INFORMATION EXTRACTION FROM SCANNED INVOICE DOCUMENTS USING DEEP LEARNING METHODS

by

Ufuk İlke Avcı

Submitted to Graduate School of Natural and Applied Sciences
in Partial Fulfillment of the Requirements
for the Degree of Master of Science in
Computer Engineering

Yeditepe University

2023

INFORMATION EXTRACTION FROM SCANNED INVOICE DOCUMENTS USING DEEP LEARNING METHODS

APPROVED BY:

Assoc. Prof. Dr. Dionysis Goularas .....
(Thesis Supervisor)
(Yeditepe University)

Prof. Dr. Emin Erkan Korkmaz .....
(Yeditepe University)

Prof. Dr. Cem Ünsalan .....
(Marmara University)

DATE OF APPROVAL: ..... / ..... / 20.....

I hereby declare that this thesis is my own work and that all information in this thesis has been obtained and presented in accordance with academic rules and ethical conduct. I have fully cited and referenced all material and results as required by these rules and conduct, and this thesis study does not contain any plagiarism. If any material used in the thesis requires copyright, the necessary permissions have been obtained. No material from this thesis has been used for the award of another degree.

I accept all kinds of legal liability that may arise in case contrary to these situations.

Name, Last Name Ufuk İlke Avcı

Signature .....

ACKNOWLEDGEMENTS

I would like to express my gratitude to my supervisor Assoc. Prof. Dr. Dionysis Goularas and Prof. Dr. Emin Erkan Korkmaz. Their guidance and support were essential in the completion of this thesis.

I'm thankful to Hande Altunordu and Hasan Üstün. Hande's support and encouragement were invaluable, and Hasan's technical help was instrumental in overcoming challenges in my project.

I also want to thank to Intecon Informatics and Consultancy and Barış Deveci for their support on this study.

Lastly, my deep gratitude goes to my family; my mother, Fatma Avcı, my father, Seyfettin Avcı, and my sister, İlkay Avcı. Their love and support have been my foundation throughout this journey.

ABSTRACT

INFORMATION EXTRACTION FROM SCANNED INVOICE DOCUMENTS USING DEEP LEARNING METHODS

In this thesis, a comprehensive comparison was conducted between a Graph Convolutional Network (GCN) model and two transformer based models for the tasks of node classification and sequence labeling respectively. These deep learning models were trained to extract information from invoice documents, with the aim of improving the automated processing of such data. A dataset of 1000 invoices was utilized for training the models, while a separate dataset consisting of 250 invoices was employed for evaluating their performance. The results showed that the one of the transformer based models achieved better results than GCN model, by achieving an F1-score of 0.65 for LayoutLMv1 model and 0.72 for LayoutLMv3 model, compared to the GCN model's 0.32. The results of this study shows the effectiveness of transformer based neural network models in information extraction tasks from scanned invoice documents and provide information about success of transformers and graph convolutional networks for this task.

ÖZET

DERİN ÖĞRENME YÖNTEMLERİ KULLANILARAK, TARATILMIŞ FATURA DÖKÜMANLARINDAN BİLGİ ÇIKARIMI

Bu tezde, düğüm sınıflandırma ve sıralı etiketleme görevleri için bir Graf Konvolüsyonel Ağı (GCN) modeli ile iki dönüştürücü tabanlı model arasında kapsamlı bir karşılaştırma yapıldı. Bu derin öğrenme modelleri, fatura belgelerinden bilgi çıkarmak üzere eğitildi, böylece bu tür verilerin otomatik işlenmesinin iyileştirilmesi amaçlandı. 1000 faturalık bir veri seti modellerin eğitiminde kullanılırken, performanslarını değerlendirmek için 250 faturalık ayrı bir veri seti kullanıldı. Sonuçlar, dönüştürücü tabanlı modellerden birinin GCN modelinden daha iyi sonuçlar elde ettiğini gösterdi. LayoutLMv1 modeli için F1-puanı 0.65 ve LayoutLMv3 modeli için 0.72 elde edilirken, GCN modeli için bu değer 0.32 oldu. Bu çalışmanın sonuçları, taranmış fatura belgelerinden bilgi çıkarma görevlerinde dönüştürücü tabanlı sinir ağı modellerinin etkinliğini ve transformerlar ile graf konvolüsyonel ağların bu görevdeki başarısını göstermektedir.

TABLE OF CONTENTS

ACKNOWLEDGEMENTS iv
ABSTRACT v
ÖZET vi
TABLE OF CONTENTS vii
LIST OF FIGURES ix
LIST OF TABLES xi
LIST OF SYMBOLS/ABBREVIATIONS xiii
1. INTRODUCTION 1
1.1. INVOICES 2
1.2. INFORMATION EXTRACTION 2
1.3. MOTIVATION OF THE STUDY 3
2. BACKGROUND 5
3. METHODOLOGY 11
3.1. DEEP LEARNING 11
3.2. DEEP LEARNING METHODS 12
3.2.1. Transformers 12
3.2.2. Graph Convolution 16
3.3. ADAMW OPTIMIZER 18
3.4. COSINE SIMILARITY 18
3.5. SOFTMAX FUNCTION 18
3.6. EVALUATING METHODS 19
4. ANALYSIS AND DESIGN 21
4.1. DATASET 21
4.1.1. Definition of Classes 22
4.1.2. Tokenization and Vectorization 25
4.1.3. Enrichment of Test Dataset with Invoice Variations 27
4.2. MODELS 27
4.2.1. Custom Graph Convolutional Network 28
4.2.2. LayoutLMv1 29
4.2.3. LayoutLMv3 31

4.2.4. LayoutLM Version Differences....32
5. IMPLEMENTATION....37
5.1. GRAPH CONSTRUCTION FROM INVOICE IMAGE....37
5.2. INVOICE LABELING APPLICATION....38
6. RESULTS AND DISCUSSION....40
6.1. CUSTOM GRAPH CONVOLUTIONAL NETWORK....41
6.2. LAYOUTLMV1....43
6.3. LAYOUTLMV3....49
6.4. MODEL COMPARISON....53
7. CONCLUSION AND FUTURE WORK....56
REFERENCES....57

LIST OF FIGURES

Figure 2.1. Attend, Copy, Parse model architecture [1]. 6

Figure 2.2. Multimodal architecture of BERT and GCN model [2]. 7

Figure 2.3. PICK model architecture [3] 7

Figure 2.4. Architecture of fully convolutional model [4]. 8

Figure 2.5. First stage predictions of finance office and company logo detectors [5]. 9

Figure 3.1. Training of BERT [6] 14

Figure 3.2. Transformer Architecture [7] 15

Figure 3.3. Architecture of an example GCN model with fully connected classification layer. 17

Figure 4.1. Regions of invoice for class grouping. 23

Figure 4.2. Tokenization and vectorization of data. 27

Figure 4.3. Architecture of custom GCN model. 28

Figure 4.4. Processing of visual properties in LayoutLM [8]. 30

Figure 4.5. LayoutLM architecture. 30

Figure 4.6. Detailed diagram of operations in LayoutLM models. 34

Figure 4.7. Diagram of calculation of input embeddings using sequences of token vectors and bounding boxes 35

Figure 4.8. Diagram of calculation of visual embeddings. 35

Figure 5.1. Invoice labeling application structure. 39

Figure 5.2. Screenshot of application. 39

Figure 6.1. Improvement over metrics during dataset enlargement 40

Figure 6.2. Confusion matrix of GCN model on 58 classes 41

Figure 6.3. Confusion matrix of LayoutLMv1 model on 58 classes 45

Figure 6.4. Confusion matrix of LayoutLMv3 model on 58 classes 49

Figure 6.5. Comparison of precision, recall, f1-sore and accuracy over all three models. 55

LIST OF TABLES

Table 4.1. Header group classes. 24

Table 4.2. Table group classes. 24

Table 4.3. VAT table group classes. 25

Table 4.4. Summary group classes. 25

Table 4.5. Other classes. 26

Table 4.6. Input output pair of model. 36

Table 6.1. GCN Grouped Metric Averages 41

Table 6.2. GCN metrics for header group classes 42

Table 6.3. GCN metrics for summary group classes. 42

Table 6.4. GCN metrics for table group classes. 43

Table 6.5. GCN metrics for VAT table group classes. 43

Table 6.6. GCN metrics for other classes. 44

Table 6.7. LayoutLMv1 Grouped Metric Averages 46

Table 6.8. LayoutLMv1 metrics for header group classes. 46

Table 6.9. LayoutLMv1 metrics for summary group classes 46

Table 6.10. LayoutLMv1 metrics for table group classes. 47

Table 6.11. LayoutLMv1 metrics for VAT table group classes. 47

Table 6.12. LayoutLMv1 metrics for other classes. 48

Table 6.13. Average elapsed time for each invoice 50

Table 6.14. LayoutLMv3 Grouped Metric Averages 51

Table 6.15. LayoutLMv3 metrics for header group classes. 52

Table 6.16. LayoutLMv3 metrics for summary group classes. 52

Table 6.17. LayoutLMv3 metrics for table group classes. 53

Table 6.18. LayoutLMv3 metrics for VAT table group classes. 53

Table 6.19. LayoutLMv3 metrics for other classes. 54

LIST OF SYMBOLS/ABBREVIATIONS

ACP	Attend, Copy, Parse
AI	Artificial Intelligence
AIE	Automated Information Extraction
BERT	Bidirectional Encoder Representations from Transformers
CNN	Convolutional Neural Networks
DCNN	Deep Convolutional Neural Networks
EATEN	Entity-aware Attention Text Extraction Network
FP	False Positive
FN	False Negative
GCN	Graph Convolutional Networks
IE	Information Extraction
LSTM	Long short-term memory
NLP	Natural Language Processing
OCR	Optical Character Recognition
PBM	Part-Based Modeling
PO	Purchase Order
RNN	Recurrent Neural Networks
TN	True Negative
ToI	Text of Interest
TP	True Positive
VAT	Value Added Tax
VLT	Visual Layout Transformer

1. INTRODUCTION

Extracting information from invoices is an important task in Finance and administration as it makes it easy for organizations to process, organize and analyze financial data efficiently [9]. In recent years, the development of deep learning methods, especially Natural Language Processing (NLP) and computer vision, has significantly improved information extraction systems [10]. This thesis presents a study of training and testing Graph Convolutional Networks (GCN), LayoutLMv1 and LayoutLMv3 models in the field of information extraction from the invoice in sequential node classification, sequence classification and sequence classification tasks respectively.

Invoice documents are semi-structured documents, with textual and non-textual information located spatially over the page [9]. Traditional approaches to information extraction, such as rule-based methods and shallow machine learning techniques, are frequently incapable of dealing with the complexity of these texts [11]. As a result of their capacity to model complicated connections between characteristics and develop appropriate data representations, deep learning approaches have come forward as a possible alternative [12].

In this thesis, an approach to information extraction from scanned invoice documents by leveraging the capabilities of LayoutLMv1 [8] and LayoutLMv3 [13] models, which combine NLP and computer vision techniques to capture both textual and layout information was proposed. Also GCN [14] employed for node classification tasks, which allows us to model the relationships between different elements in the document, and thereby enhance the overall extraction performance.

Further evaluation of models was done on a set of invoice documents. Moreover, the success of approach, and potential areas for future research and development was analyzed.

The thesis starts with an introduction to information about invoices, information extraction (IE) and motivation of study. Background section describes previous work related to this study. Methodology section explains deep learning and deep learning techniques, which have shown great potential in solving IE problems. It then describes the models used in the study, including GCNs, LayoutLMv1, and LayoutLMv3, their differences and evaluation methods used in the study. The analysis and design section explains in detail the dataset which used in

this study and models trained. Implementation section details hardware and used libraries. The results and discussion section presents the success of each model on the invoice IE task and compares results of models. Finally, the conclusion and future work section summarizes the findings of the study and outlines potential areas for further research. Overall, the thesis presents a comprehensive analysis of the performance of different deep learning models on a document-level IE task, with practical implications for real-world applications.

1.1. INVOICES

An invoice provides a clear list of services received or products purchased, the total charge, and the date of payment. This document is important for both the consumer and the provider or seller as it indicates to the consumer how much they need to pay, and it signifies to the provider how much payment they should receive for the service given or product sold. Invoices also function as tax documents. Businesses often use these documents to determine their tax obligations and to receive necessary tax deductions. Likewise, for individual consumers, invoices can serve as proof of specific expenditures. In recent years, digital invoicing has gained importance. Digital invoices, besides helping to protect the environment and reducing paper waste, provide a faster and more efficient payment process. However, traditional paper invoices are still in use and are preferred in some instances. Invoices are a crucial tool for documenting financial transactions, determining tax obligations, and tracking costs. Digitalization accelerates invoicing processes while offering a more sustainable solution [9].

In this study all invoice documents are provided by Intecon Informatics and Consultancy. These invoices are collected before to represent different types of invoices from different companies. This situation makes invoices in dataset more variable which is better for deep learning model training and testing.

1.2. INFORMATION EXTRACTION

The identification and extraction of structured information from unstructured or semi-structured sources, such as scanned documents, texts, and images, is a crucial step in the automated information extraction (AIE) process from scanned materials [15]. Transforming

large volumes of written data into a format that computers can understand is the main task of AIEs for researchers and organizations. This allows the extracted data to be processed in studies such as decision making, trend analysis and knowledge discovery [15].

Due to the enormous amounts of useful data that are still present in physical or scanned formats but are not easily accessible or searchable, AIE from scanned documents is required [15]. Additionally, manually extracting information from these sources can be labor-intensive, time-consuming, and error-prone [15]. AIE provides a more effective and accurate solution to these problems.

OCR, NLP, and machine learning algorithms are important methods used in AIE from scanned texts [16]. OCR technology transforms scanned text pictures into machine-readable formats that may be processed using NLP approaches to identify and extract particular information based on syntactic and semantic analysis. Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN) are two examples of deep learning algorithms that have shown to significantly increase the accuracy and efficiency of information extraction from scanned texts [16].

1.3. MOTIVATION OF THE STUDY

The rising digitization of enterprises has increased the demand for efficient information extraction from scanned invoice papers. These papers, which account for a sizable part of business transactions, include essential data that may be used to improve financial management, process automation, and decision-making. Despite the widespread use of electronic invoicing systems, many businesses still rely on paper-based invoices or receive them from partners, necessitating the translation of these documents into digital formats [17]. Manual data input is not only time-consuming, but it is also prone to human mistake, resulting in inconsistencies in financial records and increased operating expenses [15]. As a result, in today's fast-paced corporate climate, developing accurate and effective information extraction procedures for scanned invoice papers is critical. Identifying and extracting relevant data fields such as invoice number, date, supplier details, item descriptions, quantities, prices, and taxes from scanned invoices is known as IE [9]. The obstacles in this domain are numerous,

including invoice format variations, the existence of noise in scanned documents, and the complexity of data extraction procedures. Recent advances in OCR and NLP technology have shown encouraging results in terms of automating the extraction process and lowering the requirement for manual intervention [17]. The purpose of this study is to contribute to the ongoing research on IE from scanned invoice papers by presenting a unique technique that tackles the aforementioned issues, thereby aiding enterprises in simplifying their financial procedures and increasing overall efficiency.

2. BACKGROUND

A literature review was conducted for this study. In this review, studies that attempted to solve the problem using different methods were identified, and some necessary information for conducting this study was obtained. In this section of thesis, a descriptive list of similar studies and studies which aimed to solve information extraction problem.

LayoutLM is a revolutionary pre-training approach that tackles the constraints of existing NLP models by modeling text and layout information in scanned document pictures simultaneously. LayoutLM integrates both layout and style information, which is critical for analyzing document pictures and extracting essential information, as opposed to earlier techniques that concentrated exclusively on text-level modification. The model collects visual information about words by incorporating picture characteristics, making it the first framework to combine text and layout learning in a single document-level pre-training process. LayoutLM achieves new state-of-the-art outcomes in a variety of downstream tasks, including form comprehension and document picture categorization. The authors have made the code and pre-trained models available to the public, further benefiting the research community [8].

Another work introduces EATEN (Entity-aware Attention Text Extraction Network), a trainable end-to-end system for extracting Text of Interest (ToI) from pictures without the need for post-processing. EATEN is the first single-step method for extracting entities from photos, solving the shortcomings of previous systems that rely on extensive engineering pipelines including OCR and structural information extraction. The proposed system parses each entity using entity-aware decoders and proposes a unique state transition technique to improve the robustness of visual ToI extraction. To aid in evaluation, the authors created and made publicly available a dataset of 0.6 million photos from real-world scenarios such as train tickets, passports, and business cards. The experimental findings show that EATEN outperforms previous approaches, showing its potential for OCR applications [18].

Similar to this study, in "Automatic invoice interpretation: Invoice structure analysis" paper, writers aimed to extract information from invoices using graphical information of document. They are using classical image processing methods to detect visual structures in document to interpret obtained data by using OCR [19].

