ΟΙΚΟΝΟΜΙΚΟ ΠΑΝΕΠΙΣΤΗΜΙΟ ΑΘΗΝΩΝ

ATHENS UNIVERSITY OF ECONOMICS AND BUSINESS

School of Information Sciences and Technology
Department of Informatics
Athens, Greece

Master Thesis
in
Data Science

Automatic detection of sections and paragraphs in legal documents

Christos Vlachos

Academic Supervisor: Prof. Ion Androutsopoulos
Department of Informatics
Athens University of Economics and Business

Supervisors: Mr. Manolis Papageorgiou
Cognitiv Plus

Mr. Delio D'Anna
Cognitiv Plus

November 2022

Christos Vlachos

Automatic detection of sections and paragraphs in legal documents

November 2022

Supervisor: Prof. Ion Androutsopoulos

Athens University of Economics and Business

School of Information Sciences and Technology

Department of Informatics

Information Processing Laboratory, Natural Language Processing Group

Athens, Greece

Abstract

Document analysis is a procedure focusing on the processing of documents in order to extract details concerning information such as transactions, involved parties, objectives etc. It is a topic most relevant in the modern society due to the increasing need for fast, easy and clear understanding of the content of documents, as is the case with legal documents. Nevertheless, the incorporation of Deep Learning (DP) techniques that could simplify procedures such as layout analysis, section classification, etc., has only been a topic of relatively recent origin [Yih+20]. Up to that point, document analysis was mostly carried out either manually or by the use of predefined rules. This project aims to broaden the research on the field while also providing its services to Cognitiv+, a company carrying out legal document analysis. Thus, this project is split into two tasks. The first task pertains to the detection of paragraphs and titles, called text blocks, in images of legal documents, using Computer Vision and Natural Language Processing (NLP) techniques. For the purpose of the task, focus will be given on two Computer Vision models, namely YOLOv5 and RetinaNet, that will be trained to detect regions of text blocks. After that, a post-processing phase will be included in order to further classify these texts as either "paragraph" or "title", by means of a simple rule-based NLP approach. The best model will be selected to work along a similar one that is trained to detect tables and is developed by the company mentioned. As per the second task, it revolves around the classification of document text zones (not to be confused with text blocks) as one of the classes "Table of contents", "Recitals", "Cover page", "Introduction" or "Main body", by utilizing their textual content. This task can exist stand-alone or in combination with the first one. For the completion of the task two NLP models will be trained. These models are a RoBERTa and a hierarchical model. The hierarchical model will be based on the RoBERTa output and will try to further improve its results by adding a second model on top. In this thesis, insight over the data used, methodologies and results will be given along with an overview of each one's strengths and weaknesses. Finally, further possible improvements or alternatives will be discussed.

Keywords
Document Analysis; Computer Vision; Natural Language Processing; Deep Learning; Object Detection

Περίληψη

Η ανάλυση εγγράφων είναι μια διαδικασία που εστιάζει στην επεξεργασία εγγράφων, με σκοπό να εξάγει λεπτομέρειες που σχετίζονται με πληροφορίες όπως συναλλαγές, εμπλεκόμενα μέλη, στόχους κ.λπ. Πρόκειται για ένα θέμα, όλο και ποιο επίκαιρο στη σύγχρονη κοινωνία, λόγω της αυξανόμενης ανάγκης για μια γρήγορη, εύκολη και σαφή επεξήγηση των περιεχομένων τους, όπως στη περίπτωση των νομικών εγγράφων. Όμως, η χρήση τεχνικών Βαθιάς Μάθησης, οι οποίες θα μπορούσαν να απλοποιήσουν διαδικασίες όπως η ανάλυση διάταξης, η ταξινόμηση τομέων, κ.λπ., αποτελεί ένα σχετικά πρόσφατο θέμα [Yih+20]. Μέχρι πρότινος, η ανάλυση εγγράφων γινόταν κατά κύριο λόγο χειρωνακτικά ή μέσω χρήσης προκαθορισμένων κανόνων. Η εργασία αυτή στοχεύει στην διεύρυνση του συγκεκριμένου πεδίου έρευνας προσφέροντας ταυτόχρονα τις υπηρεσίες της στην Cognitiv+, μια εταιρία που πραγματοποιεί ανάλυση νομικών εγγράφων. Η εργασία είναι χωρισμένη σε δύο μέρη. Το πρώτο αφορά τον εντοπισμό παραγράφων και τίτλων σε εικόνες νομικών εγγράφων με τη χρήση τεχνικών Υπολογιστικής Όρασης και Επεξεργασίας Φυσικής Γλώσσας. Για τους σκοπούς του, έμφαση θα δοθεί σε δύο μοντέλα Υπολογιστικής Όρασης, συγκεκριμένα στα YOLOv5 και RetinaNet, τα οποία θα εκπαιδευτούν στον εντοπισμό συγκεκριμένων περιοχών κειμένου. Μετά τη διαδικασία εντοπισμού θα ακολουθήσει ένα βήμα μετα-επεξαργασίας κατά το οποίο οι περιοχές κειμένου θα ταξινομηθούν επιπλέον σε μία από τις κατηγορίες "παράγραφος" ή "τίτλος", μέσω εφαρμογή μία απλής προσέγγισης Επεξεργασίας Φυσικής Γλώσσας με τη χρήση κανόνων. Το καλύτερο μοντέλο θα κληθεί να συνεργαστεί με ένα δεύτερο μοντέλο που κατασκευάστηκε από την ίδια την εταιρία με σκοπό τον εντοπισμό πινάκων. Το δεύτερο μέρος σχετίζεται με την ταξινόμηση ζωνών κειμένων (το οποίο δεν πρέπει να συγχέεται με τις περιοχές κειμένου) σε μία από τις κατηγορίες "Table Of contents", "Recitals", "Cover page", "Introduction" ή "Main body". Αυτό το μέρος μπορεί να λειτουργήσει σε συνεργασία με τα μοντέλα που προαναφέρθηκαν ή ανεξάρτητα. Για την επίτευξη του, δύο μοντέλα Επεξεργασίας Φυσικής Γλώσσας θα εκπαιδευτούν. Αυτά τα μοντέλα περιλαμβάνουν το RoBERTa καθώς και ένα ιεραρχικό. Το ιεραρχικό μοντέλο θα βασιστεί στις προβλέψεις του RoBERTa και θα προσπαθήσει να βελτιώσει περεταίρω τα αποτελέσματα του, συνδυάζοντάς τες με ένα επιπλέον μοντέλο. Στη διπλωματική αυτή, λεπτομέρειες θα δοθούν σχετικά με τα δεδομένα που χρησιμοποιήθηκαν, τη μεθοδολογία που ακολουθήθηκε και τα αποτελέσματα του εκάστοτε μοντέλου, σε

συνδυασμό με τις δυνατότητές τους. Τέλος, επιπλέον πιθανές βελτιώσεις ή εναλλακτικές θα προταθούν.

Λέξεις Κλειδιά
Ανάλυση Εγγράφων· Υπολογιστική Όραση· Επεξεργασία Φυσικής Γλώσσας· Βαθειά Μάθηση· Ανίχνευση Αντικειμένων

Acknowledgements

First and foremost, I would like to thank my academic supervisor, Professor Ion Androutsopoulos, Cognitiv+'s supervisors, Mr. Manolis Papageorgiou and Mr. Delio D'Anna and the director of the company, Mr. Vassilis Tsolis, for their active support throughout the capstone. Their suggestions, advice and resources provided, were truly invaluable, being the catalysts for the completion of this capstone. I would also like to thank Cognitiv+'s employees, who wholeheartedly offered their help with the company's equipment and data, provided advice of their own and made me feel as a member of the company myself. Finally, I would like to thank my loving family and friends for their constant support and care, helping me achieve one more milestone.

Contents

1 Introduction

1.1 Motivation and Problem Statement

With the vast increase of data that is observed in the last years, the use of Artificial Intelligence (AI) is more relevant than ever. AI can be used as a solution for the extraction of meaningful information from data, but also as a facilitation for cumbersome tasks that were previously carried out manually. The rise of AI in combination with the benefits that come with its use found their way in almost every field and document analysis is no exception. Tasks relevant to document analysis include document segments classification, layout detection, section detection etc. [She+21]. Until relatively recently, these tasks were mostly carried out manually or using rule-based approaches as there weren't enough studies providing advanced solutions at the time. With the recent advances of NLP and Computer Vision the introduction of Deep Learning in document analysis was made possible.

Cognitiv+ is a company involved in this field with many contributions and ongoing projects, focusing on the legal domain. One of the ongoing projects is a service that provides visual exploration of documents to users. For the purpose of the task, a backbone Deep Learning model is needed, trained to detect paragraphs and titles, referred to as text blocks, that will work in combination with an already implemented component, that is able to detect tables. This model is to focus on the detection of the layout of the text blocks rather than the context of each one, since the model needs to be language agnostic. Given a document, it needs to output coordinates linking to the regions mentioned. Lastly, the model, since it is to be integrated to a live service, needs to be as lightweight, in terms of memory usage, and fast as possible without undermining accuracy. A second objective was also added later on, requiring an NLP model able to classify specific text zones (text zones are comprised of one or more text blocks) stemming from legal documents. These zones include the cover page, introduction, table of contents, recitals and main body of the documents. Contrary to the first task, the NLP models will focus on English documents only, since they are to be further tested, using the output of a model implemented by the company. The existing company's model is aimed towards text zone detection using regular expressions, and will provide the input that will later be classified by the NLP models that are presented in this thesis.

For the purpose of the first task and keeping in mind its requirements and requests of the company, focus will be given on a specific type of DL pretrained model, called one-stage detector, that mostly favors speed rather than accuracy, while still maintaining good performance, such as the YOLOv5 and RetinaNet models, but a baseline using alternative techniques will also be developed for reference purposes. In order to train these models, a dataset is provided by the company including annotated coordinates for different regions on document images. The regions of interest for the task are labeled as "text", "anonymity" or "form" and are all considered text blocks, but the dataset also includes other regions as well, such as tables. These text blocks may include text of any form, meaning that there is no distinction between titles and paragraphs. As a result, the task will have to be divided in two subtasks. The first one, named text block detection, revolves around the use of the Computer Vision models for the detection of text blocks in documents, while the second, named text block classification, will implement rules in order to classify each text block as either title or paragraph. As per the second task, two NLP oriented models will be implemented focusing entirely on English documents, that will be trained on a different dataset that is also provided by the company. Any of the two Computer Vision models can then be combined with the NLP models created in order to classify each text block or table extracted, although this is not among the objectives of the thesis. To the knowledge of the author there are no official studies focusing on layout detection and classification in legal documents, utilizing the combination of the Computer Vision and NLP models that will be featured in this thesis. As a result this thesis is also written with the hopes of adding to the knowledge of the field.

1.2 Thesis Structure

The chapters to follow further explain the techniques and processes involved in the creation of these models. These chapters are :

Chapter 2 - Background and Related Work

In this chapter insight is given over the pretrained models that are used. Furthermore, fundamental definitions are provided for terms that are closely linked with each task. Finally, relevant work will be discussed pointing to ideas that could have been implemented, if the results of the models presented in this thesis proved to be unsatisfactory.

Chapter 3 - System Design and Implementation

This chapter gives a detailed overview of the way each implemented model works. For each model its architecture will be explored along with its use. Finally, all the necessary preprocessing and postprocessing procedures linked with each model or technique implemented will be explained.

Chapter 4 - Data Exploration and Experimental Setup

In the 4th chapter the datasets used will be explored. For each dataset the labels, annotated cases and other information and statistics will be provided. An overview of the metrics with which each model will be evaluated, will be given as well.

Chapter 5 - Results and Error Analysis

In the 5th chapter the results of each model developed will be provided, with the use of the metrics mentioned in the previous chapter. For each model the strengths and weaknesses will be discussed, and possible explanations of the results will be given.

Chapter 6 - Conclusions and Future Work

In the final chapter, the key points of the thesis will be summarized. Possible improvements and ideas for future work that were not explored in this thesis either due to time restrictions or the company's requirements, will also be provided.

2 Background and Related Work

2.1 Background

In the process of solving the text block detection and classification and the text zone detection tasks a plethora of techniques were used stemming from different fields. The tasks revolve around the use of both Computer Vision and NLP. Since the documents used are of legal nature, the end results have to abide by legal outlines as well. For the sake of a clearer view of the legal outline for document structure as viewed by Cognitiv+, that will be used throughout the thesis, a basic explanation of the structure of a legal document will be given. Afterwords, the core aspects of NLP and Computer Vision relevant to this thesis will be made clear.

2.1.1 Document structure

Legal documents, according to Cognitiv+'s guidelines, are comprised of text zones that include text serving a specific purpose. Out of the 7 zone types, namely "Contract cover", "Table of contents", "Introduction", "Recitals", "Main body", "Signatures" and "Attachments", only the first five will be used, with the rest being merged with the "Main body" zone. The document structure is the following :

1. **Contract Cover**
   Contract cover, or Cover page, refers to the first page of the document and usually includes information such as the document's title, the names of the contract parties, start or effective date etc. A cover page may not be present in every document, in which case, the document starts directly from the introduction.

2. **Table of Contents**
   A table of contents may be present at any point before the main body. It includes page indexing of the main document clauses and usually starts with the phrases "Table of Contents" or "Contents". As is the case with cover pages, tables of contents may not be featured in every document.

3. **Introduction**
   Introduction is yet another optional text zone present usually after the cover page

and table of contents. It includes information about the contract parties, dates etc., that summarize the document, usually at a span of a single paragraph.

4. **Recitals**
   The "Recitals" zone is usually present after the introduction and is often marked by one of the headers "RECITALS", "BACKGROUND", "INTRODUCTION" (if an introduction is not already present), "PREAMBLE" and features information found in the introduction, expressed in detail, in a list form, usually starting with capital English letters (A,B,..,Z). From the company's perspective, the title of each text zone does not always point towards the correct label. For example, the header "INTRODUCTION" may refer to either the "Introduction" or the "Recitals" text zones. As a result, the label will have to be define based on the content of each text zone.

5. **Main Body**
   The main body is the main part of the document organized by theme and numbering. For the purpose of the project it also includes any text regarded as a signature or an attachment.

The above format represents a basic example of a document's structure. It is by no means final as most of the text zones are optional and most importantly may appear with different order, to an extent. It is also possible that a text zone class appears multiple times throughout the document, which is usually the case with the "Main body" text zone.

Another important terminology for the purpose of the project is the text block, the detection of which is the main focus of the first task. Text blocks are components of text zones and are split in 3 different classes, namely "text", "anonymity" and "form", based on the dataset provided by the company. Text is considered anything that exists between the header and the footer of the document and is not a column, table, image or handwriting. Anonymity is a form of text block, containing text that is partially or entirely crossed by a red line or covered by a white box with black or blue outline making the text difficult (but not impossible in some cases) to read. An example of the "anonymity" text block is present in Figure 2.1. Finally, the "form" text block acts as a placeholder for information such as dates, signatures, names etc., that will be filled in the future by the parties, which the document refers to. Unless necessary, these 3 types will be simply referred to as text blocks.

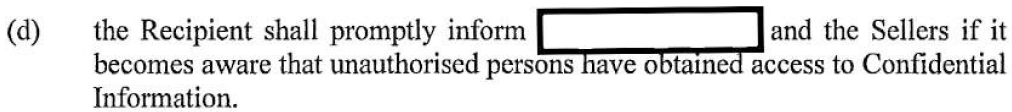


Fig. 2.1: An "anonymity" text block.

2.1.2 Text block detection

The first objective of the project is to build a model able to detect text blocks in images of any given legal document. These text blocks can be either titles or paragraphs, the detection and extraction of which are subject to the Computer Vision approach that is explored. On the Computer Vision part several models were created stemming from two main object detection approaches, namely one-stage detection and two-stage detection. Regardless of being one-stage or not, the majority of the models implement an Object Classification and an Region Proposal step. Their basic differences stem from the model's point in which each step is implemented [Yan+22]. For example two-stage detectors execute their Object Classification step only after the execution of the Region Proposal step, while one-stage detectors execute these steps in parallel. Basic terms related to each type are explained below.

Bounding box – Bounding box regression: In cases where the location of an object in an image rather that its simple existence is the objective, bounding boxes are an essential aspect. They are the means of locating a specific object, by setting a border around it usually in the form of a rectangle, and are specified through a set of coordinates. An example of bounding boxes on an image is shown in Figure 2.2. These objects are present in areas called Regions Of Interest (ROIs). Bounding box regression is a popular technique used by models, in order to predict such bounding boxes or improve their prediction accuracy. Using this technique models try to regress from their output, to the golden bounding boxes presented as input [LKC19]. Depending on the task or the model, bounding box regression is implemented differently and many alternatives have been proposed to combat frequent problems such as the ambiguity of some cases where golden labels are set in an inaccurate manner [He+19].

Non maximum suppression (NMS) is a technique broadly used in the Computer Vision field to eliminated bounding boxes "targeting" the same object [HBS17; HBS16; Sym+19]. In the classification stage multiple boxes may be classified as containing the desired object, which happens frequently in practice. The acceptable approach in such cases is to keep the box that encapsulates the object the closest. NMS offers a means of doing so by assuming that box to be the one with the highest class probability among overlapping bounding boxes. It is favored due to its simplicity, execution speed and accurate results [HBS16]. The basic NMS algorithm is presented in Algorithm 1's pseudo-code. Its effect is depicted in Figure 2.3.

