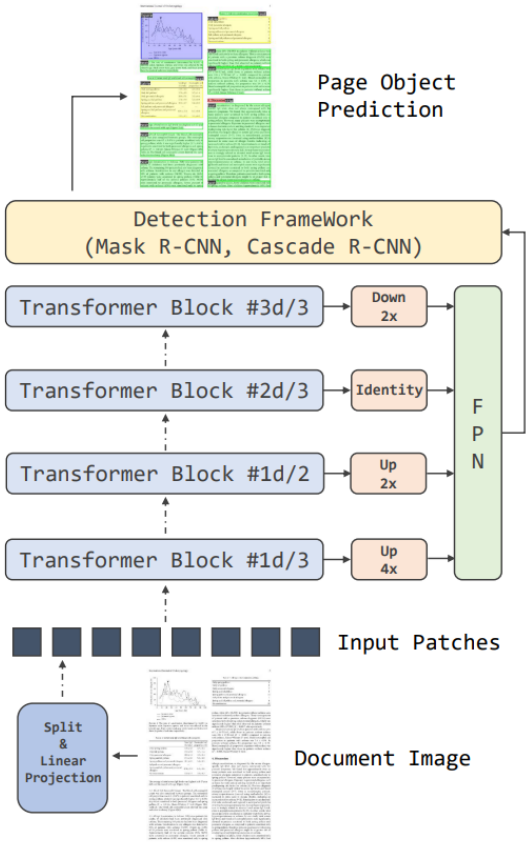


Figure 4.7: Applying DiT as the backbone network in different detection frameworks

Table Classification

Due to the difference in the type of tables in documents as we have mentioned, the need for classification is inevitable for further reconstruction. Since this is a binary classification, there is no need for a complex model approach. We construct a heuristic process for table classification(bordered and borderless). Because of the difference in the quantity, and distribution of lines(borders) in the two variations, there is a considerable impact on the structure and layout of a table and its content. This can lead to very distinctive signals and features when going through a deep learning pipeline aiming to understand the table structure. Hence our decision to perform a classification step to better optimize a solution for each kind of table.

Our proposed solution follows a simple and straightforward principle. A table, in most cases, consists of many vertical and horizontal lines and the main difference

between bordered and borderless tables lies in the distribution and quantity of those lines with respect to the original picture ratio. Thus, if we can identify and locate the majority of those lines and their distribution using filters or transformation, we can produce a classification with high confidence.

After a table has been localized and extracted from a page of a document, it will next go through the classification process. The image will first be converted to grayscale and then undergo morphological image processing operations such as erosion and dilation to denoise and improve the distinction of borders in the image. The borders, consisting of vertical and horizontal lines, will be retrieved using appropriate structure elements (also called kernels). A filter can be added to only keep lines that are relatively close to the dimensions of the picture, the ones that do not meet the length requirements are usually noises and artifacts. With the coordinates of the retrieved lines, we can go on to calculate their distribution in the picture. The picture can be imagined as consisting of multiple horizontal or vertical sections. Were the detected line to cover a significant number of areas (80% of the total areas) with adequate density per area, we can conclude that the picture is depicting a bordered table or a borderless table otherwise. We also collect and label a small dataset for testing, further details will be provided in section 4.1.3.

Table Structure Recognition

The table recognition model handles the task of "understanding" and recreating the grid structure of tables. Due to the high degree of intra-class variabilities such as different layouts, the erratic use of ruling lines, structure delineation, and diverse table contents, there is no universal definition of what a table looks like. There are also factors of varied page quality, light distortion, image noises,... This leads to a lack of a formal pattern for table structure which makes the task of table recognition considerably challenging.

To determine the best approach for this task we perform research and comparisons among the most promising state-of-the-art deep learning models including: DeepDeSRT [4], Deep Splitting and Merging [45], TableNet [**?**], GraphTSR [3].

Some basic information about our table recognition models of choice are available in Table 5.9. Due to various reasons, most table recognition models do not have an

Model	Dataset	Architecture	Pretrained	Number of parameters	Year
Deep Splitting and Merging	Private dataset	Pair of convolutional models with a novel pooling regions	True	No official implementation	2019
TableNet	Marmot Extended	Encoder/decoder model for semantic segmentation	True	No official implementation	2020
DeepDeSRT	PASCAL VOC 2011	Augmented fully convolutional network	True	No official implementation	2017

Table 4.1: Table Recognition Models of Choice

official public code base which leads to some shortage of information regarding the models.

Deep Splitting and Merging(SPLERGE)

SPLERGE composed of a Split model to predict the fine grid structure of the table ignoring span cell by focusing on detecting signals of the space between columns and rows (also called row and column separator). Then either a Merge model or some heuristics methods are used to predict where cells span multiple columns or rows. The key idea put forth by this paper is to pool information over large regions of the table image such as entire rows/columns of pixels or previously predicted cell regions to help learned features propagate across large images. The models are pretrained on a private dataset that boasts larger variety of complexity, visual appearance, and structure than ICDAR 2013 training set. Thus, this approach performs exceptionally well and are much better than others when dealing with tables with partial or no borders(less presence of border means an increase in the presence of column and row separators). The SPLERGE model has been proven to be highly effective as it can achieve state-of-the-art performance on the ICDAR 2013 Table Competition dataset. The model achieved an impressive F-measure of 95.15% for adjacency relationships, surpassing the previously published best result of 94.6%. This marks a significant improvement in performance and highlights the model's ability to accurately extract and recognize table structures. It was also evaluated on a larger and more diverse private dataset of table images, where it continued to outperform other state-of-the-art deep learning models and a major commercial software system.

TableNet

The author proposed TableNet: a novel deep multi-task architecture for both

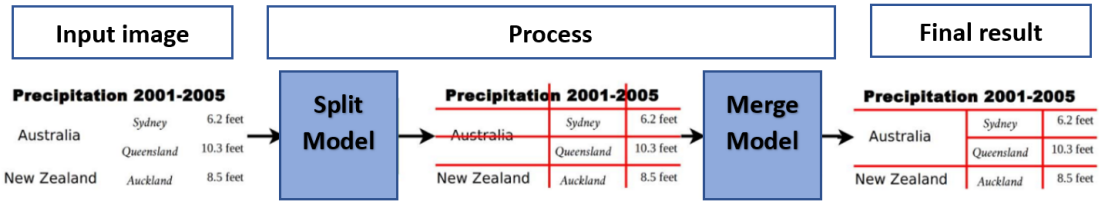


Figure 4.8: Overview of SPLERGE

table detection and structure recognition. They also demonstrated that adding spatial semantic features to TableNet during training helps in improving performance. The Marmot dataset for table data extraction is also annotated and released for the community [23]. With all previous works, table detection and column detection are considered separately as two different problems which can be solved independently. However, intuitively if all the columns present in a document are known apriori, the table region can be determined easily. It is based on this notion that the author of TableNet built their approach. They successfully implement and achieve state-of-the-art result with an encoder-decoder model for semantic segmentation combined with convolutional filters utilized to detect tables reinforced by column-detecting filters. After processing the documents using TableNet, masks for table and column regions are generated. These masks are used to filter out the table and its column regions. Then using an OCR engine(the author chose TesseractOCR), they retrieved all word positions of the document, and only the word patches lying inside table and column regions were filtered out. Now, using these filtered words and column positions, a row can be located and defined as a collection of words from multiple columns, which are at similar horizontal levels. The row segmentation process also utilized heuristic rules to help with outliers. While the results achieved are not conclusively better than previous methods (such as DeepDSert), they are on the same level, and the fact that this is an end-to-end pipeline means that it can further be improved.

DeepDeSRT

The authors presented a deep learning-based solution for table detection in document images and a novel deep learning based approach for table structure recognition. For table structure recognition, general purpose domain object detectors are adapted

and transfer leaning is utilized by augmenting and fine-tuning an FCN semantic segmentation model by Shelhamer et al. [58] pre-trained on Pascal VOC 2011 [59]. Based on the notion that the basic features detected by early network layers such as edges and changes in color, can facilitate boundary detection. Modifications were performed on the original model in order to extract extra details in the shallower layers which can help with obtaining cleaner delineation results for rows and columns. Scale layers are also swapped in for normalization layers which provide learnable scaling capabilities for the models. This gives the model the ability to learn the scaling factors by itself during training. Evaluation results of DeepDeSRT outperform all of the existing methods at its time of publication. By successfully implemented their idea, the authors presented another proof for the efficacy of fine-tuning deep neural networks even when source and target domains are highly dissimilar and the target training set is rather small.

GraphTSR

In this paper, the authors proposed a novel graph neural model for complicated table structure recognition in PDF files. They reformulated the problem as a graph problem with each cell in a table viewed as a vertex, and an adjacent relation (i.e., vertical, horizontal) can be viewed as a labeled edge. By that definition, a table can be represented as a graph with labeled edges T = (V,R), where V is a set of vertices, and $R \subseteq V \times V \times \{vertical; horizontal\}$ is the set of relations representing the table that we need to obtain. The problem is now framed as follows: given a input in the form of a set of vertices V of table T, find out an approximation to the real relations R as visualized in 4.9. The model computes the vertex and edge features representations by N edge-to-vertex graph attention blocks and N vertex-to-edge graph attention blocks, respectively. Finally, a classification is performed on these edges. Additionally, they also constructed and released a large-scale dataset for table structure recognition numbering close 15000 samples. While the results obtained are not definitively better than other state of the art solutions (as they are comparing to unofficial implementation of previous works), this proves to be a promising new angle to tackle the problem of table structure recognition.

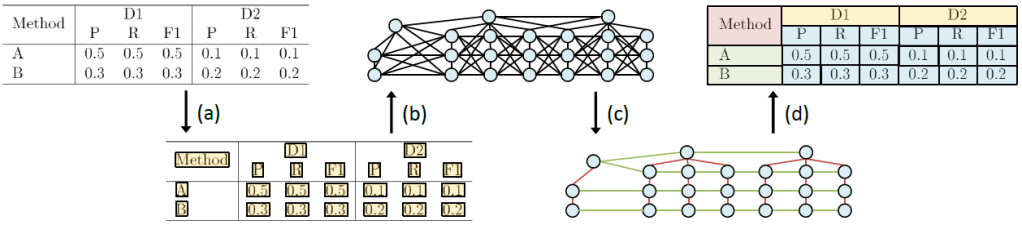


Figure 4.9: Overview of GraphTSR method. Given a table in PDF as input, recognize its structure by the following steps: (a) Pre-processing: obtaining cell contents and their corresponding bounding box; (b) Graph construction: building an undirected graph on these cells; (c) Relation prediction: predicting adjacent relations by the proposed GraphTSR; (d) Post-processing: recovering table structure from the labeled graph

OCR of cell content

Optical Character Recognition (OCR) models are pivotal in converting scanned documents into digital text. Several prominent open-source OCR models have made significant contributions in this field. As this is a support module and not the main focus of our pipeline, our research work regarding this area is not as extensive as the main components(table detection and structure recognition)

Tesseract OCR

Tesseract OCR, developed by Google, is a widely recognized OCR engine. Its workflow encompasses image preprocessing, text recognition utilizing deep learning models, and post-processing steps to refine the recognized text. Tesseract leverages a combination of CNNs and RNNs within its model structure, enabling accurate text recognition. Throughout its many iterations and upgrades, it has added many new and powerful features, such as an LSTM focused on line recognition in a recent update. Extensive benchmark tests have showcased Tesseract's exceptional performance, consistently achieving high accuracy rates and competitive results in various OCR tasks. It is optimized for CPU and provides wrappers in multiple programming languages. It also support multiple languages and other configuration options. Since it uses the classical page layout analysis technique, however, text detection is not as accurate, and it is not optimized for GPU and batch processing.

EasyOCR

EasyOCR, an open-source Python-based OCR engine, has gained attention for its simplicity and ease of integration. It accepts scanned images as input and follows a workflow encompassing image preprocessing, text detection, and recognition. EasyOCR employs CRAFT algorithm **??** for text detection and CRNN **??** for recognition, leveraging their ability to achieve state-of-the-art accuracy. The output of EasyOCR is the recognized text extracted from the input image. While specific benchmark tests may vary, EasyOCR has demonstrated robust performance in recognizing text across multiple languages. In a benchmark study conducted on the challenging RIMES dataset, EasyOCR achieved an accuracy rate of 89.4%, outperforming other open-source OCR engines. The engine also has GPU support and batch prediction making it comparatively faster compared to Tesseract's OpenCL version. It also supports multiple languages, vertical text and other options. However, easyOCR is slower than Tesseract in CPU Mode and not better than Tesseract for scanned documents.

Paddle-OCR

PP-OCR is a prominent open-source OCR engine known for being a practical ultra lightweight OCR system. It leverages deep learning architectures such as Differentiable Binarization **??** as a text detector which is based on a simple segmentation network and CRNN **??** as a text recognizer. Through extensive benchmark testing on datasets like ICDAR 2015, PP-OCR has demonstrated impressive character accuracy, placing among the top-performing OCR engines. It has also excelled in the Competition for Chinese and English Text Recognition. It is considered lightweight as the overall model size is only 3.5M for recognizing 6622 Chinese characters and 2.8M for recognizing 63 alphanumeric symbols. A collection of strategies to either enhance the model's ability or lighten the model is also provided in the paper alongside the codebase. For its strengths, this is an end-to-end OCR model with data synthesis and semi-automated OCR annotation solutions. It also has multilingual OCR support and options for inference and serving. Paddle-OCR, however, suffers some complications with dependencies and needs external libraries for Paddle to Pytorch / ONNX conversion.

OCR conclusion

All aforementioned models achieved state-of-the-art results and are all widely

used for their own strengths. In our research, as the emphasis is not on the OCR component, we chose easyOCR as part of the pipeline for its ease of integration, GPU support, and batch prediction.

Spreadsheet generator

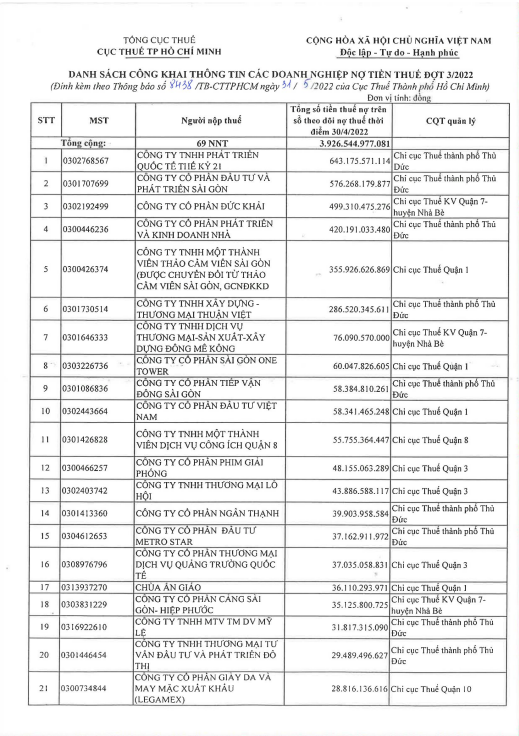
Our model produces a raw output that contains crucial cell information along with a list of potential spanning cells. The list of cells includes various attributes, such as the specific columns and rows they occupy, as well as their OCR content. To ensure accuracy and completeness, we employ a set of heuristic rules that consider factors like column and row widths, as well as other statistical measures to filter out any inaccurate or incomplete cells effectively. Additionally, we perform sorting and other post-processing steps on the cell list. After that, the list of spanning cells is iterated to merge any overlapping or adjacent cells accordingly.

Once the necessary processing steps have been completed, the raw output is now in a clean and refined format, ready for further manipulation. At this stage, the output is converted into a nested array representation which is then utilized to construct a data frame with the help of powerful libraries such as pandas. Finally, the resulting data frame is saved as a CSV/XLSX format file, ensuring compatibility with various data analysis tools and facilitating seamless integration into downstream tasks.

Sample output of each step in the pipeline

Compared to the prototype in Phase 1, the table classification component is now fully implemented and some tuning has been conducted on the pre-processing and post-processing modules. The end-to-end pipeline has finished and is ready to transform an image table into a spreadsheet. Figure 4.10 illustrates a sample input image.

After passing through the TD component, the detected tables are then classified into bordered or borderless tables so that a suitable TSR model can be called to perform recognition. Finally, each detected cell is then pre-processed and passed into OCR model for recognition. After obtaining all cell contents, spreadsheet generator puts all the cells together into a Pandas data frame and generates the final output spreadsheet. The detection and recognition results of the two sample input images are



TỔNG CỤC THUẾ
CỤC THUẾ TP HỒ CHÍ MINH

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM
Độc lập - Tự do - Hạnh phúc

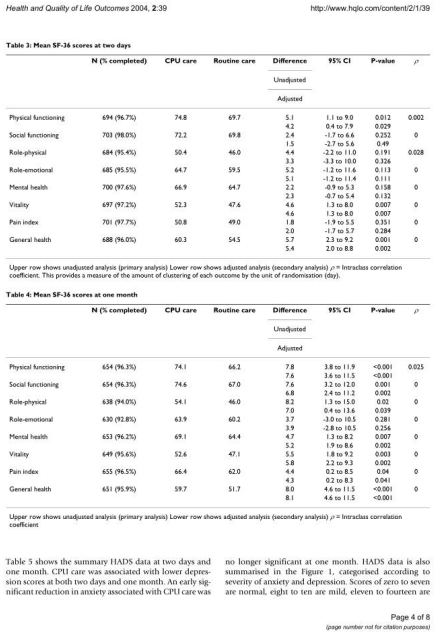
DANH SÁCH CÔNG KHAI THÔNG TIN CÁC DOANH NGHIỆP NỢ TIỀN THUẾ ĐỢT 3/2022

(Đính kèm theo Thông báo số 9458 /TB-CTTPHCM ngày 31 / 5 /2022 của Cục Thuế Thành phố Hồ Chí Minh)