Attend, copy, parse end-to-end information extraction from documents papers authors address the constraints of current word classification approaches for document information extraction, which rely on expensive word-level labels that are frequently unavailable for real-world applications, in this work. The Attend, Copy, Parse (ACP) architecture is proposed, which is a deep neural network model that can be trained directly on end-to-end data without the requirement for word-level labels. The ACP architecture is tested on a large, diversified collection of bills and outperforms a state-of-the-art production system based on word categorization. The authors suggest that their proposed architecture is more applicable to real-world information extraction tasks where standard word classification approaches may be impractical due to a lack of needed word-level labels [1]. In Figure 2.1, diagram of the model is shown. While attend, copy and parse modules are designed separately, they can be trained at the same time since tho all operations are differentiable an model [1].

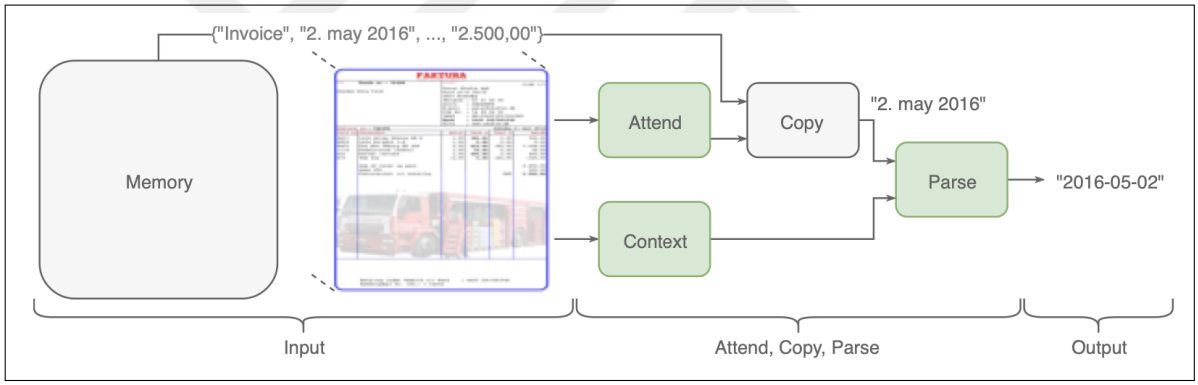

Figure 2.1. Attend, Copy, Parse model architecture [1].

Another study in information extraction field was published as Robust Layout-aware IE for Visually Rich Documents with Pre-trained Language Models. In this study, a multimodal architecture was presented for information extraction. Model was designed as a combination of BERT model and a custom graph convolutional network [2]. Diagram of the model is shown in Figure 2.2. Model predicts by feeding outputs of BERT model to a custom graph convolutional network to process font properties of text in image [2].

Similar to previous study, Processing Key Information Extraction from Documents using improved Graph Learning Convolutional Networks (PICK) study introduces another multimodal architecture using transformer and graph convolutional models [3]. In this study, unlike the previous one, model does not feed output of transformer to graph convolutional

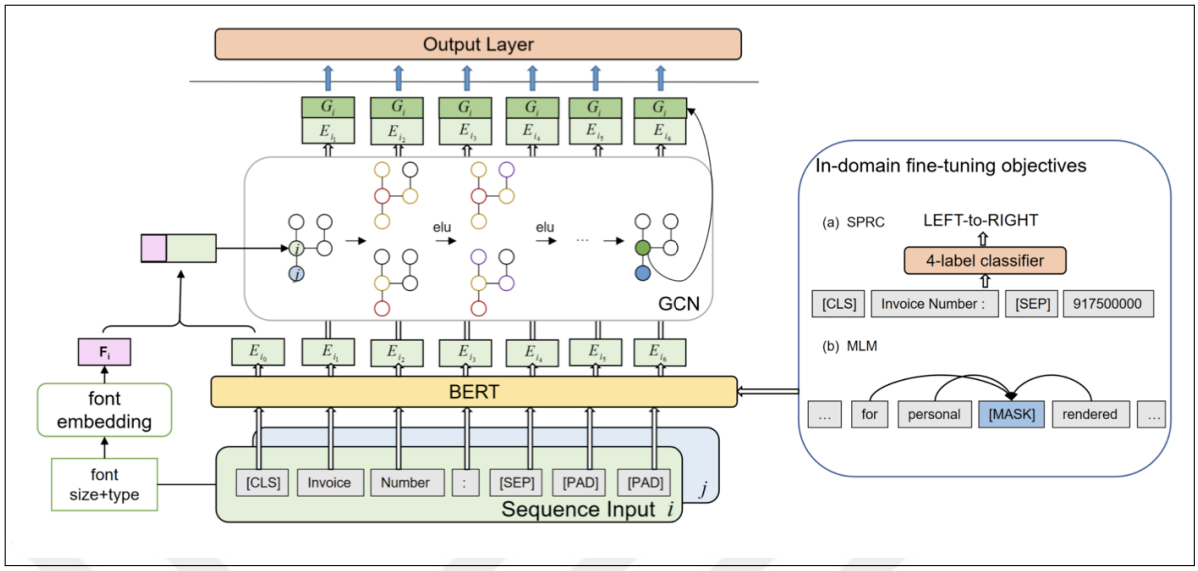

Figure 2.2. Multimodal architecture of BERT and GCN model [2].

model, instead they are preparing three set of inputs for a given document image. One of the inputs is a sequence of words in document, second one is cropped images of words in document and the last input is graph of words to represent spatial relations of words in document. Model process these inputs by using a transformer model, a convolutional neural network and a graph convolutional network respectively [3]. After these three models process data, combination of outputs are fed to a decoder network which build with bidirectional Long short-term memory (LSTM) model to predict output classes of words [3]. Diagram of PICK model is shown in Figure 2.3.

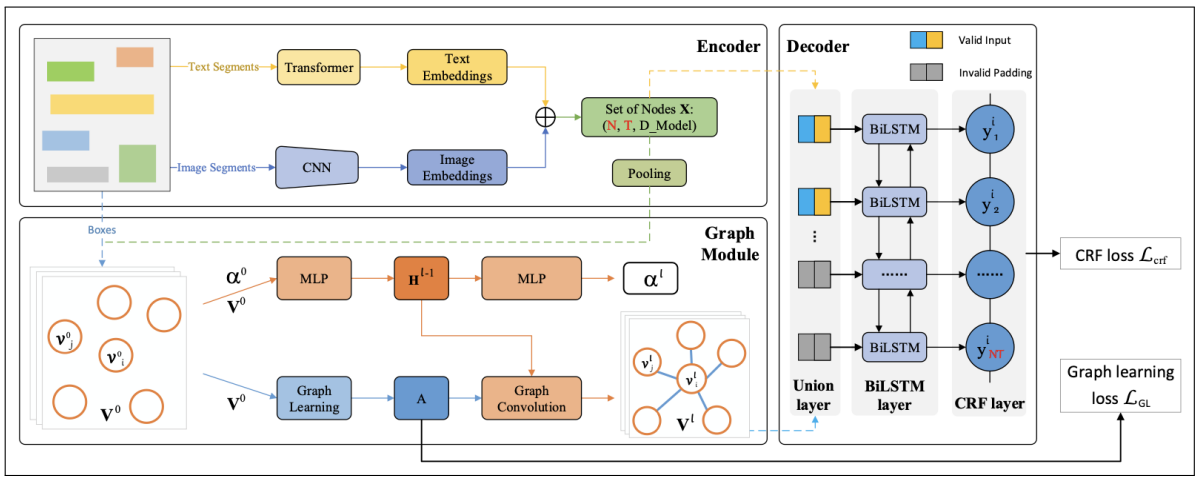

Figure 2.3. PICK model architecture [3].

In paper named Table Detection in Invoice Documents by Graph Neural Networks, a type

of automated preprocessing operation was aimed on invoice documents [20]. In this study, a custom GCN model was built to classify entities in a document as table or not. Resulting model for the study was tested on a dataset and reached 0.70 f1-score value on table detection task [20].

In another study for information extraction from document images, segmentation of document image was aimed by masking document image using fully convolutional network. In paper Learning to Extract Semantic Structure from Documents Using Multimodal Fully Convolutional Neural Networks, a fully convolutional model was designed and trained to segmentize document images into figure , table , section heading , caption , list and paragraph segments in image [4]. Model was designed to generate an image as same size as input to represent segments in image as colors [4]. As can be seen from Figure 2.4, the model produces output as an image and uses only convolutional operations to process input image. Model reaches 86% of average intersection over union ratio for all these segments [4].

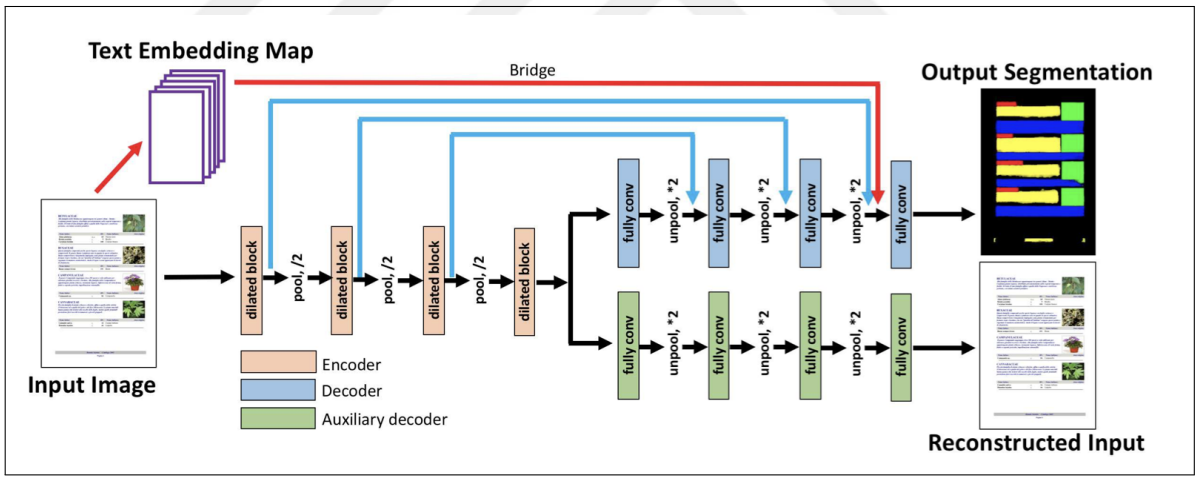

Figure 2.4. Architecture of fully convolutional model [4].

In the article titled "CloudScan - A configuration-free invoice analysis system using recurrent neural networks", details about the invoice analysis system called CloudScan are presented. A standout feature of CloudScan is that it does not require any initial configuration or annotation and does not adhere to the templates of invoice layouts [21]. Instead, it uses a single universal model that can naturally generalize to unseen invoice layouts [21]. The structure of the model is based on the LSTM architecture. As stated in the article, the model has an average F1-score of 0.84 [21].

A Part Based Modeling Approach for Invoice Parsing article focuses on a new method that brings an automated approach to invoice processing and information extraction [5]. In this new approach, a two-stage structure is used [5]. In the first stage, support vector machine, maximum entropy, and histogram of oriented gradients are used to identify different sections of the invoice. In Figure 2.5, outputs of first stage is shown for finance office logo and company logo along with invoice image with bounding boxes for these objects. In this stage, potential candidates for invoice parts are generated [5]. In the second stage, a modeling approach is used to bring together the parts of the invoice [5]. This is similar to the process of face or human body detection from digital images [5]. The main advantage of this part-based modeling (PBM) approach is its ability to handle all kinds of invoices [5]. Finally, the proposed system has been tested on real invoices, and experimental results confirm the effectiveness of the proposed approach [5]. This study presents a new way of effectively performing automated invoice processing and information extraction [5].

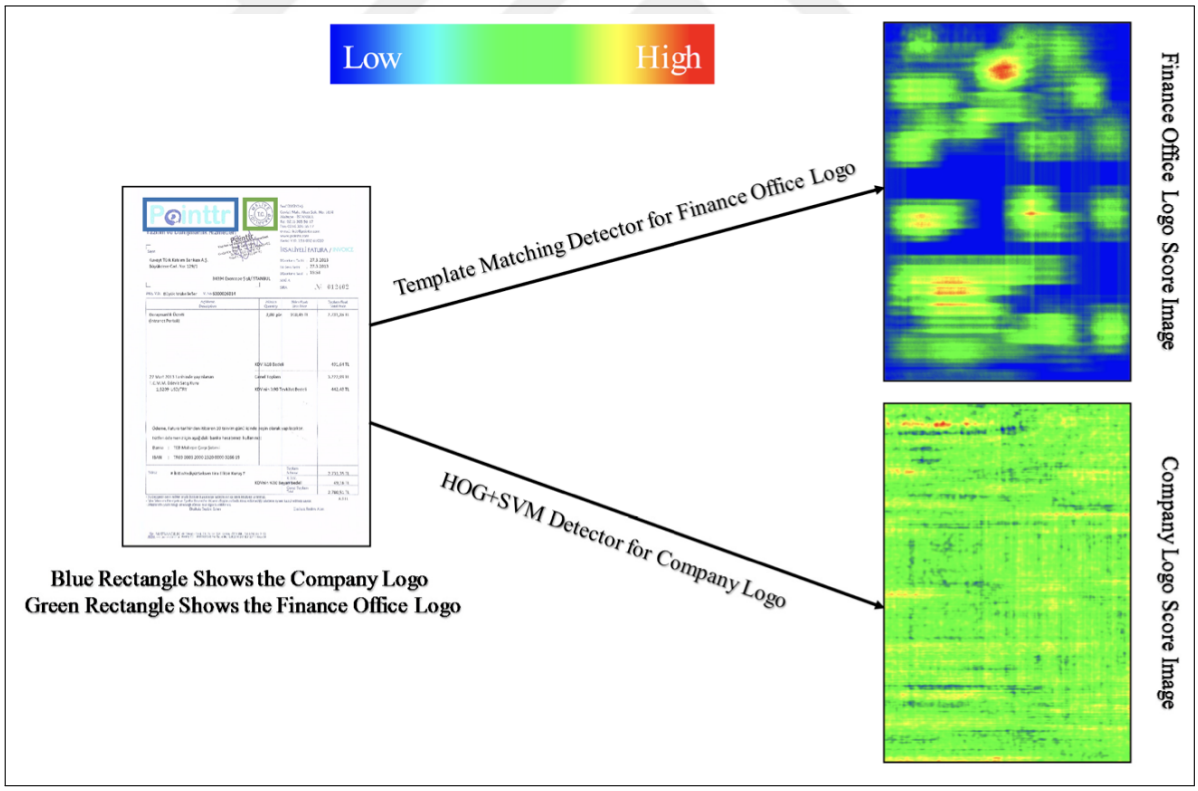

Figure 2.5. First stage predictions of finance office and company logo detectors [5].

the article Deep Learning for automatic sale receipt understanding discusses the industrial context of scanned document analysis, focusing on the automatic comprehension of sales receipts and providing access to consumption statistics. The goal of the paper is to extract

high-confidence information such as the store brand, purchased products, and related prices from an image taken with a smartphone. To accomplish this, even if scanning is not perfectly controlled, the authors propose a dual-check process utilizing Deep Convolutional Neural Networks (DCNN) and traditional image and text processing techniques. The novelty of this study lies in this dual-check process and the joint use of DCNNs for different applications and text analysis. The article aims to develop a tool that could potentially assist a company in better understanding its customers' purchasing habits. This could help a company more effectively target its products and services. Notably, despite challenges that may arise during the scanning process, this tool aims to provide accurate information with a high level of confidence [22].

Every study for information extraction from scanned documents require characters in document to detected by computer. Yindumathi presents an in-depth examination of Optical Character Recognition (OCR) technology, which is required for identifying printed or handwritten characters from pictures such as scanned invoices and bills, as well as allowing document analytics through intelligent text extraction and storage. The authors explain the usage of an OCR engine for processing intermediate pictures as they investigate the methods for extracting any information from printed invoices. They look at segmentation strategies that work with different typefaces and languages, as well as image classification methodologies that help with decision-making based on classification results. The article compares different methodologies in terms of measurements, algorithms, and outcomes, providing important insights into the current status of OCR technology and its applications in information extraction [19].

3. METHODOLOGY

In this section, recent deep learning methodologies which can be used in the deep learning-based invoice information extraction study are presented. It begins by brief definition of methods. After explanation method details and their advantages over different cases represented. After deep learning methods, evaluating methods which are used to evaluate results of trained models are described.