Improvements have been proposed featuring the introduction of neural NMS approaches, in which the NMS process is handled by a neural network which is trained utilizing image features, apart from the standard class probability criterion, or sharing information between detections which form the input of the model. In such cases the input of the model

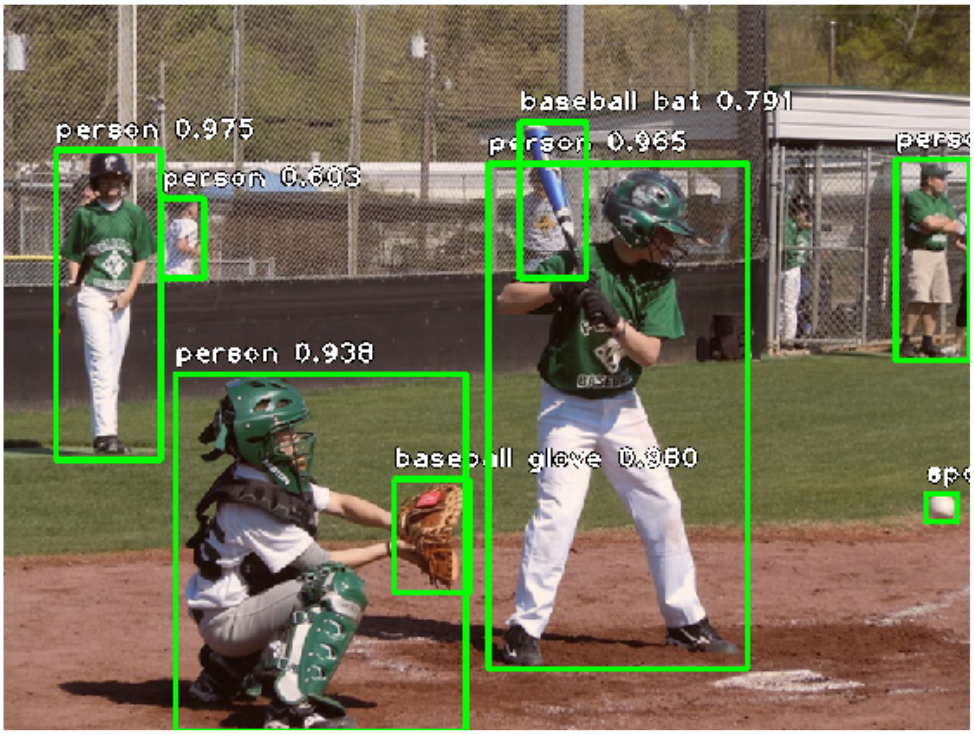


Fig. 2.2: Bounding boxes on a Coco dataset image.[1]

is comprised of a set of detected bounding boxes. Each detection is encoded and is later given a score. This score, though separate for each detection, is partially formed by taking into account the features of the other detections of the input. Sharing the information of each detection with the rest, creates a more context-aware and accurate score for each one. Cases of neural network-oriented NMS implementations are the GossipNet [HBS17; HBS16] and the network created by Charalampos Symeonidis et al. [Sym+19].

"**Intersection over Union(IoU)**, or Jaccard index, is the most popular evaluation metric used in the object detection benchmarks" [Rez+19]. Given two bounding boxes usually in the form of their coordinates, IoU is calculated as the quotient between the overlapping area of the two boxes and the sum of their areas.

Mean Average Precision (mAP) is an evaluation metric extensively used in Computer Vision and especially in object detection tasks, measuring the effectiveness of a model. This metric is usually calculated in combination with a pre-specified IoU threshold, which, in turn, is calculated between a predicted bounding box and its respective ground truth. Only the bounding boxes that surpass this threshold are considered correct for the mAP. The calculation of mAP on a specific threshold, for example 0.5 IoU, is represented as mAP@0.5, while mAP@0.5:0.95 represents the average mAP calculated over the IoU thresholds between 0.5 and 0.95, usually with a step of 0.05. It is important to note that as

[1]The image is part of the Coco dataset, which is publicly available from https://cocodataset.org/.

Algorithm 1 Non-Maximum Suppression

```
procedure GET(BB, t)                          ▷ bbox coordinates, and overlapping threshold
    Results ← []
    BB ← sorted(BB)                           ▷ Sort BBoxes from highest prediction to lowest
    while BB not empty do                     ▷ While there are still boxes
        r ← BB.pop(0)
        Results.add(r)                        ▷ Get the first box and add it to the result
        i ← 0
        while i < len(BB) do                  ▷ Iterate over the remaining boxes
            if overlap(r,BB[i]) > t then      ▷ If there is significant overlap
                BB.pop(i)                     ▷ Remove the bounding box from the initial list
                i ← i − 1
            i ← i + 1
    return Results
```

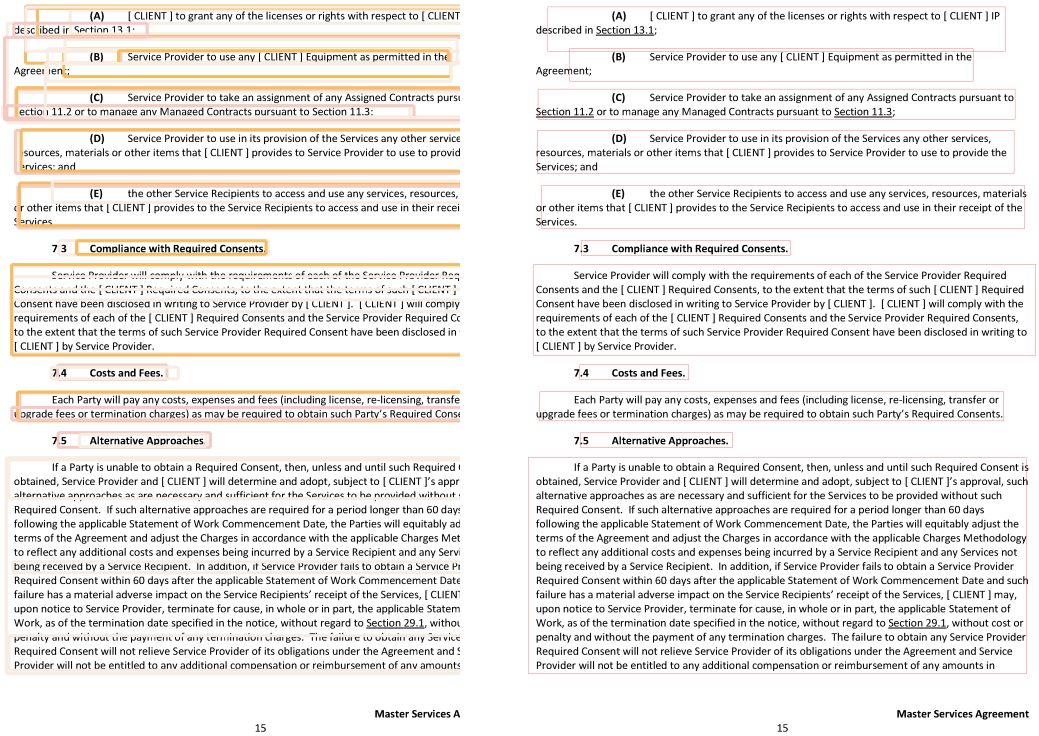


Fig. 2.3: Before and after the application of NMS.

the mAP represents the mean of the Average Precision (AP) per class, mAP and AP are equal in cases where there is only one class to be detected, as is the case in this project.

Region Proposal algorithms aim to detect regions that will later be classified by the object classification model [SI18; Ren+15a]. There are many Region Proposal techniques implemented such as Selective Search, CPMC and Objectness just to name a few [Gir+14]. The majority of them function by taking into account image attributes such as color contrast, edge maps etc. [ADF12]. Taking Selective Search as an example, it is an algorithm that defines pixel regions in images based on their structure. Older approaches proposed taking into account every possible location in an image (called exhaustive search). Exhaustive search is achieved by the sliding window method in which a window is applied from the start to the end of the image, creating candidate bounding boxes. This process normally takes a tremendous amount of time as every candidate box has to be classified. In order to combat its effects, Selective Search implements a hierarchical algorithm to gradually merge regions that are similar with each other so as to minimize the proposed regions. As a result, Selective Search is able to capture regions regardless of scale and take into account various image features such as color, lighting etc., while simultaneously reducing the computational time [Uij+13].

Anchors (or "priors") define a popular region proposal technique. They are a predefined set of fixed bounding boxes. They differ in size, aspect ratio and position on images and are assigned per class [Yan+18]. Typically, for each image, several anchors are created across it. During training, the ratio and size of these anchors may be refined based on the golden bounding boxes, meaning that the models that incorporate them, usually learn to offset the predetermined anchor box coordinates as needed, for a better performance [Zho+20]. Their use is a more efficient answer to the exhaustive search problem. By means of anchors every proposed region on an image can be processed and classified in parallel thus speeding the whole process up, usually at the expense of accuracy. They are typically present in one-stage detectors, but there are cases such as the Faster R-CNN that features such a methodology as well [Zho+20].

Object Classification is a step in which the candidate bounding box is classified as having a desired object or being part of the backround. In this step, rather than calculating the weights for every input (pixel) of an image, which is the case with traditional Feed Forward neural networks (FFNN), CNNs are used to minimize the required parameters by introducing kernels [AMA17]. Kernels are windows that are slid over the image multiplying their values with the corresponding window of the image thus creating new features. Their weights are the ones that are trained and are shared across the input space. CNNs may be present in both the classification stage and the region proposal one. With an image as an input, during the region proposal stage, bounding box coordinates are created. In turn, these coordinates offer the input for a Deep Learning model, that is trained to classify the region proposal output [Gir+14]. There are two cases for the proposed region. It either

belongs to one or more classes and as such the bounding box is valid, or it belongs to the background and is excluded.

Two-stage detector actions are split into two steps: region proposals generation and classification. Typically, the region proposal step needs to finish in order to move to the classification step, which is not the case with one-stage detectors. Earlier two-stage detectors feature a pure rule-based region proposal approach but for the most recent and accurate ones there are usually two neural networks involved, one for each task. Two-stage detectors typically suffer from a major drawback; they are computationally slow [SI18]. Furthermore, as "their training process includes two networks that have to be trained separately they are harder to optimize" [Red+16]. Finally, as the regions represent a new sub-image each, they typically tend to make more mistakes when it comes to the background of an image compared to other techniques [Red+16], as they treat each sub-image independently. Despite these drawbacks they typically are favored for their accuracy, outperforming the one-stage ones in most cases [Lin+20]. Moreover, recent advances on the field have made it possible to further minimize the speed-accuracy trade off, by accelerating the region proposal stage and minimizing the classification time [Li+17]. Some well-known approaches are the Fast R-CNN, Faster R-CNN [Ren+15a], Masked R-CNN [He+17] and Light-Head R-CNN [Li+17].

One-stage detectors treat region proposal and classification as a single step and thus are able to perform faster, usually at the expense of accuracy [Yan+22; Red+16]. They excel in areas where two-stage detectors usually pale in comparison, as is the case with real time object detection where execution speed is key, and have witnessed a rise in popularity as seen through publications in the resent years [Loh+21]. Finally, the region proposal step of the one-stage detectors is executed by means of anchors.

Focal Loss is an alternative version of the standard Cross Entropy Loss, used in order to train a one-stage detector model, called RetinaNet, which was created with the intent of bridging the gap between the accuracy of one-stage and two-stage detectors without sacrificing the speed aspect [Lin+20]. This is achieved by the introduction of an addition of a term to the Cross Entropy Loss as shown in Figure 2.4.

The idea behind the inclusion of the extra term stems from class imbalance. Typically, two-stage detectors utilize their region proposal implementation (e.g. Selective Search) to narrow down the possible bounding boxes. On the other hand, one-stage detectors must process every available anchor box the majority of which comprise the background [Lin+20]. The inclusion of the new term, depending on its value, is able to help the model focus on harder examples (having a low prediction probability) in terms of classification while down-weighting easier ones. The higher the γ value the lower the impact of easily classified examples is. This in turn increases the accuracy of the predictions, to the point of reaching two-stage detectors in some cases, without sacrificing speed [Lin+20].

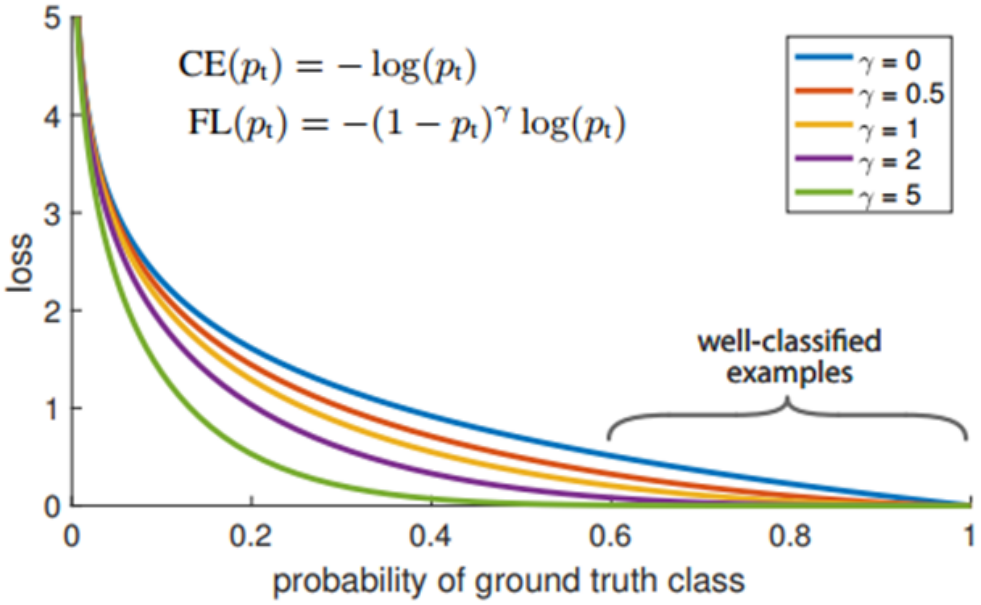


Fig. 2.4: The addition of the factor $(1 - p_t)^{\gamma}$ to the standard cross entropy criterion. Setting $\gamma > 0$ reduces the relative loss for well-classified examples ($p_t > 0.5$), putting more focus on hard, misclassified examples. In the case of $\gamma = 0$ Focal loss turns into the regular Cross Entropy Loss [Lin+20].

YOLO (You Only Look Once) is a family of one-stage detectors, highly praised for their speed and lightweight architecture. Each implementation revolves around the application of a grid, splitting the image into grid cells. For each cell a set of anchor boxes is generated, having the same center as the cell but featuring different sizes. The final bounding box coordinates are predicted along with a probability for each bounding box through a single network [Red+16]. In other words, the region proposal and classification steps take place simultaneously [Yan+22] having both bounding box coordinates and class predictions as output without the need of task specific sub-networks [Red+16]. Thanks to its simple yet effective architecture, the YOLO model family features the model with the highest known inference speed (YOLOv7) [WBL22]. Moreover, being able to process the whole image, they can create features based on the whole context and thus are able to avoid background misclassification as was the case with two-stage detectors. What this means for the current task is that they may be able to ignore ROIs based on their position and not just their content. Finally, "YOLO learns generalizable representations of objects" [Red+16] being able to predict cases of objects in different contexts. When tested on artwork against two-stage detectors, the implemented YOLO model was able to outperform its opponents in terms of both accuracy and speed.

As is the case with every model of this architecture, earlier YOLO model versions suffer in terms of accuracy compared to the top two-stage detectors but also to some one-stage cases. On top of that they struggle when it comes to detecting small object [Red+16]. In

the latest version improvements were made, featured in every new version, reducing the prediction errors to a point of outperforming R-CNN implementations (as is the case with YOLOv7-E6) [WBL22] while maintaining high execution speed.

It is important to note that the YOLOv5 model used for the project, is not considered an official YOLO model since it was not created by the minds behind the first four implementations (YOLOv1 - YOLOv4). Regardless, it is almost identical to YOLOv4. Furthermore, it is unclear whether YOLOv5 is superior to YOLOv4 as the accuracy of each one is task dependent, with no clear victor. The use of the YOLOv5 model for the task is justified from its lower memory usage and training time, compared to YOLOv4, but is also based on the requests of the company.

2.1.3 Text block and Text zone classification

In this thesis, NLP is present both during the classification of text blocks as titles or paragraphs and text zone classification. For the accomplishment of these tasks, a variety of methods were used, ranging from simple rule-based approaches to fine-tuning advanced pretrained models. These implementations feature the use of the pytesseract Optical Character Recognition (OCR) tool and the RoBERTa pretrained model.

Python-tesseract (pytesseract library) is an OCR tool for python, meaning that is able to recognise and read the text included in an image.[2] It is built around Google's tesseract which includes the OCR engine and the command line program responsible for the character recognition. The tesseract implementation by Google is a neural network trained to identify characters from images for more than 100 languages and can be further trained to support new ones.

Robustly optimized BERT Pretraining approach (RoBERTa) is a Transformer-based model featuring the same architecture as the BERT model. Like BERT, RoBERTa features Transformer blocks with bidirectional self-attention to capture the context of the input [Liu+19; Dev+19]. Their key difference stems from the pretraining approach. More specifically Yinhan Liu et al. came to the conclusion that BERT is "significantly undertrained" [Liu+19] and as such started working on modifications on the pretraining process. While BERT includes Next Sentence Prediction (NSP) as a downstream task it was excluded from RoBERTa claiming redundancy. To further improve the model, RoBERTa was pretrained using more data over a longer period and over longer sequences. Finally, the static masking of BERT was replaced by a dynamic one, generated per training sequence, to avoid having the same mask over each training instance [Liu+19]. This approach boosted the performance of the RoBERTa model over the "undertrained" BERT.

[2] Pytesseract is freely available from https://pypi.org/project/pytesseract/.