Đơn vị tính: đồng

STT	MST	Người nộp thuế	Tổng số tiền thuế nợ trên sổ theo dõi nợ thuế thời điểm 30/4/2022	CQT quản lý
Tổng cộng:		69 NNT	3.926.544.977.081	
1	0302768567	CÔNG TY TNHH PHÁT TRIỂN QUỐC TẾ TIẾ KÝ 21	643.175.571.114	Chi cục Thuế thành phố Thủ Đức
2	0301707699	CÔNG TY CỔ PHẦN ĐẦU TƯ VÀ PHÁT TRIỂN SÀI GÒN	576.268.179.877	Chi cục Thuế thành phố Thủ Đức
3	0302192499	CÔNG TY CỔ PHẦN ĐỨC KHẢI	499.310.475.276	Chi cục Thuế KV Quận 7-huyện Nhà Bè
4	0300446236	CÔNG TY CỔ PHẦN PHÁT TRIỂN VÀ KINH DOANH NHÀ	420.191.033.480	Chi cục Thuế thành phố Thủ Đức
5	0300426374	CÔNG TY TNHH MỘT THÀNH VIÊN THẢO CẦM VIÊN SÀI GÒN (ĐƯỢC CHUYỂN ĐỔI TỪ THẢO CẦM VIÊN SÀI GÒN, GCNĐKKD	355.926.626.869	Chi cục Thuế Quận 1
6	0301730514	CÔNG TY TNHH XÂY DỰNG - THƯƠNG MẠI THUẬN VIỆT	286.520.345.61	Chi cục Thuế thành phố Thủ Đức
7	0301646333	CÔNG TY TNHH DỊCH VỤ THƯƠNG MẠI-SẢN XUẤT-XÂY DỰNG ĐỒNG MỄ KÔNG	76.090.570.000	Chi cục Thuế KV Quận 7-huyện Nhà Bè
8	0303226736	CÔNG TY CỔ PHẦN SÀI GÒN ONE TOWER	60.047.826.605	Chi cục Thuế Quận 1
9	0301086836	CÔNG TY CỔ PHẦN TIẾP VẬN ĐÔNG SÀI GÒN	58.384.810.261	Chi cục Thuế thành phố Thủ Đức
10	0302443664	CÔNG TY CỔ PHẦN ĐẦU TƯ VIỆT NAM	58.341.465.248	Chi cục Thuế Quận 1
11	0301426828	CÔNG TY TNHH MỘT THÀNH VIÊN DỊCH VỤ CÔNG ÍCH QUẬN 8	55.755.364.447	Chi cục Thuế Quận 8
12	0300466257	CÔNG TY CỔ PHẦN PHIM GIẢI PHÓNG	48.155.063.289	Chi cục Thuế Quận 3
13	0302403742	CÔNG TY TNHH THƯƠNG MẠI LỘ HỘI	43.886.588.117	Chi cục Thuế Quận 3
14	0301413360	CÔNG TY CỔ PHẦN NGÂN THẠNH	39.903.958.584	Chi cục Thuế thành phố Thủ Đức
15	0304612653	CÔNG TY CỔ PHẦN ĐẦU TƯ METRO STAR	37.162.911.972	Chi cục Thuế thành phố Thủ Đức
16	0308976796	CÔNG TY CỔ PHẦN THƯƠNG MẠI DỊCH VỤ QUẢNG TRƯỜNG QUỐC TẾ	37.035.058.831	Chi cục Thuế Quận 3
17	0313937270	CHÙA ÂN GIÁO	36.110.293.971	Chi cục Thuế Quận 1
18	0303831229	CÔNG TY CỔ PHẦN CẢNG SÀI GÒN- HIỆP PHƯỚC	35.125.800.725	Chi cục Thuế KV Quận 7-huyện Nhà Bè
19	0316922610	CÔNG TY TNHH MTV TM DV MỸ LỆ	31.817.315.090	Chi cục Thuế thành phố Thủ Đức
20	0301446454	CÔNG TY TNHH THƯƠNG MẠI TƯ VẤN ĐẦU TƯ VÀ PHÁT TRIỂN ĐÔ THỊ	29.489.496.627	Chi cục Thuế thành phố Thủ Đức
21	0300734844	CÔNG TY CỔ PHẦN GIÀY DA VÀ MAY MẶC XUẤT KHẨU (LEGAMEX)	28.816.136.616	Chi cục Thuế Quận 10

(a) A sample input image of a page containing a bordered table



Health and Quality of Life Outcomes 2004, **2**:39 http://www.hqlo.com/content/2/1/39

Table 3: Mean SF-36 scores at two days

	N (% completed)	CPU care	Routine care	Difference	95% CI	P-value	ρ
				Unadjusted			
				Adjusted			
Physical functioning	694 (96.7%)	74.8	69.7	5.1	1.1 to 9.0	0.012	0.002
				4.2	0.4 to 7.9	0.029	
Social functioning	703 (98.0%)	72.2	69.8	2.4	-1.7 to 6.6	0.252	0
				1.5	-2.7 to 5.6	0.49	
Role-physical	684 (95.4%)	50.4	46.0	4.4	-2.2 to 11.0	0.191	0.028
				3.3	-3.3 to 10.0	0.326	
Role-emotional	685 (95.5%)	64.7	59.5	5.2	-1.2 to 11.6	0.113	0
				5.1	-1.2 to 11.4	0.111	
Mental health	700 (97.6%)	66.9	64.7	2.2	-0.9 to 5.3	0.158	0
				2.3	-0.7 to 5.4	0.132	
Vitality	697 (97.2%)	52.3	47.6	4.6	1.3 to 8.0	0.007	0
				4.6	1.3 to 8.0	0.007	
Pain index	701 (97.7%)	50.8	49.0	1.8	-1.9 to 5.5	0.351	0
				2.0	-1.7 to 5.7	0.284	
General health	688 (96.0%)	60.3	54.5	5.7	2.3 to 9.2	0.001	0
				5.4	2.0 to 8.8	0.002	

Upper row shows unadjusted analysis (primary analysis) Lower row shows adjusted analysis (secondary analysis) ρ = Intraclass correlation coefficient. This provides a measure of the amount of clustering of each outcome by the unit of randomisation (day).

Table 4: Mean SF-36 scores at one month

	N (% completed)	CPU care	Routine care	Difference	95% CI	P-value	ρ
				Unadjusted			
				Adjusted			
Physical functioning	654 (96.3%)	74.1	66.2	7.8	3.8 to 11.9	<0.001	0.025
				7.6	3.6 to 11.5	<0.001	
Social functioning	654 (96.3%)	74.6	67.0	7.6	3.2 to 12.0	0.001	0
				6.8	2.4 to 11.2	0.002	
Role-physical	638 (94.0%)	54.1	46.0	8.2	1.3 to 15.0	0.02	0
				7.0	0.4 to 13.6	0.039	
Role-emotional	630 (92.8%)	63.9	60.2	3.7	-3.0 to 10.5	0.281	0
				3.9	-2.8 to 10.5	0.256	
Mental health	653 (96.2%)	69.1	64.4	4.7	1.3 to 8.2	0.007	0
				5.2	1.9 to 8.6	0.002	
Vitality	649 (95.6%)	52.6	47.1	5.5	1.8 to 9.2	0.003	0
				5.8	2.2 to 9.3	0.002	
Pain index	655 (96.5%)	66.4	62.0	4.4	0.2 to 8.5	0.04	0
				4.3	0.2 to 8.3	0.041	
General health	651 (95.9%)	59.7	51.7	8.0	4.6 to 11.5	<0.001	0
				8.1	4.6 to 11.5	<0.001	

Upper row shows unadjusted analysis (primary analysis) Lower row shows adjusted analysis (secondary analysis) ρ = Intraclass correlation coefficient

Table 5 shows the summary HADS data at two days and one month. CPU care was associated with lower depression scores at both two days and one month. An early significant reduction in anxiety associated with CPU care was no longer significant at one month. HADS data is also summarised in the Figure 1, categorised according to severity of anxiety and depression. Scores of zero to seven are normal, eight to ten are mild, eleven to fourteen are

(b) A sample input image of a page containing borderless tables

Figure 4.10: Sample input images containing different kinds of tables

illustrated in figure 4.12 and 4.13. The final output spreadsheet(s) of a full-bordered table is shown in 4.11a while that of a borderless table is shown in 4.11b.

4.2 Parameter Pruning Method For Transformer-based Models

4.2.1 Overview

Parameter pruning is a deep-learning technique for compressing and optimizing large models for low-power devices. As these devices become more popular, it is crucial to make these models accessible to low-resource hardware, such as mobile phones and embedded systems. Pruning deep learning models has become necessary to overcome this challenge, and researchers have proposed various methods to achieve this.

Transformer-based models dominate many deep learning fields, including natural language processing, image recognition, and audio generation. However, pruning these models presents a unique set of challenges due to their optimal performance. Nevertheless, pruning the Transformer architecture offers numerous benefits, such as improving model efficiency and accessibility to low-resource devices.

To address this challenge, we implemented a pruning method that was proposed by Khetan et al. [8] to optimize BERT models' parameters. Our motivation for re-implementing this method is twofold. Firstly, the original authors did not provide a detailed implementation of their approach along with their publication. Secondly, we want to prove whether the approach works for a general transformer-based model on a different domain, specifically table detection and table structure recognition. By implementing this method, we aim to improve model efficiency and accessibility to low-resource devices and ultimately contribute to the development of more efficient and effective deep-learning models.

	0	1	2	3	4
0	STT	MST	Người nộp thuế	Tổng số tiền thuế nợ trên sổ theo dõi nợ thuế thời điểm 30/4/2022	CQT quản lý
1		Tông cộng:	69 NNT	3.926.544.977.081	
2		0302768567	CÔNG TY TNHH PHÁT TRIỂN QUỐC TẾ TIIẾ KỶ 21	643.175.571.114	Chi cục Thuê thành Thủ Đức phố
3	2	0301707699	CÔNG TY CỔ PHẦN ĐAU TƯ VÀ PHÁT TRIỂN SÀI GÒN	576.268.179.877	Chi cục Thuế thành Thủ Đức phố
4	3	0302192499	CÔNG TY CỔ PHÀN ĐỨC KHẢI	499.310.475.276	Chi cục Thuế KV Quận 7- huyện Nhà Bè
5	4	0300446236	CÔNG TY CO PHẦN PHÁT TRIỂN VÀ KINH DOANH NHÀ	420.191.033.480	Chi cục Thuế thành Thủ Đức phố
6	5	0300426374	[CÔNG TY TNHH MỘT THÀNH VIÊN THẢO CẦM VIÊN SÀI GÒN (ĐƯỢC CHUYỂN ĐỔI TỪ THẢO cẦM VIÊN SÀI GÒN, GCNĐKKD	355.926.626.869	Chi cục Thuế Quận 1
7	6	0301730514	CÔNG TY TNHH XÂY DỰNG THƯƠNG MAI THUÂN VIỆT	286.520.345.611	Chi cục Thuế thành Thủ Đức phố
8	7	0301646333	CÔNG TY TNHH DỊCH VỤ THƯƠNG MẠI-SẢN XUẤT-XÂY DỰNG ĐÔNG MÊ KÔNG	76.090.570.000	Chi cục Thuế KV Quận 7- huyện Nhà Bè
9	8	0303226736	CÔNG TY CỔ PHẦN SÀI GÒN ONE TOWER	60.047.826.605	Chi cục Thuế Quận 1
10	9	0301086836	CÔNG TY CO PHAN TIẾP VẬN ĐÔNG SÀI GÒN	58.384.810.261	Chi cục Thuế thànhi Thủ Đức phố
11	10	0302443664	CONG TY CO PHAN ĐAU TƯ VIỆT NAM	58.341.465.248	Chi cục Thuế Quận 1
12	11	0301426828	CÔNG TY TNHH MỘT THÀNH VIÊN DỊCH VỤ CÔNG ÍCH QUẬN 8	55.755.364.447	Chi cục Thuế Quận 8
13	12	0300466257	CÔNG TY CỔ PHAN PHIM GIAI PHÓNG	48.155.063.289	Chi cục Thuế Quận 3
14	13	0302403742	CÔNG TY TNHH THƯƠNG MẠI LO HỘI	43.886.588.117	Chi cục Thuế Quận 3
15	14	0301413360	CÔNG TY CỔ PHẦN NGÂN THẠNH	39.903.958.584	Chi cục Thuế thành Thủ Đức phố
16	15	0304612653	CÔNG TY CỔ PHẦN ĐAU TƯ METRO STAR	37.162.911.972	Chi cục Thuế thành Thủ Đức phố
17	16	0308976796	CÔNG TY CỔ PHAN THƯƠNG MẠI DỊCH VỤ QUẢNG TRƯỜNG QUỐC TẾ	37.035.058.831	Chi cục Thuế Quận 3
18	17	0313937270	CHỦA AN GIÁO	36.110.293.971	Chi cục Thuế Quận]
19	18	0303831229	CONG TY CO PHAN CANG SÀI GÒN- HIỆP PHƯỚC	35.125.800.725	Chi cục Thuế KV Quận 7 - huyện Nhà Bè
20	19	0316922610	CONG TY TNHH MTV TM DV MỸ LE	31.817.315.090	Chi cục Thuế thành Thủ Đức phố
21	20	0301446454	CÔNG TY TNHH THƯƠNG MẠI TƯ VẤN ĐẦU TƯ VÀ PHÁT TRIỂN ĐÔ THỊ	29.489.496.627	Chi cục Thuế thành Thủ Đức phố
22	21	0300734844	CÔNG TY CO PHAN GIÀY DA VÀ MAY MẶC XUẤT KHẨU (LEGAMEX)	28.816.136.616	Chi cục Thuế Quận 10

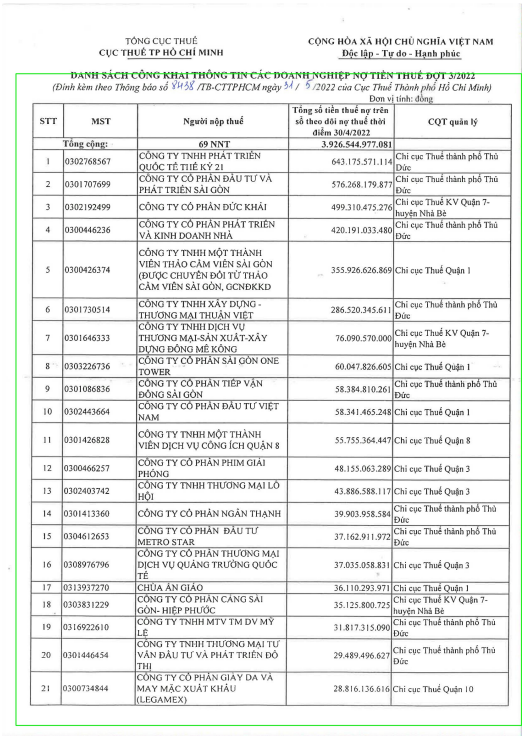
(a) Output spreadsheet of the bordered table contained in the input image 4.10a

	0	1	2	3	4	5	6	7
0		N (% completed)	CPU care	Routine care	Difference Unadjusted	95% CI	P-value	p
1								
2								
3	Physical functioning	654 (96.3%)	74.1	66.2	7.8	3.8 to | [.9	<0.00|	0.025
4					7.6	3.6 to |/.5	<0.001	
5		654 (96.3%)	74.6	67.0	7.6	3.2 to /2.0	0.00/	0
6	Role-physical				6 8	24to U 2	0 002	
7		638 (94.0%)	54.1	46.0	8.2	13 to [5.0	0.02	0
8	Role-emotional	630 (92.89)			70	0 4+0 /3 6	0.039	
9			63.9	60.2	3.7	-3.0 to /0.5	0.28 1	0
10	Mental health	653 (96.2%)		64.4	3.9	-2.8 to /0.5	0.256	
11			69.|		4.7	1.3 to 8.2	0.007	0
12		649 (95.6%)		47.|	5.2	1.9 to 8.6	0.002	
13	Vitality		52.6		5.5	1.8 to 9.2	0.003	0
14				62.0	5.8	2.2 to 9.3	0.002	
15	Pain index	655 (96.5%)	66.4		4.4	0.2 to 8.5	0.04	0
16					4.3	0.2 to 8.3	0.041	
17	General health	651 (95.9%)	59.7	51.7	8.0	4.6 to |1.5	<0.001	0
18					8.|	4.6to |L.5	<0.00|	

	0	1	2	3	4	5	6	7
0		N (% completed)	CPU care	Routine care	Difference Unadjusted	95% CI	P-value	p
1								
2								
3	Physical functioning	694 (96.7%)	74 8	69 7	5 |	LL to 9 0	0 012	0 002
4					4.2	0.4 to 7.9	0.029	
5	Social functioning Ro	703 (98 0%)	72 2	69 8	24	.7 to 6 6	0 252	0
6		n			15	027t2 5 6	0 49	
7		684 (95.4%)	50.4	46.0 59.5	44	22.2 to /L.0	0./91	0.028
8	1 1 Role-emotional	685 (95.5%)			3 3	-3 3 to |0 0	0 326	
9			64.7		5.2	-2 to /L.6	0.13	0
10				647	5.1	-[.2 to /1.4	0.1	
11	Mental health	700 (97.6%)	66.9 52.3		2)	~0 9+o 5 3	0 /58	0
12				47.6	23	-0.7to 5.4	0.132	
13	Vitality	697 (97.2%)			4.6	1.3 to 8.0	0.007	0
14				49.0	4.6	1.3 to 8.0	0.007	
15	Pain index	701 (97.7%)	50.8		L.8	-[.9to 5.5	0.351	0
16					2.0	-L7 to 5.7	0.284	
17	General health	688 (96.09)	60.3	54.5	5.7	2.3 to 9.2	0.00 /	0
18					5.4	2.0 to 8.8	0.002	

(b) Output spreadsheets of two borderless tables contained in the input image 4.10b

Figure 4.11: Output spreadsheet of different kinds of tables



TỔNG CỤC THUẾ
CỤC THUẾ TP HỒ CHÍ MINH

CỘNG HÒA XÃ HỘI CHỦ NGHĨA VIỆT NAM
Độc lập - Tự do - Hạnh phúc

DANH SÁCH CÔNG KHAI THÔNG TIN CÁC DOANH NGHIỆP NỢ TIỀN THUẾ ĐỢT 3/2022

(Đính kèm theo Thông báo số 9458 /TB-CTTPHCM ngày 31 / 5 /2022 của Cục Thuế Thành phố Hồ Chí Minh)