3.1. DEEP LEARNING

Machine learning is a form of artificial intelligence (AI), and deep learning is a subfield of machine learning that focuses on using artificial neural networks to model and solve complicated problems [12]. The hierarchical, layered architecture of these neural networks, which mimic the structure and operation of the human brain, enables the system to learn from and adapt to massive volumes of data. The "deep" in deep learning refers to the various layers of connected nodes, or neurons, used in the network to gradually extract higher-level properties and representations from the input data [23]. Deep learning models can attain state-of-art performance in a variety of tasks, including image and speech recognition, natural language processing, and computer vision [24], by training these networks on large datasets. Deep learning techniques may be used to improve the accuracy and efficiency of the extraction process in the context of AIE. Deep learning applications in AIE may be divided into two categories: enhancing OCR performance and supporting enhanced natural language interpretation and extraction. CNNs are a group of deep learning models that have shown substantial success in image recognition tasks [25]. CNNs may be used to increase the accuracy of OCR systems in the context of AIE by more successfully detecting and converting scanned pictures of text into machine-readable format. CNN-based OCR systems can handle diverse fonts, sizes, and styles of text, as well as overcoming issues related to noise, distortion, and low-resolution pictures, by learning robust features from large-scale training datasets [26]. Advancing natural language understanding and extraction: After OCR conversion, deep learning models such as RNNs [27] and Transformer-based models [7] may be used to handle and evaluate textual data. These models are capable of capturing intricate linguistic

patterns and relationships, enabling for more accurate text detection and extraction. RNNs, for example, can model word sequences to recognize syntactic and semantic relationships within a sentence, whereas transformer-based models, such as BERT [6] and GPT [28], can leverage large-scale pre-training to develop a deep understanding of language, which can then be fine-tuned for specific information extraction tasks. To summarize, deep learning approaches have the potential to significantly improve the capabilities of AIE systems by boosting OCR performance and enabling improved natural language processing and extraction. These developments may eventually lead to more accurate, efficient, and resilient AIE solutions capable of extracting important insights from scanned documents and other unstructured data sources.

3.2. DEEP LEARNING METHODS

The models used in this study are based on methods that have emerged as a result of recent developments in the field of deep learning. One of these methods is the transformer architecture. A detailed explanation of this architecture will be made later in subsection 3.2.1. The main distinguishing feature of this architecture is its ability to operate on sequential data in the form of a long series, and its performance is not affected by the length of the series. Another method is the graph convolution method. A detailed explanation of this method has also been made in 3.2.2, and its primary purpose is to process data in a graph structure.

3.2.1. Transformers

In the paper "Attention Is All You Need", the authors introduce the transformer architecture to be used on sequential data, as an alternative to models in the RNN architecture. The transformer architecture consists of a combination of an encoder and a decoder. This combined encoder-decoder structure applies operations to each element in the sequential data individually. The result of the encoder operation applied for each element is influenced by other elements within the sequence at varying rates, thanks to the attention mechanism [7]. As a result of these operations, significant progress has been made in the field of sequence processing. The model's success in language-to-language translation has attracted attention

[29]. The success that the authors particularly highlight in the paper is that the model is not significantly affected by the growth of data, unlike RNNs [7].

A pre-trained language model called Bidirectional Encoder Representations from Transformers (BERT) was created by Google, and it has attained state-of-art performance on a variety of natural language processing (NLP) tasks, such as text classification, question answering, and language generation [6]. BERT is intended to produce contextualized embeddings of words that capture their meaning in context, in contrast to typical language models that generate representations of words depending on their context in a phrase. The Transformer architecture, a class of neural network that has been demonstrated to be efficient for sequence-to-sequence tasks, is the foundation of BERT [6]. The model is pre-trained using a masked language modeling aim, which includes randomly masking off words in a sentence and then predicting the masked words based on their context, on a huge corpus of text. The next sentence prediction objective trains the model to determine whether two sentences in a given text will follow one another.

By building a task-specific layer on top of the pre-trained model and training the entire model from beginning to end on the task-specific dataset, the BERT model may be fine-tuned on a particular downstream task after pre-training. This technique of fine-tuning has produced state-of-the-art performance on numerous benchmarks and has been proven to be highly successful for a variety of NLP jobs [6]. The ability of BERT to produce contextualized embeddings that accurately capture the meaning of words in context is one of its main advantages. This enables the model to excel in tasks that call on a thorough comprehension of language, such sentiment analysis and question-answering [6]. In the paper "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding", the BERT model is introduced. The most notable point of the BERT model is that it is structured in a transformer architecture, and by adding a simple final processing layer on top of this, an NLP model that can be trained to perform complex tasks such as question answering has been created [6].

As seen from Figure 3.1, pre-training of BERT model was done by providing masked sentences and expecting whole sentences from model [6]. After pre-training of model, task specific fine tuning also presented in figure such as question answering [6]. For other

types of tasks, model can be fine-tuned by using suitable data and data format.

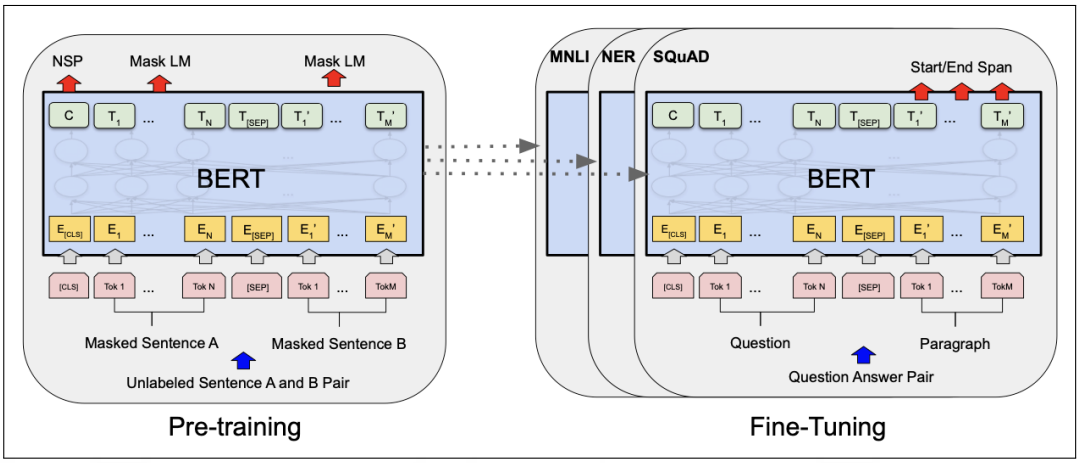

Figure 3.1. Training of BERT [6]

In the Figure 3.2, the structure of the transformer model is shown. Following the flow of data, it can be seen that the models are provided to the network with a positional embedding. This is included to allow the attention modules, named Multi-Head Attention, to identify the place of the input within the sequence and process. In the continuation of the operations, it is seen that the encoder results are obtained with a feed-forward network and transferred to the decoder module, while another input is provided to the decoder module. The data expressed as outputs in the figure represents the output produced by the model after the previous input. As can be observed in the Figure 3.2, upon the completion of the decoder module's operations, a fully connected layer process is applied to generate the result the model produces for the corresponding element in the sequence.

The attention mechanism in Transformer networks regulates the interaction between the every instance in input and the other inputs in the sequence during processing. The operation of the attention mechanism can be summarized as follows. In the first stage, Query (Q), Key (K), and Value (V) vectors are calculated for each token. These vectors are obtained by multiplying the weight matrices of the input tokens. Assuming the E is input embedding token provided to attention module, calculations of output is shown in Equations 3.1, 3.2 and 3.3 [7].

$$Q = E \times W_q \tag{3.1}$$

$$K = E \times W_k \tag{3.2}$$

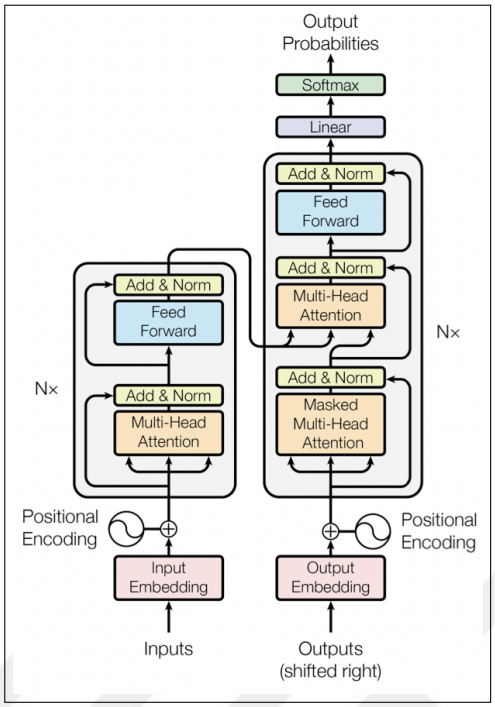

Figure 3.2. Transformer Architecture [7]

$$V = E \times W_v \tag{3.3}$$

In the second stage, the query vector is compared with each key vector and a score is obtained. The score determines how much the model's attention will focus on a token. The scores are subjected to the softmax operation and ensure that the scores represent a probability distribution. This distribution determines which tokens are important for a specific token. Calculation of score S_i for Q_i is given in Equation 3.4 [7].

$$S_i = softmax(\frac{Q_i \times K^T}{\sqrt{d_K}}) \tag{3.4}$$

The softmax scores are multiplied by the value vectors. This gives a weighted Value vector for each token. All weighted value vectors are added up. This result gives an output vector for each input token and these output vectors become the input for the next layer [7]. Calculation of output of E_i from V and S_i is shown in Equation 3.5

$$Output_i = V \times S_i \tag{3.5}$$

3.2.2. Graph Convolution

In the paper "Semi-Supervised Classification with Graph Convolutional Networks", graph convolution is introduced and by using this method, GCNs are created and made trainable to process data in graph structure. The graph convolution structure aims to modify node values by associating them with their neighbors based on the characteristics of the edges connecting nodes in the data. The authors have likened this situation to updating values in a matrix based on neighborhood relationships in the traditional convolution structure. With this created structure, the aim is to affect the result of the structure represented by the graph by enabling data units to interact with each other [30].

The neural network type known as GCNs works with graph data structures, which are frequently employed to describe intricate interactions between objects or nodes in graph. By defining convolutions on graphs, GCNs expand conventional convolutional neural networks to operate with graph data [14]. The fundamental concept underlying GCNs is to update a node's feature representation by collecting data from the node's vicinity using graph convolutions [14]. This is accomplished by giving each node's features and the features of its neighbors a weight matrix, summing the resulting products, and then calculating the outcome. To create the new node features, the result of this process is then run through a non-linear activation function. This procedure is repeated over a number of layers, each of which gathers data from a bigger and farther community. GCNs have found use in numerous tasks, for instance node classification, link prediction, and graph classification [31]. GCNs are able to learn representations of nodes that capture both their local and global context in the graph. This makes GCNs predominantly effective for tasks which the presence of relationships and relation details between nodes are important, such as social network analysis, recommender systems, and drug discovery.

Graph convolution operation in graph convolution layers are done by calculating weighted averages of connected nodes to a given node and updating of node value [14]. Assuming H_l vector defines node values for all nodes at the beginning of layer l and H_{l+1} defines node values for all nodes at the end of layer l, W_l is differentiable weight matrix for layer l, A is adjacency matrix for graph and σ is activation function, equation 3.6 and 3.7 show calculation

of $\hat{A}$ and $\hat{D}_{ii}$.

$$\hat{A} = A + I_N \tag{3.6}$$

$$\hat{D}_{ii} = \sum_j \hat{A}_{ij} \tag{3.7}$$

After $\hat{A}$ and $\hat{D}_{ii}$ are calculated, output of layer l, H_{l+1}, is defined in Equation 3.8 as activation function applied on multiplication of $\hat{D}^{-\frac{1}{2}}$, A, $\hat{D}^{-\frac{1}{2}}$, H_l and W_l. As the result of these operations, a new graph, with same adjacency matrix and different node values, is obtained. Output of graph convolution layer can be provided to another graph convolution layer for more processing or node values can be fed to a fully connected layer for further operations such as classification [14]. In Figure 3.3, architecture and description of an example GCN model with node classifier fully connected layer at the end.

$$H_{l+1} = \sigma(\hat{D}^{-\frac{1}{2}} A \hat{D}^{-\frac{1}{2}} H_l W_l) \tag{3.8}$$

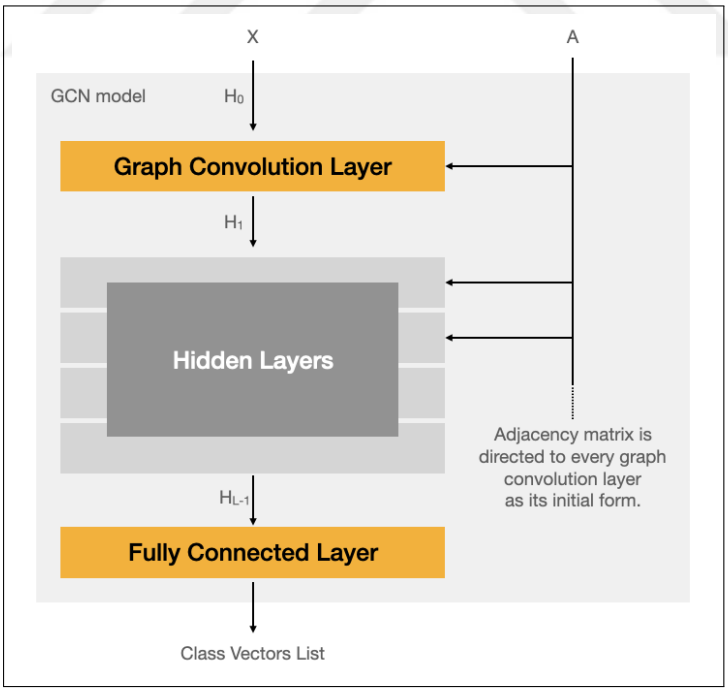

Figure 3.3. Architecture of an example GCN model with fully connected classification layer.

3.3. ADAMW OPTIMIZER

AdamW was presented as a solution to some of the problems of Adam optimizer by separating weight decay from optimization processes[32]. This change improves performance and makes convergence more reliable. The weight decay is implemented after the gradients have been calculated in conventional Adam [33]. This may appear to be a trivial point, but it can have a substantial impact on the model's performance. The concern is that the scale of the gradients influences the weight decay, which might result in irregular learning behavior [33]. AdamW alters the update rule such that weight decay is applied before computing gradients [32]. AdamW is more successful at discovering optimal solutions after detaching weight decay from optimization phases [32]. AdamW's 'W' stands for weight decay fix [32].

3.4. COSINE SIMILARITY

Cosine similarity is a metric for calculation of similarity between two vectors in a multi-dimensional space. As seen from Equation 3.9, its defined as ratio between dot product of two vectors to multiplication of their lengths.

$$CosineSimilarity(A, B) = \frac{A \cdot B}{||A|| \times ||B||} \tag{3.9}$$

Resulting value of cosine similarity gives cosine of angle between vectors and range from -1 to 1. Higher cosine similarity value indicates similar vectors in space.

3.5. SOFTMAX FUNCTION

Softmax is a function used to convert a set of real numbers into a probability distribution. It is commonly used as a decision function in multi-class classification problems in machine learning and particularly in deep learning applications [12]. Equation 3.10 shows calculation of softmax function on z vector where K is length of z vector.

$$Softmax(z)_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \tag{3.10}$$

3.6. EVALUATING METHODS

Precision, recall, F1 score, and accuracy are essential performance metrics in the field of machine learning, particularly for evaluating the quality of classification algorithms [34]. These metrics allow researchers to determine the effectiveness of a model by comparing its predictions to ground truth labels.

Precision, also known as positive predictive value, measures the proportion of true positive instances among those classified as positive by the model [35]. It is calculated as the ratio of true positives (TP) to the sum of true positives and false positives (FP), as seen in Equation 3.11. High precision indicates that the model is reliable in identifying positive instances, but it does not account for the number of false negatives (FN) [34].

$$Precision = \frac{TP}{TP + FP} \tag{3.11}$$

Recall, alternatively referred to as sensitivity or true positive rate, evaluates the proportion of true positive instances among all actual positive instances. It is computed as the ratio of true positives to the sum of true positives and false negatives, as seen in Equation 3.12. High recall implies that the model is effective in capturing positive instances but does not consider the number of false positives [34].

$$Recall = \frac{TP}{TP + FN} \tag{3.12}$$

The F1 score is a harmonic mean of precision and recall, balancing the trade-off between them. It is defined in Equation 3.13. The F1 score ranges from 0 to 1, with 1 representing perfect precision and recall, and 0 indicating the absence of either [34].

$$F1score = 2 * \frac{Precision * Recall}{Precision + Recall} \tag{3.13}$$

Accuracy measures the overall proportion of correct predictions, i.e., the ratio of true positives and true negatives to the total number of instances [36]. Definition is given in Equation 3.14. While accuracy is intuitive, it can be misleading in cases of imbalanced datasets, where one

class is significantly more prevalent than the others [37].

$$Accuracy = \frac{TP * TN}{TP + FP + FN + TN} \tag{3.14}$$

4. ANALYSIS AND DESIGN

In this chapter, details of the study expressed about dataset and used neural network models. Chapter starts with explanation of dataset, this section explains which type of data was used, which classes are labeled on dataset and which preprocessing operations was done on dataset. Second section describes details of used models which modification was done on models for this task and training process.