2.2 Related Work

The basic approach to document layout analysis involves the use of standard Computer Vision techniques, some of which were explored earlier like YOLO or Faster R-CNN or even rule-based approaches [She+21]. Depending on the data, accuracy or speed restrictions, one may be favored over the other, but regardless there weren't many studies focusing on document analysis-oriented models compared to other fields. It's only since the past few years, that new models started being developed tackling this specific task. Having been trained in large scale document datasets, their purpose is, given an image of a document, to define the areas where sections are present and classify them properly. Some well-known, open-source approaches are the following:

LayoutParser is a tool that incorporates Deep Learning techniques to tackle document image analysis (DIA). It is able to perform operations such as layout detection and classification, character recognition, visualization and more [She+21]. Moreover, it incorporates a selection of many different pretrained models, that specialize on different layout formats based on 6 datasets (HJDataset, PubLayNet, PrimaLayout, NewspaperNavigator, TableBank and MFD datasets).[3] Furthermore, it supports fine-tuning and multiple customization options, enabling better performance for custom data as the layouts of most tasks defer by a lot. Finally, it offers a tool for data annotations featuring active learning. Utilizing its detection model only the important layout objects are needed in terms of annotations minimizing the annotations cost [She+21].

LayoutParser can be further configured to produce results according to a confidence score threshold. By setting a low threshold, multiple regions can be generated. On this set up, LayoutParser will detect bounding boxes that it would categorize as "text", "title", "list", "table" or "figure" with very low confidence based on its pretraining; meaning that the output will not be irrelevant but an actual candidate region containing some form of text, even on documents of potentially different format that those of its training. For the purpose of the project, LayoutParser, being able to generate candidate areas such as these even though it was not trained on the dataset provided by the company, will be used as the region proposal architecture of a custom two-stage detector. The region proposals of LayoutParser will then be passed to a CNN for classification.

LayoutParser can be used directly to detect text blocks. For that to happen though, a training process is needed on the company's dataset. Nevertheless, LayoutParser utilizing two-stage detectors, comes with prohibitively long training and inference times both in terms of the project's deadline and the company's requirements. Thus, the LayoutParser training approach was not explored in this thesis.

[3] The models are available from https://layout-parser.readthedocs.io/en/latest/notes/modelzoo.html.

LayoutLM is a model that combines information of both textual and layout origin, extracted from document images and aimed towards document analysis [Yih+20]. The LayoutLM model builds upon the basic BERT model adding a Faster R-CNN architecture, thus enabling the extraction of both textual and visual embeddings from documents. The visual embeddings feature 2D position and image embeddings [Yih+20]. A 2D embedding treats an element (e.g. a collection of one or more words) in terms of its position in the document rather than a sequence which is the case with BERT's positional embeddings. This is achieved by a surrounding bounding box for each element featuring upper left and lower right coordinates using an OCR tool. Its aim is to utilize the position of a text to infer the kind of information that its surroundings offer. On the other hand, the image embeddings are extracted by utilizing the Faster R-CNN which treats each bounding box as an image. Treating each text visually but also the image as a whole, features such as text font, size, table formats, layouts etc. can be used to maximize efficiency when handling downstream tasks [Yih+20].

The two featured tasks involve the training of a Masked Visual-Language Model and Multi-label Document Classification. The Masked Visual-Language Model is inspired by the BERT language model architecture. In the same fashion LayoutLM is pretrained by randomly masking text embeddings and predicting the masked features by utilizing their 2D positional embeddings. As a second task and in order to create high level and accurate representations of the document elements, LayoutLM is also pretrained by minimizing the classification loss of image elements, using each document's golden labels for supervision during pretraining, much like the usual classification tasks [Yih+20].

3 System Design and Implementation

In this section insight is given over the functioning of the models developed or used. For each model, information will be given over its architecture and the methodology followed in order to prepare the data and configure or process its output. Finally, training details and results will be shown so as to provide a hint for their final evaluation.[4]

3.1 Text block detection

For the purpose of text block detection two custom two-stage detectors were created, both featuring different region proposal implementations. The classification step is carried out, in both cases, by the same CNN architecture which is trained on different input images, based on each of the region proposal approaches. Nevertheless, as per the company's request, focus will be given on one-stage detectors. For the one-stage detector approach two pretrained models are featured, namely RetinaNet and YOLOv5. For the YOLOv5 model multiple variations were trained as well. For each model insight will be given over its respective architecture and training procedure. Finally, due to the lack of necessary data, a rule-based approach was implemented in order to further classify text blocks as paragraphs or titles, instead of a Deep Learning one.

3.1.1 Region Proposal

The objective of the region proposal method in this case is, given an image, to output coordinates of candidate ROI bounding boxes containing text blocks. There is a plethora of methods developed and used, the predictions of which, along with a well-trained CNN, have impressive results. Nevertheless, most such cases are not suitable for the specific task. Such is the case with Selective Search. The process of grouping pixels based on color, texture, size and shape that the specific method implements tends to work better in cases where real life objects are present. For the purpose of the task at hand though, its results are far from optimal. In this case, intuitively following the idea of the algorithm, one would argue that the objects on documents are the letters or numbers which are clearly defined, in most cases, by their black color in white background. This is exactly the case

[4]Every model was trained and tested using a single Quadro RTX 6000 GPU.

with Selective Search. The method tends to draw bounding boxes around characters or in the best case a collection of them. Capturing a whole text block seems far beyond its intended purpose as seen in Figure 3.1.

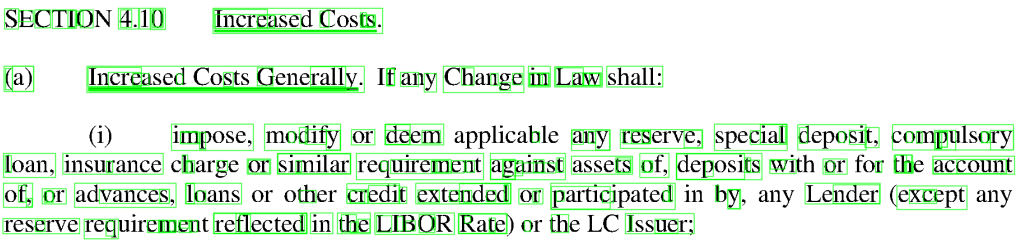


Fig. 3.1: The Selective Search algorithm's output on a test case.

As such, alternative approaches needed to be found. There are two approaches that were followed. The first approach is based on OpenCV image processing methods, that serves as a baseline, while the other relies on the use of LayoutParser predictions for candidate text blocks.

3.1.2 Object classification - CNN

The CNN's objective is, given an image, to output the probability of it being a text block. To this extent, the input for the model provides the bounding box set associated with each document, rather than the image of the document as a whole. The bounding boxes that are extracted during the region proposal step will be "fed" to the CNN, for each of which a probability will be reported, much in the same way as the R-CNN developed by Ross Girshick et al. [Gir+14]. If the probability exceeds a specified threshold the bounding box will be present in the original image. Following the architectural orientation that Ross Girshick et al. followed when creating their R-CNN [Gir+14], a CNN having similar architecture was developed in the work of this thesis, for the object classification stage. More specifically the model of choice for the development of their R-CNN [Gir+14] was the VGG_ILSVRC_16_layers model available in Caffe Model Zoo.[5] The basic VGG 16-layer architecture is the one seen in Figure 3.2. Similarly, in the implementation of the custom R-CNN in this thesis, a VGG architecture was used. The VGG version selected for the task is VGG19. The VGG19 model, described in the work of Karen Simonyan et al. [SZ14] is a neural network model featuring 16 convolutional layers (compared to the 13 convolutional layers of VGG16), famous for its simplicity and accuracy. It uses the smallest filter size (3x3) able to capture the surroundings of each pixel, along with a size 1x1 filter for linear transformations. After the CNN layers, 3 Fully Connected Layers (FCL) follow,

[5]The Caffe Model Zoo is publicly available from https://github.com/BVLC/caffe/wiki/Model-Zoo.

acting as classifiers for the convolution layer output, which is in the form of a probability per class, through a softmax activation function.

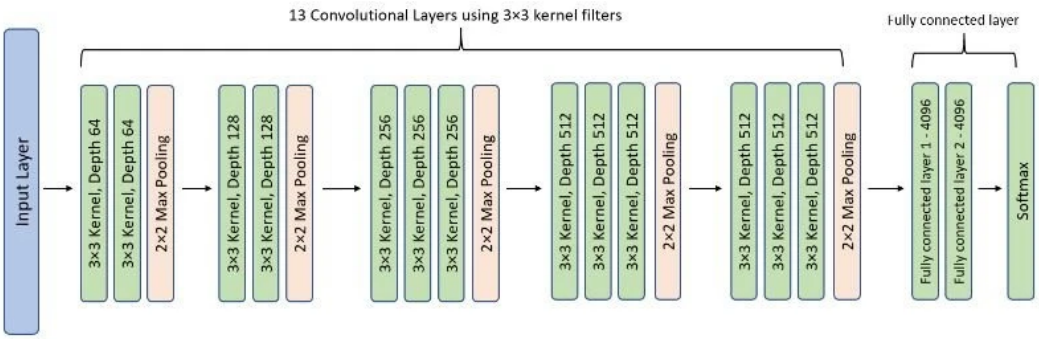


Fig. 3.2: VGG16's basic architecture [Tam19].

For the purpose of the project, the most recent version of VGG available in the Keras library was used. Contrary to the region proposal step where the two-stage detectors to be developed, will accent different approaches, the same CNN architecture will be used in both cases, accepting input images of size 224x224. Finally, as the VGG model is pretrained on the ImageNet dataset, in order to maintain the acquired knowledge but also specialize it to the task at hand, instead of the original three FCLs, two new FCLs were introduced. The first one manipulates the base model's extracted features by means of 1024 nodes with a Rectified Linear Unit (ReLu) activation function, while the second one outputs the probability that the image is a text block through a sigmoid activation function.

3.1.3 Text Block Detection Baseline

The OpenCV approach was chosen as the baseline as it was less dependent on neural networks, compared to the other approaches. Moreover, it is used as a means of comparison between rule-based and network-based region proposal methods. The OpenCV approach relies solely on the application of rules in order to propose regions and involves three main steps.[6] The first two focus on the necessary preprocessing of the images while the last one is responsible for the detection. The result of each step is depicted in Figure 3.3.

1. **Binarization**
   The objective of this step is to remove the possible noise and get an image with characters, as foreground, while anything else as background. The foreground and backround need to be separated by color; ideally the characters need to be in white and the rest in black. In order for the binarization of the image to be more accurate two filters were used. The first one is a simple gray-scale filter applied to the image. This color transformation removes unnecessary coloring that will

[6]The inspiration came from a post found on https://stackoverflow.com/questions/57249273/how-to-detect-paragraphs-in-a-text-document-image-for-a-non-consistent-text-stru.

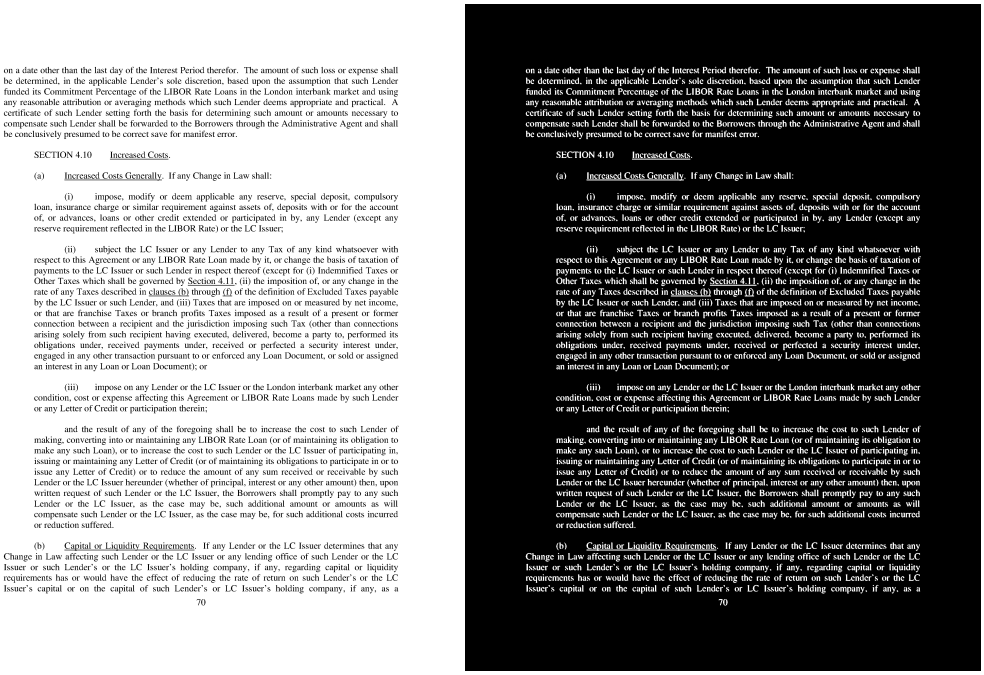
make the desired distinction more difficult. Afterwards a blurring filter is applied in order for the edges of the characters to be smoother and thus easily separable from the background. From this point on, the image is passed to the actual binarization step. Since the image may contain any number of colors in the white-black color spectrum a threshold needs to be set, depending on which, each color is transformed to black or white. The threshold in the specific case stems from Otsu's Binarization technique [Ope00a]. This technique enables the automatic detection of the optimal threshold for the task. After applying the threshold, the resulting image is finally binarized.

2. **Dilation**
   Dilation is a technique implemented in the OpenCV library that enlarges the boundaries of the foreground, meaning characters in this case [Ope00b]. Dilation was used for the task with the aim of "stretching" each character in a way that characters grouped together (as is the case with text blocks) overlap. The dilation was set to take effect more in the x axis of each character compared to the y axis to avoid grouping characters belonging in different text blocks. In the resulting image the dilated characters now form a rough rectangle where the text blocks are.

3. **Contour Detection**
   In the final step the shapes created during dilation need to be pinpointed. OpenCV utilizes the Border Following algorithm [SA85; Ope00c] to detect these shapes and return the coordinates of their corresponding bounding boxes. The original image is later cropped according to these coordinates, offering the input for the object detection CNN.

Training

Since the OpenCV region proposal step is not trainable, the training revolves around the CNN of the object classification step. It is important to note that, the training data used for the model are not the golden bounding boxes of the annotations. A common practice in such cases is to use the output of the region proposal technique. The results of the region proposal step are then compared to the golden annotations. Given a threshold, if the IoU between the output and a golden annotation is greater, the output bounding box is treated as a positive case. If not, it is considered negative. This enables the model to learn by utilizing its own predictions rather than the golden truth which no region proposal is able to achieve. For the purpose of the task, the threshold was set to be 0.6 as it balances the positive and negative cases. The resulting distribution of the data is 50.000 positive cases and 50.000 negative cases. The use of the whole dataset was avoided mainly because both the region proposal generation, and CNN training times were prohibitive. The validation data size is 20.000 and includes the same amount of positive and negative cases.



on a date other than the last day of the Interest Period therefor. The amount of such loss or expense shall be determined, in the applicable Lender's sole discretion, based upon the assumption that such Lender funded its Commitment Percentage of the LIBOR Rate Loans in the London interbank market and using any reasonable attribution or averaging methods which such Lender deems appropriate and practical. A certificate of such Lender setting forth the basis for determining such amount or amounts necessary to compensate such Lender shall be forwarded to the Borrowers through the Administrative Agent and shall be conclusively presumed to be correct save for manifest error.

SECTION 4.10 Increased Costs.

(a) Increased Costs Generally. If any Change in Law shall:

(i) impose, modify or deem applicable any reserve, special deposit, compulsory loan, insurance charge or similar requirement against assets of, deposits with or for the account of, or advances, loans or other credit extended or participated in by, any Lender (except any reserve requirement reflected in the LIBOR Rate) or the LC Issuer;

(ii) subject the LC Issuer or any Lender to any Tax of any kind whatsoever with respect to this Agreement or any LIBOR Rate Loan made by it, or change the basis of taxation of payments to the LC Issuer or such Lender in respect thereof (except for (i) Indemnified Taxes or Other Taxes which shall be governed by Section 4.11, (ii) the imposition of, or any change in the rate of any Taxes described in clauses (b) through (f) of the definition of Excluded Taxes payable by the LC Issuer or such Lender, and (iii) Taxes that are imposed on or measured by net income, or that are franchise Taxes or branch profits Taxes imposed as a result of a present or former connection between a recipient and the jurisdiction imposing such Tax (other than connections arising solely from such recipient having executed, delivered, become a party to, performed its obligations under, received payments under, received or perfected a security interest under, engaged in any other transaction pursuant to or enforced any Loan Document, or sold or assigned an interest in any Loan or Loan Document); or

(iii) impose on any Lender or the LC Issuer or the London interbank market any other condition, cost or expense affecting this Agreement or LIBOR Rate Loans made by such Lender or any Letter of Credit or participation therein;

and the result of any of the foregoing shall be to increase the cost to such Lender of making, converting into or maintaining any LIBOR Rate Loan (or of maintaining its obligation to make any such Loan), or to increase the cost to such Lender or the LC Issuer of participating in, issuing or maintaining any Letter of Credit (or of maintaining its obligations to participate in or to issue any Letter of Credit) or to reduce the amount of any sum received or receivable by such Lender or the LC Issuer hereunder (whether of principal, interest or any other amount) then, upon written request of such Lender or the LC Issuer, the Borrowers shall promptly pay to any such Lender or the LC Issuer, as the case may be, such additional amount or amounts as will compensate such Lender or the LC Issuer, as the case may be, for such additional costs incurred or reduction suffered.

(b) Capital or Liquidity Requirements. If any Lender or the LC Issuer determines that any Change in Law affecting such Lender or the LC Issuer or any lending office of such Lender or the LC Issuer or such Lender's or the LC Issuer's holding company, if any, regarding capital or liquidity requirements has or would have the effect of reducing the rate of return on such Lender's or the LC Issuer's capital or on the capital of such Lender's or LC Issuer's holding company, if any, as a

70

(a) Initial image.