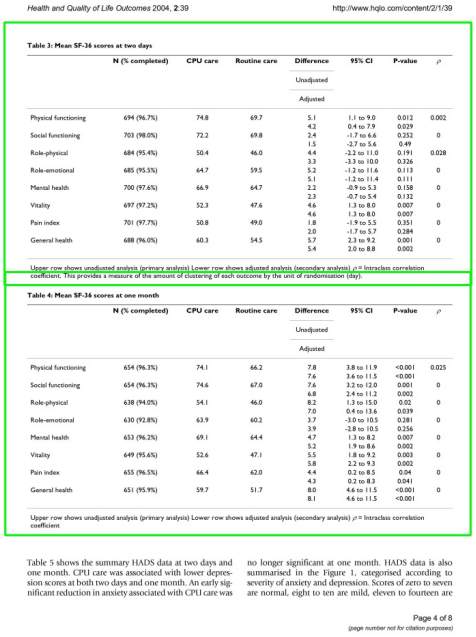
Đơn vị tính: đồng

STT	MST	Người nộp thuế	Tổng số tiền thuế nợ trên sổ theo dõi nợ thuế thời điểm 30/4/2022	CQT quản lý
Tổng cộng:		69 NNT	3.926.544.977.081	
1	0302768567	CÔNG TY TNHH PHÁT TRIỂN QUỐC TẾ TIẾ KỲ 21	643.175.571.114	Chi cục Thuế thành phố Thủ Đức
2	0301707699	CÔNG TY CỔ PHẦN ĐẦU TƯ VÀ PHÁT TRIỂN SÀI GÒN	576.268.179.877	Chi cục Thuế thành phố Thủ Đức
3	0302192499	CÔNG TY CỔ PHẦN ĐỨC KHẢI	499.310.475.276	Chi cục Thuế KV Quận 7-huyện Nhà Bè
4	0300446236	CÔNG TY CỔ PHẦN PHÁT TRIỂN VÀ KINH DOANH NHÀ	420.191.033.480	Chi cục Thuế thành phố Thủ Đức
5	0300426374	CÔNG TY TNHH MỘT THÀNH VIÊN THẢO CẦM VIÊN SÀI GÒN (ĐƯỢC CHUYỂN ĐỔI TỪ THẢO CẦM VIÊN SÀI GÒN, GCNĐKKD	355.926.626.869	Chi cục Thuế Quận 1
6	0301730514	CÔNG TY TNHH XÂY DỰNG - THƯƠNG MẠI THUẬN VIỆT	286.520.345.611	Chi cục Thuế thành phố Thủ Đức
7	0301646333	CÔNG TY TNHH DỊCH VỤ THƯƠNG MẠI-SẢN XUẤT-XÂY DỰNG ĐÔNG MÊ KÔNG	76.090.570.000	Chi cục Thuế KV Quận 7-huyện Nhà Bè
8	0303226736	CÔNG TY CỔ PHẦN SÀI GÒN ONE TOWER	60.047.826.605	Chi cục Thuế Quận 1
9	0301086836	CÔNG TY CỔ PHẦN TIẾP VẬN ĐÔNG SÀI GÒN	58.384.810.261	Chi cục Thuế thành phố Thủ Đức
10	0302443664	CÔNG TY CỔ PHẦN ĐẦU TƯ VIỆT NAM	58.341.465.248	Chi cục Thuế Quận 1
11	0301426828	CÔNG TY TNHH MỘT THÀNH VIÊN DỊCH VỤ CÔNG ÍCH QUẬN 8	55.755.364.447	Chi cục Thuế Quận 8
12	0300466257	CÔNG TY CỔ PHẦN PHIM GIẢI PHÓNG	48.155.063.289	Chi cục Thuế Quận 3
13	0302403742	CÔNG TY TNHH THƯƠNG MẠI LỘ HỘI	43.886.588.117	Chi cục Thuế Quận 3
14	0301413360	CÔNG TY CỔ PHẦN NGÂN THẠNH	39.903.958.584	Chi cục Thuế thành phố Thủ Đức
15	0304612653	CÔNG TY CỔ PHẦN ĐẦU TƯ METRO STAR	37.162.911.972	Chi cục Thuế thành phố Thủ Đức
16	0308976796	CÔNG TY CỔ PHẦN THƯƠNG MẠI DỊCH VỤ QUẢNG TRƯỜNG QUỐC TẾ	37.035.058.831	Chi cục Thuế Quận 3
17	0313937270	CHÙA ÂN GIÁO	36.110.293.971	Chi cục Thuế Quận 1
18	0303831229	CÔNG TY CỔ PHẦN CẢNG SÀI GÒN- HIỆP PHƯỚC	35.125.800.725	Chi cục Thuế KV Quận 7-huyện Nhà Bè
19	0316922610	CÔNG TY TNHH MTV TM DV MỸ LỆ	31.817.315.090	Chi cục Thuế thành phố Thủ Đức
20	0301446454	CÔNG TY TNHH THƯƠNG MẠI TƯ VẤN ĐẦU TƯ VÀ PHÁT TRIỂN ĐÔ THỊ	29.489.496.627	Chi cục Thuế thành phố Thủ Đức
21	0300734844	CÔNG TY CỔ PHẦN GIÀY DA VÀ MAY MẶC XUẤT KHẨU (LEGAMEX)	28.816.136.616	Chi cục Thuế Quận 10

(a) TD result on input image

(b) TSR result on the detected table

Figure 4.12: Visualized outputs of TD and TSR components with an input image containing a bordered table



Health and Quality of Life Outcomes 2004, **2**:39 http://www.hqlo.com/content/2/1/39

Table 3: Mean SF-36 scores at two days

	N (% completed)	CPU care	Routine care	Difference	95% CI	P-value	ρ
				Unadjusted			
				Adjusted			
Physical functioning	694 (96.7%)	74.8	69.7	5.1	1.1 to 9.0	0.012	0.002
				4.2	0.4 to 7.9	0.029	
Social functioning	703 (98.0%)	72.2	69.8	2.4	-1.7 to 6.6	0.252	0
				1.5	-2.7 to 5.6	0.49	
Role-physical	684 (95.4%)	50.4	46.0	4.4	-2.2 to 11.0	0.191	0.028
				3.3	-3.3 to 10.0	0.326	
Role-emotional	685 (95.5%)	64.7	59.5	5.2	-1.2 to 11.6	0.113	0
				5.1	-1.2 to 11.4	0.111	
Mental health	700 (97.6%)	66.9	64.7	2.2	-0.9 to 5.3	0.158	0
				2.3	-0.7 to 5.4	0.132	
Vitality	697 (97.2%)	52.3	47.6	4.6	1.3 to 8.0	0.007	0
				4.6	1.3 to 8.0	0.007	
Pain index	701 (97.7%)	50.8	49.0	1.8	-1.9 to 5.5	0.351	0
				2.0	-1.7 to 5.7	0.284	
General health	688 (96.0%)	60.3	54.5	5.7	2.3 to 9.2	0.001	0
				5.4	2.0 to 8.8	0.002	

Upper row shows unadjusted analysis (primary analysis) Lower row shows adjusted analysis (secondary analysis) ρ = Intraclass correlation coefficient. This provides a measure of the amount of clustering of each outcome by the unit of randomisation (day).

Table 4: Mean SF-36 scores at one month

	N (% completed)	CPU care	Routine care	Difference	95% CI	P-value	ρ
				Unadjusted			
				Adjusted			
Physical functioning	654 (96.3%)	74.1	66.2	7.8	3.8 to 11.9	<0.001	0.025
				7.6	3.6 to 11.5	<0.001	
Social functioning	654 (96.3%)	74.6	67.0	7.6	3.2 to 12.0	0.001	0
				6.8	2.4 to 11.2	0.002	
Role-physical	638 (94.0%)	54.1	46.0	8.2	1.3 to 15.0	0.02	0
				7.0	0.4 to 13.6	0.039	
Role-emotional	630 (92.8%)	63.9	60.2	3.7	-3.0 to 10.5	0.281	0
				3.9	-2.8 to 10.5	0.256	
Mental health	653 (96.2%)	69.1	64.4	4.7	1.3 to 8.2	0.007	0
				5.2	1.9 to 8.6	0.002	
Vitality	649 (95.6%)	52.6	47.1	5.5	1.8 to 9.2	0.003	0
				5.8	2.2 to 9.3	0.002	
Pain index	655 (96.5%)	66.4	62.0	4.4	0.2 to 8.5	0.04	0
				4.3	0.2 to 8.3	0.041	
General health	651 (95.9%)	59.7	51.7	8.0	4.6 to 11.5	<0.001	0
				8.1	4.6 to 11.5	<0.001	

Upper row shows unadjusted analysis (primary analysis) Lower row shows adjusted analysis (secondary analysis) ρ = Intraclass correlation coefficient

Table 5 shows the summary HADS data at two days and one month. CPU care was associated with lower depression scores at both two days and one month. An early significant reduction in anxiety associated with CPU care was no longer significant at one month. HADS data is also summarised in the Figure 1, categorised according to severity of anxiety and depression. Scores of zero to seven are normal, eight to ten are mild, eleven to fourteen are

(a) TD result on input image

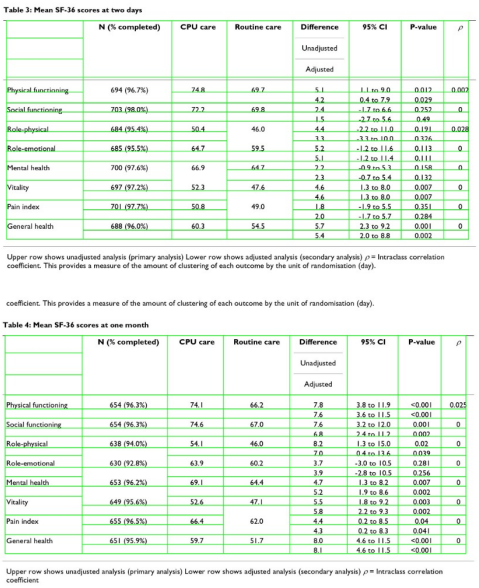


Table 3: Mean SF-36 scores at two days

	N (% completed)	CPU care	Routine care	Difference	95% CI	P-value	ρ
				Unadjusted			
				Adjusted			
Physical functioning	694 (96.7%)	74.8	69.7	5.1	1.1 to 9.0	0.012	0.002
				4.2	0.4 to 7.9	0.029	
Social functioning	703 (98.0%)	72.2	69.8	2.4	-1.7 to 6.6	0.252	0
				1.5	-2.7 to 5.6	0.49	
Role-physical	684 (95.4%)	50.4	46.0	4.4	-2.2 to 11.0	0.191	0.028
				3.3	-3.3 to 10.0	0.326	
Role-emotional	685 (95.5%)	64.7	59.5	5.2	-1.2 to 11.6	0.113	0
				5.1	-1.2 to 11.4	0.111	
Mental health	700 (97.6%)	66.9	64.7	2.2	-0.9 to 5.3	0.158	0
				2.3	-0.7 to 5.4	0.132	
Vitality	697 (97.2%)	52.3	47.6	4.6	1.3 to 8.0	0.007	0
				4.6	1.3 to 8.0	0.007	
Pain index	701 (97.7%)	50.8	49.0	1.8	-1.9 to 5.5	0.351	0
				2.0	-1.7 to 5.7	0.284	
General health	688 (96.0%)	60.3	54.5	5.7	2.3 to 9.2	0.001	0
				5.4	2.0 to 8.8	0.002	

Upper row shows unadjusted analysis (primary analysis) Lower row shows adjusted analysis (secondary analysis) ρ = Intraclass correlation coefficient. This provides a measure of the amount of clustering of each outcome by the unit of randomisation (day).

coefficient. This provides a measure of the amount of clustering of each outcome by the unit of randomisation (day).

Table 4: Mean SF-36 scores at one month

	N (% completed)	CPU care	Routine care	Difference	95% CI	P-value	ρ
				Unadjusted			
				Adjusted			
Physical functioning	654 (96.3%)	74.1	66.2	7.8	3.8 to 11.9	<0.001	0.025
				7.6	3.6 to 11.5	<0.001	
Social functioning	654 (96.3%)	74.6	67.0	7.6	3.2 to 12.0	0.001	0
				6.8	2.4 to 11.2	0.002	
Role-physical	638 (94.0%)	54.1	46.0	8.2	1.3 to 15.0	0.02	0
				7.0	0.4 to 13.6	0.039	
Role-emotional	630 (92.8%)	63.9	60.2	3.7	-3.0 to 10.5	0.281	0
				3.9	-2.8 to 10.5	0.256	
Mental health	653 (96.2%)	69.1	64.4	4.7	1.3 to 8.2	0.007	0
				5.2	1.9 to 8.6	0.002	
Vitality	649 (95.6%)	52.6	47.1	5.5	1.8 to 9.2	0.003	0
				5.8	2.2 to 9.3	0.002	
Pain index	655 (96.5%)	66.4	62.0	4.4	0.2 to 8.5	0.04	0
				4.3	0.2 to 8.3	0.041	
General health	651 (95.9%)	59.7	51.7	8.0	4.6 to 11.5	<0.001	0
				8.1	4.6 to 11.5	<0.001	

Upper row shows unadjusted analysis (primary analysis) Lower row shows adjusted analysis (secondary analysis) ρ = Intraclass correlation coefficient

(b) TSR results on the detected tables

Figure 4.13: Visualized outputs of TD and TSR components with an input image containing borderless tables

4.2.2 Model of Choice

In order to find the most suitable pruning approach, we analyze the characteristics of each method presented in section 2.1.3.

The classical methods are the most traditional approaches to pruning, and they have been applied to CNN and ANN models. However, they are outdated approaches that are not as effective in modern deep learning networks. Unstructured pruning, on the other hand, produces sparse model parameters, which may not be efficient for GPU inference, and this method is more tailored for CNNs.

Modern methods are a more recent approach to pruning that focuses on modern deep learning networks. These methods are tailored for different kinds of architectures, including Transformer-based ones. They are more up-to-date with modern deep learning networks and offer more effective and efficient pruning techniques for modern models.

Neural Architecture Search is a modern and powerful approach to pruning. However, it requires very high computational capability, making it more suitable for larger teams with access to powerful computing resources.

Finally, quantization is a method that can be combined with any pruning method but does not reduce the number of parameters. It is a technique used to reduce the precision of model parameters, making them more efficient for GPU inference. This method can be combined with any other parameter pruning method.

In conclusion, the key factors that make modern pruning the optimal approach for Transformer compression are: (1) their focus on state-of-the-art deep networks like Transformers, rather than classical models; (2) their ability to remove parameters through structured pruning by exploiting Transformer architectures; (3) their direct optimization of model efficiency through parameter reduction, unlike quantization alone; and (4) their lower resource requirements compared to neural architecture search. Given these significant benefits of modern pruning methods for Transformers with minimal downsides relative to alternatives, they are the most compelling choice for compressing and speeding up Transformer-based models while maintaining accuracy.

Therefore, we have chosen the DEtection TRansformer (DETR) as our primary

Table Structure Recognition (TSR) model and the target model for parameter pruning. Our selection of DETR over other TSR models is based on several key factors. Firstly, it is capable of accommodating variable numbers of objects in an image, ensuring a consistent level of detection quality across a wide range of cell numbers in tables. Secondly, it is built on Transformer architecture, which is highly optimized and has significant potential for future development. However, this also presents a challenge when it comes to parameter pruning. Thirdly, the DETR model is trained on the PubTables-1M dataset, which is the most recent and formal table dataset published by Microsoft and has better generalization capabilities with real-life data. Finally, the original source code published by the authors is up-to-date and systematic and employs new versions of deep learning frameworks such as PyTorch and OpenCV, among others. This not only makes it more convenient for us to implement the pruning methods onto this baseline but also to evaluate the pruned models on PubTables-1M.

It is worth noting that the DEtection TRansformer model has demonstrated a high degree of efficacy in table structure recognition tasks, making it a solid choice for our research. In addition, the Transformer architecture used in the model is highly optimized for processing sequential data and has been shown to produce state-of-the-art results in a wide range of natural language processing and computer vision tasks, further reinforcing our decision to utilize it as our primary model.

We anticipate that our proposed parameter pruning methods will further enhance the performance of the DETR model while maintaining its high level of accuracy.

4.2.3 Details of algorithm

Annotations and Transformer Layers Revisit

The previous section provided a detailed explanation of the original Transformer architecture, which forms the basis of the study. In this section, we will discuss how the Transformer architecture was annotated, as well as the forward pass of a Transformer encoder. We followed the notation conventions of Transformer, denoting the number of encoder layers as ℓ, the hidden size as h, and the number of attention heads as a. In the original Transformer implementation described in Vaswani et

al. [55], the key-query dimension for multi-head attention k was set to ℓ/a, and the value dimension for multi-head attention v was set to k. The feed-forward filter size f was set to 4h. In order words, there are three design dimensions in Transformer: ℓ, h, and a, which are listed in Table 4.2. The original Transformer has ℓ=6, h=512, and a=8. The other three dimensions f, k, v are a function of h and a. Each layer is identical and uses the same value of a, f, k, v.

It is important to note that Transformer does not require all encoder/decoder layers to be identical. This aspect of the design can be optimized, resulting in non-identical layers. A generalized Transformer will have a_1, a_2, ..., a_ℓ number of heads, f_1, f_2, ..., f_ℓ filter sizes in the feedforward networks, k_1, k_2, ..., k_ℓ key sizes, and v_1, v_2, ..., v_ℓ value sizes in the attention heads, in the layers 1, 2, ..., ℓ respectively. All the design dimensions of Transformer that can be optimized without changing the architecture are given in Table 4.3.

The goal of the study is to optimize (by pruning) all these dimensions to maximize accuracy for a given size of the model. Each design dimension is tied to more than one parameter matrix. The parameter matrices associated with each design dimension are explained by providing a detailed view of an encoder layer of Transformer.

Figure 4.14 demonstrates the structure of a Transformer encoder layer. The figure's symbols are subscripted with 1, indicating the first encoder layer. The input to an encoder layer is the hidden representation of a token, which is of dimension h. The input first passes through a multi-head attention cell. Note that the multi-head attention cell processes the hidden representation of all the tokens in a combined way, but for simplicity, we show only one hidden representation in Figure 4.14. The multi-head attention cell is made up of three parameter tensors - key K_1, query Q_1, and value V_1. K_1 has a size of $k_1 \times a_1 \times h$. The key vector for each head of the attention is of dimension k_1, and a_1 represents the number of heads. The hidden representation of dimension h is projected onto the key tensor K_1 to get a_1 key vectors, each of dimension k_1. Similarly, the query tensor Q_1 is used to get a_1 query vectors, each of dimension k_1, for the a_1 heads of the multi-head attention cell. The value tensor V_1 has a size of $v_1 \times a_1 \times h$. The hidden representation is projected onto the value tensor V_1 to get a_1 value vectors, each of dimension v_1. Note that k_1 and v_1 can be different. The inner product of the key and query vectors, after passing through the

Transformer		
number of encoder layers	ℓ	12
hidden size	h	768
number of self-attention heads	a	12
feed forward dimension	f	$4h$
key-query dimension for attention	k	h/a
value dimension for attention	v	h/a

Table 4.2: Transformer hyper-parameters

Pruned Transformer	
ℓ	ℓ
h	h
a	$a_1, a_2, \cdots, a_\ell$
f	$f_1, f_2, \cdots, f_\ell$
k	$k_1, k_2, \cdots, k_\ell$
v	$v_1, v_2, \cdots, v_\ell$

Table 4.3: Pruned Transformer hyper-parameters

softmax layer, gives weights for combining value vectors. For details on the multi-head attention cell, we refer readers to Vaswani et al. [55]. In summary, using three parameter tensors - K_1, Q_1, V_1 - a multi-head attention cell transforms the hidden representation of size h to a vector of dimension $(v_1 \times a_1)$.

The vector is then projected back to dimension h using a projection matrix P1 and element-wise added to the input of the encoder cell. Layer norm is applied to this sum. The result is passed through two fully-connected layers, D_1 and G_1. D1 has a parameter matrix of size $f_1 \times h$ and G1 has a parameter matrix of size $h \times f_1$. The output of G_1 is added to the input of D_1 and layer norm is applied. This is the output of the encoder cell and is fed into the next encoder cell.

In Figure 4.14, the color coding shows which vectors must have the same dimension. The hidden representation size h is constant across all encoder layers. In a multi-head attention cell, key and query vectors in each head must have the same dimension, so key and query tensors K_1 and Q_1 are of size $k_1 \times a_1 \times h$. The value vector can have a different dimension k_1, so the value tensor V_1 is of size $v_1 \times a_1 \times h$. The filter size f1 in the fully-connected layers D_1 and G_1 can be any integer.