4.1. DATASET

The dataset was used for training models, consists of 1000 invoice document pages as PNG images which is provided by Intecon Informatics and Consultancy. These images were labeled by selecting locations of desired fields of invoices. After labeling images a few steps preprocessing was applied to labeled data to feed and train models using this dataset. first step of preprocessing is extracting textual information from document images using TesseractOCR. TesseractOCR provides information as words in document and spatial locations of rectangular bounding boxes of these words in document image as coordination of top-left pixel of bounding box as x and y coordinate, width and height of bounding box as pixel values. After all data gathered from OCR, words were processed to split into word pieces, then word context vectors was calculated by summing context vectors of word pieces that word has. After preprocessing data can be considered as a sequence of pair of word context vectors and spatial locations as input data and class name for corresponding word belongs to as output data. Data is used in this form for training LayoutLMv1 and LayoutLMv3 as it is. In order to train GCN, data converted to graph forms for every invoice. Conversion from raw data to graph form algorithm is described in section 4.2.1. For classification of words, 57 different classes were decided to be. These classes are considered in three main groups which are desired classes, label classes and common unnecessary classes. Desired classes is defined as information represented in invoice and required to apply further operations on invoice data. Label classes are labels represented in invoice as title or description for other desired data that can be extracted from invoice. Title for phone number can be given as example for label class. Common unnecessary classes are common information that can be

gathered from invoices but not required for other analysis operations that can be applied to invoice such as bank account information of one of the parties to the invoice. Classification of words were required to be modified to feed LayoutLM models since the architecture of models requires additional classes to train. In order to classify word groups, LayoutLM classifies words in group one by one. To indicate that these words belong to same word group, classification of words are done by using subclasses. By that conversion every class turned into 4 different classes with B, I, E and S prefixes. These prefixes consecutively defines if the word is beginning word, inner word or ending word in word group or if it is a single word information. This subclassing method gives ability to group classified words as paragraph, sentence or a short phrase. Since output of LayoutLM models are sequences of classes, this subclassing is the only method for grouping words.

For evaluating results of model, 57 classes with additional OTHER named class were grouped to understand results of model based on regions of invoice. In Figure 4.1, regions of an invoice are shown. For evaluation classes are grouped based on which invoice region contains them. These groups are defined as table group, header group, summary group and VAT table group.

4.1.1. Definition of Classes

In this study, classification of words in invoice documents is aimed to extract information from invoices. To train models by using dataset labeling based on classes is required. In this subsection of thesis, description of classes are presented. These classes are separated to different groups to understand results of models better. Fist group is header group and contains classes invoice customer information, invoice date, invoice due date, invoice number, invoice purchase order (PO) number, supplier name, supplier address, supplier fax number, supplier phone number and supplier Value Added Tax (VAT) number. As can seen from included classes, these group is related to general information about invoice and its parties. List of all classes are shown in Table 4.1 which included in header group.

Second selected group is table group. Instances of this group contains information about products or services. Classes belong to this group can be listed as invoice line item number, invoice line number, invoice line quantity, invoice line subtotal, invoice line unit price, invoice

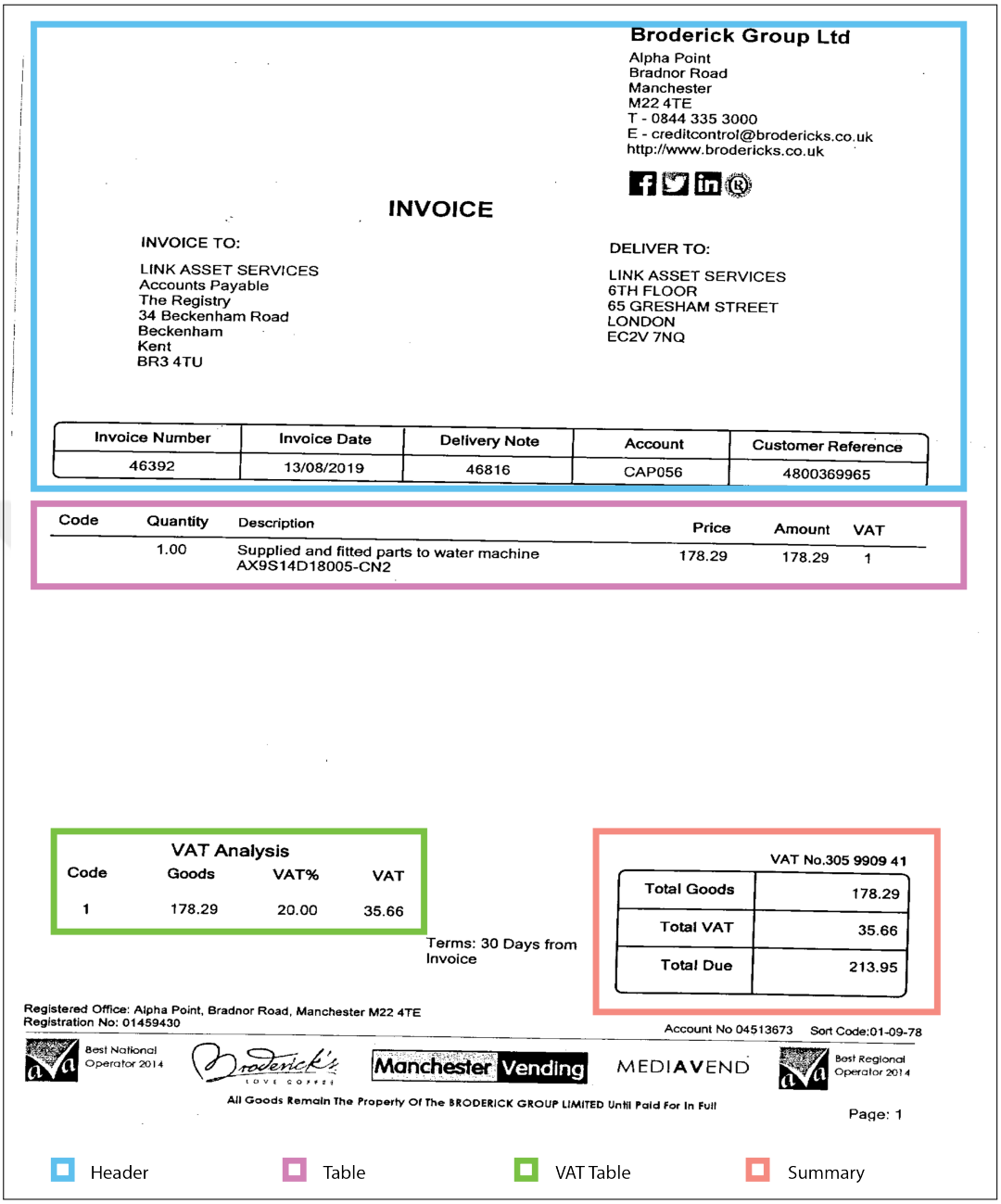

Figure 4.1. Regions of invoice for class grouping.

line VAT and invoice line VAT rate. List of all classes in table group is shown in Table 4.2.

Table 4.1. Header group classes.

Class Name
invoice customer info
invoice date
invoice po number
supplier phone number
invoice due date
invoice number
supplier fax number
supplier name
supplier vat number
supplier address

Table 4.2. Table group classes.

Class Name
invoice line vat rate
invoice line unitprice
invoice line item no
invoice line quantity
invoice line subtotal
invoice line number
invoice line vat

Another group is VAT table group. This group contains least common class instances among others. Information stored in instances of these class describes details about taxes. VAT table header, VAT table code, VAT table rate, VAT table subtotal and VAT table VAT classes are considered in this group. Classes of this group are presented in Table 4.3.

Fourth class group is names as summary group and contains invoice summary information in its instances. Classes in summary group are invoice summary subtotal, invoice summary total, invoice summary VAT and invoice summary VAT rate. Summary group classes are shown in Table 4.4.

Other than these four groups 32 other classes are labeled in invoices. These classes are created to get mode information from model and provide to other downstream tasks. These classes listed as invoice date label, invoice due data label, invoice extra information, invoice footer,

Table 4.3. VAT table group classes.

Class Name
invoice vat table code
invoice vat table header
invoice vat table vat
invoice vat table rate
invoice vat table subtotal

Table 4.4. Summary group classes.

Class Name
invoice summary vat rate
invoice summary vat
invoice summary total
invoice summary subtotal

invoice information header, invoice line item description, invoice line item description label, invoice line item number label, invoice line number label, invoice line quantity label, invoice line subtotal label, invoice line unit price label, invoice line VAT label, invoice line VAT rate label, invoice line extra information, invoice number label, invoice payment information, invoice PO number label, VAT table code label, VAT table rate label, VAT table subtotal label, VAT table VAT label, invoice summary label, invoice summary subtotal label, invoice summary total label, invoice summary vat label, invoice summary extra information, supplier fax number label, supplier phone number label, supplier VAT number label and supplier extra information. Classes that contains label in their names are information indicating words in document similar to "phone:". Extra information classes are designed to classify objects in different sections of invoice document and its instances are rarely seen in invoice documents. List of other classes are shown in Table 4.5

4.1.2. Tokenization and Vectorization

After document images are scanned using OCR, raw text data is obtained. This text data is needed to be processed before they fed to networks. In order to tokenize and vectorize text, WordPiece[38] method was used. This method first takes whole text as input and then split

Table 4.5. Other classes.

Class Name
invoice payment info
supplier extra info
invoice line item no label
invoice vat table subtotal label
other
invoice line vat rate label
supplier fax number label
invoice summary extra info
invoice vat table code label
invoice line extra info
invoice summary total label
invoice line number label
invoice footer
invoice summary vat label
invoice due date label
invoice number label
invoice vat table vat label
invoice date label
invoice line quantity label
invoice summary subtotal label
invoice extra info
invoice po number label
invoice vat table rate label
invoice summary label
supplier phone number label
invoice line subtotal label
invoice info header
invoice line item description
invoice line vat label
invoice line item description label
invoice line unitprice label
supplier vat number label

into word pieces which were defined before by examining large text corpus. As continuation of tokenization process, these word pieces are converted into vectors of length 768, using a lookup table that created by another network which is trained on same corpus. Tokenization and vectorization process is described with a diagram of an example in Figure 4.2 After these operations done sequence of token vectors are ready to fed into both GCN and transformer networks. GCN is using this sequence as list of node values.

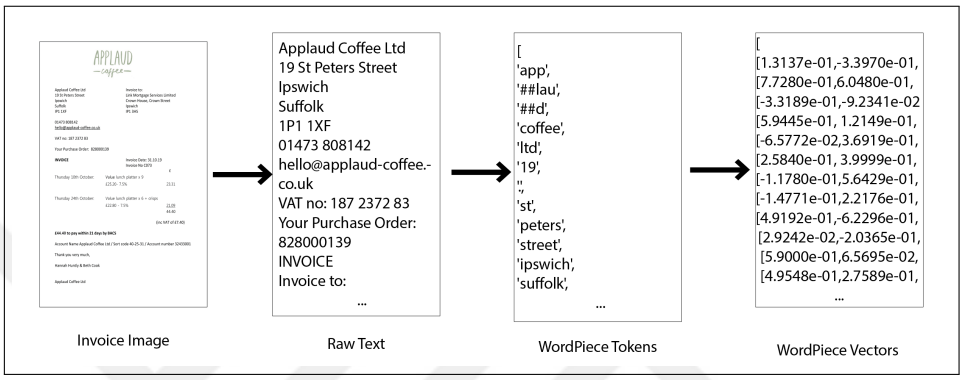

Figure 4.2. Tokenization and vectorization of data.

4.1.3. Enrichment of Test Dataset with Invoice Variations

To test models success on variety of invoices, 50 of the invoices vrom dataset was created using made up invoices. These invoices are generated using random information including invoice number, supplier name, product price etc. These randomly created invoice information is turned into document images by constructing documents in various randomly generated layouts.

4.2. MODELS

In this section, detailed information is provided about the training of the models used. As mentioned in the model details, the training and testing of the models were carried out by keeping the created dataset constant. In this way, the success of the model structures on this process could be evaluated.

4.2.1. Custom Graph Convolutional Network

To utilize GCN architecture for information extraction from invoice documents, A GCN model was designed from scratch using pyTorch. The model was designed as union of two graph convolutional layers as feature extraction module and two fully connected layers as classifier module of model. To convert document data to graph form, nodes of graph constructed as context vectors of words. Edges are connected between words if bounding boxes of words are horizontally or vertically aligned, and these words are neighbors. With this method node of every word is connected to the nearest words nodes in four main directions. This connection gives model ability of understanding in sentence relations between words if they are horizontally connected or table-like structure relations if they are vertically aligned. Model was designed to predict which class words are belong to. Output from model obtained as a graph with same node connection as input graph and instead of Word vectors, nodes store class information. The model was trained by using 1000 annotated invoice images for 300 epochs. Model architecture is shown in Figure 6.2.

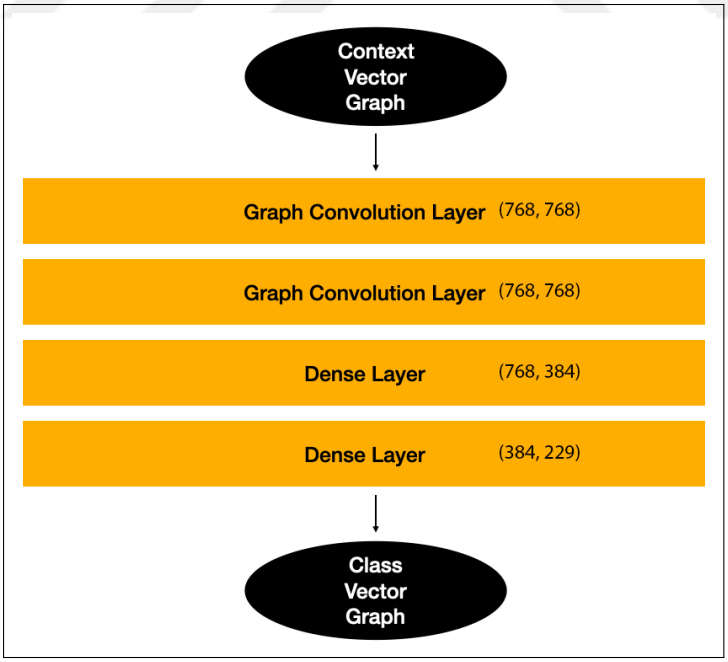

Figure 4.3. Architecture of custom GCN model.

4.2.2. LayoutLMv1

In addition to advancements in field of transformer models, one of the most important studies is LayoutLM. LayoutLM uses BERT model as its base and most important modification on bert is position embedding [8]. LayoutLM replaces positional embedding in traditional transformer model with complex location information [8]. In addition to indexes for sequence instances LayoutLM also provides bounding boxes of words in document to transformer network as position embedding [8].

In order to recognize and comprehend the arrangement of document images, including PDFs, scans, and pictures, LayoutLM is a pre-trained language model [8]. The model is built on top of the well-known BERT architecture [6], but it has been expanded to include layout embedding, which records details on the physical positioning of text and graphics within a document. This enables the model to carry out a variety of document analysis-related tasks, such as word recognition, table detection, and form recognition. Transformer layers plus a layer for integrating layouts make up the LayoutLM architecture [8]. While the layout embedding layer is in charge of encoding the document's spatial organization, the Transformer layers are in charge of learning contextual representations of the text. The model is honed for particular downstream tasks such named entity recognition, key-value extraction, and document categorization after being trained on a sizable corpus of documents [8]. On a number of benchmark datasets, including the FUNSD dataset for form understanding [39] and the DocBank dataset for document understanding [40], LayoutLM has demonstrated state-of-the-art performance. Invoice processing, receipt recognition, and document picture categorization are only a few examples of the tasks for which it has been employed [8]. The capacity of LayoutLM to use both textual and spatial information to enhance document understanding is one of its main advantages. This makes it especially helpful in industries like law, finance, and medicine where the design and organization of the document are crucial.