(b) Binarization.

(c) Dilation.

(d) Contour Detection.

Fig. 3.3: Text block detection baseline region proposal approach.

The VGG19 architecture utilized for the purpose of the task includes pretrained weights for the convolutional layers. In order for the VGG19 model to adapt to a specific classification task, the custom FCLs need to be trained from scratch, independently from the rest of the model. Afterwards the whole model needs to be jointly trained in order for the convolutional layer weights to be tuned, thus achieving maximum effectiveness. For this to happen the model was trained in two phases.

The aim of the first phase is to train the FCLs without altering the weights of the rest of the model. This was accomplished by a method called freezing that allows for partial model training. Training the model unfrozen would have had catastrophic results as the pretrained weights would be tarnished during the first training epochs when the top layer results would be far from optimal. When the results of the training reach a satisfactory level, the first phase is complete and the model passes to the next step. The model in its current state was trained for 16 epochs, with the best weights being formed in the 6th and with each epoch being completed after 8.5 minutes on average. The validation loss at that point is 0.19.

The second phase is carried out with the hopes that the model still has the capacity to learn by fine tuning the weights of every layer. This process involves the "unfreezing" of the layer weights and the continuation of the training process from the 6th epoch. To avoid catastrophic alterations of the convolutional layer weights the second training process was carried out using a very small learning rate. This training process went on for 13 epochs reaching 19 epochs in total. The finest model weights were formed in the 9th epoch. Due to the model having to backpropagate through more layers, the average training time per epoch was increased to 13.3 minutes. The resulting training and validation losses are present in Figure 3.4 reaching a minimum validation loss of 0.14 compared to the 0.19 loss achieved while the layers were frozen. The continuation of the training processes with unfrozen layers has reduced the validation loss by about 0.05, justifying the use of the method.

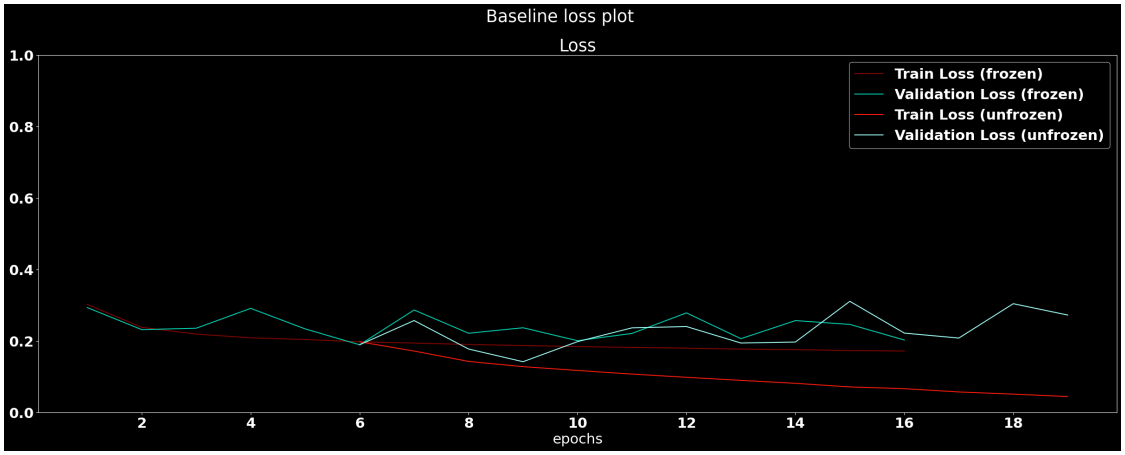


Fig. 3.4: Text block detection baseline training and validation loss per epoch.

Post-processing

Usually there would be a post-processing step before the final output. It's quite possible that there may be several bounding boxes correctly predicted, that contain the desired object. As such during post-processing, NMS (as seen in Subsection 2.1.2) is usually applied to keep the bounding box with the highest probability per detected object. Due to the nature of the region proposal algorithm though, there is only one possible bounding box at most per ROI and as such NMS is not required in this case.

3.1.4 LayoutParser as a Region Proposal method

LayoutParser Architecture

LayoutParser [She+21] incorporates the Detectron2 library, made by Facebook AI team, that provides a variety of model choices.[7] As such LayoutParser can be configured to use a subset of them. The difference between the LayoutParser and Detectron2 models is that LayoutParser features models pretrained solely on document datasets. The model utilized for the specific task is Faster R-CNN FPN with PubLayNet dataset pretraining. The basic architecture of the model is featured in Figure 3.5.

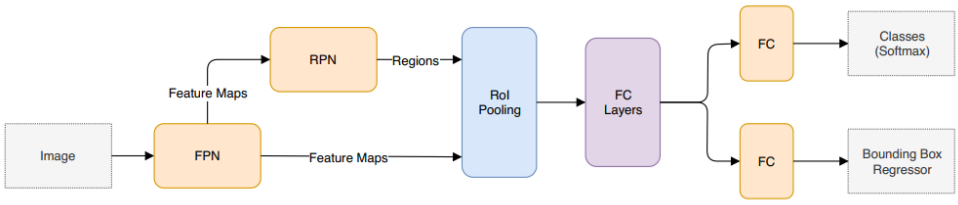


Fig. 3.5: Faster R-CNN FPN architecture [Luc+19].

What Facebook AI's implementation does is to replace the original backbone of the Faster R-CNN with a Feature Pyramid Network (FPN) in order to provided a richer collection of features. More specifically, the way an FPN works is by incorporating three main aspects as shown in Figure 3.6. The first one is a bottom up pathway which creates features by resizing the input image's feature map to half the original size. The initial image's feature map is usually created through a backbone model. Each new feature map instance is downscaled compared to the previous one in which features are created in a fully convolutional manner. The second aspect is the top down approach and does the exact opposite process. Having the top feature map, which is semantically richer yet remarkably smaller than the input image's initial feature map, for each level created in the bottom up stage, an upscaled version of the previous map is created, starting from the top level. In other words for each scale of the image two features maps are created, one starting from the bottom while the other from the top. Each scale has exactly two feature maps which are merged through a lateral connection that applies an element-wise addition of the two maps, to create a

[7] Detectron2 is publicly available from https://github.com/facebookresearch/detectron2.

representation boasting of rich information. From this representation, a final feature map is created per level [Lin+17].

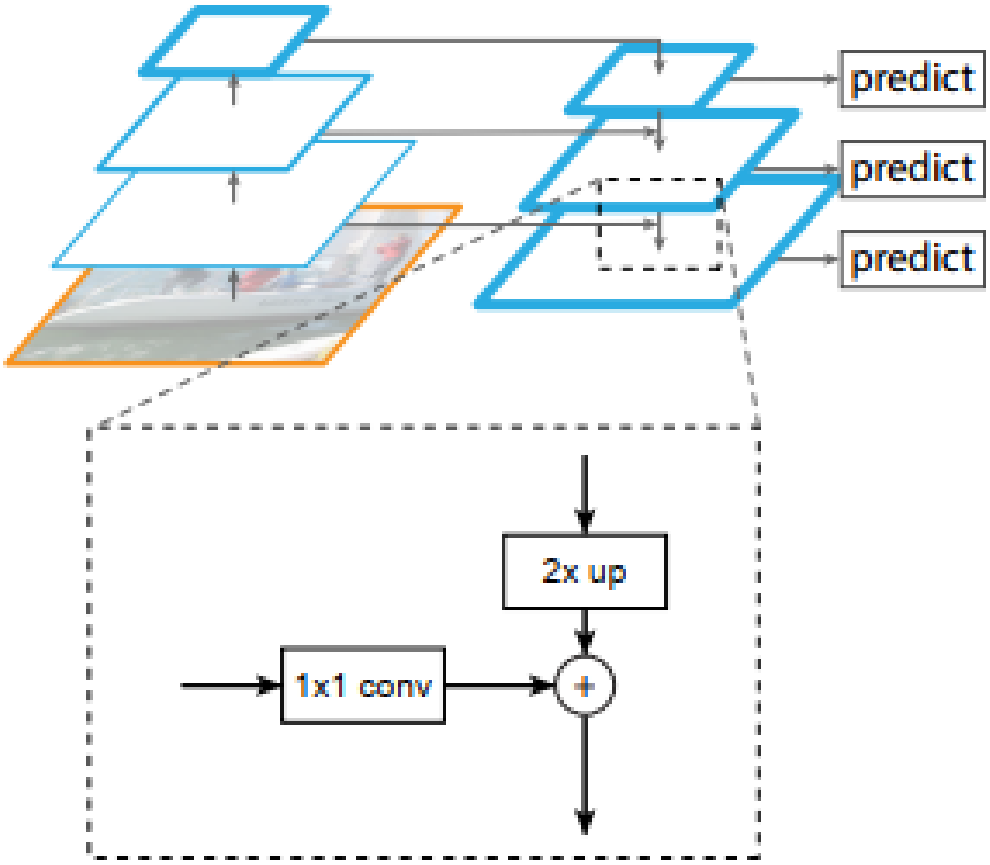


Fig. 3.6: The FPN's 3 main aspects. The block represents the lateral connection that merges features of the same level [Lin+17].

The architecture mentioned also incorporates, a Region Proposal Network (RPN), which is a sub-model responisble for generating regions of interest directly from an image input and is created by utilizing CNN layers [Ren+15b]. The outputs of the model are the coordinates of the detected bounding boxes along with the respective class probability.

Training

Under normal circumstances, the CNN that will classify the LayoutParser's output would have to be trained using the LayoutParser predicted regions. Due to the immense time it takes to generate a proportional amount of data to match the baseline, LayoutParser was only used during inference. Instead the data used as positive cases were directly the golden annotations. The original dataset contained annotations for bounding boxes that were not considered text blocks (e.g. images, signatures, headers etc.) even though most did contain text. As a result for negative cases a mixture of annotations belonging to one of these classes and automatically generated bounding boxes not belonging to any class were employed. The same CNN as in the baseline was trained using the data specified.

The training process, while the convolutional layers are frozen, took 84 epochs with an early stopping callback of 10 epochs. Since the CNN architecture, input size and training configuration is the same as the baseline model, so is the efficiency of the model in terms of execution speed. The model state with the lowest validation loss was detected at epoch 74. From this point on, the model was completely unfrozen and kept on being trained for another 13 epochs. While being unfrozen the model kept improving in terms of loss, justifying the procedure. The model with the best performance was detected 3 epochs after the unfreezing. The model while being frozen reached a validation loss 0.093. Its perfromance continued to improve after unfreezing reaching 0.089 validation loss as depicted in Figure 3.7. The difference between frozen and unfrozen base model is not significant since the model already has low loss in this case, but still stands up for the specific approach. Furthermore, it would seem that it has achieved lower loss than the baseline in terms of the validation set, but as the inputs for the model are directly the golden labels, rather that the LayoutParser outputs, there is no indication of whether it will be able to outperform the baseline in the final test case.

Post-processing

Contrary to the baseline model, LayoutParser is able to predict multiple bounding boxes for the same RIO. For this reason, the output will have to pass through the NMS algorithm to exclude redundant bounding boxes. As previously mentioned, NMS requires a threshold above which overlapping bounding boxes with predicted probability lower than the highest one, are excluded. The threshold that worked best in this case and with the current configuration, is 0.3. The application of NMS in Figure 2.3 is an example of its use, applied directly to the output of the current model.

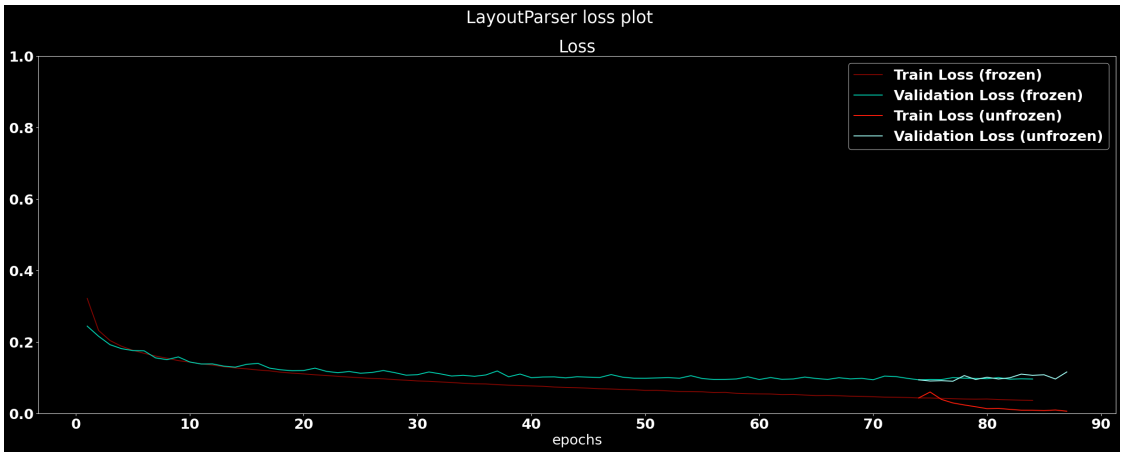


Fig. 3.7: LayoutParser/CNN Training and validation loss per epoch.

3.1.5 RetinaNet

RetinaNet being an one-stage detector but also providing high accuracy, is highly favored and widely used when it comes to object detection. It is also one of the two main choices for the completion of the text block detection task.

Architecture

RetinaNet is comprised of a backbone network and two task specific sub-networks, as depicted in Figure 3.8. For the backbone a ResNet architecture is used. ResNet is a Deep Learning model pretrained on the ImageNet dataset. The ResNet architecture is combined with the FPN which provides a rich collection of features stemming from the same image, as was the case with Faster R-CNN FPN architecture.

A Box regression and a Classification sub-net comprise the task specific sub-networks. The box regression sub-net includes an Fully Convolutional Network (FCN) used to regress from each anchor boxes offsets to their corresponding ground truth bounding boxes by means of a vector of length 4 (one value per coordinate). One the other hand, the Classification sub-net is an FCN responsible for predicting class probabilities per anchor (after applying the offset) and per class. It features a similar architecture as the box regressor but they both use separate parameters [Lin+20].

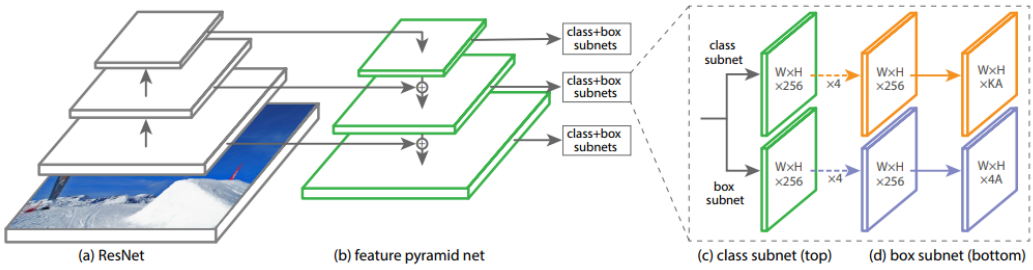


Fig. 3.8: RetinaNet's architecture [Lin+20].

Implementation details

The Keras implementation of RetinaNet offers ready to use scripts, able to preprocess the data and train, evaluate and test the model as well as plenty of examples and on point documentation, making the use of RetinaNet straight forward.[8] On the other hand, its usage requires careful handling of the input. In order to train and validate the results per epoch, two folders are required, each containing their respective images. Contrary to the two-stage detectors, the training and validation data for RetinaNet are the document images themselves. To recognise each bounding box, apart from the image folder, it requires a file including such information. The format chosen for the task that is also compatible with RetinaNet is CSV. In the CSV format, each line represents one bounding box, meaning that each document can be represented by multiple lines. Each line includes

[8]The implementation is available from https://github.com/fizyr/keras-retinanet.

6 values; the path of the image, along with the top left and bottom right coordinates and the class of the object. In case that an image does not include any bounding boxes, the path to the image must be included and the other values must be left empty. Like the image folders, one CSV file was created per set. A final CSV file is required that maps the classes of the task to a number. For this task, since the detection refers to one class only, a single line was required, that included the name of the class and an id number.

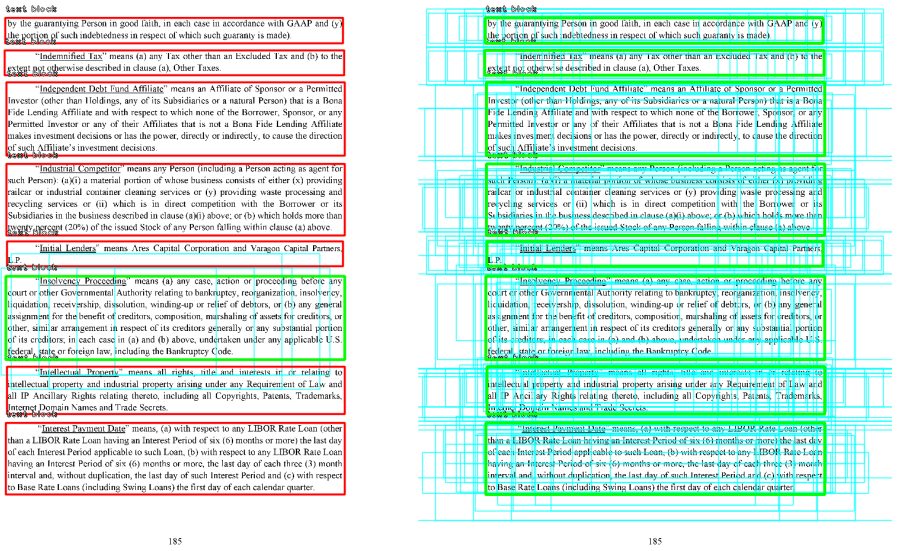
Training

RetinaNet offers the options of both fine-tuning and training from scratch. Due to the nature of the dataset of this thesis, which is far from what RetinaNet is trained on, but also due to the sufficient amount of data available, the training process was opted. Downloading RetinaNet's files, along with the model, a training script is included that offers a variety of choices ranging from backbone options to training history output formats. In order to train the model, Restnet101 was chosen as a backbone, which is an upgrade from Resnet50 (suggested model by the creators of RetinaNet). Restnet101 pretrained weights were also loaded. The training was set to complete after 200 epochs, but the early stopping option of 10 epochs was also implemented to avoid overfitting.