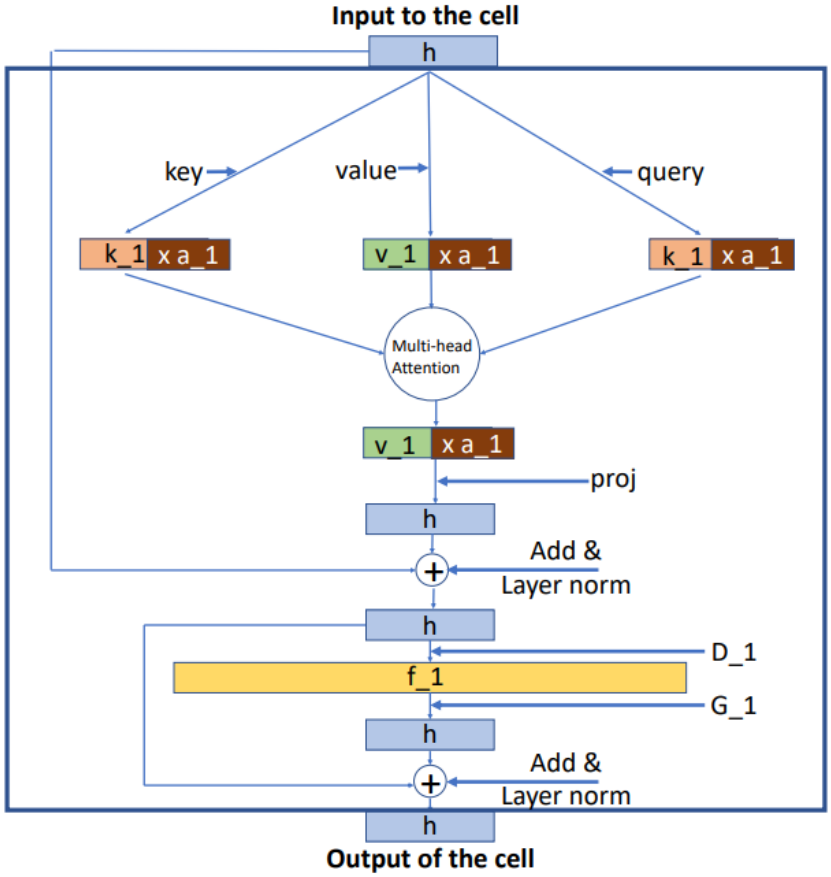


Figure 4.14: A Transformer Encoder Layer

Pruning method

In order to prune the size of the original Transformer architecture, we optimize the Transformer design dimensions, which are provided in Table 4.3. All design dimensions have an upper bound listed in Table 4.2, which is the same as their original value in Transformer. Since we kept the architecture the same, we did not remove any layers, and the design dimensions have a lower bound of one.

For each design dimension we optimized, we introduced a prune-parameter vector α of the same size as the original dimension. The prune-parameter vector was then multiplied by the parameter tensors/matrices associated with that specific design dimension. The original Transformer architecture was left intact, which means no layers were removed. As a result, the design dimensions had a minimum value of one.

$$\alpha_h \in \mathbf{R}^h$$

$$\left\{\alpha_{f_i} \in \mathbf{R}^f\right\}_{i=1,2,\cdots,\ell}$$

$$\left\{\alpha_{a_i} \in \mathbf{R}^a\right\}_{i=1,2,\cdots,\ell}$$

$$\left\{\alpha_{k_i} \in \mathbf{R}^k\right\}_{i=1,2,\cdots,\ell}$$

$$\left\{\alpha_{v_i} \in \mathbf{R}^v\right\}_{i=1,2,\cdots,\ell}$$

Table 4.4: Prune-parameters

To provide an example, the filter size of the feed-forward layer in the first encoder layer is $f_1 = 3072$, and to optimize this dimension, a prune-parameter vector $\alpha_{f_1} \epsilon R^{3072}$ was introduced and initialized with all ones. In the original Transformer, the two parameter matrices D_1 and G_1 were tied to the design dimension f_1. The matrices are now replaced by $diag(\alpha_{f_1}) \cdot D_1$ and $G_1 \cdot diag(\alpha_{f_1})$ in the Transformer pre-trained model.

Table 4.4 lists all the prune-parameters, and Table 4.5 lists all the parameter tensors/matrices for which design dimensions are optimized by multiplying prunable parameters on all sides. Key and query tensors K_i, Q_i for $i\epsilon 1, 2, ..., \ell$ are multiplied on all three sides with prunable parameters corresponding to key-vector, number of attention heads, and hidden size. The value tensor V_i is multiplied in a similar manner using a different value-vector prunable parameter. The proj tensor has the same multiplication as the value tensor. The two feedforward matrices The multiplications of the two feedforward matrices D_i and G_i are identical. The prunable tensors resulting from the optimization process were denoted with a tilde on their top.

We do not have prune parameters for pruning number of encoder/decoder layers. We keep the default number of encoder/decoder layers, ℓ.

Our approach involves identifying the individual prunable parameters that can be set to zero with minimal increase in the pretraining loss. These parameters include α_h, $\{\alpha_{a_i}, \alpha_{v_i}, \alpha_{k_i}, \alpha_{f_i}\}$ $i \epsilon [1, 2, \ldots, \ell]$. Once we have these sparse prunable parameter vectors, we determine the appropriate value of each dimension in the Transformer parameter matrices, including K_i, Q_i, V_i, P_i, D_i, G_i $i \epsilon [1, 2, \ldots, \ell]$, to obtain a smaller

$$\mathrm{K}_i \in \mathbf{R}^{k \times a \times h} \rightarrow K_i \left[\mathcal{D}\left(\alpha_{k_i}\right) \mathcal{D}\left(\alpha_{a_i}\right) \mathcal{D}\left(\alpha_h\right)\right] \equiv \widetilde{K_i}$$

$$\mathrm{Q}_i \in \mathbf{R}^{k \times a \times h} \rightarrow Q_i \left[\mathcal{D}\left(\alpha_{k_i}\right) \mathcal{D}\left(\alpha_{a_i}\right) \mathcal{D}\left(\alpha_h\right)\right] \equiv \widetilde{Q}_i$$

$$\mathrm{V}_i \in \mathbf{R}^{v \times a \times h} \rightarrow V_i \left[\mathcal{D}\left(\alpha_{v_i}\right) \mathcal{D}\left(\alpha_{a_i}\right) \mathcal{D}\left(\alpha_h\right)\right] \equiv \widetilde{V}_i$$

$$\mathrm{P}_i \in \mathbf{R}^{h \times v \times a} \rightarrow P_i \left[\mathcal{D}\left(\alpha_h\right) \mathcal{D}\left(\alpha_{v_i}\right) \mathcal{D}\left(\alpha_{a_i}\right)\right] \equiv \widetilde{P}_i$$

$$\mathrm{D}_i \in \mathbf{R}^{f \times h} \rightarrow D_i \left[\mathcal{D}\left(\alpha_{f_i}\right) \mathcal{D}\left(\alpha_h\right)\right] \equiv \widetilde{D_i}$$

$$\mathrm{G}_i \in \mathbf{R}^{h \times f} \rightarrow G_i \left[\mathcal{D}\left(\alpha_h\right) \mathcal{D}\left(\alpha_{f_i}\right)\right] \equiv \widetilde{G_i}$$

Table 4.5: Prunable Transformer parameter matrices/tensors

and faster Transformer model. The algorithm for finding these sparse prunable parameters is explained in detail below.

First, we use the prunable parameters given in Table 4.4 and initialize them with all ones. These prunable Transformer parameter matrices are created by multiplying the prunable parameters to the learned Transformer parameter matrices, as given in Table 4.5. We then optimize the prunable parameters α's with a sparsity-inducing loss with a regularization coefficient γ incorparated into it, together with the original loss, as given in Equation 4.1. The $L1$ penalty induces sparsity in the parameters, and we multiply the $L1$ loss terms with the cost terms β's to minimize the number of parameters. This is because each element of prune parameters α, when set to zero, reduces a different number of Transformer parameters.

After training the prunable Transformer model for a fixed number of steps, we truncate the smallest prune parameters to zero to identify the appropriate value of each dimension of the Transformer parameter matrices, including K_i, Q_i, V_i, P_i, D_i, G_i $i \ \epsilon \ [1, 2, \ldots, \ell]$. Finally, we proceed to fine-tune the newly pruned Transformer model.

$$\begin{aligned}
&\arg \min_{\left\{\alpha_h, \left\{\alpha_{a_i}, \alpha_{v_i}, \alpha_{k_i}, \alpha_{f_i}\right\}_{i \in [\ell]}\right\}} \\
&\textit{original loss} \\
&+ \gamma \left\{\beta_h \left\|\alpha_h\right\|\right\} + \gamma \sum_{i=1}^{\ell} \left\{\beta_{a_i} \left\|\alpha_{a_i}\right\| + \beta_{v_i} \left\|\alpha_{v_i}\right\| \right. \\
&\left. + \beta_{k_i} \left\|\alpha_{k_i}\right\| + \beta_{f_i} \left\|\alpha_{f_i}\right\|\right.\!.
\end{aligned} \tag{4.1}$$

Algorithm 1 summarizes the pruning method.

Algorithm 1: Parameter pruning method

Input: A Transformer-based model, target fraction η, number of epochs N

Output: An optimally pruned Transformer model

/* Initialization */

Initialize prunable parameters $\alpha_h, \alpha_{a_i}, \alpha_{k_i}, \alpha_{v_i}, \alpha_{f_i}$

Multiply prunable parameters with network parameter $K_i, Qi, Vi, P_i, D_i, G_i \rightarrow \tilde{K}_i, \tilde{Q}_i, \tilde{V}_i, \tilde{P}_i, \tilde{D}_i, \tilde{G}_i$

repeat N **times**

| Train the network using the *Pruning Loss*

end

/* After training */

Set the ζ smallest prunable parameters to zero to achieve η reduction in the target objective value

Create smaller model parameter tensors after knowing the exact values of the hyperparameters in each block. $K_i, Qi, Vi, P_i, D_i, G_i \rightarrow \hat{K}_i, \hat{Q}_i, \hat{V}_i, \hat{P}_i, \hat{D}_i, \hat{G}_i$

Finetune the new model using the Original Loss Function

Chapter 5

Experimental results and discussion

In Chapter 5, we present the results of our experiments on various Table Detection (TD) and Table Structure Recognition (TSR) models, using multiple defined metrics to evaluate their performance. We provide a detailed discussion on the advantages and disadvantages of each candidate model, and draw conclusions regarding the pipelines, TD/TSR models, and the efficiency of our parameter pruning methods.

5.1 Table OCR pipeline

5.1.1 Experimental results

Table Detection

Experimental results from authors

In order to compare different models, we analyzed the results published by their original authors across various evaluation datasets. However, due to differences in the evaluation datasets chosen for each model, only some of the models were evaluated on specific datasets.

For ICDAR 2019, we found that DIT outperformed other models at all IoU thresholds with a weighted average F1-score of 0.9629, as shown in Table 5.1. CDeC-Net also achieved a competitive F1-score of 0.944. However, it is important to note that while

Model	IoU				
	@0.6	@0.7	@0.8	@0.9	WAvg.
CascadeTabNet	0.943	0.934	0.925	0.901	0.901
DiT	**0.9789**	**0.9722**	**0.97**	**0.9388**	**0.9629**
CDeCNet	NaN	NaN	NaN	NaN	0.944

Table 5.1: Experimental results on ICDAR 2019 (Modern)

Model	Recall	Precision	F1
CascadeTabNet	1	1	1
CDeCNet	1	1	1

Table 5.2: Experimental results on ICDAR 2013

these models performed well on this particular dataset, we cannot make a general statement about their overall performance.

Moving on to ICDAR 2013, this dataset has been published for some time now, and knowledge of it has been transferred through many generations of models. As a result, many models are now able to achieve a maximum score on benchmarking this dataset, as shown in Table 5.2. While it is still important to continue evaluating models on this dataset, it is clear that the difficulty of this dataset has decreased over time.

We then turned our attention to TableBank, where we found that CDeC-Net outperformed CascadeTabNet on this dataset by achieving much higher recall (0.979 vs 0.9299), precision (0.995 vs 0.9571), and F1-score (0.987 vs 0.9433), as shown in Table 5.3. This suggests that CDeC-Net may be a more suitable model for this specific task. Finally, in Table 5.4 we found that DiT achieved a slightly higher mAP than CDeC-Net.

Experimental results from our testing datasets.

In order to conduct our experiment, we constructed two distinct testing datasets to

Model	Recall	Precision	F1
CascadeTabNet	0.9299	0.9571	0.9433
CDeCNet	**0.979**	**0.995**	**0.987**

Table 5.3: Experimental results on TableBank

Model	mAP
DIT	**0.978**
CDeCNet	0.967

Table 5.4: Experimental results on PubLayNet

evaluate the performance of various tabular object detection (TD) models. The first dataset, known as the public testing dataset, was carefully sampled from five publicly available TD datasets, namely TNCR, TableBank, PubTable-1M, PubLayNet, and ICDAR 2019 Track A. To be specific, we collected 2000 images and their corresponding labels from each of the first four datasets, and 1329 images from ICDAR 2019 Track A, resulting in a total of around 9000 images in the public testing dataset. Meanwhile, the second testing dataset, referred to as the private testing dataset, consisted of 211 images of Vietnamese financial statements and Japanese repository management spreadsheets, which were selected to simulate real-world scenarios.

To evaluate the performance of various TD models, we employed recall, precision, and F1-score metrics at different intersection-over-union (IoU) thresholds. Specifically, we presented the experimental results on the public testing dataset in Table 5.5. As shown in the table, DETR, CDeC-Net, and CascadeTabNet achieved the highest precision, while Multi-Type-TD-TSR and Cascade Mask RCNN trained on TNCR achieved the highest recall. This suggests that no single model can outperform others in all aspects. To further examine the generalization ability of these models on real-life data, we evaluated them on our private testing dataset.

Model	IoU 0.5			IoU 0.6			IoU 0.7			IoU 0.8			IoU 0.9		
	Recall	Precision	F1-Score	Recall	Precision	F1-Score	Recall	Precision	F1-Score	Recall	Precision	F1-Score	Recall	Precision	F1-Score
DETR	17.22	91.72	29.00	16.51	87.90	27.79	14.59	77.71	24.57	11.36	60.51	19.13	2.39	12.74	4.03
CDeCNet	31.34	94.93	47.12	31.34	94.93	47.12	30.98	93.84	46.58	25.60	77.54	38.49	10.77	32.61	16.19
CascadeTabNet	31.22	96.31	47.15	31.22	96.31	47.15	30.98	95.57	46.79	29.78	91.88	44.99	25.12	77.49	37.94
faster_rcnn	75.72	54.06	63.08	75.12	53.63	62.58	74.52	53.20	62.08	71.89	51.32	59.89	39.59	28.27	32.98
Cascade RCNN on ResNeXt-101-32x4d_1x trained on TNCR	55.38	59.9	57.55	67.46	50.63	57.85	66.99	50.27	57.44	66.51	49.91	57.03	52.87	39.68	45.33
Cascade RCNN on ResNeXt-101-64x4d_1x trained on TNCR	63.4	50.91	56.47	63.4	50.91	56.47	62.8	50.43	55.94	62.08	49.86	55.3	51.2	41.11	45.6
Cascade Mask RCNN on ResNeXt-101-64x4d_1x trained on TNCR	75.6	58.46	65.94	75.48	58.37	65.83	75.12	58.09	65.52	74.16	57.35	64.68	64.23	49.68	56.03
Modern Cascade RCNN DIT Base	75.72	54.06	63.08	75.12	53.63	62.58	74.52	53.2	62.08	71.89	51.32	59.89	39.59	28.27	32.98
Modern Cascade RCNN DIT Large	75.72	54.06	63.08	75.12	53.63	62.58	74.52	53.2	62.08	71.89	51.32	59.89	39.59	28.27	32.98
Modern Mask RCNN DIT Base	75.72	54.06	63.08	75.12	53.63	62.58	74.52	53.2	62.08	71.89	51.32	59.89	39.59	28.27	32.98
Modern Mask RCNN DIT Large	75.72	54.06	63.08	75.12	53.63	62.58	74.52	53.2	62.08	71.89	51.32	59.89	39.59	28.27	32.98

Table 5.5: Experimental results on our public testing dataset

Table 5.6 presents the experimental results of TD models on our private testing dataset. We observed that CDeC-Net achieved the highest precision and F1-score at every threshold, with DETR coming in a close second for precision. On the other hand, CascadeTabNet achieved the highest recall, but was associated with extremely low precision, resulting in low F1-scores. This means that CascadeTabNet tended to produce multiple false positives. Interestingly, we found that there was no significant difference between cascade RCNNs and cascade mask RCNNs. All cascade (mask) RCNN-based models achieved similar scores and maintained a balance between recall and precision. Moreover, we did not observe significant drops in the performance of these models across different IoU thresholds.

Paper	Model	IoU 0.5			IoU 0.6			IoU 0.7			IoU 0.8			IoU 0.9		
		Recall	Precision	F1-Score	Recall	Precision	F1-Score	Recall	Precision	F1-Score	Recall	Precision	F1-Score	Recall	Precision	F1-Score
PubTable-1M	DETR	45.98	77.34	57.67	43.17	72.61	54.14	39.89	67.1	50.04	35.72	60.09	44.81	29.65	49.87	37.19
CDeCNet	CDeCNet	57.72	**77.78**	**66.27**	55.28	**74.5**	**63.47**	52.19	**70.33**	**59.92**	48.87	**65.86**	**56.11**	43.15	**58.14**	**49.54**
CascadeTabNet	CascadeTabNet	**69.09**	4.47	8.39	**65.65**	4.24	7.97	**61.8**	3.99	7.5	**57.39**	3.71	6.97	**44.2**	2.86	5.37
Multi-Type-TD-TSR	Multi-Type-TD-TSR	56.03	64.6	60.01	52.87	60.96	56.63	48.6	56.03	52.05	43.57	50.24	46.67	35.45	40.88	37.97
TNCR	Cascade RCNN on ResNeXt-101-32x4d_1x trained on TNCR	52.06	69.98	59.7	49.4	66.41	56.66	46.62	62.67	53.47	43.51	58.49	49.9	38.36	51.57	44
	Cascade RCNN on ResNeXt-101-64x4d_1x trained on TNCR	52.4	70.62	60.16	49.86	67.19	57.24	47.18	63.57	54.16	44.25	59.62	50.8	39.66	53.44	45.53
	Cascade Mask RCNN on ResNeXt-101-64x4d_1x trained on TNCR	51.68	67.18	58.42	49.08	63.81	55.49	46.31	60.21	52.35	43.74	56.87	49.45	39.87	51.84	45.07
DiT	Modern Cascade RCNN DiT Base	56.03	64.6	60.01	52.87	60.96	56.63	48.6	56.03	52.05	43.57	50.24	46.67	35.45	40.88	37.97

Table 5.6: Experimental results on our private testing dataset

Clone Name	NLS1	NLS2	Nucleus	Cytoplasm
TWIST^{wt}	RKRR	KRGKK	+	
K38R	R**R**RR	KRGKK		+
K73R	RKRR	**R**RGKK		+
K76R	RKRR	KRG**R**K	+	
K77R	RKRR	KRGK**R**		+

Figure 5.1: Borderless

Based on our experimental results and the results from the original authors, we selected DETR, CDeC-Net, and one of the Cascade RCNNs trained on TNCR as the candidates for our baseline TD model. For a more comprehensive analysis of the advantages and disadvantages of these models, please refer to the section 5.1.2.