In Figure 4.4 from original LayoutLM paper, utilization of visual properties of invoices are shown in a diagram. This diagram shows that after transformer model predicts results for sequence as a hidden layer result, visual properties of instances affects results by help of a convolutional feature extractor. Model combines hidden layer resulting vectors with feature vectors to process visual properties of text such as font weight, to use this information

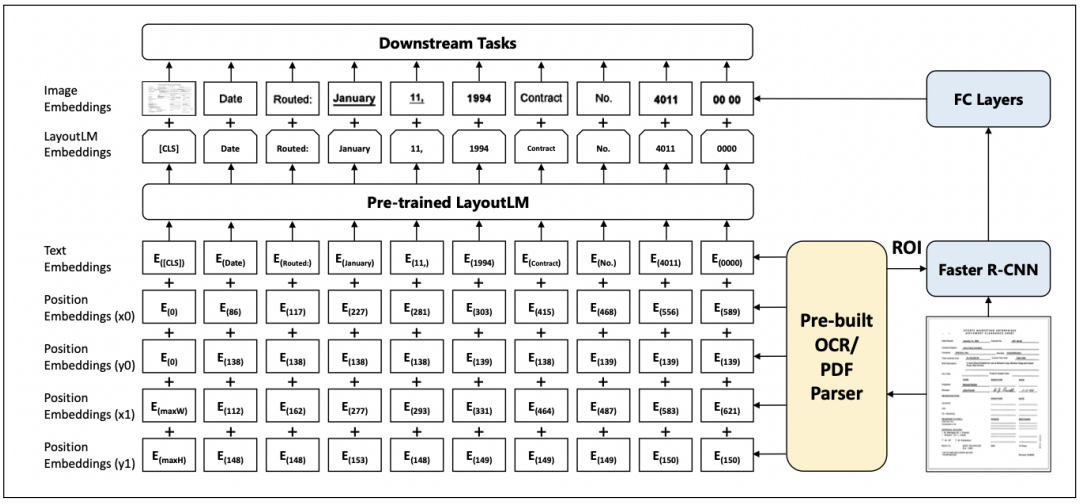

Figure 4.4. Processing of visual properties in LayoutLM [8].

when deciding for class prediction of model. Also from parsed information, it can be seen that model feed first layer of transformer with four position embeddings along with text embeddings.

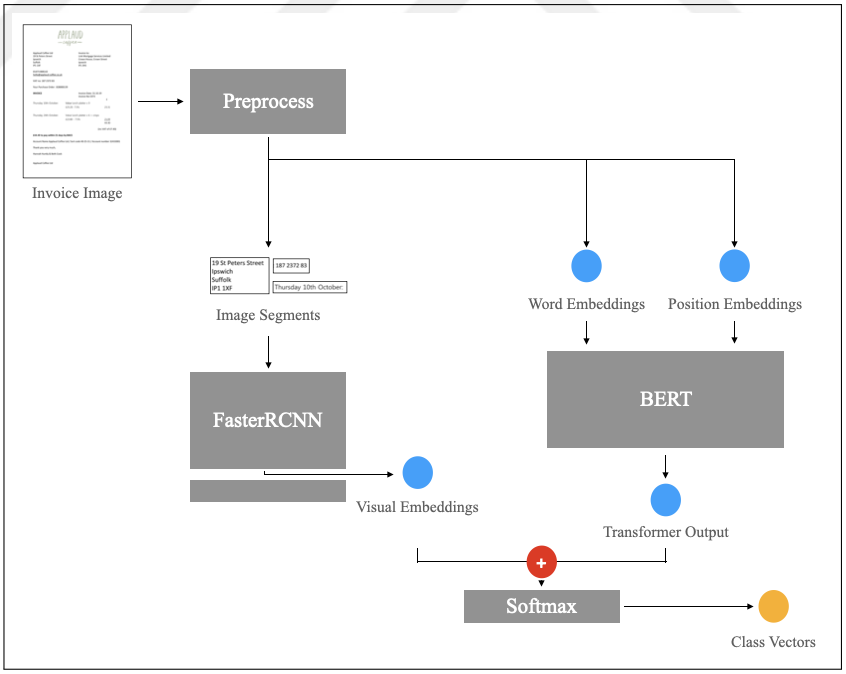

Figure 4.5. LayoutLM architecture.

In addition to this description, more detailed representation of operations are shown in Figure 4.5. Operations are represented as a diagram to show end-to-end calculations for a given input. As seen from figure, operations start with a preprocessing module to construct desired data from invoice images. Preprocess operation gives word and position embeddings using

OCR and then image segments are cropped from invoices image based on word bounding boxes which are provided by OCR. In continuation of calculations, FasterRCNN model is used to extract feature vector from image segments to add to transformer output which is calculated by BERT network based on word and position embeddings. At the end softmax operation rearranges resulting vector to indicate predicted class for given word in sequence [8].

In LayoutLM architecture, position embeddings are calculated similar to transformer architecture but instead of one position embedding to represent order in sequence, LayoutLM also adds four additional position embedding to represent top-left and bottom-right coordinates of word in document image to process layout information [8].

The existing source code of the model was used to customize it for the specific objective of using LayoutLM for information extraction. Model modification was done on final layer of model. Layer redesigned to classify into 228 classes. Other than last layer system kept same. The model was fine-tuned using a dataset of 1000 annotated invoice images, and the training procedure lasted 100 epochs.

4.2.3. LayoutLMv3

In order to increase performance on tasks like information extraction, named entity recognition, and document categorization, LayoutLMv3 is a state-of-art pre-trained model for document understanding tasks [13]. LayoutLMv3 makes various improvements to achieve greater performance on benchmark datasets, building on the success of its predecessors LayoutLM [8] and LayoutLMv2 [41]. Similar to LayoutLMv1, the model expands the original BERT model [6] by incorporating visual embeddings from both the text areas and layout components in a document. It is based on the Transformer architecture [1]. This gives the model the ability to take into account both the spatial and visual interactions between textual pieces inside a document in addition to the text's semantic information. LayoutLMv3 uses a 2D position embedding in addition to the conventional token, segment, and position embeddings to store the spatial position of each token in the document [13].

As one of the major advancements in LayoutLMv3, the VLT, which focuses on learning

the layout and visual structure of a document was introduced [13]. To improve the model's comprehension of the document layout, the VLT takes into account visual elements from the page, such as font size and color. The multi-modal fusion approach used by LayoutLMv3 effectively merges visual and textual information, improving performance on a variety of document understanding tasks [13]. On benchmark datasets like FUNSD, SROIE, and CORD, the LayoutLMv3 model has displayed outstanding results [13]. It is a useful tool for both researchers and practitioners involved in the fields of natural language processing and computer vision because its efficacy has been demonstrated across a variety of document interpretation tasks.

LayoutLMv3 model was fine-tuned using the same collection of 1000 annotated invoice images that were used for the preceding models. Hugging Face's Python transformers package was used to obtain the pre-trained model. To maximize the model's performance, the fine-tuning method included 100 epochs. Model modification done on model by a procedure as same as LayoutLMv1 model.

4.2.4. LayoutLM Version Differences

In order to perform better on tasks like information extraction, named entity recognition, and document categorization, LayoutLMv1 and LayoutLMv3 are both pre-trained models for document comprehension. Although both models are based on the BERT architecture and have a similar framework, there are important distinctions in their strategies and innovations that enhance performance in LayoutLMv3 [13]. By adding visual embeddings from the text regions and layout elements in a document, LayoutLMv1 expands the basic BERT model [8]. Standard token, segment, and position embeddings are included, along with a 2D position embedding to store each token's spatial location within the document. This enables LayoutLMv1 to take into account both the spatial and visual interactions between textual elements inside a document in addition to the semantic content of the text [8]. On the other hand, LayoutLMv3 adds to the improvements made by LayoutLMv2 [41] while also making a number of new ones [13]. The VLT has been incorporated into LayoutLMv3 as one of its major improvements [13]. Through incorporation of visual elements from the document, such as font size and color, the VLT focuses on learning the layout and

visual structure of a document. Compared to LayoutLMv1, this considerably improves the model's comprehension of the page layout. Additionally, LayoutLMv3 uses a multi-modal fusion mechanism that successfully combines visual and textual information, enhancing performance on a variety of document understanding tasks [13].

In Figure 4.6, detailed explanation of operations done during process of model is represented for LayoutLM models. As seen from the diagram, operations starts by scanning invoice images with OCR and extracting text data along with position data. This raw data is converted to sequence of token vectors of text data using WordPiece vectorization method and 2D position embeddings. As continuation of operations, 2D position vectors, 1D position vectors and segment vectors are added to token vectors. After segment vectors added LayoutLMv3 model adds visual embeddings calculated by VLT module which depends on a convolutional network to these vectors. Then obtained input embedding vectors are provided to BERT model.

After BERT model outputs a sequence vectors in size of 768, LayoutLMv1 model adds visual embeddings to these vectors. As last operations on both models sequence of vectors are processed through a classification layer of linear layer to produce sequence of vectors in size 229 to represent 228 subclasses and one additional other class. Then softmax operation is applied over these vectors to produce class index vectors for each token provided to model.

In Figure 4.7 calculation of input embeddings without visual embeddings is represented with a diagram. After OCR scan applied to image, a sequence of words and their bounding boxes are obtained. Then sequence of token vectors are calculated using WordPiece tokenization method. With this method some words some words are represented with more than one token, in this situation bounding box information is shared between word pieces of word. After Sequence of token vectors and normalized bounding boxes obtained, position embeddings are calculated for sequence of indexes to get 1D position embeddings, sequence of sentence indexes for segment embeddings and corner coordinates of bounding box for 2D position embeddings using Equation 4.1. After all embeddings are calculated all of them added to token vector to obtain input embeddings for model. These operations are same for LayoutLMv1 and LayoutLMv3 models. For better understanding input and output system of network, a slice from sequence of word pieces is given with corresponding class names

output from model is given in Table 4.6

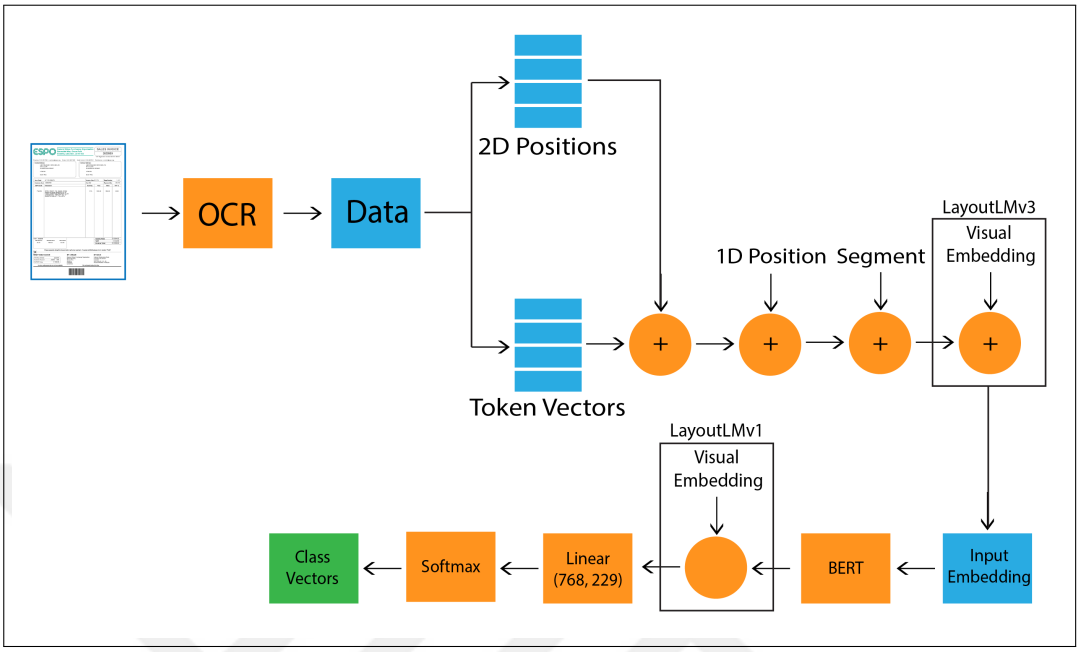

Figure 4.6. Detailed diagram of operations in LayoutLM models.

$$PositionEmbedding(pos) = \begin{cases} sin(\frac{pos}{10000^{\frac{2i}{d_{model}}}}), & \text{if } i \text{ is even} \\ cos(\frac{pos}{10000^{\frac{2i}{d_{model}}}}), & \text{if } i \text{ is odd} \end{cases} \quad (4.1)$$

For both models visual embeddings are calculated using similar method. In Figure 4.8, calculation of visual embedding with FasterRCNN model is represented. This method is the exact method used in LayoutLMv1 model. LayoutLMv3 model adds more layer of process after hidden layer output is obtained from FasterRCNN model instead of single layer fully connected network.

In conclusion, LayoutLMv3 incorporates the Visual Layout Transformer and a multi-modal fusion mechanism, leading to higher performance on benchmark datasets and a larger range of document understanding tasks, while LayoutLMv1 and LayoutLMv3 both extend the BERT model for document understanding tasks.

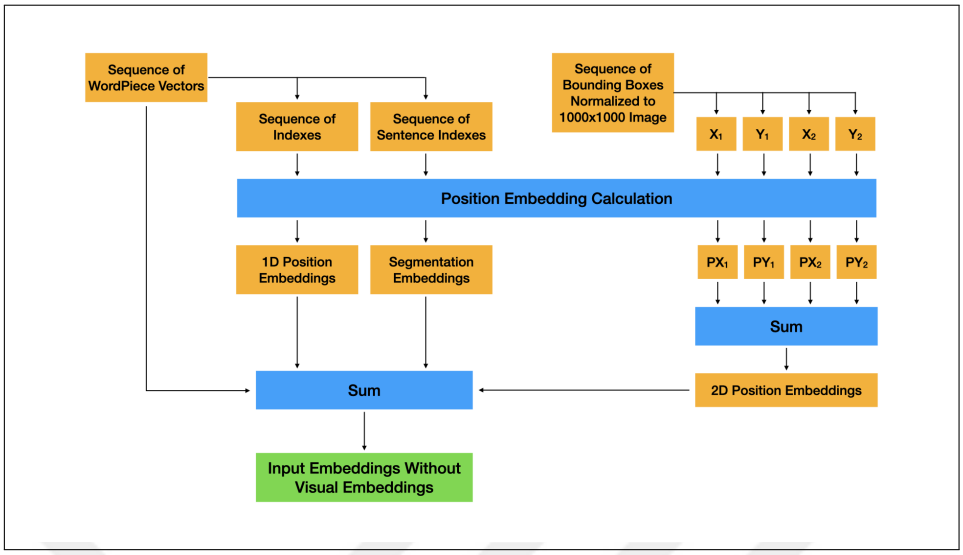

Figure 4.7. Diagram of calculation of input embeddings using sequences of token vectors and bounding boxes.

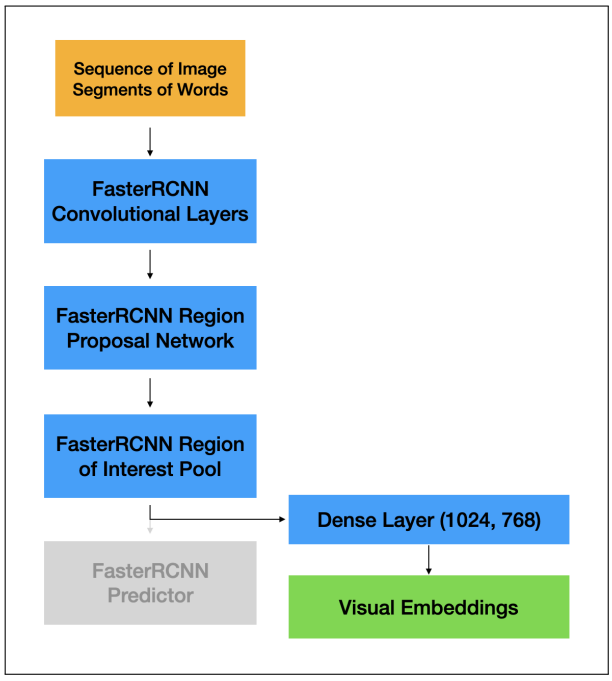

Figure 4.8. Diagram of calculation of visual embeddings.

Table 4.6. Input output pair of model.

Word Pice	**Class Name**
goods	B_invoice_summary_subtotal_label
total	E_invoice_summary_subtotal_label
va	B_invoice_summary_vat_label
##t	E_invoice_summary_vat_label
in	B_invoice_summary_total_label
##vo	I_invoice_summary_total_label
##ice	I_invoice_summary_total_label
total	E_invoice_summary_total_label
£	B_invoice_summary_subtotal
1664	I_invoice_summary_subtotal
.	I_invoice_summary_subtotal
00	E_invoice_summary_subtotal
£	B_invoice_summary_vat
332	I_invoice_summary_vat
.	I_invoice_summary_vat
80	E_invoice_summary_vat
£	B_invoice_summary_total
1996	I_invoice_summary_total
.	I_invoice_summary_total
80	E_invoice_summary_total

5. IMPLEMENTATION

All operations over models were done on a computer with Ubuntu 20.04 OS, 12 Core 3.7 GHz CPU, Nvidia RTX3090 GPU and 64 GB RAM. For creating custom GCN model, torch-geometric library was used along with PyTorch. For LayoutLM models PyTorch and transformers library was used.