During both the training and testing phases the model's predictions were inadequate. Its results were even worse that the baseline even though the model is pretrained. It was later found that the problem was caused by the anchor box configuration which was predefined by the creators of RetinaNet in order to perform best in most cases, focusing especially on small objects. This comes in contrast to the current task as there are cases where the region of interest covers up to 90% of the image. As a result there were no anchor boxes able to capture the majority of the training instances. Golden bounding boxes, which RetinaNet is not able to detect by means of anchor boxes, are excluded during training, thus the training instances were very few and restricted in terms of position. After altering the anchor box configuration of RetinaNet's scripts in order to include anchor boxes of larger size and ratio, the model was able to capture the majority of the training instances and thus perform remarkably better. An example is provided in Figure 3.9.

This example encapsulates two of the basic issues of using the default configuration of anchor boxes. Firstly, without tuning the anchor box hyperparameters, in cases such as this one it would seem that the model is not able to capture the essence of the data, which is definitely not the case. Secondly as the hyperparameters have to be tuned there is a number of different possible combinations, each of which offer different results. In order to find the optimal result these combinations must be explored, each one requiring its own training phase, meaning that better configurations may be possible for the current task.

For this case, after some preliminary experimentation, the anchor boxes were set to include boxes of scale (size of the anchor box in pixels) up to 1.5 times the original size while increasing the ratio (comparison between the width and height of the box) to 4. With these



(a) Document with default anchor boxes. **(b)** Document with altered anchor boxes.

Fig. 3.9: The difference of default and altered anchor boxes. Red boxes represent golden boxes with few to no corresponding anchor boxes which will be ignored during training. Boxes in green represent golden boxes having an adequate number of corresponding anchor boxes. The anchor boxes are the rectangles in blue.

changes the majority of golden boxes were detected thus increasing the training data that would contribute to the model's training. With the increase in the data came the increase in training time as well. The training folder in this case contains 19122 images while the validation 4096 which lead to a training time of 48 minutes per epoch, mostly due to the number of layers RetinaNet includes. In order to reduce the training time, the amount of documents in the training set was reduced to a third (to about 6500 documents), dropping the training time to 23 minutes per epoch. Overall, the model was trained over 27 epochs with early stopping. The best model was observed in the 17th epoch having a stable AP as depicted in Figure 3.10. Since the text block detection task revolves around the detection of one class only, AP is equal to mAP in this case.

Post-processing

RetinaNet does not provide a script that outputs predictions, at least for the Keras implementation that was used. Instead, it offers a script that converts the final model to a format that can be later used in a custom fashion to provide the predictions. This custom procedure involves the loading of the model in Keras format. During inference, the images have to be prepossessed beforehand, in the same fashion as the training which was done automatically. For the model to work accordingly each image has to be rescaled to 800x1300 pixels. The pixels are also zero-centered. With the current image preprocessing, the model, given a preprocessed image as an input, outputs the predicted box coordinates, labels and probabilities. Having the coordinates, the final output is formed by drawing bounding boxes in the document image where their respective probability is higher than 0.65.

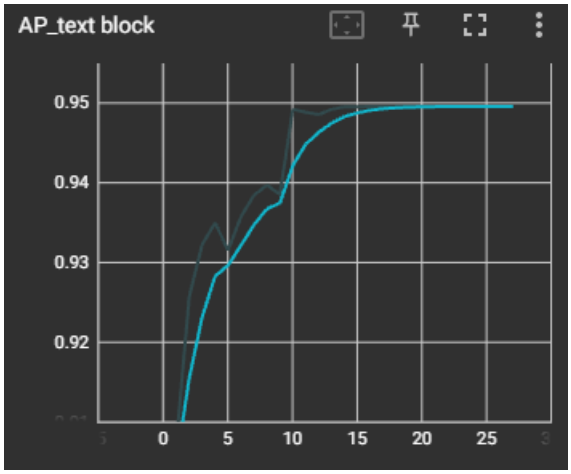


Fig. 3.10: RetinaNet's AP per epoch.

3.1.6 YOLOv5

For the purpose of the task YOLOv5 was also used. YOLOv5 offering high accuracy and even higher speed was the main focus for the completion of the text block detection task along with RetinaNet. YOLOv5 is also highly praised for the surprising low training needs in terms of execution speed and hardware. It's due to this fact that 3 alternative configurations of YOLOv5 were created along with the main model.

Architecture

YOLOv5 consists of four main parts; namely the input, backbone, neck and head. Its architecture is depicted in Figure 3.11 [NE22]. The backbone architecture includes a Focus block. It consists of a single layer rather than three, which was the case with YOLOv3, in order to minimize memory usage and speed (hence the name), without a noticeable difference in terms of accuracy. This speed increase is attributed to the fact that forward propagation as well as backpropagation is done through fewer layers, compared to the previous versions.

Following the Focus block there is a stacked combination of simple convolutional layers and C3 blocks. C3 blocks themselves are a combination of convolutional layers and Cross Stage Partial Network-like (CSP) layers, which are featured in YOLOv4 [NE22]. CSP layers are part of the plugin modules, called bag of specials, used in YOLOv4 in order to increase the training quality and time, while just slightly increasing inference speed. The CSP layers featured in YOLOv5 are comprised of a mixture of three convolutional and several bottleneck layers which help reduce the dimensionality of the features and size of the model. The last block of the backbone is a Spatial Pyramid Pooling (SPP) layer which "significantly increases the receptive field, separates out the most significant context features and causes almost no reduction of the network operation speed" [AWL20] and is also part of the bag of specials. SPP's purpose is to eliminate the need for a fixed size input. Traditional

CNNs, consisting of convolutional and dense layers require an input of fixed size. This requirement stemms from the dense layers themselves rather than the convolutional ones that can function with arbitrary sized inputs [He+14]. This requirement of a fixed size input leads, in turn to the need for input resizing, re-scaling etc. In the case of images, such actions may distort them enough to lead to unwanted results. With the addition of the SPP layer between the convolutional and dense layers this image preprocessing need is eliminated, as SPP accepts inputs of any size and provides fixed size output for the dense layers. On top of that, SPP is able to generate different feature maps, by processing multiple instances of the image (finer or coarser cases compared to the original). The features extracted are later pooled (using average pooling) in order to produce the final output that will later be passed to the dense layers as depicted in Figure 3.12.

The neck of YOLOv5 is comprised of the Path Aggregation Network (PANet) architecture, as depicted in Figure 3.11. PANet takes the place of the FPN featured in RetinaNet and Faster R-CNN. Feature maps produced in the backbone stage are combined with feature maps of the PANet through the Concat blocks that perform concatenation of the feature maps. Similar to the top down aspect of the FPN, the initial feature map of this stage passes through an Upsample block increasing the size of the feature map. The feature map may be upsampled multiple times generating multiple levels, for which there is an equal number of combined C3, convolutional and Concat blocks that the last upsampled feature map passes through. Each level, provides a distinct feature map as was the case with the FPN. Contrary to the FPN though, no down-scaling procedure is involved. Each feature map is finally processed be a CNN block that serves as a detection module [NE22]. These blocks form the head of the model and are identical to both YOLOv3 and YOLOv4 architectures.

Implementation details

Compared to RetinaNet, YOLOv5 demands a stricter directory hierarchy and data format. More specifically, a data folder needs to be created with two sub-folders; a folder containing images and a folder containing the annotations. Apart from the contents of each folder their structure is otherwise identical. The data need to be split into training, validation and testing sets. Each one of these sets form the main folders of their parents. Their contents are where their structures diverge. Each sub-folder of the image folder contains images belonging to a specific set. For the annotations folder each sub-folder contains the exact same file names as in the image folder, with a .txt file type. Files with names that are not present in the respective location of the image folder are disregarded. Each file of the annotations folder represents an image and has a number of rows that matches the text block annotations per image. If an image contains no bounding boxes an empty .txt file still needs to be present. Each annotation is represented in the file by five values. These are the id of the label, the x and y coordinates of the center of the bounding box, relative to the top left corner of the document image, and the width and height of the bounding box. Each number is scaled on a range of 0-1 so that image preprocessing will not affect

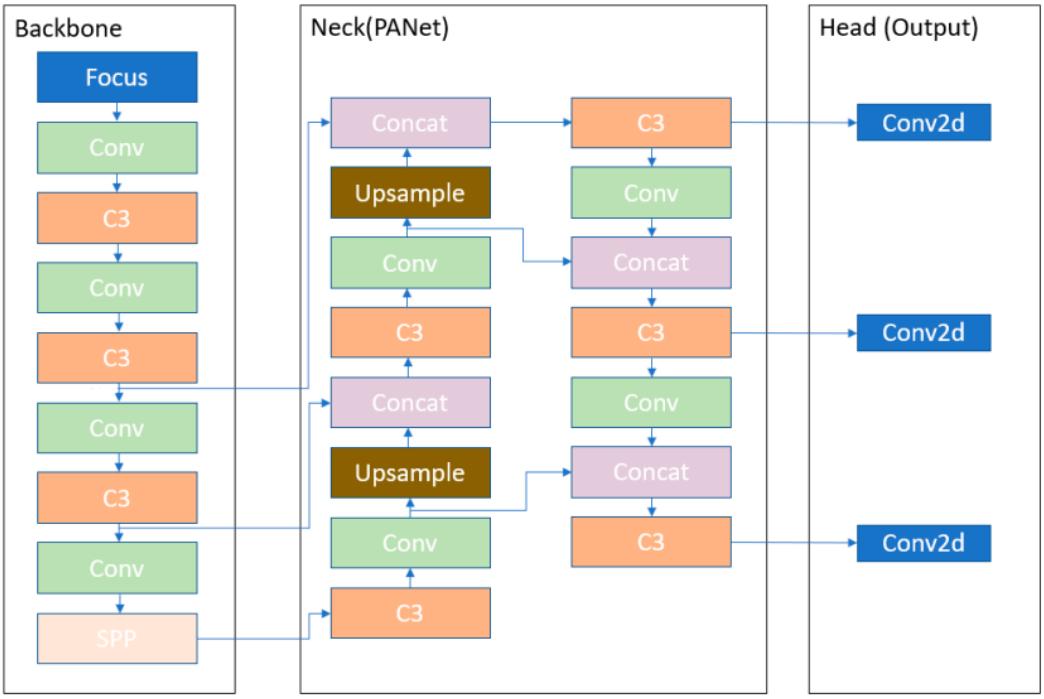


Fig. 3.11: Backbone, neck and head of YOLOv5.

the annotation values. Finally a .yaml file needs to be created. The specific file contains the paths to the image sub-folders, the number of classes and a list of the class names.

Training

Just like RetinaNet, YOLOv5 offers already implemented scripts for training. Different training configurations are possible by simply using the training script with specific arguments. The model was configured to resize the images to 640x640 as per the YOLOv5 creators suggestion and since the data bounding boxes are sufficiently large and well annotated. A small-scale augmentation step was also initialized in hopes of increasing the efficiency of the model. The augmentation techniques utilized include saturation of the image as well as left-right flips and finally mosaicing. Image mosaicing was introduced by YOLOv4, and is a process of combining 4 training images into one in order to train the model to contexts not yet explored while maintaining the annotations intact [AWL20]. Since the algorithm should be language agnostic such augmentations should also help the model focus more on the context of the text block rather than the content. Some examples of the augmentation process are present in Figure 3.13.

Every YOLOv5 model was configured to be trained over 200 epochs with an early stopping callback of 10 epochs measuring fitness. Fitness is a weighted linear combination of metrics [Ult21]. These metrics are precision, recall, mAP@0.5 and mAP@0.5:0.95. The weights of precision and recall were set to 0, while the weights for mAP@0.5 and mAP@0.5:0.95 were 0.1 and 0.9 respectively. For the initial model, YOLOv5m (medium) was chosen which has 21.2 million parameters and takes up to 41 MB of space. The model was trained over

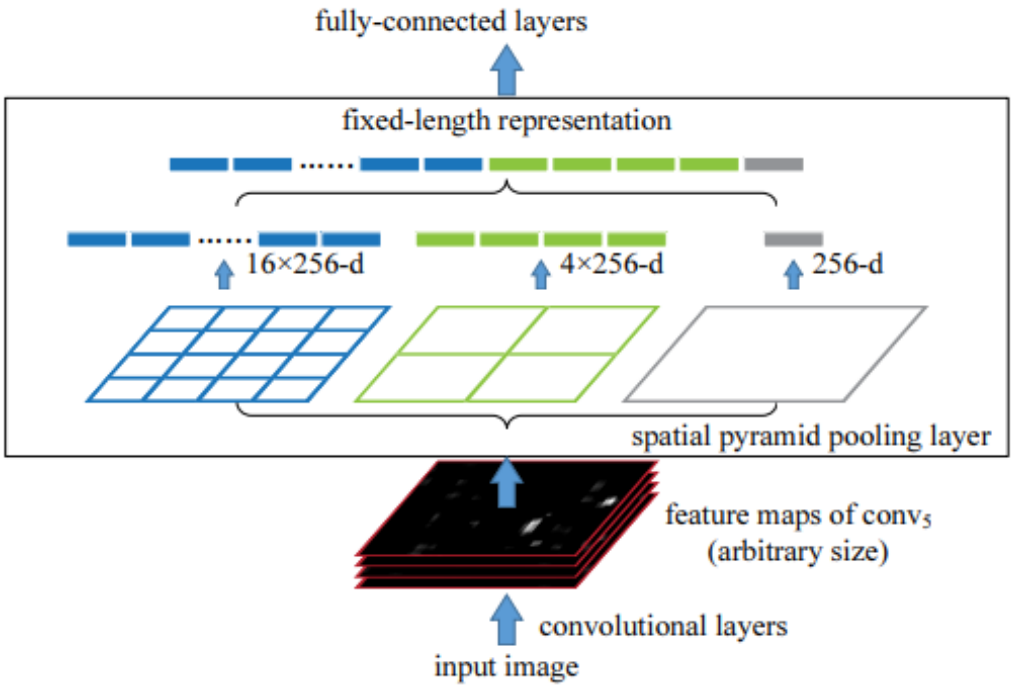


Fig. 3.12: A CNN model featuring an SPP layer between convolutional and dense layers [He+14].

103 epochs, with each epoch taking 3.3 minutes on average. There were also 2 additional models trained for comparison; YOLOv5l (large) and YOLOv5s (small) model. YOLOv5s has 7.2 million parameters and takes 14 MB of space. The training went on for 178 epochs with each epoch being completed after 2 minutes on average. Normally smaller models perform worse but in order to compare the accuracy difference and since both size and execution speed were important for the purpose of the task, this model was also considered. On the other hand, YOLOv5l includes 46.5 million parameters and takes 90 MB of space. It was trained using the same images resized to 1280x1280 pixels and over 111 epochs with the best state of model being in epoch 101. Each epoch took 6 minutes to complete, on average. The resulting mAPs on the validation set are as depicted in in Figure 3.14.