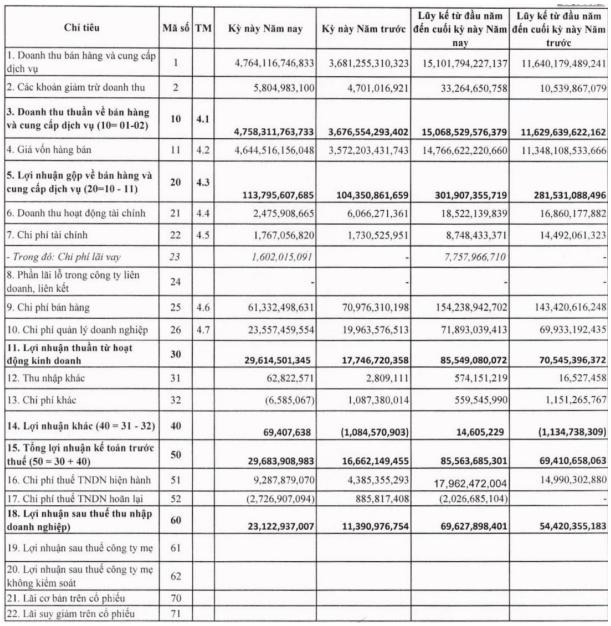
Table Classification

For this part of the pipeline, we collected and labeled 1200 samples for testing. The data is a mix of Vietnamese and English documents spanning various domains taken from the public datasets we have listed. Each sample is classified as Bordered or Borderless (visualization available at 5.2 and 5.1).

Our method achieved 97.16% accuracy in classifying the tables. Notably, while this method can capture a large proportion of borderlines thus making it suitable for classification class, they are not reliable for capturing every line thus making them unsuitable to be a reconstruction method. After performing an error analysis, we were able to determine the main percentage of the errors come in the form of tables that have thin and fuzzy borderlines which do not get pickup by the classifier process(5.3)

Table Recognition

Experimental results from authors Because of the complications of defining table structure and the lack of labeled data for this particular problem, we mostly have to rely on experimental results provided by the authors of their respective publications. The most common and reliable dataset of choice when testing for table structure recognition is ICDAR 2013 Table Competition dataset (task 2) [33]. With



Chỉ tiêu	Mã số	TM	Kỳ này Năm nay	Kỳ này Năm trước	Lũy kế từ đầu năm đến cuối kỳ này Năm nay	Lũy kế từ đầu năm đến cuối kỳ này Năm trước
1. Doanh thu bán hàng và cung cấp dịch vụ	1		4,764,116,746,833	3,681,255,310,323	15,101,794,227,137	11,640,179,489,241
2. Các khoản giảm trừ doanh thu	2		5,804,983,100	4,701,016,921	33,264,650,758	10,539,867,079
3. Doanh thu thuần về bán hàng và cung cấp dịch vụ (10= 01-02)	**10**	**4.1**	**4,758,311,763,733**	**3,676,554,293,402**	**15,068,529,576,379**	**11,629,639,622,162**
4. Giá vốn hàng bán	11	4.2	4,644,516,156,048	3,572,203,431,743	14,766,622,220,660	11,348,108,533,666
5. Lợi nhuận gộp về bán hàng và cung cấp dịch vụ (20=10 - 11)	**20**	**4.3**	**113,795,607,685**	**104,350,861,659**	**301,907,355,719**	**281,531,088,496**
6. Doanh thu hoạt động tài chính	21	4.4	2,475,908,665	6,066,271,361	18,522,139,839	16,860,177,882
7. Chi phí tài chính	22	4.5	1,767,056,820	1,730,525,951	8,748,433,371	14,492,061,323
- Trong đó: Chi phí lãi vay	*23*		*1,602,015,091*	-	*7,757,966,710*	-
8. Phần lãi lỗ trong công ty liên doanh, liên kết	24		-	-	-	-
9. Chi phí bán hàng	25	4.6	61,332,498,631	70,976,310,198	154,238,942,702	143,420,616,248
10. Chi phí quản lý doanh nghiệp	26	4.7	23,557,459,554	19,963,576,513	71,893,039,413	69,933,192,435
11. Lợi nhuận thuần từ hoạt động kinh doanh	**30**		**29,614,501,345**	**17,746,720,358**	**85,549,080,072**	**70,545,396,372**
12. Thu nhập khác	31		62,822,571	2,809,111	574,151,219	16,527,458
13. Chi phí khác	32		(6,585,067)	1,087,380,014	559,545,990	1,151,265,767
14. Lợi nhuận khác (40 = 31 - 32)	**40**		**69,407,638**	**(1,084,570,903)**	**14,605,229**	**(1,134,738,309)**
15. Tổng lợi nhuận kế toán trước thuế (50 = 30 + 40)	**50**		**29,683,908,983**	**16,662,149,455**	**85,563,685,301**	**69,410,658,063**
16. Chi phí thuế TNDN hiện hành	51		9,287,879,070	4,385,355,293	17,962,472,004	14,990,302,880
17. Chi phí thuế TNDN hoãn lại	52		(2,726,907,094)	885,817,408	(2,026,685,104)	-
18. Lợi nhuận sau thuế thu nhập doanh nghiệp)	**60**		**23,122,937,007**	**11,390,976,754**	**69,627,898,401**	**54,420,355,183**
19. Lợi nhuận sau thuế công ty mẹ	61					
20. Lợi nhuận sau thuế công ty mẹ không kiểm soát	62					
21. Lãi cơ bản trên cổ phiếu	70					
22. Lãi suy giảm trên cổ phiếu	71					

Figure 5.2: Bordered

Method	F1-score	Recall	Precision
Deep Splitting and Merging	0.9300	0.9224	0.9378
TableNet	0.9098	0.8987	0.9215
GraphTSR	0.8370	0.8550	0.8190
DeepDeSRT	0.9144	0.8736	0.9593

Table 5.7: Collected table recognition results on ICDAR 2013 dataset

word-level annotations provided for collected public domain documents, this is the standard benchmark dataset for evaluating the task of reconstructing table structure. The results of the aforementioned models on this dataset have been compiled into one table 5.7. Unlike table detection, table structure recognition task still proves to be a challenging problem with the lack of labeled data both for testing and training as well as no models approaching optimal score.

We can see that except for GraphTSR which scores a considerable lower score, the other 3 methods prove to have relatively similar performance. Based on results above, TableNet, DeepDeSRT and Deep Splitting and Merging are the candidates for our table recognition model and a deeper look at their respective architect and details

Chỉ tiêu	Số Báo cáo	Số Kiểm toán	Chênh lệch
A	1	2	3=2-1
I. Thuế	**2.913.798.978**	**6.948.251.373**	**4.034.452.395**
1.Thuế GTGT	1.239.438.883	1.336.737.746	97.298.863
2.Thuế thu nhập doanh nghiệp	1.135.649.807	3.293.378.096	2.157.728.289
3.Thuế thu nhập cá nhân	56.489.528	56.489.528	
4.Thuế nhà đất và tiền thuê đất		1.779.425.243	1.779.425.243
5. Thuế tài nguyên	482.220.760	482.220.760	
6. Các loại thuế khác		0	
II. Các khoản phải nộp khác	**151.360.440**	**172.060.440**	**20.700.000**
1. Phí bảo vệ môi trường	151.360.440	151.360.440	
2. Các khoản phải nộp khác		20.700.000	20.700.000
		0	
Tổng cộng =I+II	**3.065.159.418**	**7.120.311.813**	**4.055.152.395**

Figure 5.3: A fail classification case

are available in the section 5.1.2.

5.1.2 Discussion

Overall Pipeline

Combining multiple approaches to tackle complex problems is a common practice in machine learning research. By doing so, researchers can leverage the different strengths and advantages that are offered by each approach. This approach is particularly useful in the context of building a pipeline for table detection and recognition tasks. Those strengths, however, also come with a cost.

Advantages

The reason for this approach is that we can research and implement the most specialized solution that offers the best performance for each individual task. As presented in our document, solutions for table detection and recognition tasks still focus on modular individual problems. By dividing the pipeline into modules, we can turn the lack of a universal solution into an advantage. Furthermore, the modularity of this approach makes it easier to improve results. Although the modules are connected, they work separately and can be altered or upgraded without affecting the whole pipeline. This modularity also helps when the project progresses, and changes or experiments are made. Researchers can always have a working system for

Model	Dataset	Architecture	Pretrained	Number of parameters	Year
DETR	PubTables-1M	Transformer	True	28,642,567	2022
CDeCNet	All public dataset	Cascade RCNN	True	144,098,452	2020
Cascade RCNN on ResNeXt-101-32x4d_1x	TNCR	Cascade RCNN	True	87,569,453	2021

Table 5.8: Table Detection Models of Choice

demonstration separate from the experimental ones.

Another advantage of this approach is that by categorizing tables into two types - border and borderless - we can divide and conquer each type with the deep learning models that are best suited for the task. This strategy helps to optimize the performance of the pipeline and improves the accuracy of the output.

Disadvantages

However, there are also some disadvantages to this approach. For example, in dividing the pipeline into many parts, we increase the chance of each part malfunctioning. This can make it more challenging to set up and get all the parts to work in harmony. Execution time is another challenge, as the more modules there are, the more time the whole pipeline will need to finish. This can lead to slower processing times and may not be suitable for real-time applications.

Furthermore, there are other points of concern that, while not serious problems, are worth considering. For instance, using multiple deep learning models can increase the workload, and various conversions may be needed from one component's output into another's input.

In summary, the modular approach to building a pipeline for table detection and recognition tasks offers several advantages, such as leveraging the strengths of different solutions and improving the accuracy of the output. However, there are also some challenges associated with this approach, such as increasing the chance of component failure and slower processing times. Researchers need to consider these pros and cons when designing a pipeline and tailor it to their specific needs.

Table Detection Models

Table 5.8 shows some basic information about the table detection models of our choice, including the dataset(s) they are trained on, their architecture, whether they have been pretrained or not, the number of parameters, and the year of publication.

Each of these models has its own set of strengths and weaknesses.

DEtection TRansformer (DETR).

Advantages:

- Able to handle variable numbers of objects in an image
- Highly interpretable, with attention maps that can be used to visualize its decision-making process
- Fast and does not require expensive computational capability
- Trained on PubTables-1M, which contains detailed header and location information for table structures and tends to generalize better in real-life datasets

Disadvantages:

- May not perform well on distorted images since there are no such images in the PubTables-1M dataset that it is trained on

CDeC-Net.

Advantages

- Unique multi-stage extension of Mask R-CNN and a dual backbone with deformable convolution, making it highly flexible and adaptable.
- Composite backbone improves detection accuracy without adding too much computational cost.
- Deformable convolution captures features using a variable receptive field and makes detection independent of fixed geometric transforms.
- Excellent performance across multiple thresholds of IoU and is trained on many available datasets, resulting in good performance on the public testing dataset.

Disadvantages

- Number of parameters (144,098,452) is the largest among the three models discussed, which makes the inference time of CDeC-Net slow compared to the other two.

- Source code is not easy to develop and retrain since it uses an outdated version of PyTorch.

Cascade RCNN on ResNeXt-101-32x4d_1x
Advantages:

- The Cascade RCNN on ResNeXt-101-32x4d_1x is a deep learning-based method for table detection that achieves state-of-the-art performance on the TNCR dataset, which contains 9428 high-quality labeled images.
- The cascade architecture of the model allows for stable performance at different thresholds.
- The number of parameters (87,569,453) of cascade RCNN provides stable detection quality at a sufficient speed.

Disadvantages:

- The number of more than 87 million parameters is still quite expensive for most GPUs nowadays, making the training procedure take a while on popular commercial GPUs.
- The model requires anchor boxes and non-maximum suppression, which can lead to inefficiencies and inaccuracies.

Conclusion

In summary, each of the three models has its own set of strengths and weaknesses. The DETR model is able to handle variable numbers of objects and is highly interpretable, but may not perform well on distorted images. The CDeC-Net model has a unique design that makes it well-suited to address the problems of noisy detection and large transformations but has a large number of parameters and slow inference time. The Cascade RCNN on ResNeXt-101-32x4d_1x achieves state-of-the-art performance on the TNCR dataset but requires anchor boxes and non-maximum suppression. Overall, the choice of model depends on the specific needs of the user, including the dataset, computational resources, and desired performance metrics.

Model	Dataset	Architecture	Pretrained	Number of parameters	Year
Deep Splitting and Merging	Private dataset	Pair of convolutional models with a novel pooling regions	True	No official implementation	2019
TableNet	Marmot Extended	Encoder/decoder model for semantic segmentation	True	No official implementation	2020
DeepDeSRT	PASCAL VOC 2011	Augmented fully convolutional network	True	No official implementation	2017

Table 5.9: Table Recognition Models of Choice

Table Classification

We have managed to build a heuristic method to classify table types and achieve high accuracy on our collected dataset. As compared to using a model approach, a logic-based one requires less computational power and much less effort to develop. With the aforementioned benefits and high accuracy, a heuristic approach proves to be a correct choice in this context. However, we are fully aware of the fact that logic-based processes might not be able to generalize well and require a lot of hand-engineering which can prove to be a problem on a larger scale. Thus, given a larger system with more resources and time available, a deep-learning-based method might prove to be more beneficial.

Table Recognition

Some basic information about our table recognition models of choice are available in Table 5.9. Due to various reasons, most table recognition models do not have an official public code base which leads to some shortage of information regarding the models.

Deep Splitting and Merging(SPLERGE). SPLERGE composed of a Split model to predict the fine grid structure of the table ignoring span cell by focusing on detecting signals of the space between columns and rows (also called row and column separator). Then either a Merge model or some heuristics methods are used to predict where cells span multiple columns or rows. The key idead put forth by this paper is to pool information over large regions of the table image such as entire rows/columns of pixels or previously predicted cell regions to help learned features propagate across

large images. This approach performs exceptionally well and are much better than others when dealing with tables with partial or no borders(less presence of border means an increase in the presence of column and row separators).

TableNet. The author proposed a novel end-to-end multi-task architecture for both table detection and structure recognition. The novel idea of this approach is instead of solving table detection and column detection separately, the authors made use of convolutional filters utilized to detect tables reinforced by column detecting filters. After the first step and masks for table and column regions are generated, table row extraction are done by combining words position in document obtained via OCR, position of column and some heuristic process. Trained on Marmot [23] dataset that has been further enriched, TableNet is able to obtained state-of-the-art score on ICDAR 2013 - main benchmark for table structure recognition test.

DeepDeSRT DeepDeSRT is deep learning-based solution for table structure recognition which utilized transfer leaning by augmenting and fine-tuning a fully convolutional network for semantic segmentation model by Shelhamer et al [58]. By making suitable changes to the base model, DeepDeSRT is capable of extracting extra details in shallower layers and is provided learnable scaling capabilities compared to the original work. These augmentations help with obtaining cleaner rows and columns delineation. DeepDeSRT is amongst the first deep learning table structure recognition solution and has been very to be influential eversince. Out of the three solutions we consider for this task, DeepDeSRT is the oldest solution and has only been pretrained on PASCAL VOC 2011 dataset [59], the model has great potential to be further improved with the help of recent datasets.

Conclusion

It is clear that each approach has its own set of strengths and weaknesses. DeepDeSRT and TableNet are tried and true models but do not benefit from recent advancements, especially with regard to the amount of available data. Deep Splitting and Merging(SPLERGE) on the other hand is a more recent development and while it boasts promising results, the codebase is not publicly available so we would have to rely on unofficial implementations. Another promising option is DETR, which is trained on the biggest and most formal table dataset, PubTables-1M, and also the most up-to-date model that uses the latest versions of deep learning frameworks. The

number of parameters of the DETR model is also an advantage. Overall, the final choice of model would depend on the resources, the final desired performance, and the data distribution. With this in mind, we can have a modular approach that can be flexible in choosing which model for a specific situation. However, the current pipeline uses the TableNet for performing TSR task on bordered tables as it is carefully tailored for this type of table only and DETR for borderless tables. The detailed reasons for choosing DETR are already presented in section 4.2.2.

Implications from each component's output and some improvements

According to the result and brute force testing, the overall performance of TD and TSR on tables with simple structure is acceptable with minor mistakes. OCR performance still needs further improvement and optimization.

Table Detection Although the experimental results of TD models on the two testing datasets are not too high and still need improvement, they can detect almost all tables in images that have obvious tables. However, through brute force testing with a couple of handpicked images, the detected regions of tables in the input images significantly affect the TSR results afterward. The detected tables and their corresponding TSR results are shown in Figures 5.4 and 5.5 to illustrate this problem. To solve this problem, we simply post-process the detected tables by padding them in 4 directions.

Another minor problem mentioned in Phase 1 is adjacent tables detection. There is still a lack of a formal evaluation method to compare the performance between the two phases' pipelines. Some fail cases in Phase 1 are fixed in Phase 2 and vice versa. This poses a need for adjacent tables detection evaluation in future development. However, this problem is rare and not popular, the overall TD is very good. An example of this failure case is shown in figure 5.6. The green bounding box illustrates the prediction while the red ones are ground truth tables. Note that there is no improvement in the TD task between the two phases.