For implementation of custom GCN model, GCNConv [30], GENConv [42] and GCN2Conv [43] was tested as graph convolution layers. Since there were no important success difference observed, model implementation continued with GCNConv layer which is the fastest of all three. GCNConv layer is used from torch-geometric library as an extension of message passing class of library. For dense layers used in GCN model, standard Linear layer from pytorch was used.

For the imlementation of LayoutLMv1 model, github repository of model was used to access source code of model. The model which written using pytorch library was modified for sequence labeling task for 58 classes. LayoutLMv3 was obtained from huggingface transformers library. Similar to LayoutLMv1, same modification process was applied to model.

5.1. GRAPH CONSTRUCTION FROM INVOICE IMAGE

As mentioned before, invoice images are required to be preprocessed to construct graph representation of image to feed data to GCN model. To represent image to graph an algorithm was developed to create graph nodes from words and edges based on spatial locations in image. While nodes are calculated as a sequence of word vectors from document, connection between nodes are calculated based on following conditions. If two words are aligned horizontally or vertically and words are neighbours, edge is constructed between nodes of these two words. In Algorithm 5.1, adjacency matrix calculation for graph conversion is explained where "W" is a list of words and "P" is list a of positions in form of [x_0, y_0, x_1, y_1]. Algorithm constructs adjacency matrix of graph to use in initialization of graph object from torch geometric library. Word embedding are calculated using tokenizer module

of BERT model by converting list of words into single string with separation with space. Horizontal-Alignment and Vertical-Alignment procedures check if bounding boxes of two words overlap in an axis more than half of the size of shorter bounding box edge size and if there is no other word between two words.

Algorithm 5.1. Graph construction algorithm.

$W \leftarrow$ the set of all words
$P \leftarrow$ the set of all positions
$N \leftarrow$ the number of words
$A \leftarrow$ the $N \times N$ zero matrix
for $i \leftarrow 1$ to N **do**
    for $j \leftarrow 1$ to N **do**
        if $i \neq j$ **then**
            $(x_0, y_0, x_1, y_1) \leftarrow P[i]$
            $(x'_0, y'_0, x'_1, y'_1) \leftarrow P[j]$
            $h \leftarrow$ HORIZONTAL-ALIGNMENT$(x_0, y_0, x_1, y_1, x'_0, y'_0, x'_1, y'_1)$
            $v \leftarrow$ VERTICAL-ALIGNMENT$(x_0, y_0, x_1, y_1, x'_0, y'_0, x'_1, y'_1)$
            if h or v **then**
                $A[i, j] \leftarrow 1$
            end if
        end if
    end for
end for

5.2. INVOICE LABELING APPLICATION

To use trained LayoutLMv1 in invoice labeling in industry, an application was developed with Intecon Informatics and Consultancy. Structure of developed application is presented at Figure 5.1. Developed application takes invoice images as input. After input images are fed to application, known invoice type detector module takes invoice images and checks if any other invoices were labeled before with same layout. This module simply calculates a feature vector from image using feature extractor layers of MobileNetV3 [44], then compares it with feature vectors which has been calculated from invoices with known layout by using cosine similarity. If there is an invoice from system that has more than 0.99 cosine similarity, invoice skips layout detection and use existing layout information from database. If invoice type is not found, module forwards invoice image to invoice labeling module with LayoutLMv1.

Whether invoice gets labels from existing layout or labeling module, it goes to user control. If invoice labeled with existing layout, relabeling with LayoutLMv1 option is provided to user at this point. Also user can modify labels to correct errors or add new labels for missing parts. After operations done, user saves layout and information of invoice to database.

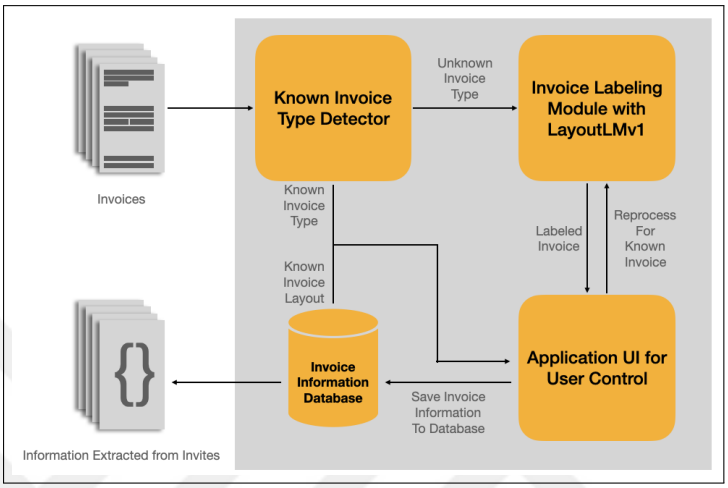

Figure 5.1. Invoice labeling application structure.

In Figure 5.2, a screenshot of application is shown in invoice labeling page. After invoice gets label information from database or labeling module, image of invoice is displayed at the right hand side of the web page. Extracted information is listed on the left hand side of page and indicates location of the fields while focused. Also changing and adding information is provided at this side of page.

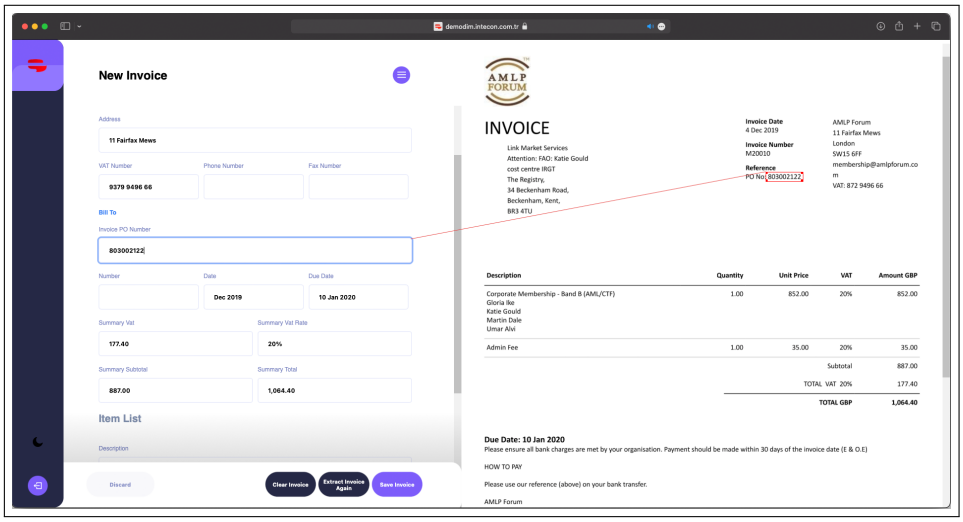

Figure 5.2. Screenshot of application.

6. RESULTS AND DISCUSSION

During the dataset preparation process, the training of LayoutLMv1 was carried out using dataset subsets of 250, 350, 450, 550, 650, and 720 invoices with the aim of practically testing the training success of the dataset model. During these trainings, the success levels of the model on different classes were examined, and the aim was to balance the training by making changes to the dataset. The models created as a result of the trainings were compared based on f1-score, precision, recall, and accuracy values, as shown in the Figure 6.1. As expressed in the graph, while the training success rapidly increased in the initial dataset subsets, the impact of the newly added invoice images on the result decreased as the dataset grew.

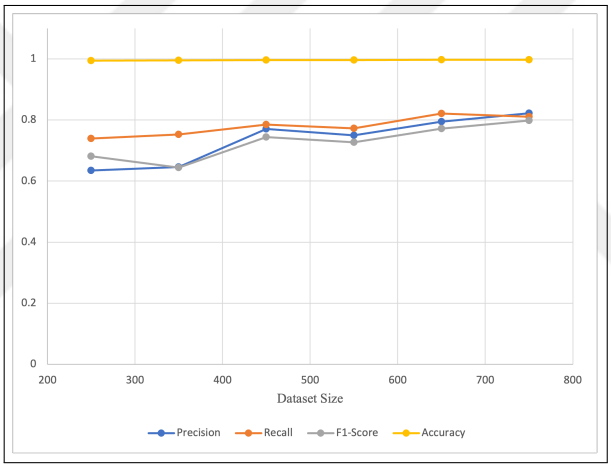

Figure 6.1. Improvement over metrics during dataset enlargement

Due to their structures, models perform classification within a large number of classes, which makes it difficult to evaluate the criteria of these classes. In the study, these classes are grouped relationally to facilitate the comparison process, and the average metric values of the classes belonging to these relational groups are expressed in the figures. In this way, the success of the model on different sections serving different purposes within the invoices can be seen and made comparable. The defined class groups are as follows; the "header" group consists of classes containing general information about the invoice parties and the invoice, such as the supplier's name and invoice number. The "table" group contains classes related to the table where products and services are listed. Classes such as product name and quantity are among the classes in this group. The "VAT Table" group includes classes related to taxes, such as VAT code and VAT rate, while the "Summary" class contains classes related to the

overall total of the invoice.

6.1. CUSTOM GRAPH CONVOLUTIONAL NETWORK

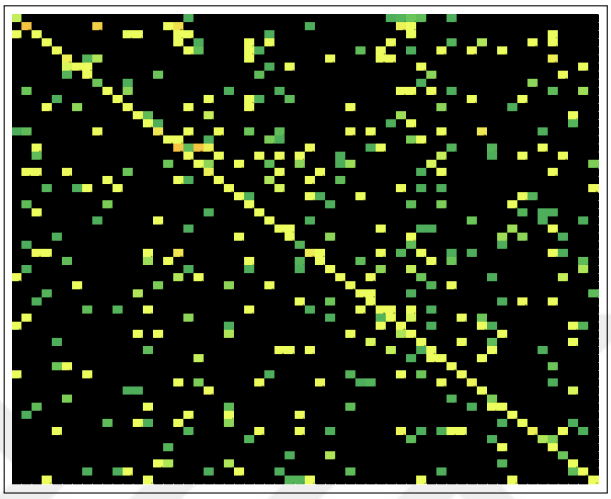

Figure 6.2. Confusion matrix of GCN model on 58 classes

After training the GCN model, it was run on a test set of 250 invoices, and the model's predictions were compared with the correct data, resulting in the confusion matrix shown in the Figure 6.2. As can be seen from the Figure 6.2, although some classes were correctly predicted in some cases, the model generally failed to make accurate predictions. As seen on the confusion matrix, there is no pattern in the model's errors, and the model's performance is quite low. In the Table 6.1, the average precision, recall, f1-score, and accuracy values are specified for the grouped classes. The row labeled as 'All' represents the average of all classes.

Table 6.1. GCN Grouped Metric Averages

Group	Precision	Recall	F1 Score	Accuracy
Header	0.32	0.40	0.30	0.97
Table	0.46	0.41	0.37	0.98
Summary	0.25	0.34	0.21	0.99
VAT Table	0.45	0.39	0.38	0.97
Other	0.45	0.39	0.38	0.97
Average	0.38	0.40	0.32	0.98

In Table 6.2, model metrics are shown for every class in header group. While invoice po

number class achieve 0.87 precision, overall success of model is lower than 0.5 F1-score. For most of the classes model reaches extremely low precision values with recall values around 0.4. As similar to other tables accuracy values are high even if other metrics extremely low.

Table 6.2. GCN metrics for header group classes.

Precision	Recall	F1 Score	Accuracy	Class Name
0.02	0.43	0.03	0.99	INVOICE_CUSTOMER_INFO
0.00	0.33	0.00	0.93	INVOICE_DATE
0.87	0.49	0.63	1.00	INVOICE_PO_NUMBER
0.40	0.39	0.39	1.00	SUPPLIER_PHONE_NUMBER
0.50	0.43	0.47	0.81	INVOICE_DUE_DATE
0.36	0.48	0.41	0.98	INVOICE_NUMBER
0.12	0.43	0.19	0.96	SUPPLIER_FAX_NUMBER
0.08	0.33	0.13	1.00	SUPP_NAME
0.48	0.38	0.43	1.00	SUPPLIER_VAT_NUMBER
0.36	0.33	0.34	1.00	SUPPLIER_ADDRESS

Metrics for classes in in summary group are given in Table 6.3. 0.04 and 0.03 precision values indicates extremely low success rate for invoice summary VAT and invoice summary total classes. On the other hand, evaluation of invoice summary subtotal achieves quite high success with precision value of 0.77 contrary to the overall success of the model. Recall values are not differ much on different classes, this situation makes f1-scores also closely related to precision value.

Table 6.3. GCN metrics for summary group classes.

Precision	Recall	F1 Score	Accuracy	Class Name
0.16	0.31	0.21	1.00	INVOICE_SUMMARY_VAT_RATE
0.04	0.32	0.08	0.99	INVOICE_SUMMARY_VAT
0.03	0.35	0.06	0.96	INVOICE_SUMMARY_TOTAL
0.77	0.36	0.49	1.00	INVOICE_SUMMARY_SUBTOTAL

Metrics for table group classes are shown in Table 6.4. As shown from table while invoice line unitprice class achieves 0.93 precision, invoice line quantity class achieves 0.87 precision value and invoice line item description class achieves 0.89 precision value, other classes in this group can be considered as unacceptable. Even model success is insufficient these results shows that GCN model is successfull over table structured data compared to other

class groups.

Table 6.4. GCN metrics for table group classes.

Precision	Recall	F1 Score	Accuracy	Class Name
0.40	0.33	0.36	0.99	INVOICE_LINE_VAT_RATE
0.93	0.49	0.65	1.00	INVOICE_LINE_UNITPRICE
0.58	0.33	0.42	0.99	INVOICE_LINE_ITEM_NO
0.87	0.46	0.60	1.00	INVOICE_LINE_QUANTITY
0.02	0.32	0.03	0.91	INVOICE_LINE_SUBTOTAL
0.05	0.34	0.09	1.00	INVOICE_LINE_NUMBER
0.21	0.44	0.28	0.99	INVOICE_LINE_VAT
0.89	0.47	0.62	0.97	INVOICE_LINE_ITEM_DESCRIPTION

When Table 6.5 is examined, it can be seen that models success is extremely low on VAT table group classes. This may be the result of low number of samples for these classes in training dataset since usage of VAT table structure in invoices is not frequent.

Table 6.5. GCN metrics for VAT table group classes.

Precision	Recall	F1 Score	Accuracy	Class Name
0.48	0.47	0.48	0.95	INVOICE_VAT_TABLE_CODE
0.71	0.32	0.44	0.94	INVOICE_VAT_TABLE_HEADER
0.09	0.38	0.15	0.99	INVOICE_VAT_TABLE_VAT
0.67	0.33	0.44	0.99	INVOICE_VAT_TABLE_RATE
0.32	0.45	0.38	0.99	INVOICE_VAT_TABLE_SUBTOTAL

Table 6.6 shows metrics for classes other than grouped classes. Some of the classes in Table 6.6 achieves unusually high values for precision considering average metrics of custom GCN model. On the other hand most of the classes are predicted with unacceptable scores. After all groups and classes are evaluated it can reached that results of model does not show any pattern other than slightly better results on classes which its instances are located in tabular structure in document.

6.2. LAYOUTLMV1

Similar to the GCN model, for the LayoutLMv1 model, the same test dataset was used for evaluation after training, and a confusion matrix was created to assess the model's

Table 6.6. GCN metrics for other classes.