Considering solely the mAP score, the large model outperforms the rest, while performing slightly better than the medium. Of course this is the one that takes the most amount of time per epoch which will also be taken into consideration during inference. Regardless of the model size it would appear that every YOLO model considered, outperforms the previous detectors (baseline, LayoutParser and RetinaNet). Contrary to the RetinaNet's case the YOLOv5 training set containts more than 19.000 document images. For RetinaNet's case, due to the excessively long training time, fewer training instances had to be included (6500 images were used for RetinaNet's training). On the other hand, YOLOv5 had no such training issue requiring up to 15 times less training time. Thus, the better results of the YOLOv5 models are partially to be attributed to the number of training instances used and to the fact that YOLOv5s' anchors are task specific, meaning that their configuration is generated automatically based on the training instances. In RetinaNet's case, the anchors

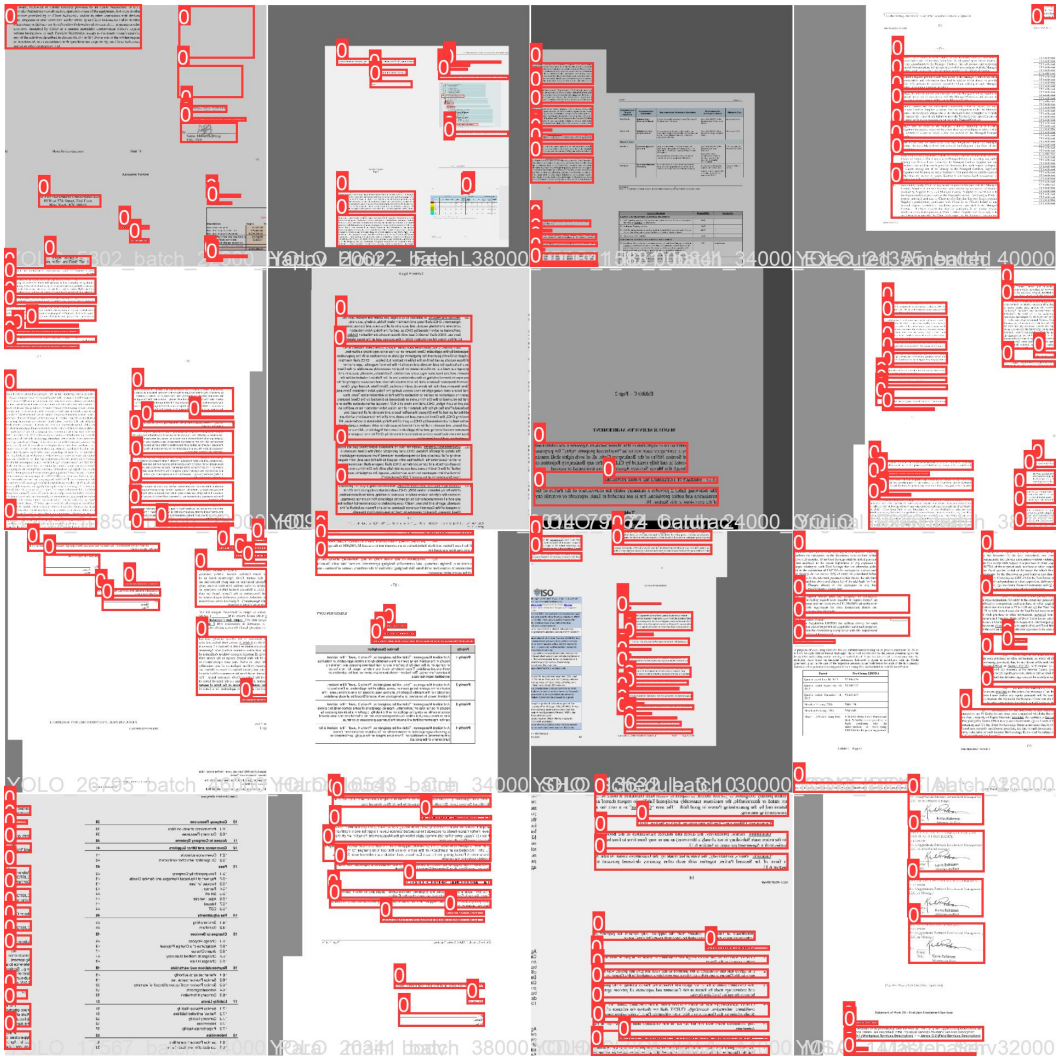
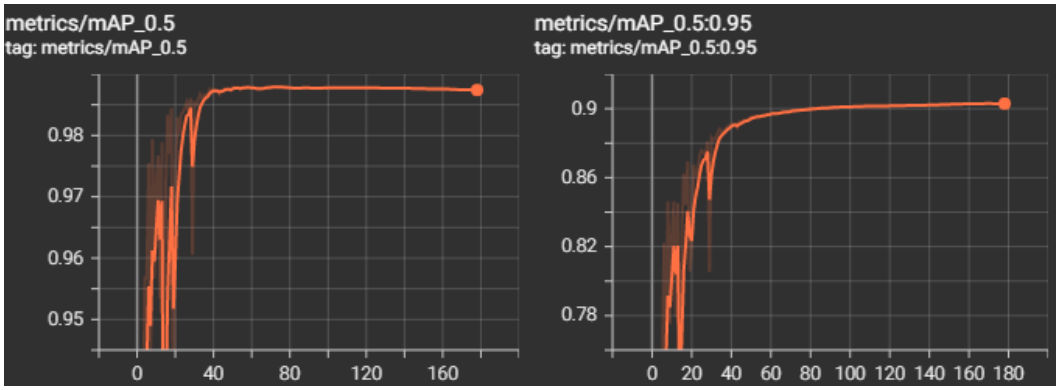


Fig. 3.13: YOLOv5 augmentation output.

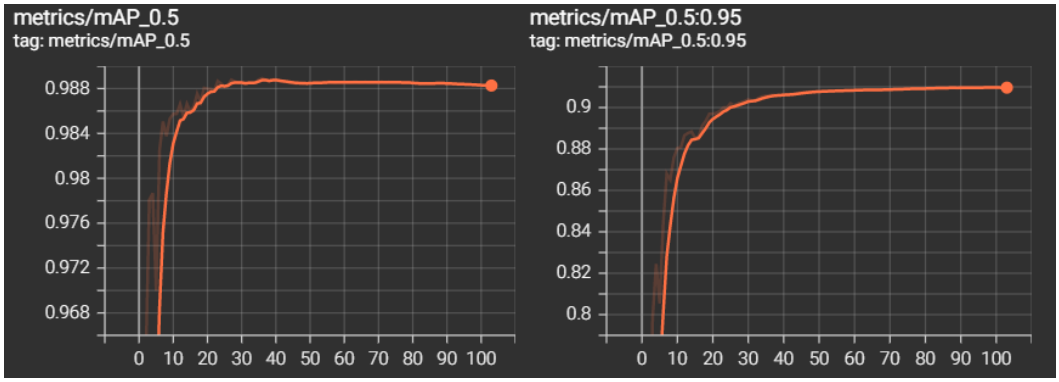
had to be manually tuned, meaning that a better configuration for the RetinaNet's anchors may have been possible.

3.1.7 Rule-based text block classification

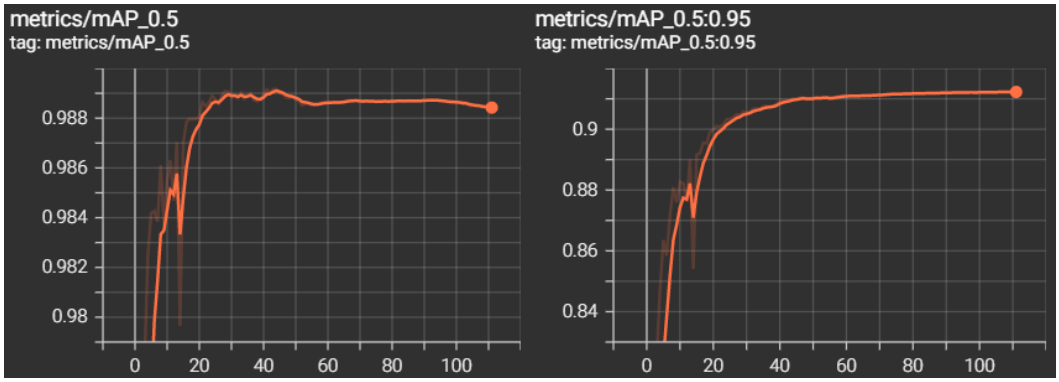
As previously mentioned, the initial data provided did not contain the appropriate annotations for a task so specific as to classify text blocks as paragraphs or titles. Thus the use of a rule-based approach was suggested by the company. These rules are to focus on the text contained in each bounding box. In order to extract the text, pytesseract OCR tool was used. After extracting and carefully examining a subset of the cases from a training set consisting of 50 document images, 7 rules were created with the addition of a default case. The rules are final, meaning that if a rule is satisfied, no subsequent rules are checked. These rules are described below.



(a) YOLOv5small.



(b) YOLOv5medium.



(c) YOLOv5large.

Fig. 3.14: mAP scores per epoch and per YOLO version.

1. **No text available**
   There are cases where pytesseract was not able to detect text. This is possible when the text is underlined, which is usually the case with titles. Of course the majority of underlined texts were correctly detected. It is also possible that some texts of the dataset belong to the anonymity class as explained in Subsection 2.1.1, making the text that much harder to read for pytesseract. The majority of the cases where text was not detected in a block, either due to the text being underlined or belonging to the anonymity class, were title cases and as such a title is assumed by the rule.

2. **More than 20 tokens**
   Keeping in mind the company's legal document guidelines and noticing that, from the extracted title text blocks of the training set no title extended more that 15 words, a rule was formed. This rule dictates than any text having more than 20 tokens (extra tokens were added for good measure) be considered a paragraph and not a title.

3. **Part of a list**
   Applying the above rule does not automatically classify the cases with fewer than 20 tokens as titles. There are cases of text being part of a list that belongs to a paragraph with as little as two words. The common attribute of list items is the use of brackets and a form of enumeration. More specifically, list items on the training set tend to appear with a letter or number as a first token followed by a right bracket. The existence of the right bracket is what distinguishes list items from titles. As such, a rule was created to detect right brackets in the first token of each text. If the rule is satisfied a paragraph is assumed.

4. **Fewer than 5 tokens**
   With the majority of small texts being classified by the previous rule, texts with fewer than 5 tokens are considered titles. Of course this rule, like the previous ones, stems from the documents of the training set. That also does not mean that every title has 5 or fewer tokens, but the rule manages to capture small titles well enough. From this point on, numbers, punctuation and stopwords are removed.

5. **Word capitalization**
   A common feature of titles, of the training set at least, is capitalization. The words of a title in these documents are either fully capitalized or have the first character of each word capitalized, with the exception of very few cases. With that in mind, a rule was created able to detect only whether the first character of each word in a text block is capitalized, as the capitalization of the rest of the word characters is irrelevant. If indeed this is the case, then a title is assumed. This rule is meant to capture a title, without taking into account the number of its tokens.

6. **Part of speech (POS) tagger**
   After removing possible stopwords, it is very uncommon for titles to include verbs (verbs removed as stopwords include the words is, be, etc.). As a result, a POS tagger implemented by Spacy was used. Spacy is a python library extensively used for advanced NLP. Its POS tagger's purpose is, given a sentence, to detect the part of speech each word belongs to. Thus, text blocks where no verbs are detected are considered titles. Its use proved very applicable for texts that were not classified up to this point.

7. **Default case**
   If no rules are applicable at this point it means that they are texts of length between 5 and 20 tokens that are most likely not titles. Every title is expected to have been detected up to this point, so the default case is that the text block that reached this point is a paragraph.

3.2 Text zone Classification

For the text zone classification task two classifiers were developed along with a baseline for comparison purposes. Their objective is, given the texts of a document's text zones, to classify each one as either "Cover page", "Introduction", "Table of contents", "Recitals" or "Main body". The text zones used for the training and testing of the model, were created as a result of manual annotation from the company's employees, or from a model able to split the document in text zones using regular expressions. The text zone classifiers' use is broader than that of the text block classifiers, meaning that they can be applied to many a task, stand-alone, but also work with some cases of the text block detecting models. Contrary to the object detectors though, the models developed for this task are strictly English oriented, per the company's request. The baseline is a combination of a TF-IDF vectorization approach and a logistic regression model. On the other hand, the two main models are a fine-tuned RoBERTa model and a hierarchical one that combines the outputs of the fine-tuned RoBERTa with stacked Recursive Neural Network (RNN) layers, which will be further explored.

3.2.1 Text zone classification baseline

For comparison purposes a baseline was created using a combination of TF-IDF vectorization and a logistic regression model. The purpose of the TF-IDF is to create an embedding of each text zone, by creating a vocabulary. For each word of a text zone, if the token is part of the vocabulary (the user may opt to keep a certain number of words in TF-IDF's vocabulary), the token is transformed in a numerical value based on its frequency in the text zone and the number of text zones that include the specific token. Having transformed

the zones to vectors of fixed size, they provide the input for the logistic regression model, which aim is to output the probability of a text zone belonging to each of the classes. Before the vectorization step, the text of each text zone was lowercased and stopwords were removed along with any punctuation to reduce the size of the vocabulary. The vocabulary size for the TF-IDF vectorizer was set to include 1000 tokens as it produced the best results on the validation set.

Training

For the model's training approximately 75.000 (60% of the dataset) were used for training and 25.000 for validation. To visualize the ability of the logistic regression model to learn from the training set, the model was trained multiple times using more data each time. As depicted in Figure 3.15, the model was trained using from 10%, up to 100% of the training data, with a step of 5% and measuring the macro f1 score of its predictions. The model seems to rapidly improve as the number of training instances increases, reaching 0.85 macro f1 score when using 100% of the data. From the visualization, it seems the model will not be able to improve much more given even more data.

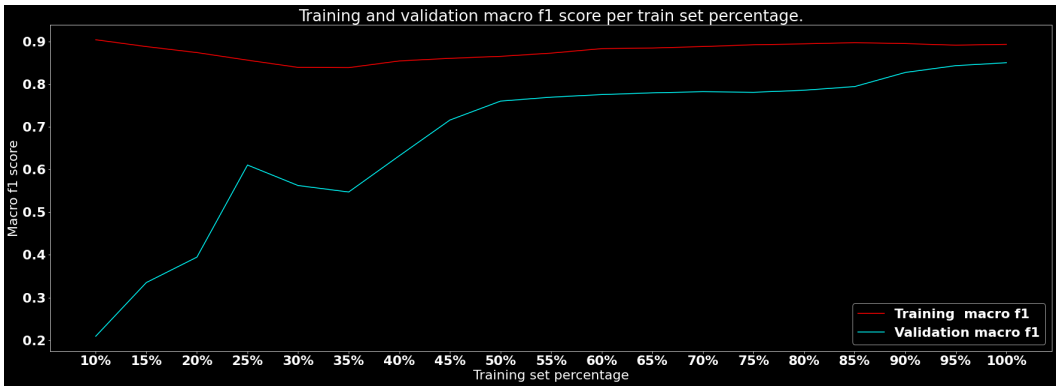


Fig. 3.15: Visualization of the text zone classification baseline's training.

3.2.2 RoBERTa

RoBERTa was chosen for the task due its combination of state of the art architecture and robust training, effectively increasing the capabilities of the model.

Architecture

RoBERTa is based on the BERT model [Liu+19], which is featured in Figure 3.16. BERT utilizes stacked Transformer blocks, which accept embeddings as input and with the use of several layers encodes the content. A Transformer block is comprised of a self attention mechanism, which transforms the input sequence to a new one giving weight on specific objects of the sequence. The input for the attention mechanism provides a combination of positional embeddings and the actual input. Positional embeddings are crucial, since without those, the model would not be able to keep track of the position of each word in context with the rest, as it processes the whole sequence. The new embeddings created

pass through a layer normalization block in order to decrease the amount of time it takes for convergence and stabilize the training process. Each embedding then passes from a fully connected layer to a new layer normalization stage. Residual connections are also present to help prevent the vanishing gradient problem that could cause the weights to stop being updated. The basic Transformer block is depicted in Figure 3.17.

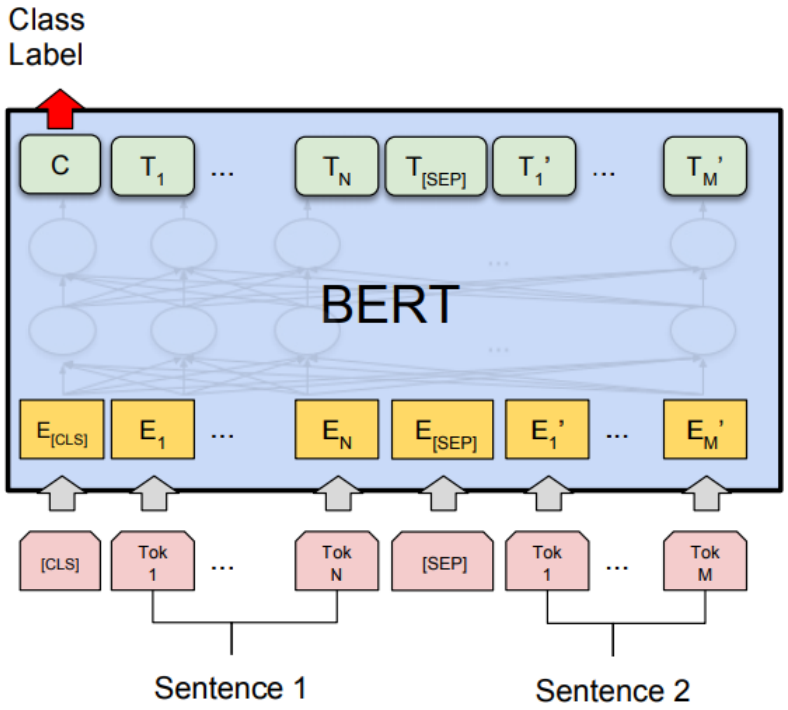


Fig. 3.16: The BERT model [Dev+19].

Training

Since the RoBERTa model is pretrained and in order to "specialize" the model to the needs of the task the fine-tuning approach was utilized. For this part, each section of the document was treated independently. The textual content of each block was preprocessed, so as to remove any punctuation except for dots. Dots were intentionally left in the text since their presence easily distinguishes some of the "Table of contents" cases with the rest of the labels. The text was also lower-cased. The preprocessed texts provide the input for the tokenizer, which maps each token to an embedding, that will later provide the input for the RoBERTa model itself. The output of the tokenized texts along with their labels can then be used to train the model. The model was set to be fine-tuned for 200 epochs with patience of 10 evaluations, using the same number of text zones as the baseline for training and validation. Each evaluation takes place after 100 steps. Steps are subsets of epochs and for the current case each epoch is comprised of 625 steps. The training process was completed at 4000 steps which is 6,4 epochs and the best state of the model was during step 3000 (epoch 4,8), barely reaching 0.95 macro f1 score. It takes 35 minutes on average for the model to be trained for an epoch (5.5 mins/100 steps). The results of the training

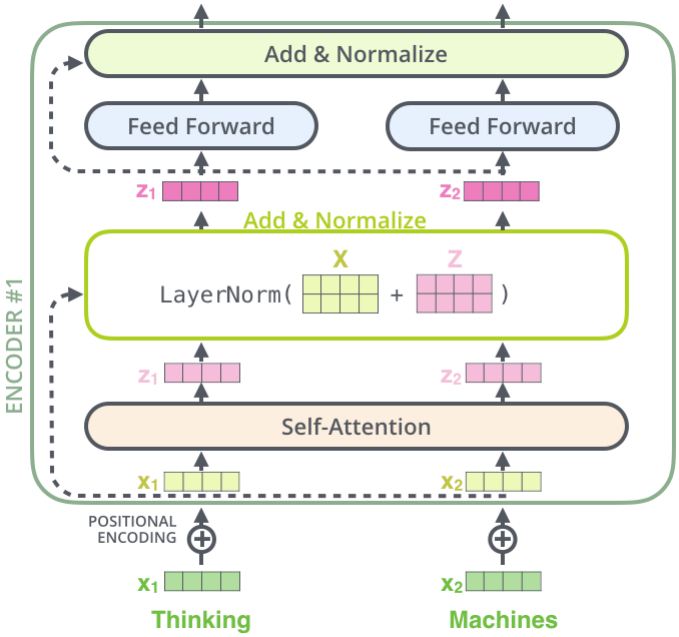


Fig. 3.17: A Transformer block [Ala20].

process are depicted in Figure 3.18. The evaluation suggests that the model was able to adjust easily and accurately to the task with minimal loss and high f1 score.