Table Structure Recognition In the case of bordered tables, the TSR model tailored for recognizing the structure of these tables tends to provide correct predictions. When it comes to borderless tables, compared to the results from Phase 1, the problem of dummy columns detected has been mitigated by retraining models

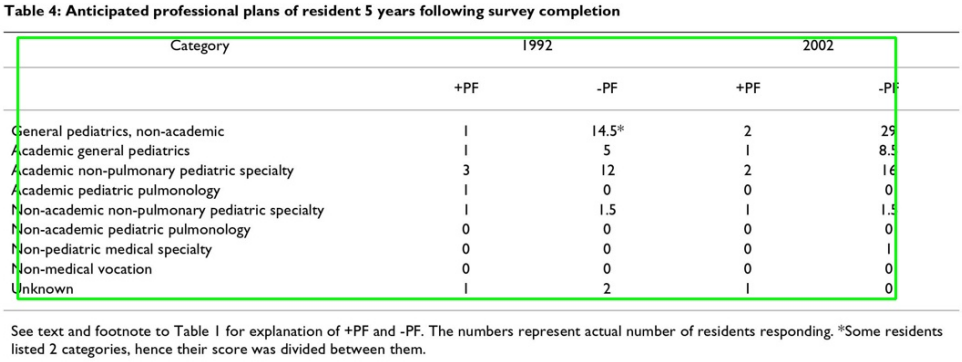
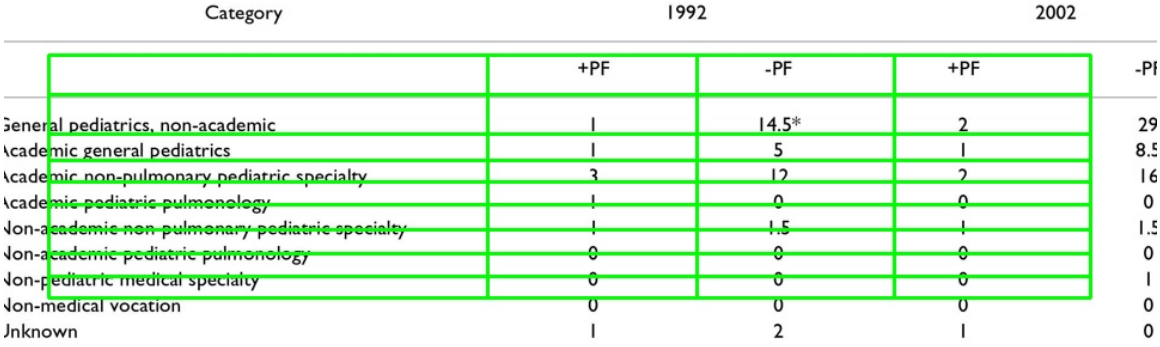


Table 4: Anticipated professional plans of resident 5 years following survey completion

Category	1992		2002	
	+PF	-PF	+PF	-PF
General pediatrics, non-academic	1	14.5*	2	29
Academic general pediatrics	1	5	1	8.5
Academic non-pulmonary pediatric specialty	3	12	2	16
Academic pediatric pulmonology	1	0	0	0
Non-academic non-pulmonary pediatric specialty	1	1.5	1	1.5
Non-academic pediatric pulmonology	0	0	0	0
Non-pediatric medical specialty	0	0	0	1
Non-medical vocation	0	0	0	0
Unknown	1	2	1	0

See text and footnote to Table 1 for explanation of +PF and -PF. The numbers represent actual number of residents responding. *Some residents listed 2 categories, hence their score was divided between them.

(a) Example of close table detection



Category	1992		2002	
	+PF	-PF	+PF	-PF
General pediatrics, non-academic	1	14.5*	2	29
Academic general pediatrics	1	5	1	8.5
Academic non-pulmonary pediatric specialty	3	12	2	16
Academic pediatric pulmonology	1	0	0	0
Non-academic non-pulmonary pediatric specialty	1	1.5	1	1.5
Non-academic pediatric pulmonology	0	0	0	0
Non-pediatric medical specialty	0	0	0	1
Non-medical vocation	0	0	0	0
Unknown	1	2	1	0

(b) TSR result of the closely detected table

Figure 5.4: TD and TSR results in case of close table detection

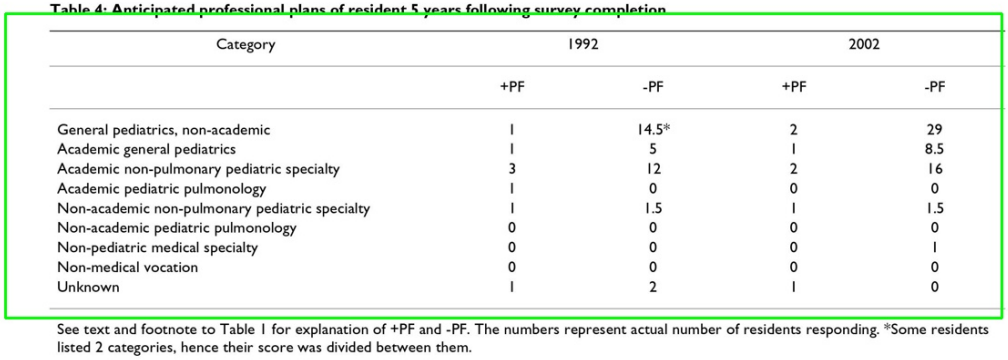


Table 4: Anticipated professional plans of resident 5 years following survey completion

Category	1992		2002	
	+PF	-PF	+PF	-PF
General pediatrics, non-academic	1	14.5*	2	29
Academic general pediatrics	1	5	1	8.5
Academic non-pulmonary pediatric specialty	3	12	2	16
Academic pediatric pulmonology	1	0	0	0
Non-academic non-pulmonary pediatric specialty	1	1.5	1	1.5
Non-academic pediatric pulmonology	0	0	0	0
Non-pediatric medical specialty	0	0	0	1
Non-medical vocation	0	0	0	0
Unknown	1	2	1	0

See text and footnote to Table 1 for explanation of +PF and -PF. The numbers represent actual number of residents responding. *Some residents listed 2 categories, hence their score was divided between them.

(a) Example of loose table detection

Category	1992		2002	
	+PF	-PF	+PF	-PF
General pediatrics, non-academic	1	14.5*	2	29
Academic general pediatrics	1	5	1	8.5
Academic non-pulmonary pediatric specialty	3	12	2	16
Academic pediatric pulmonology	1	0	0	0
Non-academic non-pulmonary pediatric specialty	1	1.5	1	1.5
Non-academic pediatric pulmonology	0	0	0	0
Non-pediatric medical specialty	0	0	0	1
Non-medical vocation	0	0	0	0
Unknown	1	2	1	0

(b) TSR result of the loosely detected table

Figure 5.5: TD and TSR results in case of loose table detection

and post-processing outputs. Figure 5.7 shows an example of this problem and the improvement on this problem of the latest version.

Another popular failure case of TSR models in Phase 1 is missing row separators. When the line spaces between rows are too close, TSR models merge these rows into one big row. Figure 5.8 shows this type of error and its improvement in Phase 2. As illustrated in Figure 5.8, there has been a significant improvement in this problem in the latest pipeline thanks to finetuning. Note that the TSR models used for conducting inference in this section is the pruned DETR with our parameter pruning method.

Optical Character Recognition (OCR) The OCR engine still needs improvement in general. Since our focus is not on OCR, the OCR performance is only improved by pre- and post-processing the input images and output texts. The OCR performance mostly depends on the update from the original authors of EasyOCR.

5.2 Parameter Pruning Method

5.2.1 Datasets

Data for training and evaluation of our parameter pruning method are extracted from PubTables-1M by Microsoft **??**. It contains almost one million tables from scientific articles, supports multiple input modalities, and contains comprehensive header and location information for table structures. This dataset is arguably the most extensive dataset in this area both in terms of quality and quantity. Hence, we hope to accurately evaluate our experiments with this dataset. For each task, we sampled a number of samples from the original PubTables-1M and divide them into a training data for finetuning the models and a testing dataset for evaluation. Specifically:

- For *Table Detection* task, the training set contains 100,000 samples for finetuning the pruned models and 30,000 samples for evaluation
- For *Table Structure Recognition* task, the training set contains 94,000 samples for finetuning the pruned models and 20,000 samples for evaluation.

We also benchmarked the improvement in inference time when plugging the pruned models into our pipeline. The datasets used in this experiment were also extracted from PubTables-1M, and divided into three sets:

- *Borderless*: This set contains 100 images, sampling only borderless tables.
- *Bordered*: This set contains 54 images, sampling only bordered tables.
- *Mixed*: This set contains 100 images, sampling both borderless and bordered tables.

5.2.2 Experimental Results

We apply the pruning algorithm to create pruned models with different pruning configurations resulting in different numbers of parameters. The pruned models are finetuned for both TD and TSR tasks.

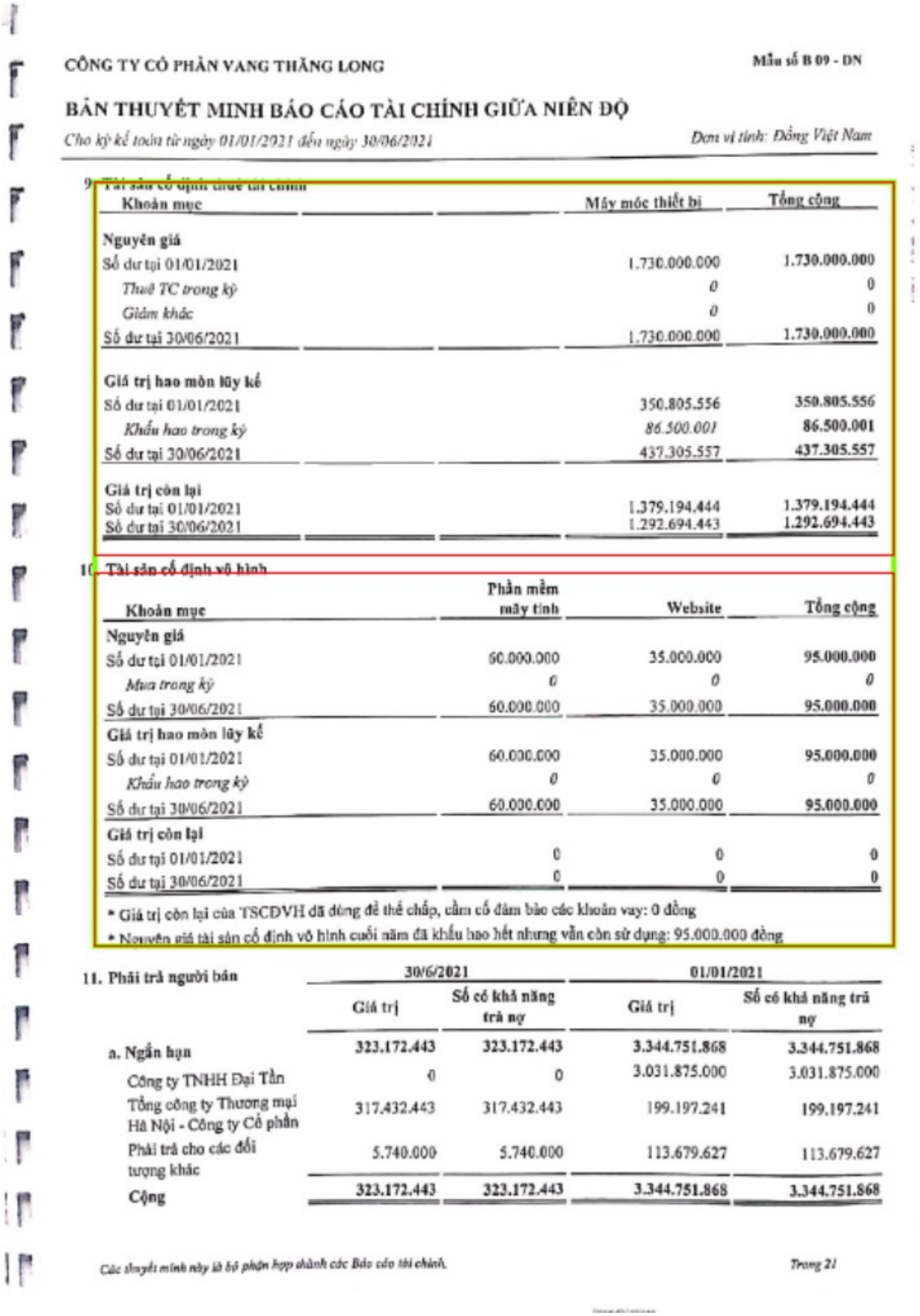
Table Structure Recognition

Table 5.10 shows the pruning configurations of (pruned) models that are used to analyze and evaluate the efficiency of our pruning method. Experimental results of these models on the testing dataset are shown in Table 5.11. The metrics of choice are the general detection metrics (including AP50, AP75, AP, AR) and the GriTS as introduced in section 2.3. Note that the GriTS_Con and Average_Con metrics are not taken into account in this evaluation since we only want to analyze the efficiency and correctness of the TSR task, not the OCR task.

The first three lines compare three models including the original model Table Transformer, the pruned Table Transformer at all dimensions with pruning ratio of 0.25, and the uniformly pruned Table Transformer at all dimensions with pruning ratio of 0.25. Firstly, it can be seen that the pruned version achieves lower evaluation results at all metrics compared to the original model. However, this drop ratio of approximately 2%-4.3% in performance is much less than the compression ratio of nearly 12%. Secondly, the model pruned using our method outperforms the one using the uniform pruning method in general detection metrics. However, the uniformly pruned model achieves higher GriTS_Top when it comes to GriTS metrics. This places

a question about the efficiency of pruning all dimensions. The three dimensions of k, v, and h directly impact the attention mechanism of Transformer and their inconsistency in values between encoder/decoder layers can cause drop and instability in the prediction. We then try to only prune the f dimension to see whether we can mitigate the drop in performance and also prove the efficiency of our pruning method compared to the conventional one that uniformly prunes the dimension(s).

The remaining lines show alternately the models pruned by our method and the ones that are uniformly pruned. All of these models are only pruned on f dimension. In general, all of the models pruned with our method achieve better results than the ones that are uniformly pruned. Regarding both general detection and GriTS metrics, it can be seen that although there is not much difference between the two kinds of models at small pruning ratios such as 0.25 and 0.5, there is a significant difference between them at high pruning ratios, e.g. 0.8 and 0.9. This means our method makes a difference at high pruning ratios. Moreover, the relative performance between the extremely pruned models, and the original model is not much compared to its relatively small size. For example, the pruned model on f dimension at the pruning ratio of 0.9 is 66% smaller than the original model but can achieve nearly 97.7% performance of the original model in AP score. For the GriTS_Top score, for example, the pruned model on f dimension at the pruning ratio of 0.9 can achieve nearly 99% performance of the original model.



CÔNG TY CỔ PHẦN VANG THĂNG LONG

Mẫu số B 09 - DN

BẢN THUYẾT MINH BÁO CÁO TÀI CHÍNH GIỮA NIÊN ĐỘ

Cho kỳ kế toán từ ngày 01/01/2021 đến ngày 30/06/2021

Đơn vị tính: Đồng Việt Nam

9. Tài sản cố định thuê tài chính

Khoản mục	Máy móc thiết bị	Tổng cộng
Nguyên giá		
Số dư tại 01/01/2021	1.730.000.000	1.730.000.000
Thuê TC trong kỳ	*0*	0
Giảm khác	*0*	0
Số dư tại 30/06/2021	1.730.000.000	1.730.000.000
Giá trị hao mòn lũy kế		
Số dư tại 01/01/2021	350.805.556	350.805.556
Khấu hao trong kỳ	*86.500.001*	86.500.001
Số dư tại 30/06/2021	437.305.557	437.305.557
Giá trị còn lại		
Số dư tại 01/01/2021	1.379.194.444	1.379.194.444
Số dư tại 30/06/2021	1.292.694.443	1.292.694.443

10. Tài sản cố định vô hình

Khoản mục	Phần mềm máy tính	Website	Tổng cộng
Nguyên giá			
Số dư tại 01/01/2021	60.000.000	35.000.000	95.000.000
Mua trong kỳ	*0*	*0*	*0*
Số dư tại 30/06/2021	60.000.000	35.000.000	95.000.000
Giá trị hao mòn lũy kế			
Số dư tại 01/01/2021	60.000.000	35.000.000	95.000.000
Khấu hao trong kỳ	*0*	*0*	*0*
Số dư tại 30/06/2021	60.000.000	35.000.000	95.000.000
Giá trị còn lại			
Số dư tại 01/01/2021	0	0	0
Số dư tại 30/06/2021	0	0	0

* Giá trị còn lại của TSCĐVH đã dùng để thế chấp, cầm cố đảm bảo các khoản vay: 0 đồng

* Nguyên giá tài sản cố định vô hình cuối năm đã khấu hao hết nhưng vẫn còn sử dụng: 95.000.000 đồng

11. Phải trả người bán

	30/6/2021		01/01/2021	
	Giá trị	Số có khả năng trả nợ	Giá trị	Số có khả năng trả nợ
a. Ngắn hạn	**323.172.443**	**323.172.443**	**3.344.751.868**	**3.344.751.868**
Công ty TNHH Đại Tần	0	0	3.031.875.000	3.031.875.000
Tổng công ty Thương mại Hà Nội - Công ty Cổ phần	317.432.443	317.432.443	199.197.241	199.197.241
Phải trả cho các đối tượng khác	5.740.000	5.740.000	113.679.627	113.679.627
Cộng	**323.172.443**	**323.172.443**	**3.344.751.868**	**3.344.751.868**

Các thuyết minh này là bộ phận hợp thành các Báo cáo tài chính.

Trang 21

Figure 5.6: A failure case of TD models when there are adjacent tables

Table 4: The number of genes and the average expression level of genes (only genes having more than 100 residues) which have percent composition of positively correlated residues (e.g. Ala, Gly, Arg & Val) in different bin/range.

Percent Composition		Ala		Gly		Arg		Val	
	Genes *	E. Level **	Genes	E. Level	Genes	E. Level	Genes	E. Leve
1 – 3	152	2.15	333	2.82	543	3.70	543	3.70
3 – 5	1063	2.05	1166	2.39	1728	2.86	1728	2.86
5 – 7	1204	2.75	1176	3.61	798	2.85	798	2.85
7 – 9	613	4.86	510	6.14	194	7.52	194	7.52
9 – 11	242	9.19	155	7.39	55	19.58	55	19.58
11 – 13	61	15.19	33	12.64	23	17.30	23	17.30
13 – 15	30	15.81	15	12.73	5	29.86	5	29.86
> 15	32	16.52	7	13.54	2	12.85	2	12.85

* Total number of genes in this range

(a) The problem of dummy columns detected in Phase 1

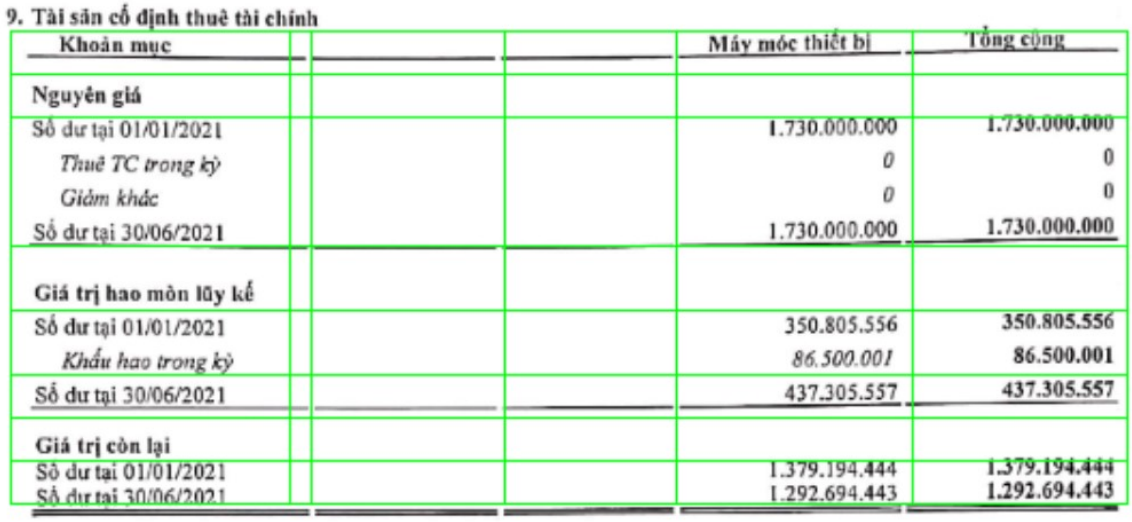
Table 4: The number of genes and the average expression level of genes (only genes having more than 100 residues) which have percent composition of positively correlated residues (e.g. Ala, Gly, Arg & Val) in different bin/range.