Precision	Recall	F1 Score	Accuracy	Class Name
0.97	0.39	0.56	0.87	INVOICE_PAYMENT_INFO
0.06	0.33	0.11	0.98	SUPPLIER_EXTRA_INFO
0.90	0.38	0.54	0.99	INVOICE_LINE_ITEM_NO_L
0.68	0.43	0.53	0.96	VAT_TABLE_SUBTOTAL_L
0.62	0.46	0.53	0.99	OTHER
0.93	0.37	0.53	0.94	SUPPLIER_FAX_NUMBER_L
0.37	0.39	0.38	0.99	INVOICE_SUMMARY_EXTRA_INFO
0.24	0.37	0.29	0.99	INVOICE_VAT_TABLE_CODE_L
0.90	0.43	0.58	0.99	INVOICE_LINE_EXTRA_INFO
0.16	0.38	0.22	1.00	INVOICE_SUMMARY_TOTAL_L
0.21	0.46	0.29	0.99	INVOICE_LINE_NUMBER_L
0.02	0.48	0.04	0.99	INVOICE_FOOTER
0.07	0.43	0.11	1.00	INVOICE_SUMMARY_VAT_L
0.15	0.39	0.22	0.99	INVOICE_DUE_DATE_L
0.59	0.34	0.44	1.00	INVOICE_NUMBER_L
0.01	0.29	0.01	0.99	INVOICE_VAT_TABLE_VAT_L
0.98	0.31	0.47	0.99	INVOICE_DATE_L
0.73	0.50	0.59	1.00	INVOICE_LINE_QUANTITY_L
0.30	0.30	0.30	0.99	INVOICE_SUMMARY_SUBTOTAL_L
0.39	0.46	0.42	0.98	INVOICE_EXTRA_INFO
0.08	0.36	0.13	0.99	INVOICE_PO_NUMBER_L
0.31	0.47	0.37	1.00	INVOICE_VAT_TABLE_RATE_L
0.72	0.44	0.55	0.97	INVOICE_SUMMARY_L
0.08	0.42	0.13	1.00	SUPPLIER_PHONE_NUMBER_L
0.29	0.43	0.35	1.00	INVOICE_LINE_SUBTOTAL_L
0.06	0.35	0.11	0.99	INVOICE_INFO_HEADER
0.24	0.42	0.30	1.00	INVOICE_LINE_VAT_L
0.01	0.41	0.01	0.97	LINE_ITEM_DESCRIPTION_L
0.01	0.31	0.01	0.99	INVOICE_LINE_UNITPRICE_L
0.34	0.47	0.40	0.99	SUPPLIER_VAT_NUMBER_L

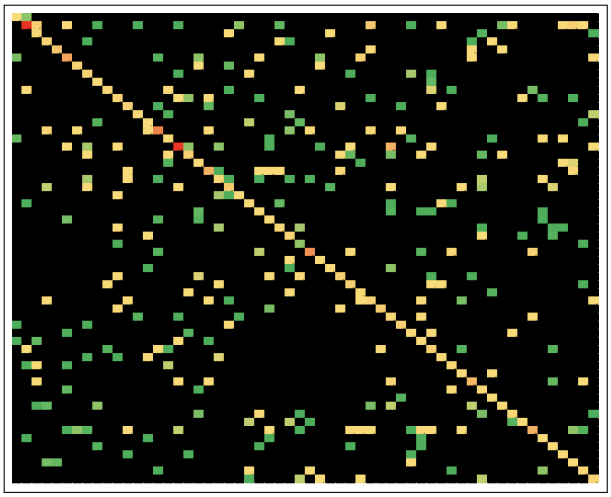

Figure 6.3. Confusion matrix of LayoutLMv1 model on 58 classes

performance for the classes. As seen in the Figure 6.3, the model predictions are mostly concentrated on the main diagonal of the matrix. Although the erroneous predictions show an irregular distribution on the matrix, the total error being considerably less than the correct answers positively affects the criteria used for model evaluation. In Table 6.7 which is created for the class group criteria, it can be seen that the precision, recall, and f1-score values are significantly higher compared to the GCN model. While the model performance remains below its average for the header group, high success has been achieved in the table and summary groups. In addition, the model evaluation cannot be performed based on accuracy. As can be seen by evaluating Table 6.1 and Table 6.7, the accuracy value does not change in a manner related to model success. The reason for this is that the accuracy criterion uses the true negative value in the calculation. Having many classes to be separated in the dataset causes the true negative value used in the model evaluation to be much higher compared to other values. When the accuracy criterion is calculated with this high true negative value, the impact of true positive, false positive, and false negative values on the result decreases significantly, and the models are brought closer to each other in terms of accuracy.

In Table 6.8, metrics for classes in header group are shown. While model shows really good performance on most of the classes, achieves 0.22 and 0.16 F1-score values on supplier name and supplier address classes. This low precision values are showing that model predicts other classes as supplier name and supplier address frequently. When recall values are evaluated it can be seen that values are similar for every class and the average value is acceptable like

Table 6.7. LayoutLMv1 Grouped Metric Averages

Group	Precision	Recall	F1 Score	Accuracy
Header	0.56	0.80	0.61	0.99
Table	0.73	0.79	0.73	1.00
Summary	0.78	0.81	0.79	1.00
VAT Table	0.63	0.80	0.63	0.98
Other	0.63	0.80	0.63	0.98
Average	0.62	0.80	0.65	0.99

precision value. This result shows that model correctly classified most of the instances of these classes in header group.

Table 6.8. LayoutLMv1 metrics for header group classes.

Precision	Recall	F1 Score	Accuracy	Class Name
0.64	0.76	0.70	1.00	INVOICE_CUSTOMER_INFO
0.47	0.75	0.58	1.00	INVOICE_DATE
0.52	0.83	0.64	0.99	INVOICE_PO_NUMBER
0.63	0.75	0.68	1.00	SUPPLIER_PHONE_NUMBER
0.99	0.81	0.89	0.96	INVOICE_DUE_DATE
0.90	0.82	0.86	1.00	INVOICE_NUMBER
0.76	0.78	0.77	0.99	SUPPLIER_FAX_NUMBER
0.13	0.79	0.22	0.99	SUPP_NAME
0.50	0.82	0.62	1.00	SUPPLIER_VAT_NUMBER
0.09	0.84	0.16	0.99	SUPPLIER_ADDRESS

Resulting metrics for classes in summary group are shown in Table 6.9. In this class group model reaches quite high success except invoice summary VAT class with precision of 0.58. For other classes model reaches 0.826 average F1-score value. Due low number of classes in this group, resulting metrics for invoice summary VAT class lowers average F1-score value to 0.79.

Table 6.9. LayoutLMv1 metrics for summary group classes.

Precision	Recall	F1 Score	Accuracy	Class Name
0.93	0.76	0.84	1.00	INVOICE_SUMMARY_VAT_RATE
0.58	0.79	0.67	1.00	INVOICE_SUMMARY_VAT
0.72	0.84	0.77	1.00	INVOICE_SUMMARY_TOTAL
0.88	0.85	0.87	1.00	INVOICE_SUMMARY_SUBTOTAL

When considering group averages, model performs moderate success on table group classes with 0.73 average F1-score value. While averages shows these results, more detailed metric information for classes in this group can be seen in Table 6.10. While model achieves good results for most of the classes success on invoice line VAT rate is moderate with 0.65 F1-score value.

Table 6.10. LayoutLMv1 metrics for table group classes.

Precision	Recall	F1 Score	Accuracy	Class Name
0.56	0.79	0.65	0.99	INVOICE_LINE_VAT_RATE
0.88	0.83	0.85	1.00	INVOICE_LINE_UNITPRICE
0.86	0.79	0.82	1.00	INVOICE_LINE_ITEM_NO
0.96	0.77	0.86	1.00	INVOICE_LINE_QUANTITY
0.75	0.76	0.75	1.00	INVOICE_LINE_SUBTOTAL
0.63	0.83	0.72	1.00	INVOICE_LINE_NUMBER
0.91	0.77	0.83	1.00	INVOICE_LINE_VAT
0.84	0.79	0.82	0.98	INVOICE_LINE_ITEM_DESCRIPTION

As can seen from Table 6.11, model achieves average and above average results for classes of this group with two exception with low 0.52 F1-score value on invoice VAT table subtotal class and extremely low 0.02 F1-score value on invoice VAT table VAT class. This result indicates that model predicts most of the instances of these classes incorrectly. For other classes models results can be considered as accurate predictions.

Table 6.11. LayoutLMv1 metrics for VAT table group classes.

Precision	Recall	F1 Score	Accuracy	Class Name
0.93	0.76	0.84	0.99	INVOICE_VAT_TABLE_CODE
0.97	0.84	0.90	0.99	INVOICE_VAT_TABLE_HEADER
0.05	0.81	0.09	0.96	INVOICE_VAT_TABLE_VAT
0.82	0.80	0.81	1.00	INVOICE_VAT_TABLE_RATE
0.39	0.79	0.52	0.99	INVOICE_VAT_TABLE_SUBTOTAL

In Table 6.12, metrics for other non-grouped classes are shown. As can be seen from table, results are mostly high values for all metrics with a few exceptions as quite low F1-score on some classes. This situation may be an indicator of that the model can be trained to outperform this results by adding new invoices to training dataset with more instances of these classes.

Table 6.12. LayoutLMv1 metrics for other classes.

Precision	Recall	F1 Score	Accuracy	Class Name
0.99	0.80	0.88	0.95	INVOICE_PAYMENT_INFO
0.18	0.77	0.29	0.99	SUPPLIER_EXTRA_INFO
1.00	0.84	0.91	1.00	INVOICE_LINE_ITEM_NO_L
0.98	0.81	0.89	0.99	VAT_TABLE_SUBTOTAL_L
0.73	0.81	0.77	0.99	OTHER
0.99	0.83	0.90	0.98	SUPPLIER_FAX_NUMBER_L
0.93	0.82	0.87	1.00	INVOICE_SUMMARY_EXTRA_INFO
0.92	0.77	0.84	1.00	INVOICE_VAT_TABLE_CODE_L
0.83	0.81	0.82	0.99	INVOICE_LINE_EXTRA_INFO
0.95	0.83	0.88	1.00	INVOICE_SUMMARY_TOTAL_L
0.48	0.84	0.61	1.00	INVOICE_LINE_NUMBER_L
0.11	0.76	0.20	1.00	INVOICE_FOOTER
0.17	0.80	0.28	1.00	INVOICE_SUMMARY_VAT_L
0.61	0.78	0.69	1.00	INVOICE_DUE_DATE_L
0.82	0.82	0.82	1.00	INVOICE_NUMBER_L
0.06	0.71	0.10	1.00	INVOICE_VAT_TABLE_VAT_L
0.73	0.84	0.78	0.99	INVOICE_DATE_L
0.22	0.81	0.35	0.99	INVOICE_LINE_QUANTITY_L
0.28	0.80	0.41	0.98	INVOICE_SUMMARY_SUBTOTAL_L
0.98	0.81	0.89	1.00	INVOICE_EXTRA_INFO
0.24	0.78	0.36	1.00	INVOICE_PO_NUMBER_L
0.40	0.78	0.53	1.00	INVOICE_VAT_TABLE_RATE_L
0.90	0.82	0.86	0.99	INVOICE_SUMMARY_L
0.17	0.83	0.28	1.00	SUPPLIER_PHONE_NUMBER_L
0.58	0.79	0.67	1.00	INVOICE_LINE_SUBTOTAL_L
0.79	0.81	0.80	1.00	INVOICE_INFO_HEADER
0.30	0.79	0.43	1.00	INVOICE_LINE_VAT_L
0.63	0.84	0.72	1.00	LINE_ITEM_DESCRIPTION_L
0.05	0.77	0.09	1.00	INVOICE_LINE_UNITPRICE_L

When the LayoutLMv1 model is evaluated overall, successful results have been obtained in practice. After this success, to test whether the model can be used for commercial purposes as a product, information has been extracted from invoices with and without the help of the model. During the test, 20 different invoices were labeled in the first round as completely unlabeled invoices by the participants on an invoice labeling interface provided by Intecon Informatics and Consultancy. The time spent per invoice was recorded. In the second round of the test, instead of labeling the invoices on the same platform, the participants reviewed the invoices labeled by the model, corrected erroneous results, and entered missing information. In the second round, the time between each invoice being handed to the participant and the participant finishing their work with the invoice was also recorded. Table 6.13 shows the average time spent by the participants for the selected 20 invoices, comparing the time for labeling invoices from scratch and the time for correcting model labels. As can be seen in the Table 6.13, the model predictions have provided an advantage in average times for processing 20 invoices. Although the final results may be limited in interpretation since the tests were conducted with a small dataset of 20 invoices by a few participants, such as a team of 3 people, the study has reduced the average invoice processing time by 46.1%.

6.3. LAYOUTLMV3

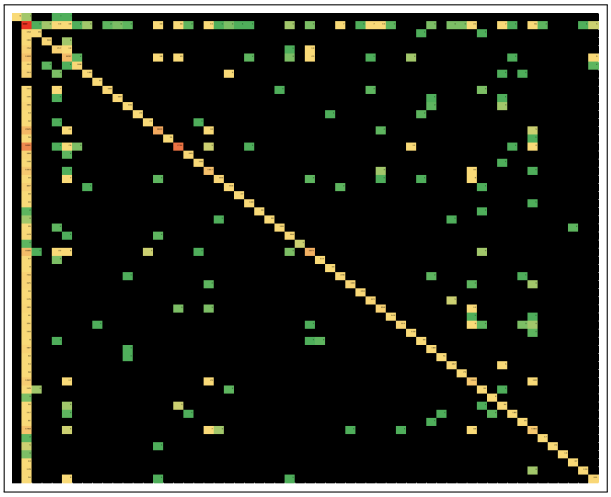

Figure 6.4. Confusion matrix of LayoutLMv3 model on 58 classes

After the fine-tuning process, the LayoutLMv3-based model was subjected to the same tests as the other models, resulting in the obtained Figure 6.4 and Table 6.14. When examining

Table 6.13. Average elapsed time for each invoice

Invoice Number	Invoice Labeling	Error Correction
1605	04:59	02:10
1606	03:33	01:28
1607	04:53	03:25
1615	03:54	02:30
1620	03:01	01:27
1624	05:27	03:27
1627	05:16	02:00
1628	04:25	01:19
1642	04:48	01:43
1643	05:52	03:25
1650	05:26	03:57
1662	04:17	03:12
1667	03:47	02:32
1670	03:37	01:01
1681	04:28	02:13
1698	04:04	02:39
1709	05:28	04:23
1711	05:14	02:37
1715	04:24	01:59
1720	03:57	01:35

the confusion matrices, it can be seen that the predictions of LayoutLMv3 are mainly concentrated on the main diagonal, similar to LayoutLMv1. However, it is observed that the model frequently makes incorrect predictions, especially in the false positive direction and secondarily in the false negative direction, when evaluating the class named "Invoice Payment Info". As can be seen from the Table 6.14, this situation significantly reduces the overall recall value of the model and decreasing the model's average success. When the classes excluding the invoice payment info class are evaluated, it is found that the model has very few erroneous predictions. When these results are evaluated, model outperforms other models on this task. To obtain more consistent results, the model's training dataset can be enriched with invoices containing examples from the invoice payment info class. No significant inference can be made when examining the model's success by looking at the average success of class groups in the Table 6.14. It can be said that the success of the model is homogeneously distributed over the invoice regions. While the precision value is high, the reason for the recall and f1-score values being significantly lower compared to the precision value is the high number of false negative predictions.

Table 6.14. LayoutLMv3 Grouped Metric Averages

Group	**Precision**	**Recall**	**F1 Score**	**Accuracy**
Header	0.98	0.68	0.79	0.99
Table	0.94	0.55	0.68	0.99
Summary	0.96	0.56	0.70	1.00
VAT Table	0.97	0.62	0.73	0.99
Other	0.97	0.62	0.73	0.99
Average	0.95	0.61	0.72	0.99

In Table 6.15, metrics for classes in header group for LayoutLMv3 model are shown. Success of the model on classes of this group is outstanding with 0.79 F1-score value with exceptionally low F1-score value of 0.64 on supplier phone number class. For precision value model reaches extremely high values of average 0.98 and minimum value as 0.92. With average recall value of 0.68, success rate differs on classes in group and invoice due date, supplier phone number, invoice number and supplier fax number classes falls behind average success in terms of recall.

Resulting metrics for summary group classes are average compared to overall success of

Table 6.15. LayoutLMv3 metrics for header group classes.

Precision	Recall	F1 Score	Accuracy	Class Name
1.00	0.71	0.83	1.00	INVOICE_CUSTOMER_INFO
0.92	1.00	0.96	1.00	INVOICE_DATE
0.99	0.74	0.85	1.00	INVOICE_PO_NUMBER
1.00	0.47	0.64	1.00	SUPPLIER_PHONE_NUMBER
0.99	0.56	0.71	0.91	INVOICE_DUE_DATE
1.00	0.48	0.65	0.99	INVOICE_NUMBER
1.00	0.53	0.69	0.99	SUPPLIER_FAX_NUMBER
0.94	0.83	0.88	1.00	SUPP_NAME
0.98	0.60	0.74	1.00	SUPPLIER_VAT_NUMBER
0.98	0.84	0.90	1.00	SUPPLIER_ADDRESS

model as shown in Table 6.16. As an exception to other classes of the group, results of invoice summary total class are 0.38 recall value and 0.53 F1-score value. This results indicates that most of the instances belongs to this class are classified correctly but some other instances are labeled as invoice summary total class frequently.

Table 6.16. LayoutLMv3 metrics for summary group classes.

Precision	Recall	F1 Score	Accuracy	Class Name
1.00	0.57	0.73	1.00	INVOICE_SUMMARY_VAT_RATE
0.98	0.85	0.91	1.00	INVOICE_SUMMARY_VAT
0.86	0.38	0.53	1.00	INVOICE_SUMMARY_TOTAL
0.99	0.46	0.63	1.00	INVOICE_SUMMARY_SUBTOTAL

When results are evaluated for classes which belongs to table group, extremely high precision values and low recall values are observed from Table 6.17. This can be interpreted as that most of the predictions made by model for this classes are correct but model missing most of the instances for these classes. Due to similar and high precision values for all classes in this group, f1-score values are proportional to recall values of these classes.