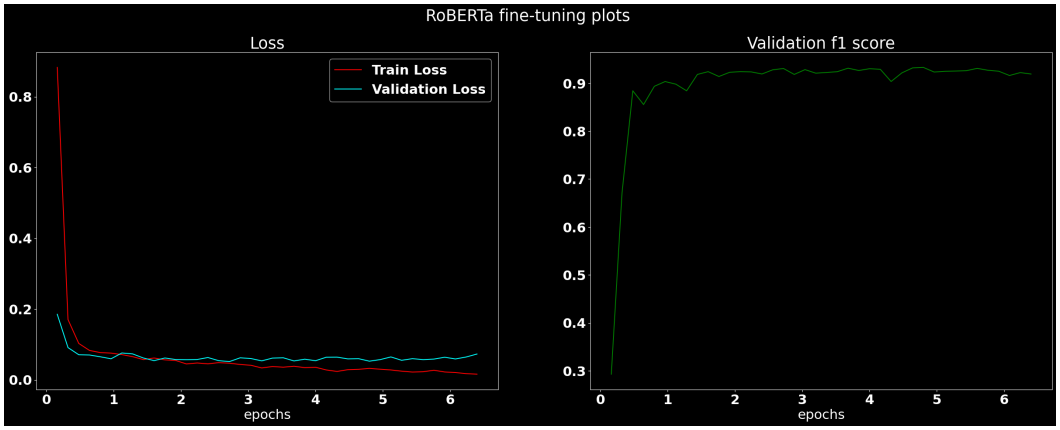


Fig. 3.18: RoBERTa fine-tuning loss and macro-averaged f1 score.

3.2.3 Hierarchical approach

In an effort to improve the results, a hierarchical approach was investigated, inspired by the work of Ilias Chalkidis et al. "Obligation and Prohibition Extraction Using Hierarchical RNNs" [CAM18]. Their work aimed towards the detection of contractual obligations and prohibitions in legal documents. For the completion of their task, two RNN models were involved. The first one was responsible for extracting an embedding for each sentence or clause. The second one having as input the results of the first RNN would try to predict classes with the hopes of capturing the context of the whole document to which it

succeeded. A similar approach was attempted for the purpose of the project's text zone classification task.

Architecture

For the role of the sentence encoder the already fine-tuned model was used. Instead of sentences though the input is still a text zone. Having embedded a text zone, the top model will try to classify it keeping in mind the context of its surroundings. The top model consists of two stacked Bidirectional LSTM (Bi-LSTM) layers of 512 and 256 nodes each. Each layer features a recurrent dropout of 0.2 and a layer normalization step as was the case with the Transformer blocks. Two fully connected layers follow with the purpose of classifying each text zone of the document. The first one includes 128 nodes with a ReLU activation function while the last features 5 nodes (one for each class) with a softmax activation function.

Training

In contrast to the RoBERTa's approach, which treats each text zone as input, the input of this model is based on documents. More specifically, the input comprises a 3D-array. The first dimension of the array represents a document, while the second represents a specific text zone of a document. Finally, the third dimension, represents the embedding of each text zone. For the 3D-array to be generated each input document needs to be pruned or padded. Four of the target classes belong to zones present in the beginning of each document apart from the "Main body" which makes up the rest of the document. Due to this fact and keeping in mind that the average length of each document is about 40 text zones, each document was pruned or padded to 40 zones. Some "Main body" zones were removed as a result, but that is of no consequence since the class of the last text zones can be inferred without the help of a model. After all, there is no point for an introduction to be at the end of a legal document. Fixing the size of each zone, the second dimension of the array was set. The size of the first dimension is irrelevant as the model accepts any number of documents (memory allowing). The third dimension is where the fine-tuned model comes into play. Each text zone is fed to the RoBERTa model from which a vector of size 768 is generated. This vector is effectively the last hidden state of the model that summarizes the context of the zone.

The training set includes the same training documents as the RoBERTa fine-tuning, this time with a different format. The top model is relatively lightweight and thus the training time for each epoch was 26 seconds on average. The training of the model was interrupted at 37 epochs from an early stopping callback, measuring epochs this time, meaning that the best weights were determined to be in epoch 27 as there wasn't any improvement in the validation set's macro-averaged f1 score after 10 epochs.

From Figure 3.19 it is obvious that the model managed to improve its predictions compared to the baseline, but most importantly there is an improvement compared to the RoBERTa

as well, having a macro f1 consistently above 0.95. What this means is that the model is able to predict the classes more accurately, after taking into consideration the surrounding text zones. Of course this model adds on execution time and memory usage which may contradict the needs of the company. Finally, although the model achieved a high macro f1 score from the very first epochs, there doesn't seem to be a major improvement in later epochs.

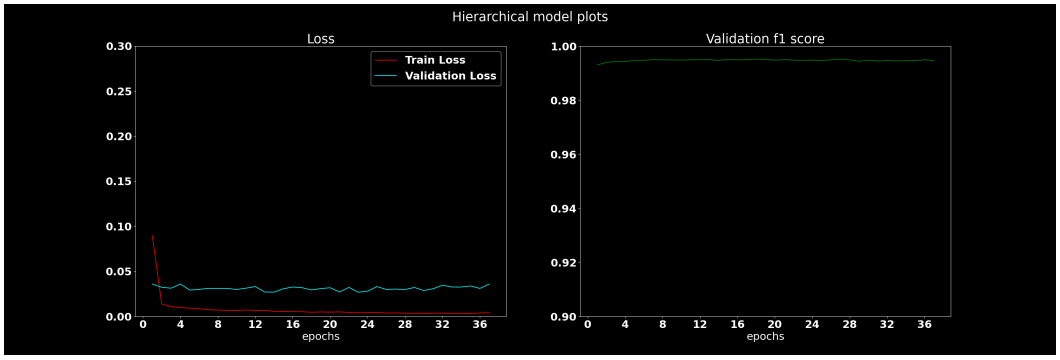


Fig. 3.19: Hierarchical model loss and macro-averaged f1 score.

4 Data Exploration and Experimental Setup

4.1 Data exploration

Prior to the creation of the models mentioned, a data exploration and analysis took place. Data exploration was crucial in order to efficiently design the custom models or define the parameters of the pretrained ones. Such an example was already given in Figure 3.9, where it was shown that RetinaNet was having trouble detecting the golden labels with the default anchor box settings. As a result they had to be set manually. This manual definition of parameters was a result of careful examination of the annotation features. This process will be explored in this section for both datasets used. Moreover, the groundwork for the model evaluation process will be set and explanations of the evaluation metrics used as well as the reasons behind their choosing will be given. Each dataset comes with its own characteristics since they stem form different sources. Moreover depending on the dataset and their intended use, different statistics were measured. Both datasets used, belong to Cognitiv+.

4.1.1 Text block detection

The dataset for the text block detection contains 27.314 legal document images along with their respective annotations. These are the data that will be used to train the text block detection models, whose aim is, given a document image, to detect regions where text blocks are present. In the dataset, out of the 12 labels in total, only the "anonymity", "text" and "form" labels are considered text blocks, as discussed in Subsection 2.1.1. The rest are considered backround. The annotations include the coordinates of each bounding box present in a document, the label of the bounding boxes, the width and height of the documents along with their skew angle. The coordinates of the bounding boxes are the x, y values of its four corners out of which only the top left and bottom right were kept. Furthermore, the skew angle was found to always be 0 so it was excluded. An example of the visualization of these annotations is featured in Figure 4.1.

While processing the annotations, 63 blank or unannotated documents were found, along with 282 erroneous bounding boxes (annotations with contradicting coordinates). An

example again is featured in Figure 4.1, where the "header" bounding box is not correctly defined. After the removal of such bounding boxes 250.567 bounding boxes remained in total. The remaining bounding boxes follow the distribution presented in Figure 4.2.

From Figure 4.2 it appears that the "text" class is overwhelming with more that 177.000 bounding boxes belonging to the class. Adding to the count the "anonymity" and "form" labels which are also considered text blocks, increases the count to over 190.000 boxes making the dataset extremely imbalanced. Nevertheless in computer vision balanced datasets are rarely the case, and as a result every model has its countermeasures as explained in Chapters 2 and 3.

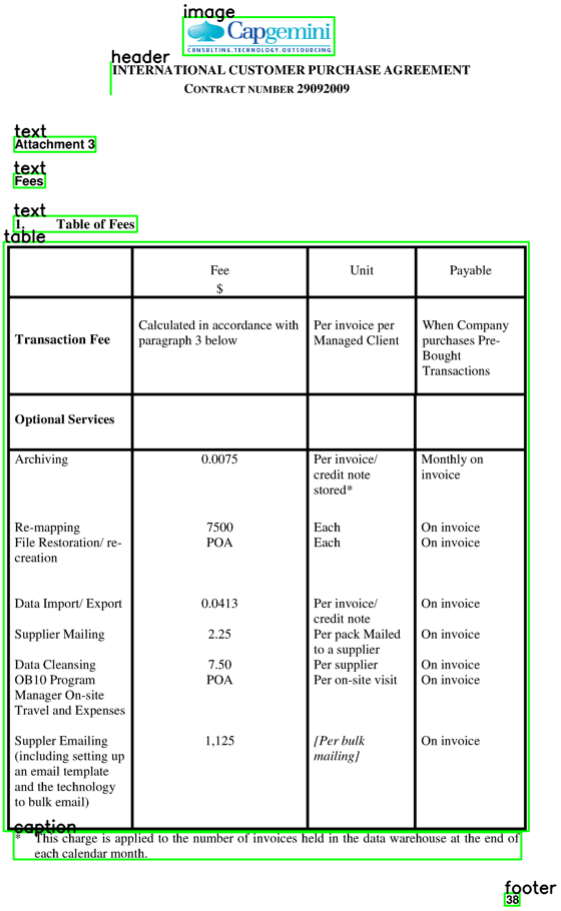


Fig. 4.1: An example of an annotated document, with its gold bounding boxes and an erroneous annotation belonging to the "header" class.

In Figure 4.3 the average count of each class in a document is depicted, where once again the text block classes are the most frequent ones. The "text" class appears 6 times per document image on average, whereas the rest of the classes are very rare. This is a result of the document structure. Labels such as "header" and "footer" appear at most once per document. On the other hand, "text" labels appear multiple times per image, as there are multiple paragraphs and titles. On top of that, the results hint to the existence of documents spanning over multiple pages, rather than a single document per image, as the

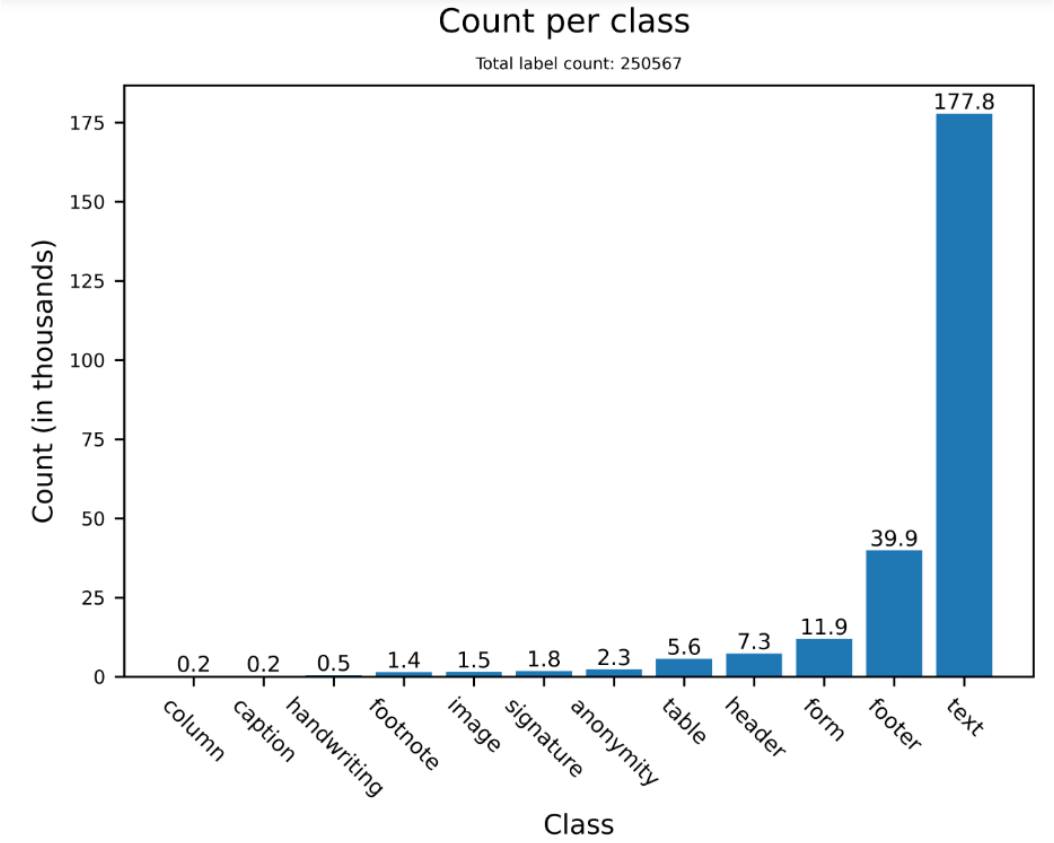


Fig. 4.2: Bounding boxes per class.

number of documents (27.314 in total) vastly outnumbers the header count, considering that each complete document has at least one.

Finally, on Figure 4.4 the average width and height of the bounding boxes are depicted. Regarding width, the length of each class is pretty standard is most cases. The "table", "footnote" and "text" classes come on top mostly because, if they are present, they usually cover the majority of the page's width, compared to the "header" class for instance, which may be smaller.

In terms of height the "table" and "column" classes come on top as expected, since a table of contents usually takes up the whole of the image as is the case with columns. In terms of the text block classes ("text", "anonymity" and "form"), although their medians, which are depicted as orange lines in each box of the same figure, are relative small, there are plenty of outliers, which is justified by their variety. For instance a small title and a large paragraph spanning half the image belong to the same class, regardless of their size. This may point towards a possible issue with small texts. For instance, titles, which are part of the text blocks are almost identical to headers, one of the 12 classes of the dataset, which is considered backround. Titles and headers can usually be told apart based on their positions on the document. Regardless, it may prove difficult for the models to make such a

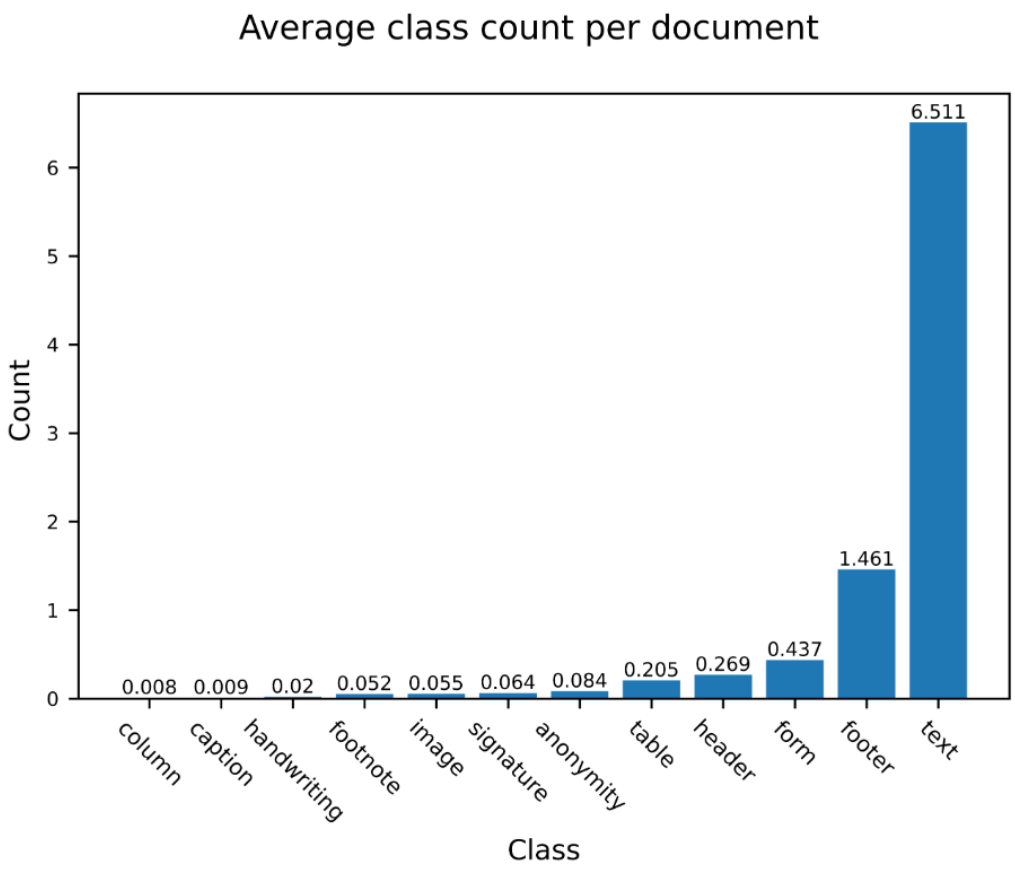
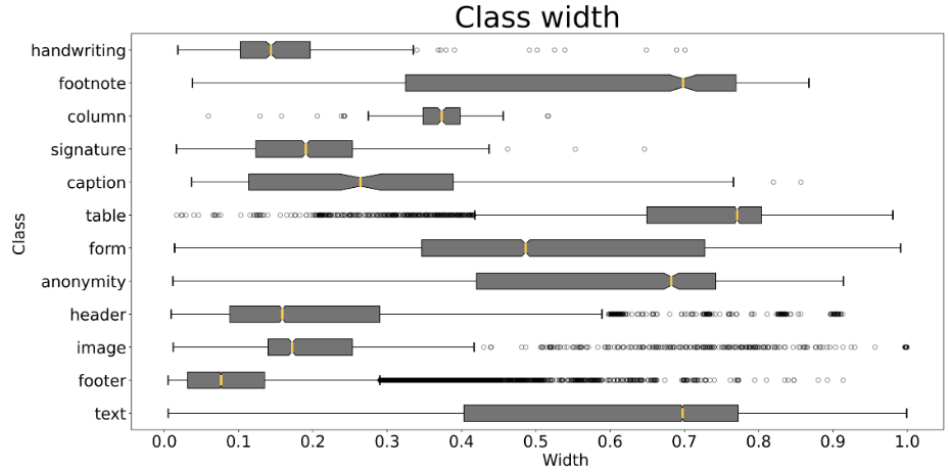


Fig. 4.3: Average class count per document.