Percent Composition	Ala		Gly		Arg		Val	
	Genes *	E. Level **	Genes	E. Level	Genes	E. Level	Genes	E. Level
1 – 3	152	2.15	333	2.82	543	3.70	543	3.70
3 – 5	1063	2.05	1166	2.39	1728	2.86	1728	2.86
5 – 7	1204	2.75	1176	3.61	798	2.85	798	2.85
7 – 9	613	4.86	510	6.14	194	7.52	194	7.52
9 – 11	242	9.19	155	7.39	55	19.58	55	19.58
11 – 13	61	15.19	33	12.64	23	17.30	23	17.30
13 – 15	30	15.81	15	12.73	5	29.86	5	29.86
> 15	32	16.52	7	13.54	2	12.85	2	12.85

* Total number of genes in this range
** Average expression level of genes in this range

(b) Recognition results of the latest version in Phase 2

Figure 5.7: Different TSR results between Phase 1 and Phase 2



9. Tài sản cố định thuê tài chính

Khoản mục				Máy móc thiết bị	Tổng cộng
Nguyên giá					
Số dư tại 01/01/2021				1.730.000.000	1.730.000.000
Thuê TC trong kỳ				*0*	0
Giảm khác				*0*	0
Số dư tại 30/06/2021				1.730.000.000	1.730.000.000
Giá trị hao mòn lũy kế					
Số dư tại 01/01/2021				350.805.556	350.805.556
Khấu hao trong kỳ				*86.500.001*	86.500.001
Số dư tại 30/06/2021				437.305.557	437.305.557
Giá trị còn lại					
Số dư tại 01/01/2021				1.379.194.444	1.379.194.444
Số dư tại 30/06/2021				1.292.694.443	1.292.694.443

(a) A failure case of TSR models merging multiple rows into one row in Phase 1

9. Tài sản cố định thuê tài chính

Khoản mục	Máy móc thiết bị	Tổng cộng
Nguyên giá		
Số dư tại 01/01/2021	1.730.000.000	1.730.000.000
Thuê TC trong kỳ	*0*	0
Giảm khác	*0*	0
Số dư tại 30/06/2021	1.730.000.000	1.730.000.000
Giá trị hao mòn lũy kế		
Số dư tại 01/01/2021	350.805.556	350.805.556
Khấu hao trong kỳ	*86.500.001*	86.500.001
Số dư tại 30/06/2021	437.305.557	437.305.557
Giá trị còn lại		
Số dư tại 01/01/2021	1.379.194.444	1.379.194.444
Số dư tại 30/06/2021	1.292.694.443	1.292.694.443

(b) Recognition results of the latest version in Phase 2

Figure 5.8: Different TSR results between Phase 1 and Phase 2 on the problem of missing row separators

Model	Annotated Name	Pruning Ratio	Configuration	Number of Parameters
Original Table Transformer	original_tableTrans	0	f = 2048 h = 256 kdim = vdim = 256	~17.4M
Pruned Table Transformer on all dimensions (f, kdim, vdim, hidden dimension)	tableTrans_pruned_all_025	0.25	encoder f's: [1159, 1204, 1235, 1294, 1349, 1593] decoder f's: [1743, 1675, 1692, 1810, 1839, 1816] encoder h: 256 decoder h: 256 encoder k: [210, 256, 255, 255, 255, 256] decoder k: [256, 256, 256, 256, 256, 256] encoder v: [31, 95, 212, 210, 52, 234] decoder v: [242, 235, 229, 234, 235, 252]	~15.3M
Uniformly pruned Table Transformer on all dimensions (f, kdim, vdim, hidden dimension)	tableTrans_uniform_all_025	0.25	encoder f: 1306 decoder f: 1762 encoder h: 256 decoder h: 256 encoder k: 248 decoder k: 256 encoder v: 139 decoder v: 238	~15.3M
Pruned Table Transformer on f dimension only	tableTrans_pruned_f_025	0.25	encoder f's: [1149, 1233, 1237, 1305, 1358, 1596] decoder f's: [1713, 1680, 1691, 1813, 1836, 1821]	~14.2M
Uniformly Pruned Table Transformer on f dimension only	tableTrans_uniform_f_025	0.25	encoder f: 1313 decoder f: 1759	~14.2M
Pruned Table Transformer on f dimension only	tableTrans_pruned_f_05	0.5	encoder f's: [338, 563, 613, 783, 853, 1179] decoder f's: [958, 1145, 1333, 1542, 1630, 1351]	~11.1M
Uniformly Pruned Table Transformer on f dimension only	tableTrans_uniform_f_05	0.5	encoder f: 722 decoder f: 1327	~11.1M
Pruned Table Transformer on f dimension only	tableTrans_pruned_f_08	0.8	encoder f's: [78, 153, 195, 319, 331, 552] decoder f's: [299, 413, 584, 687, 786, 518]	~7.3M
Uniformly Pruned Table Transformer on f dimension only	tableTrans_uniform_f_08	0.8	encoder f: 272 decoder f: 548	~7.3M
Pruned Table Transformer on f dimension only	tableTrans_pruned_f_09	0.9	encoder f's: [36, 75, 116, 194, 187, 326] decoder f's: [137, 206, 283, 296, 354, 248]	~6M
Uniformly Pruned Table Transformer on f dimension only	tableTrans_uniform_f_09	0.9	encoder f: 156 decoder f: 254	~6M

Table 5.10: TSR models configurations

Annotated Name	Number of Parameters	General Detection Metrics				GriTS (on simple tables)			GriTS (on complex tables)				GriTS (on all tables)			
		AP50	AP75	AP	AR	GriTS_Top	GriTS_Con	GriTS_Loc	Accuracy_Con	GriTS_Top	GriTS_Con	GriTS_Loc	Accuracy_Con	GriTS_Top	GriTS_Con	GriTS_Loc
original_tableTrans	~17.4M	0.971	0.948	0.909	0.94	0.9911	0.9928	0.9402	0.8795	0.9726	0.99	0.9108	0.9101	0.9809	0.9913	0.9239
tableTrans_pruned_all_025	~15.3M	0.953	0.914	0.87	0.911	0.9822	0.9839	0.9154	0.6808	0.949	0.9767	0.8698	0.7503	0.9639	0.9799	0.8902
tableTrans_uniform_all_025	~15.3M	0.951	0.906	0.86	0.903	0.9838	0.9853	0.9105	0.7151	0.9527	0.9809	0.8676	0.7756	0.9667	0.9829	0.8868
tableTrans_pruned_f_025	~14.2M	0.963	0.931	0.891	0.927	0.9871	0.9886	0.9294	0.8187	0.9596	0.9818	0.8905	0.8608	0.9719	0.9848	0.9079
tableTrans_uniform_f_025	~14.2M	0.964	0.933	0.893	0.93	0.9865	0.9879	0.93	0.8076	0.9576	0.98	0.8908	0.8525	0.9705	0.9835	0.9084
tableTrans_pruned_f_05	~11.1M	0.964	0.933	0.889	0.9201	0.9873	0.9891	0.9305	0.8435	0.9640	0.9851	0.8950	0.8778	0.9744	0.9869	0.9109
tableTrans_uniform_f_05	~11.1M	0.964	0.931	0.892	0.928	0.9873	0.9887	0.931	0.8186	0.9606	0.9827	0.894	0.8609	0.9726	0.9854	0.9105
tableTrans_pruned_f_08	~7.3M	0.964	0.933	0.895	0.931	0.9883	0.9897	0.9347	0.8218	0.9611	0.9828	0.8962	0.8663	0.9733	0.9859	0.9134
tableTrans_uniform_f_08	~7.3M	0.963	0.929	0.886	0.923	0.9875	0.989	0.927	0.8389	0.96	0.9818	0.8886	0.877	0.9723	0.985	0.9058
tableTrans_pruned_f_09	~6M	0.96	0.923	0.877	0.917	0.9877	0.9893	0.9199	0.8364	0.9603	0.9831	0.8812	0.8755	0.9726	0.9859	0.8985
tableTrans_uniform_f_09	~6M	0.949	0.897	0.852	0.901	0.9835	0.9853	0.9089	0.7941	0.9527	0.9776	0.867	0.8404	0.9665	0.9811	0.8858

Table 5.11: Experimental results of TSR models

Table Detection

Table 5.12 shows the pruning configurations of (pruned) models that are used to analyze and evaluate the efficiency of our pruning method. Most of the pruning ratios remain the same as the models presented in Table 5.10 except the models pruned at all dimensions having the pruning ratio of 0.5. Experimental results of these models on the testing dataset are shown in Table 5.13. The metrics of choice are the general detection metrics (including AP50, AP75, AP, AR).

The first three lines compare three models including the original model Table Transformer, the pruned Table Transformer at all dimensions with a pruning ratio of 0.5, and the uniformly pruned Table Transformer at all dimensions with a pruning ratio of 0.5. The same pattern can be seen that the drop in performance is not as much as the compression ratio (6% vs 37%). However, the model pruned with our method achieves lower scores compared to the uniformly pruned model. Fortunately, when it comes to the models pruned in f dimension only, there is a clear difference between the models pruned using our method and the uniformly pruned models. It strengthens the importance of k, h, and v in transformer architecture and the efficiency of our pruning method.

Model	Annotated Name	Pruning Ratio	Configuration	Number of Parameters
Original Table Transformer	original_tableTrans		f = kdim = vdim = 2048 h = 256	~17.4M
Pruned Table Transformer on all dimensions (f, kdim, vdim, hidden dimension)	tableTrans_pruned_all_05	0.5	encoder fs: [1213, 808, 546, 489, 405, 547] decoder fs: [1252, 1371, 1546, 1406, 1390, 1315] encoder h: 256 decoder h: 256 encoder k: [126, 104, 95, 73, 89, 104] decoder k: [132, 137, 120, 148, 189, 21] encoder v: [185, 124, 69, 136, 160, 185] decoder v: [171, 116, 109, 114, 79, 88]	~10.9M
Uniformly pruned Table Transformer on all dimensions (f, kdim, vdim, hidden dimension)	tableTrans_uniform_all_05	0.5	encoder f: 668 decoder f: 1380 encoder h: 256 decoder h: 256 encoder k: 99 decoder k: 125 encoder v: 143 decoder v: 113	~10.9M
Pruned Table Transformer on f dimension only	tableTrans_pruned_f_025	0.25	encoder f's: [1761, 1683, 1529, 1350, 1372, 1521] decoder f's: [1479, 1500, 1616, 1564, 1537, 1520]	~14.2M
Uniformly Pruned Table Transformer on f dimension only	tableTrans_uniform_f_025	0.25	encoder f: 1536 decoder f: 1536	~14.2M
Pruned Table Transformer on f dimension only	tableTrans_pruned_f_05	0.5	encoder f's: [1475, 1112, 917, 872, 811, 957] decoder f's: [1090, 1212, 1329, 1004, 746, 763]	~11.1M
Uniformly Pruned Table Transformer on f dimension only	tableTrans_uniform_f_05	0.5	encoder f: 1024 decoder f: 1024	~11.1M
Pruned Table Transformer on f dimension only	tableTrans_pruned_f_08	0.8	encoder f's: [36, 75, 116, 194, 187, 326] decoder f's: [137, 206, 283, 296, 354, 248]	~7.3M
Uniformly Pruned Table Transformer on f dimension only	tableTrans_uniform_f_08	0.8	encoder f: 410 decoder f: 410	~7.3M
Pruned Table Transformer on f dimension only	tableTrans_pruned_f_09	0.9	encoder f's: [427, 242, 157, 131, 57, 215] decoder f's: [410, 222, 342, 148, 53, 54]	~6M
Uniformly Pruned Table Transformer on f dimension only	tableTrans_uniform_f_09	0.9	encoder f: 205 decoder f: 205	~6M

Table 5.12: TD models configurations

Annotated Name	Number of Parameters	General Detection Metrics			
		AP50	AP75	AP	AR
original_tableTrans	~17.4M	0.995	0.995	0.959	0.978
tableTrans_pruned_all_025	~10.9M	0.995	0.955	0.899	0.942
tableTrans_uniform_all_025	~10.9M	0.995	0.966	0.898	0.947
tableTrans_pruned_f_025	~14.2M	0.995	0.995	0.952	0.98
tableTrans_uniform_f_025	~14.2M	0.995	0.986	0.948	0.976
tableTrans_pruned_f_05	~11.1M	0.995	0.995	0.962	0.983
tableTrans_uniform_f_05	~11.1M	0.995	0.982	0.944	0.974
tableTrans_pruned_f_08	~7.3M	0.995	0.99	0.962	0.982
tableTrans_uniform_f_08	~7.3M	0.995	0.99	0.954	0.981
tableTrans_pruned_f_09	~6M	0.995	0.99	0.956	0.98
tableTrans_uniform_f_09	~6M	0.995	0.982	0.955	0.979

Table 5.13: Experimental results of TD models

Pipeline Inference Time

In addition to evaluating the pruned models using the predefined metrics, we benchmarked the improvements in inference time of these pruned models when integrated into the Table Optical Character Recognition (OCR) pipeline. The models we selected for the Table Detection (TD) and Table Structure Recognition (TSR) tasks were both DETR models pruned using our method on the f dimension only at a ratio of 0.5. The timing was carefully measured as only the duration for which an input image passed through the model to produce an output. Table 5.14 presents the experimental results of inference time and the speedup percentage of the TD and TSR components on different datasets compared to using the original DETR models. Note that for the results on the *Bordered* dataset, there were no TSR inference times and its related results since the TSR model used for bordered tables was TableNet, which was not a pruned DETR model.

In general, the two pruned tables significantly accelerated the inference times of both tasks, with the highest speedup percentage of 24.87%. The speedup was more remarkable for the TSR task. For the TD task, the preferred input image size was less than 1000 pixels for faster inference, while it was more variable for TSR due to the different number of cells in each image, which played a more important role.

Table type	Input Image Size (in pixels)	Model	TD Inference Time (seconds)	TSR Inference Time (seconds)	TD + TSR Inference Time (seconds)	TD Speed-up Percentage	TSR Speed-up Percentage	TD+TSR Speed-up Percentage
Borderless	max(width, height) = 1000	Original Table Transformer	0.0209	0.0180	0.0388	12.04%	16.89%	14.28%
		Pruned Table Transformer at 0.5 ratio	**0.0184**	**0.0149**	**0.0333**			
	1000 max(width, height) = 3000	Original Table Transformer	0.0395	0.0221	0.0616	5.93%	24.87%	12.74%
		Pruned Table Transformer at 0.5 ratio	**0.0371**	**0.0166**	**0.0537**			
	3000 max(width, height) = 5000	Original Table Transformer	0.0233	0.0191	0.0425	11.20%	20.47%	15.38%
		Pruned Table Transformer at 0.5 ratio	**0.0207**	**0.0152**	**0.0359**			
Bordered	max(width, height) = 1000	Original Table Transformer	0.0167	NaN	NaN	13.38%	NaN	NaN
		Pruned Table Transformer at 0.5 ratio	**0.0144**	NaN	NaN			
	1000 max(width, height) = 3000	Original Table Transformer	0.0335	NaN	NaN	8.19%	NaN	NaN
		Pruned Table Transformer at 0.5 ratio	**0.0307**	NaN	NaN			
	3000 max(width, height) = 5000	Original Table Transformer	0.0289	NaN	NaN	3.36%	NaN	NaN
		Pruned Table Transformer at 0.5 ratio	**0.0279**	NaN	NaN			
Mixed	max(width, height) = 1000	Original Table Transformer	0.0166	0.0175	0.0340	14.64%	17.80%	16.26%
		Pruned Table Transformer at 0.5 ratio	**0.0141**	**0.0144**	**0.0285**			
	1000 max(width, height) = 3000	Original Table Transformer	0.0339	0.0200	0.0539	-0.47%	1.86%	0.40%
		Pruned Table Transformer at 0.5 ratio	**0.0340**	**0.0196**	**0.0537**			
	3000 max(width, height) = 5000	Original Table Transformer	0.0240	0.0181	0.0421	8.31%	17.90%	12.44%
		Pruned Table Transformer at 0.5 ratio	**0.0220**	**0.0149**	**0.0369**			

Table 5.14: Inference time and Speedup percentage of Table Detection (TD) and Table Structure Recognition (TSR) components on different datasets

5.2.3 Discussion

After comparing the two pruning methods in TD and TSR tasks, we can draw some important conclusions. Firstly, the k, v, and h place a crucial role in the performance of Transformer-based models. Without pruning these parameters, we can still attain a smaller model by pruning the f dimension of the feedforward layers only. Secondly, our pruning method achieves higher detection scores compared to the conventional uniform pruning method. Finally, the models pruned using our method speed up the inference process remarkably, which contributes to the overall optimization of the whole pipeline. From these conclusions, here comes the contributions of this study, which are summarized as follows:

- The attention mechanism is the key component of Transformer-based architecture. More studies for a pruning method that can efficiently optimize the attention hyperparameters (k, v, and h) are opened for research.
- Our pruning method achieves higher scores when applied to the correct dimensions. The original idea of this pruning method is yet simple and can be applied to different architectural bases, which can be further analyzed and verified its effectiveness in other domains and architectures.
- Our pruning method produces models that can significantly speed up the inference process in different tasks.

Chapter 6

Summary and future development

This chapter provides a summary of the achievements, challenges, and lessons learned throughout the course of the project. The chapter also outlines plans for future development and enhancement of the proposed solution, building on the insights gained from the research and experimentation conducted in the preceding chapters.

6.1 Summary and contribution

6.1.1 Summary

During the research and development of the proposed solution, we have been learning and inheriting knowledge and works from previous research. This provides us with an overview and general picture of the OCR table task and its rich application potential in many different fields and documents. However, the biggest challenge of this project is that although there are many datasets for TD and TSR tasks, there is no widely-adopted dataset for training and evaluation. The variation of datasets and the difference in their distributions are especially prominent in TSR tasks, where there is no off-the-shelf model to recognize the structure of all kinds of tables. Consequently, state-of-the-art models trained on one dataset could not generalize well enough to achieve good results on another. This poses the challenge of constructing a general training and evaluation dataset and also the evaluation method for fairly assessing different solutions. As there is currently no consensus on an approach that can be

considered best all around, we have also conducted extensive research to classify and identify the best solution for each type of data. There is also the problem of efficiency when using a branching pipeline approach. More components can help with overall accuracy but come at the cost of resources and time consumption. This led us to utilize the parameters pruning method inspired by Khetan et al. optimization of BERT models. We have successfully reimplemented(as there is no official code base) and expanded the approach to be generalized for a general transformer-based model. Our pipeline efficiency in terms of resources needed to achieve high accuracy was greatly improved. This further proves that it is possible to build a table OCR pipeline that is modular and capable of processing a majority number of table types and domains. It is also possible for the pipeline to remain lightweight and less resources-intensive through the use of parameter pruning.