Evaluation of classes in VAT table group shows that model achieves results at the average level of general success of the model. Success on precision values for all classes of group are similar and quite high while recall value is moderate on group average. This indicates that model frequently predicts instances as classes of this group incorrectly.

In Table 6.19, metrics for non-grouped classes are shown. Table shows that models success

Table 6.17. LayoutLMv3 metrics for table group classes.

Precision	Recall	F1 Score	Accuracy	Class Name
0.98	0.65	0.78	1.00	INVOICE_LINE_VAT_RATE
0.98	0.62	0.76	1.00	INVOICE_LINE_UNITPRICE
0.97	0.27	0.42	0.99	INVOICE_LINE_ITEM_NO
0.99	0.70	0.82	1.00	INVOICE_LINE_QUANTITY
0.99	0.64	0.78	1.00	INVOICE_LINE_SUBTOTAL
0.97	0.80	0.88	1.00	INVOICE_LINE_NUMBER
0.90	0.43	0.58	1.00	INVOICE_LINE_VAT
0.89	0.43	0.58	0.97	INV_LINE_ITEM_DESCRIPTION

Table 6.18. LayoutLMv3 metrics for VAT table group classes.

Precision	Recall	F1 Score	Accuracy	Class Name
0.96	0.58	0.73	0.98	INVOICE_VAT_TABLE_CODE
0.99	0.58	0.73	0.97	INVOICE_VAT_TABLE_HEADER
0.98	0.75	0.85	1.00	INVOICE_VAT_TABLE_VAT
0.95	0.23	0.38	0.99	INVOICE_VAT_TABLE_RATE
0.97	0.93	0.95	1.00	INVOICE_VAT_TABLE_SUBTOTAL

is mostly above average for these non-grouped classes but recall values frequently falls to medium values. When the model evaluated generally, resulting metrics for precision are generally outstanding. On the other hand recall values for most of the classes are dramatically lower than precision and resulting recall metrics of LayoutLMv1 model.

6.4. MODEL COMPARISON

After the training of all models was completed, the obtained metrics were used for comparing the models. In the Figure 6.5, the average precision, recall, f1-score, and accuracy values are shown for all three models. When comparing based on precision value, as seen in the Figure 6.5, LayoutLMv3 has a higher success with a value of 0.948. When considering the recall value, the LayoutLMv3 model falls behind the LayoutLMv1 model with 0.611 recall value compared to 0.800. When evaluating based on the F1-score, which is the harmonic mean of precision and recall values, LayoutLMv3, which is the most successful model for precision, is also the most successful model in this comparison. The GCN model is the least

Table 6.19. LayoutLMv3 metrics for other classes.

Precision	Recall	F1 Score	Accuracy	Class Name
0.38	0.98	0.55	0.65	PAYMENT_INFO
0.94	0.37	0.53	1.00	SUPPLIER_EXTRA
0.93	0.48	0.63	0.99	LINE_ITEM_NO_L
0.91	0.56	0.69	0.97	VAT_TABLE_SUBTOTAL_L
0.97	0.28	0.44	0.99	OTHER
0.98	0.59	0.73	0.96	SUPPLIER_FAX_NUMBER_L
0.99	0.41	0.58	0.99	INVOICE_SUMMARY_EXTRA_INFO
0.99	0.87	0.93	1.00	INVOICE_VAT_TABLE_CODE_L
0.97	0.28	0.43	0.99	INVOICE_LINE_EXTRA_INFO
0.96	0.21	0.35	1.00	INVOICE_SUMMARY_TOTAL_L
0.98	0.78	0.87	1.00	INVOICE_LINE_NUMBER_L
1.00	0.91	0.95	1.00	INVOICE_FOOTER
1.00	0.87	0.93	1.00	INVOICE_SUMMARY_VAT_L
0.99	0.83	0.91	1.00	INVOICE_DUE_DATE_L
0.93	0.49	0.64	1.00	INVOICE_NUMBER_L
1.00	0.71	0.83	1.00	INVOICE_VAT_TABLE_VAT_L
0.97	0.27	0.42	0.99	INVOICE_DATE_L
0.99	0.83	0.91	1.00	INVOICE_LINE_QUANTITY_L
0.98	0.72	0.83	1.00	INVOICE_SUMMARY_SUBTOTAL_L
0.97	0.62	0.76	0.99	INVOICE_EXTRA_INFO
0.98	0.44	0.61	1.00	INVOICE_PO_NUMBER_L
0.96	0.54	0.69	1.00	INVOICE_VAT_TABLE_RATE_L
0.89	0.53	0.67	0.98	INVOICE_SUMMARY_L
0.82	0.75	0.78	1.00	SUPPLIER_PHONE_NUMBER_L
0.75	0.49	0.59	1.00	INVOICE_LINE_SUBTOTAL_L
0.84	0.30	0.44	1.00	INVOICE_INFO_HEADER
0.89	0.89	0.89	1.00	INVOICE_LINE_VAT_L
1.00	0.81	0.90	1.00	LINE_ITEM_DESCRIPTION_L
1.00	0.77	0.87	1.00	INVOICE_LINE_UNITPRICE_L

successful model in terms of F1-score, while the LayoutLMv1 model is ranked second with a small difference than LayoutLMv3 model, due to its low precision value. When considering the accuracy value, results of models are similar with ranking of LayoutLMv1, LayoutLMv3, and GCN; however, as mentioned before, having a large number of classes increases the true negative value, reducing the range of possible accuracy values and making it difficult to compare and evaluate model success based on this value. Taking this into account, using the F1-score, which is affected by precision and recall values, will provide a more accurate result for comparing the models.

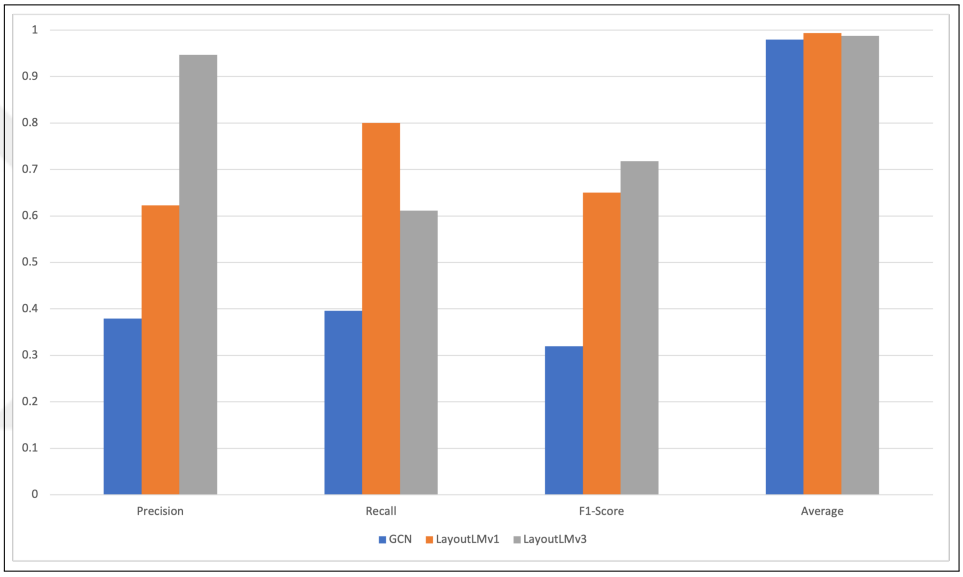

Figure 6.5. Comparison of precision, recall, f1-sore and accuracy over all three models

7. CONCLUSION AND FUTURE WORK

In this study, the automation of information extraction from scanned invoice documents for digital processing using LayoutLMv1, LayoutLMv3, and custom-made GCN deep learning models was tested in terms of model success and the potential for conversion to a commercial product. For information extraction automation, the models were trained to classify words on the document using both textual information obtained from document images read by OCR and visual information in the images.

After the training, tests showed that transformer-based LayoutLMv1 and LayoutLMv3 models achieved promising results, while the GCN model, which aimed to make inferences based on the graph of word vectors which connected based on word positions on the document, was insufficient. Among the three models, the most successful one was LayoutLMv3. It was found that the time required to correct the errors in the invoices predicted and labeled by LayoutLMv1 model, which performs very similar to LayoutLMv3 model, was approximately halved to the time needed for labeling document from the scratch.

As can be seen from the results, transformer-based models are suitable structures for the purpose of this study. The work done in this study can be advanced and used as an aid in invoice labeling processes. Different methods can be proposed for advancing the study. The first of these methods is to update the dataset and train the LayoutLMv1 and LayoutLMv3 models, thereby obtaining a low-error model capable of performing end-to-end information extraction from images. In addition, it is possible to conduct studies on automating the desired corrections by post-processing the outputs of the models.

REFERENCES

1. Palm RB, Laws F, Winther O. Attend, copy, parse end-to-end information extraction from documents. *2019 International Conference on Document Analysis and Recognition (ICDAR)*. IEEE. 2019:329-36.

2. Wei M, He Y, Zhang Q. Robust layout-aware IE for visually rich documents with pre-trained language models. *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 2020:2367-76.

3. Yu W, Lu N, Qi X, Gong P, Xiao R. PICK: processing key information extraction from documents using improved graph learning-convolutional networks. *2020 25th International Conference on Pattern Recognition (ICPR)*. IEEE. 2021:4363-70.

4. Yang X, Yumer E, Asente P, Kraley M, Kifer D, Lee Giles C. Learning to extract semantic structure from documents using multimodal fully convolutional neural networks. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017:5315-24.

5. Aslan E, Karakaya T, Unver E, Akgül YS. A Part based Modeling Approach for Invoice Parsing. *VISIGRAPP (3: VISAPP)*. 2016:392-9.

6. Wang A, Cho K. BERT has a mouth, and it must speak: BERT as a Markov random field language model. *arXiv preprint arXiv:190204094*. 2019.

7. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, et al. Attention is all you need. *Advances in neural information processing systems*. 2017;30.

8. Xu Y, Li M, Cui L, Huang S, Wei F, Zhou M. Layoutlm: Pre-training of text and layout for document image understanding. *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 2020:1192-200.

9. Hwang W, Yim J, Park S, Yang S, Seo M. Spatial dependency parsing for semi-structured document information extraction. *arXiv preprint arXiv:200500642*. 2020.

10. LeCun Y, Bengio Y, Hinton G. Deep learning. *nature*. 2015;521(7553):436-44.

11. Xu S, Song Y, Hao X. A Comparative Study of Shallow Machine Learning Models and Deep Learning Models for Landslide Susceptibility Assessment Based on Imbalanced Data. *Forests*. 2022;13(11):1908.

12. Goodfellow I, Bengio Y, Courville A. *Deep learning*. MIT press; 2016.

13. Huang Y, Lv T, Cui L, Lu Y, Wei F. Layoutlmv3: Pre-training for document ai with unified text and image masking. *Proceedings of the 30th ACM International Conference on Multimedia*. 2022:4083-91.

14. Zhang S, Tong H, Xu J, Maciejewski R. Graph convolutional networks: a comprehensive review. *Computational Social Networks*. 2019;6(1):1-23.

15. Skalickỳ M, Šimsa Š, Uřičář M, Šulc M. Business Document Information Extraction: Towards Practical Benchmarks. *Experimental IR Meets Multilinguality, Multimodality, and Interaction: 13th International Conference of the CLEF Association, CLEF 2022, Bologna, Italy, September 5–8, 2022, Proceedings*. Springer. 2022:105-17.

16. Islam N, Islam Z, Noor N. A survey on optical character recognition system. *arXiv preprint arXiv:171005703*. 2017.

17. Danner M, Maurer B, Schuh S, Greff T, Werth D. Invoice Automation: Increasing Efficiency in the Office at Satherm GmbH Using Artificial Intelligence. *Digitalization Cases Vol 2: Mastering Digital Transformation for Global Business*. 2021:45-60.

18. Guo H, Qin X, Liu J, Han J, Liu J, Ding E. Eaten: Entity-aware attention for single shot visual text extraction. *2019 International Conference on Document Analysis and Recognition (ICDAR)*. IEEE. 2019:254-9.

19. Yindumathi K, Chaudhari SS, Aparna R. Analysis of image classification for text extraction from bills and invoices. *2020 11th International Conference on Computing, Communication and Networking Technologies (ICCCNT)*. IEEE. 2020:1-6.

20. Riba P, Dutta A, Goldmann L, Fornés A, Ramos O, Lladós J. Table detection in invoice documents by graph neural networks. *2019 International Conference on Document Analysis and Recognition (ICDAR)*. IEEE. 2019:122-7.

21. Palm RB, Winther O, Laws F. Cloudscan-a configuration-free invoice analysis system using recurrent neural networks. *2017 14th IAPR International Conference on Document Analysis and Recognition (ICDAR)*. vol. 1. IEEE. 2017:406-13.

22. Raoui-Outach R, Million-Rousseau C, Benoit A, Lambert P. Deep Learning for automatic sale receipt understanding. *2017 Seventh International Conference on Image Processing Theory, Tools and Applications (IPTA)*. IEEE. 2017:1-6.

23. Rusk N. Deep learning. *Nature Methods*. 2016;13(1):35-5.

24. Dong S, Wang P, Abbas K. A survey on deep learning and its applications. *Computer Science Review*. 2021;40:100379.

25. Gu J, Wang Z, Kuen J, Ma L, Shahroudy A, Shuai B, et al. Recent advances in convolutional neural networks. *Pattern recognition*. 2018;77:354-77.

26. Ciregan D, Meier U, Schmidhuber J. Multi-column deep neural networks for image classification. *2012 IEEE conference on computer vision and pattern recognition*. IEEE. 2012:3642-9.

27. Medsker LR, Jain L. Recurrent neural networks. *Design and Applications*. 2001;5:64-7.

28. Floridi L, Chiriatti M. GPT-3: Its nature, scope, limits, and consequences. *Minds and Machines*. 2020;30:681-94.

29. Ahmed K, Keskar NS, Socher R. Weighted transformer network for machine translation. *arXiv preprint arXiv:171102132*. 2017.

30. Kipf TN, Welling M. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:160902907*. 2016.

31. Nagar O, Frydman S, Hochman O, Louzoun Y. Quadratic GCN for graph classification. *arXiv preprint arXiv:210406750*. 2021.

32. Zhuang Z, Liu M, Cutkosky A, Orabona F. Understanding adamw through proximal methods and scale-freeness. *arXiv preprint arXiv:220200089*. 2022.

33. Zhang Z. Improved adam optimizer for deep neural networks. *2018 IEEE/ACM 26th*

international symposium on quality of service (IWQoS). Ieee. 2018:1-2.

34. Sokolova M, Lapalme G. A systematic analysis of performance measures for classification tasks. *Information processing & management*. 2009;45(4):427-37.

35. Davis J, Goadrich M. The relationship between Precision-Recall and ROC curves. *Proceedings of the 23rd international conference on Machine learning*. 2006:233-40.

36. Japkowicz N, Shah M. *Evaluating learning algorithms: a classification perspective*. Cambridge University Press; 2011.

37. Boughorbel S, Jarray F, El-Anbari M. Optimal classifier for imbalanced data using Matthews Correlation Coefficient metric. *PloS one*. 2017;12(6):e0177678.

38. Wu Y, Schuster M, Chen Z, Le QV, Norouzi M, Macherey W, et al. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:160908144*. 2016.

39. Jaume G, Ekenel HK, Thiran JP. Funsd: A dataset for form understanding in noisy scanned documents. *2019 International Conference on Document Analysis and Recognition Workshops (ICDARW)*. vol. 2. IEEE. 2019:1-6.

40. Li M, Xu Y, Cui L, Huang S, Wei F, Li Z, et al. DocBank: A benchmark dataset for document layout analysis. *arXiv preprint arXiv:200601038*. 2020.

41. Xu Y, Xu Y, Lv T, Cui L, Wei F, Wang G, et al. Layoutlmv2: Multi-modal pre-training for visually-rich document understanding. *arXiv preprint arXiv:201214740*. 2020.

42. Li G, Xiong C, Thabet A, Ghanem B. Deepergcn: All you need to train deeper gcns. *arXiv preprint arXiv:200607739*. 2020.

43. Chen M, Wei Z, Huang Z, Ding B, Li Y. Simple and deep graph convolutional networks. *International conference on machine learning*. PMLR. 2020:1725-35.

44. Koonce B, Koonce B. MobileNetV3. *Convolutional Neural Networks with Swift for Tensorflow: Image Recognition and Dataset Categorization*. 2021:125-44.