6.1.2 Contribution

After conducting a comprehensive evaluation and comparison of the latest models for both TD and TSR tasks, we have acquired insightful conclusions regarding their respective strengths and weaknesses. These conclusions can be highly valuable for researchers seeking to choose the most suitable TD or TSR model, which can then be further optimized for target documents. Additionally, our proposed pipeline effectively combines the strengths of multiple solutions, which leads to an overall improvement in the accuracy of the output. With these contributions, we have laid a solid foundation for future research and development in the field of table OCR, where we hope to continue to make significant strides in advancing the technology.

As for the application of parameter pruning, the benefits of this experiment are twofold. Firstly, we would like to provide a reimplementation of Khetan et al. work [19], as the original authors did not publish a detailed implementation alongside their publication. Secondly, we want to prove whether the approach can be generalized from only BERT models to be applicable for the class of transformer-based models and for different domains, specifically table detection and table structure recognition. This can serve as a further exploration into the area of lightening transformer-based models, which can greatly improve their efficiency and range of applications.

6.2 Future development

The OCR table task is a complex challenge with very few public and commercial solutions available. To tackle this task, we break it down into subtasks such as table detection, table structure recognition, OCR, optimization, etc. We conduct thorough research on state-of-the-art models tailored for each of these subtasks and have managed to construct and optimize a complete pipeline for our proposed system that meets our goals. Moving forward, our plan is to further improve our current models to achieve even higher results on our testing datasets. We are also developing an integrated website for performing table OCR on input images. Additionally, we will conduct further research on applying the pruning method introduced in this study to other deep learning models to gain more knowledge on the efficiency of this pruning technique in the field of deep learning overall.

References

[1] N. Carion, F. Massa, G. Synnaeve, N. Usunier, A. Kirillov, and S. Zagoruyko, "End-to-end object detection with transformers," 5 2020. [Online]. Available: http://arxiv.org/abs/2005.12872

[2] C. Williamson, M. E. M. E. Zurko, and A. D. Library., *Proceedings of the 16th International Conference on World Wide Web 2007 : Banff, Alberta, Canada, May 08-12, 2007.* ACM Press, 2007.

[3] Z. Chi, H. Huang, H.-D. Xu, H. Yu, W. Yin, and X.-L. Mao, "Complicated table structure recognition," 8 2019. [Online]. Available: https://arxiv.org/abs/1908.04729v2

[4] S. Schreiber, S. Agne, I. Wolf, A. Dengel, and S. Ahmed, "Deepdesrt: Deep learning for detection and structure recognition of tables in document images." [Online]. Available: http://www.icst.pku.edu.cn/cpdp/data/marmot

[5] M. Li, L. Cui, S. Huang, F. Wei, M. Zhou, and Z. Li, "Tablebank: Table benchmark for image-based table detection and recognition," pp. 1918–1925, 2020. [Online]. Available: https://aclanthology.org/2020.lrec-1.236

[6] D. Prasad, A. Gadpal, K. Kapadni, M. Visave, and K. Sultanpure, "Cascadetabnet: An approach for end to end table detection and structure recognition from image-based documents," 4 2020. [Online]. Available: https://arxiv.org/abs/2004.12629

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

[8] A. Khetan and Z. Karnin, "schubert: Optimizing elements of bert," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 2020, pp. 2807–2818.

[9] H. Li, A. Kadav, I. Durdanovic, H. Samet, and H. P. Graf, "Pruning filters for efficient convnets," *arXiv preprint arXiv:1608.08710*, 2016.

[10] P. Molchanov, S. Tyree, T. Karras, T. Aila, and J. Kautz, "Pruning convolutional neural networks for resource efficient inference," *arXiv preprint arXiv:1611.06440*, 2016.

[11] S. Anwar, K. Hwang, and W. Sung, "Structured pruning of deep convolutional neural networks," *ACM Journal on Emerging Technologies in Computing Systems (JETC)*, vol. 13, no. 3, p. 32, 2017.

[12] H. Pham, M. Y. Guan, B. Zoph, Q. V. Le, and J. Dean, "Efficient neural architecture search via parameter sharing," *arXiv preprint arXiv:1802.03268*, 2018.

[13] H. Liu, K. Simonyan, and Y. Yang, "Darts: Differentiable architecture search," *arXiv preprint arXiv:1806.09055*, 2018.

[14] S. Singh, A. Khetan, and Z. Karnin, "Darc: Differentiable architecture compression," *arXiv preprint arXiv:1905.08170*, 2019.

[15] M. Rastegari, V. Ordonez, J. Redmon, and A. Farhadi, "Xnor-net: Imagenet classification using binary convolutional neural networks," in *European conference on computer vision*. Springer, 2016, pp. 525–542.

[16] I. Hubara, M. Courbariaux, D. Soudry, R. El-Yaniv, and Y. Bengio, "Quantized neural networks: Training neural networks with low precision weights and activations," *The Journal of Machine Learning Research*, vol. 18, no. 1, pp. 6869–6898, 2017.

[17] C. Zhu, S. Han, H. Mao, and W. J. Dally, "Trained ternary quantization," *arXiv preprint arXiv:1612.01064*, 2016.

[18] S. Han, H. Mao, and W. J. Dally, "Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding," *arXiv preprint arXiv:1510.00149*, 2015.

[19] A. Khetan and Z. Karnin, "schuBERT: Optimizing elements of BERT," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. Online: Association for Computational Linguistics, Jul. 2020, pp. 2807–2818. [Online]. Available: https://aclanthology.org/2020.acl-main.250

[20] K. Itonori, "Table structure recognition based on textblock arrangement and ruled line position," pp. 765–768, 12 2002.

[21] T. Hassan and R. Baumgartner, "Table recognition and understanding from pdf files," *Proceedings of the International Conference on Document Analysis and Recognition, ICDAR*, vol. 2, pp. 1143–1147, 2007. [Online]. Available: https://www.researchgate.net/publication/4288170_Table_Recognition_and_Understanding_from_PDF_Files

[22] E. Oro and M. Ruffolo, "Pdf-trex: An approach for recognizing and extracting tables from pdf documents," *Proceedings of the International Conference on Document Analysis and Recognition, ICDAR*, pp. 906–910, 2009.

[23] J. Fang, X. Tao, Z. Tang, R. Qiu, and Y. Liu, "Dataset, ground-truth and performance metrics for table detection evaluation," 2012, pp. 445–449.

[24] G. Harit and A. Bansal, "Table detection in document images using header and trailer patterns," *ACM International Conference Proceeding Series*, 2012. [Online]. Available: https://www.researchgate.net/publication/262252099_Table_detection_in_document_images_using_header_and_trailer_patterns

[25] T. Kieninger and A. Dengel, "The t-recs table recognition and analysis system," *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, vol. 1655, pp.

255–270, 1999. [Online]. Available: https://link.springer.com/chapter/10.1007/3-540-48172-9_21

[26] F. Cesarini, S. Marinai, L. Sarti, and G. Soda, "Trainable table location in document images," *Proceedings - International Conference on Pattern Recognition*, vol. 16, pp. 236–240, 2002.

[27] T. Kasar, P. Barlas, S. Adam, C. Chatelain, and T. Paquet, "Learning to detect tables in scanned document images using line information," *Proceedings of the International Conference on Document Analysis and Recognition, ICDAR*, pp. 1185–1189, 2013. [Online]. Available: https://dl.acm.org/doi/10.1109/ICDAR.2013.240

[28] E. Kara, M. Traquair, B. Kantarci, and S. Khan, "Deep learning for recognizing the anatomy of tables on datasheets," *Proceedings - IEEE Symposium on Computers and Communications*, vol. 2019-June, 6 2019.

[29] E. Kara, M. Traquair, M. Simsek, B. Kantarci, and S. Khan, "Holistic design for deep learning-based discovery of tabular structures in datasheet images," *Engineering Applications of Artificial Intelligence*, vol. 90, 4 2020. [Online]. Available: https://dl.acm.org/doi/10.1016/j.engappai.2020.103551

[30] S. Arif and F. Shafait, "Table detection in document images using foreground and background features," *2018 International Conference on Digital Image Computing: Techniques and Applications, DICTA 2018*, 1 2019. [Online]. Available: https://www.researchgate.net/publication/330475161_Table_Detection_in_Document_Images_using_Foreground_and_Background_Features

[31] A. Gilani, S. R. Qasim, I. Malik, and F. Shafait, "Table detection using deep learning," *Proceedings of the International Conference on Document Analysis and Recognition, ICDAR*, vol. 1, pp. 771–776, 7 2017.

[32] S. Ren, K. He, R. Girshick, and J. Sun, "Faster r-cnn: Towards real-time object detection with region proposal networks," 6 2015. [Online]. Available: http://arxiv.org/abs/1506.01497

[33] D. Karatzas, F. Shafait, S. Uchida, M. Iwamura, L. G. I. Bigorda, S. R. Mestre, J. Mas, D. F. Mota, J. A. Almazan, and L. P. D. L. Heras, "Icdar 2013 robust reading competition," *Proceedings of the International Conference on Document Analysis and Recognition, ICDAR*, pp. 1484–1493, 2013. [Online]. Available: https://www.researchgate.net/publication/261349846_ICDAR_2013_robust_reading_competition

[34] P. Fischer, A. Smajic, A. Mehler, and G. Abrami, "Multi-type-td-tsr – extracting tables from document images using a multi-stage pipeline for table detection and table structure recognition: from ocr to structured table representations," 5 2021. [Online]. Available: http://arxiv.org/abs/2105.11021

[35] M. Agarwal, A. Mondal, and C. V. Jawahar, "Cdec-net: Composite deformable cascade network for table detection in document images," *Proceedings - International Conference on Pattern Recognition*, pp. 9491–9498, 2020. [Online]. Available: https://www.researchgate.net/publication/351405982_CDeC-Net_Composite_Deformable_Cascade_Network_for_Table_Detection_in_Document_Images

[36] L. Gao, X. Yi, Z. Jiang, L. Hao, and Z. Tang, "Icdar2017 competition on page object detection," *Proceedings of the International Conference on Document Analysis and Recognition, ICDAR*, vol. 1, pp. 1417–1422, 1 2018.

[37] L. Gao, Y. Huang, H. Dejean, J. L. Meunier, Q. Yan, Y. Fang, F. Kleber, and E. Lang, "Icdar 2019 competition on table detection and recognition (ctdar)," *Proceedings of the International Conference on Document Analysis and Recognition, ICDAR*, pp. 1510–1515, 9 2019.

[38] X. Zhong, J. Tang, and A. J. Yepes, "Publaynet: largest dataset ever for document layout analysis," 8 2019. [Online]. Available: https://arxiv.org/abs/1908.07836

[39] S. Panchal, "sgrpanchal31/table-detection-dataset: This repository contains a 403 images dataset for table detection in documents." [Online]. Available: https://github.com/sgrpanchal31/table-detection-dataset

[40] A. Abdallah, A. Berendeyev, I. Nuradin, and D. Nurseitov, "Tncr: Table net detection and classification dataset," 6 2021. [Online]. Available: http://arxiv.org/abs/2106.15322http://dx.doi.org/10.1016/j.neucom.2021.11.101

[41] J. Li, Y. Xu, T. Lv, L. Cui, C. Zhang, and F. Wei, "Dit: Self-supervised pre-training for document image transformer." Association for Computing Machinery (ACM), 10 2022, pp. 3530–3539.

[42] A. W. Harley, A. Ufkes, and K. G. Derpanis, "Evaluation of deep convolutional nets for document image classification and retrieval," 2 2015. [Online]. Available: https://arxiv.org/abs/1502.07058

[43] B. Smock, R. Pesala, and R. Abraham, "Pubtables-1m: Towards comprehensive table extraction from unstructured documents," pp. 4634–4642, 2022. [Online]. Available: https://github.com/microsoft/table-transformer.

[44] S. R. Qasim, H. Mahmood, and F. Shafait, "Rethinking table recognition using graph neural networks," *Proceedings of the International Conference on Document Analysis and Recognition, ICDAR*, pp. 142–147, 5 2019. [Online]. Available: https://arxiv.org/abs/1905.13391v2

[45] C. Tensmeyer, V. I. Morariu, B. Price, S. Cohen, and T. Martinez, "Deep splitting and merging for table structure decomposition," *Proceedings of the International Conference on Document Analysis and Recognition, ICDAR*, pp. 114–121, 9 2019. [Online]. Available: https://drive.google.com/file/u/1/d/1rBGuN4v76HBtOsaNUMPw6wg-euyG1_pC/view?usp=sharing&usp=embed_facebook

[46] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel, "Backpropagation Applied to Handwritten Zip Code Recognition," *Neural Computation*, vol. 1, no. 4, pp. 541–551, 1989.

[47] B. Hassibi and D. G. Stork, "Second order derivatives for network pruning: Optimal brain surgeon," in *Advances in neural information processing systems*, 1993, pp. 164–171.

[48] K. Murray and D. Chiang, "Auto-sizing neural networks: With applications to n-gram language models," *arXiv preprint arXiv:1508.05051*, 2015.

[49] A. See, M.-T. Luong, and C. D. Manning, "Compression of neural machine translation models via pruning," *arXiv preprint arXiv:1606.09274*, 2016.

[50] Y. Kim and A. M. Rush, "Sequence-level knowledge distillation," *arXiv preprint arXiv:1606.07947*, 2016.

[51] B. Smock, R. Pesala, and R. Abraham, "Grits: Grid table similarity metric for table structure recognition," 3 2022. [Online]. Available: http://arxiv.org/abs/2203.12555

[52] N. Dalal and B. Triggs, "Histograms of oriented gradients for human detection." [Online]. Available: http://lear.inrialpes.fr

[53] P. F. Felzenszwalb, R. B. Girshick, D. Mcallester, and D. Ramanan, "Object detection with discriminatively trained part based models."

[54] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," 2001.

[55] A. Vaswani, G. Brain, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Łukasz Kaiser, and I. Polosukhin, "Attention is all you need," *Advances in Neural Information Processing Systems*, vol. 30, 2017.

[56] N. Carion, F. Massa, G. Synnaeve, N. Usunier, and A. Kirillov, "End-to-end object detection with transformers," in *European Conference on Computer Vision*, 2020.

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

[58] E. Shelhamer, J. Long, and T. Darrell, "Fully convolutional networks for semantic segmentation," 5 2016. [Online]. Available: http://arxiv.org/abs/1605.06211

[59] M. Everingham, Luc, V. Gool, Christopher, K. I. Williams, J. Winn, A. Zisserman, M. Everingham, L. V. Gool, K. U. Leuven, B. C. K. I. Williams, J. Winn, and A. Zisserman, "The pascal visual object classes (voc) challenge." [Online]. Available: http://www.flickr.com/

Appendix A

Work Assignment

Name	Student ID	Workload
Pham Bui Minh Huan	1952056	Data preparation, Table classification, Table recognition, Pipeline development
Nguyen Luat Gia Khoi	1952079	Table detection, Model pruning, Pipeline development, Data preparation

Table A.1: Workload division between members

Appendix B

Activity Log

Phase	Timeline	Task	Outcome
PROBLEM DEFINITION	05/09/2022 - 18/09/2022	- Choose and define topic: DETECTION, RECOGNITION, AND EXTRACTION OF TABLE STRUCTURE DATA - Prepare theoretical backgrounds	- Problem definition, rationale, scope, desired outcome and timeline
RESEARCH DEVELOPMENT	19/09/2022 - 02/10/2022	-Research on some approaches of dealing with table structure data -Research on available data -Research methods and propose a general pipeline that can perform on all type of tables	-Insights on current approaches -Insights on available data -Concept design and idea of pipeline
	03/10/2022 - 16/10/2022	-Define metrics -Building dataset for training and testing	-Metrics to accurately evaluate solutions -Organized dataset for training and testing
	17/10/2022 - 30/10/2022	- Research previous works: Table detection - Study the advantages and disadvantages and eproducec(if possible) the current approaches	- Detailed comparisions - Testing results reported by authors and on our own dataset
	31/10/2022 - 13/11/2022	- Research previous works: Table structure recognition - Study the advantages and disadvantages and reproducec(if possible) the current approaches	- Detailed comparisions - Testing results reported by authors and on our own dataset
	14/11/2022 - 27/11/2022	- Choose best models for each task - Implement modules and build pipeline	- Early versions of proposed table OCR pipeline
	28/11/2022 - 11/12/2022	- Further development and testing	- Early versions of proposed table OCR pipeline
	12/12/2022 - 25/12/2022	- Finish pipeline, benchmark - Further testing and compiling test results for report	- Table OCR pipeline and benchmark results
	26/12/2022 - 08/01/2023	- Write Specialized Project Report	- Specialized Project Report
	09/01/2023 - 22/01/2023	- Finish Specialized Project Report - Prepare presentation slides	- Specialized Project Report - Project presentation slides
	23/01/2023 - 05/02/2023	Tet Holiday	
	06/02/2023 - 19/02/2023	- Choose TD/TSR model to optimize. - Research and implement approaches to prune parameters in transformer-based models. - Choosing data for training/evaluating models.	- Model of choice - Dataset of choice
	20/02/2023 - 05/03/2023	- Implement method 2 for table-transformer pruning. - Prepare TSR training and testing set extracted from PubTables-1M for pruned TSR models. - Training models	- Transformer-based parameter pruning method - TSR training and testing set extracted from PubTables-1M for pruned TSR models.: + Training set: 100k images + Testing set: 30k images
EVALUATION LESSON LEARNED	06/03/2023 - 19/03/2023	- Training and evaluating 4 TSR models	- Evaluation results of the 4 models: + Pruning model - pruning f, kdim, vdim, hidden dimension (customarily) (60 epochs) + Uniformly pruning model - pruning f, kdim, vdim, hidden dimension (uniformly) (60 epochs) + Pruning model - pruning f only (customarily) - fraction = 0.25 (20 epochs) + Pruning model - pruning f only (customarily) - fraction = 0.5 (20 epochs)
	20/03/2023 - 02/04/2023	- Prepare data for training and evaluating pruned TD detection models - Seminar and fixing slides according as advised	- Data for training and evaluating pruned TD detection models. + Training set: 100k images + Testing set: 30k images
	03/04/2023 - 16/04/2023	- Include parts (preprocess, classify,...) into thesis as advised - Capstone project report writing	- Updated capstone project report
	17/04/2023 - 30/04/2023	- Further TSR model evaluation on GriTS metrics. - Capstone project report writing	- Evaluation results of all pruned TSR models on GriTS metrics. - Updated capstone project report
	01/05/2023 - 14/05/2023	- Debug pipeline - Finish capstone project report writing	- Capstone project report writing
FUTURE PLAN	Future	- Implement future plan to further develop the project	

Table B.1: Activity log since the beginning of Phase 1