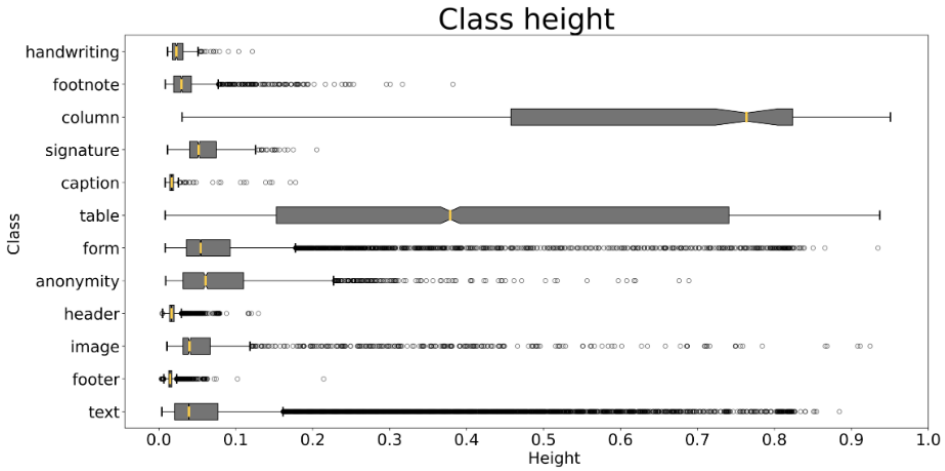
distinction. On top of that, small texts such as titles and headers are also present elsewhere in the footnotes or even images, which are also part of the backround. The models will have to make a distinction between them, based on their context. As explained in the YOLOv5's Subsection 2.1.2, according to Joseph Redmon et al. [Red+16], it is harder for two-stage detectors to make distinctions between backround and a true ROIs, as they are trained on parts of images rather than the images as a whole. On the other hand one-stage detectors, take into account the whole image, and thus are able to make prediction based on the whole context. This should make such distinctions easier for one-stage detectors, highlighting once again their choice for the task.

4.1.2 Text zone classification

The dataset used for text zone classification contains more than 2.500.000 text blocks, out of which 200.000 were used, spanning over 5100 complete documents, since the number proved to be sufficient for the needs of the task. The annotations of the dataset include the textual content of each text block, its label, the corresponding document id and its position in the document. The dataset contains texts, each one belonging to one of 36 classes. Each



(a) Average bounding box width.



(b) Average bounding box height.

Fig. 4.4: Bounding box width and height per class.

document consists of 36 text blocks on average (with a standard deviation of 45 text zones). Figure 4.5 depicts the average count of each class contained in each document.

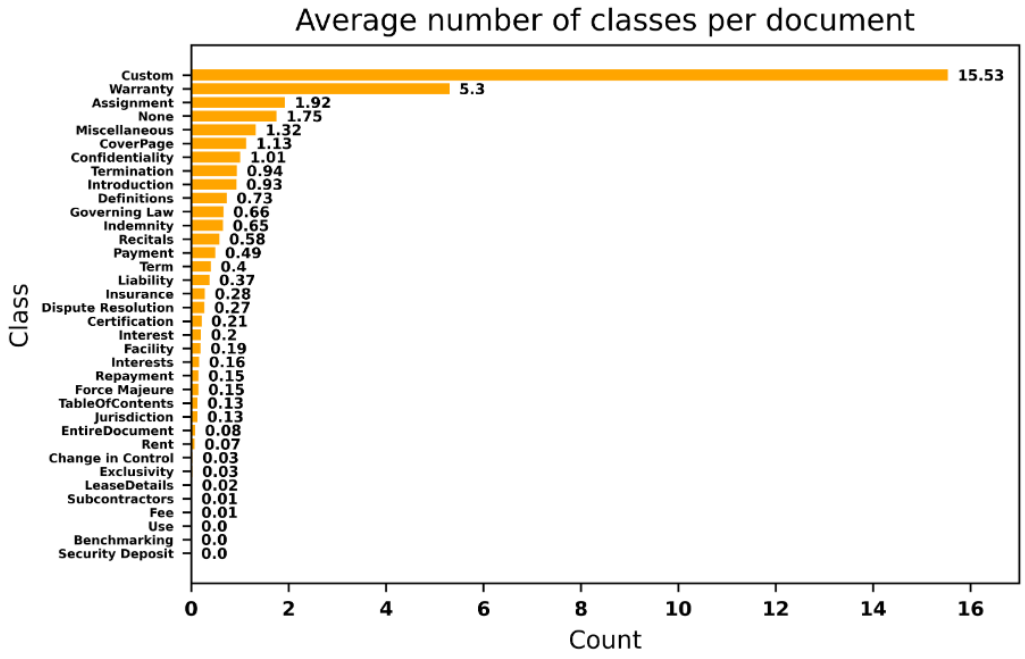
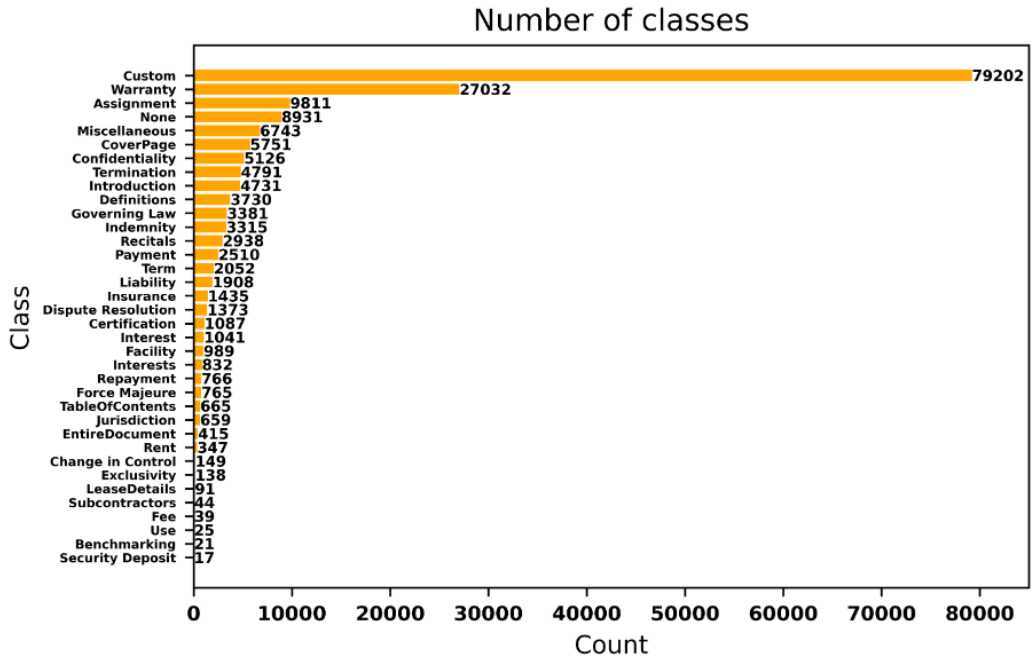


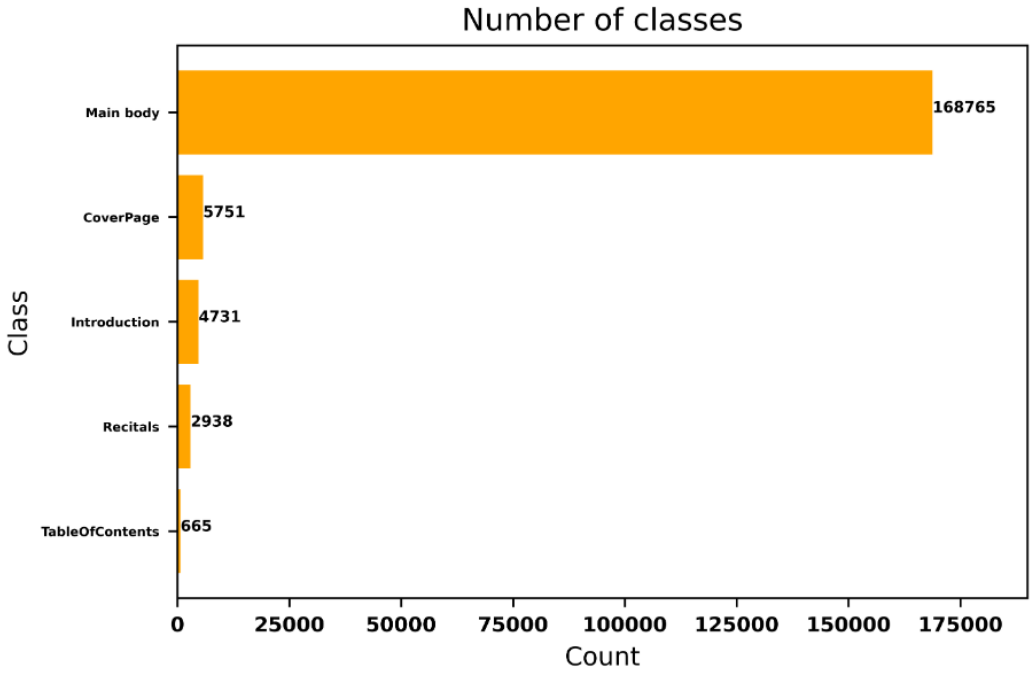
Fig. 4.5: Class count per document.

Out of the 36 classes representing the document, the models created are trained to predict 5; namely "Table of contents", "Recitals", "Cover page", "Introduction" and "Main body". Anything that does not belong to one of the first 4 classes is considered part of the "Main body" class. As such the resulting class distribution is compressed as depicted in Figure 4.6. It is obvious that the majority of each document's content belongs to the "Main body" class, and as a result this imbalance may pose a problem for the models. Since every class, apart from the "Main body" class, appears strictly as the first zones of the document, there is no point predicting the class of every single zone of a document. It was decided that every document be considered up to its first 40 text zones, keeping in mind that the average length of a document is 36 and since no other classes were present after that point, other than the "Main body" class. The rest of the zones would automatically be labeled as "Main body". As a result, the count of "Main body" zones was further reduced to 109.509 cases.

As per word count there are 19.141.439 tokens after stopword and punctuation removal from which 268.341 are unique. Every block contains on average 105 words (with a standard deviation of 332 words).



(a) Original classes count.



(b) Merged classes count.

Fig. 4.6: Class count before and after processing.

4.2 Evaluation metrics

Regarding the text block detection task, the models will be evaluated for both their prediction performance and speed. Their prediction performance will be reported in the form of mAP over multiple IoUs, as explained in Subsection 2.1.2. More specifically, there will be mAPs reported for the IoU thresholds 0.5 and 0.75 as well as the average of the thresholds between 0.5 and 0.95 with a step of 0.05. These metrics are popular in literature [Ren+15a; He+17; Red+16; WBL22; Lin+20; AWL20] aimed towards object detection and as such, they are present in this thesis. Moreover, the execution time for the inference of one image, on average, will be displayed. For the title and paragraph classification the rule-based approach will be evaluated through a macro f1 score. Macro f1 scores will be calculated for the overall performance of the NLP models as well, but f1 scores per class will also be reported.

4.2.1 Intersection over Union

IoU is calculated as shown in Formula 4.1.

$$IoU = \frac{A \cup B}{A \cap B} \tag{4.1}$$

Where A and B are bounding boxes. This calculation is strictly done between two bounding boxes. IoU can be present between two predicted bounding boxes, as is the case with the NMS algorithm (as explained in Subsection 2.1.2), or between a golden bounding box and a predicted one, as is the cases with the calculation of mAP. The output of the formula is always in a scale of $0 - 1$; 0 implies no overlap while 1, complete match. IoU encapsulates and compares many bounding box characteristics such as the location, width and height and is scale invariant. For all these properties IoU is favored and widely used as a metric in Computer Vision tasks [Rez+19].

4.2.2 Precision-Recall

Precision and recall values are not directly reported, but are crucial in the calculation of both f1 and mAP and as a result they should be mentioned. Precision and recall values are calculated as shown in Formulae 4.2 and 4.3:

$$Precision = \frac{TP}{TP + FP} \tag{4.2}$$

$$Recall = \frac{TP}{TP + FN} \tag{4.3}$$

Where TP (True Positive) is the amount of correctly classified positive cases by the model, FP (False Positive) is the amount of instances the model missclasified as positive and FN (False Negative) is the amount of instances the model missclasified as negative. Since the model outputs probabilities as predictions, these values need to be mapped to either 0 or 1. As such in order to calculate precision and recall, a threshold is defined and as a result the same predictions can have various precision and recall values. In other words by imposing a threshold, if a prediction probability is higher that the threshold then the prediction is mapped to 1. If not, the probability is mapped to 0. Having a range of thresholds results in multiple precision and recall pair values, that when plotted on x and y coordinate system, create the precision-recall curve (PR-curve).

4.2.3 mean Average Precision

Mean Average Precision, as the name suggests, is simply the mean value of Average Precision (AP) over the classes. AP represents the area under the interpolated PR-curve and is computed per class [HF17]. AP itself is calculated by obtaining precision and recall values for a range of thresholds by using Formula 4.4.

$$AP = \sum_{n=1}^{C} (R_n - R_{n-1}) * P_n \tag{4.4}$$

Where C is the number of thresholds that map the precision and recall values to either 0 or 1, R the recall values and P the precision ones at the nth threshold. The whole formula implies iteration over the thresholds in order to calculate precision and recall values. As AP is calculated per class, averaging the per class AP results in the mAP score. In order to calculate the mAP, an IoU threshold needs to be stated as well. In the case of mAP@0.5, the threshold's value is 0.5. This IoU value describes a threshold for the overlap between a prediction and its golden bounding box. If the IoU of the prediction is higher than the threshold, only then is the prediction considered correct. An example is provided in Figure 4.7. Finally, a mAP value can also be reported by averaging mAP scores over multiple thresholds, usually in the range of 0.5 - 0.95 (represented as mAP@0.5:0.95).

4.2.4 F1 score

F1 score is another metric that encapsulates both precision and recall values. Contrary to mAP though, an f1 score is computed per threshold. The threshold for any experimentation

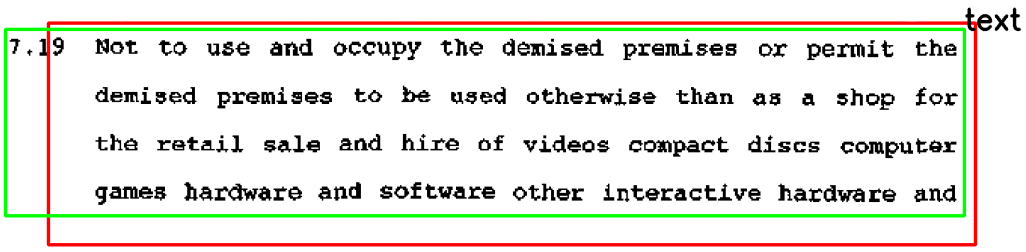


Fig. 4.7: A predicted (in red) and golden (in green) bounding box, with 0.7 IoU. The prediction will be considered correct when calculating mAP@0.5, but wrong when calculating mAP@0.75.

is fixed to 0.5, in this thesis. F1 score is the harmonic mean of precision and recall and is computed as depicted in Formula 4.5.

$$f1\ score = 2 * \frac{Precision * Recall}{Precision + Recall} = \frac{TP}{TP + 0.5 * (FN + FP)} \tag{4.5}$$

Since the datasets used feature high imbalance, a specific version of f1 score is used in which the f1 score is calculated as the average f1 of all the classes involved, called macro f1 score.

4.3 Experimental setup

In this section insight will be given over the evaluation process which will determine the most efficient models per task. For each task a different evaluation process will be followed. Regardless, the evaluation data upon which each model will be tested is the same for models build for the same purpose.

4.3.1 Text block detection

From the initial annotation dataset a subset was kept for testing purposes which is the same for every model. Extra attention was given to split the dataset so as to have a test set of images not included in any training or validation set. This set consists of 2000 images. For each text block detection model, 3 mAP scores will be provided using the formula mentioned in the previous section. Finally, the inference speed of each model for an image on average will be stated.

After determining the best model, an extra test case will be explored. For the most efficient model an identical one will be created that will be trained and validated on a more balanced dataset. The new model will be tested on two test sets. The original test case and a new more balanced one. The original model will also be tested in the latter test case. The balanced test set mentioned, features mostly far less text blocks compared to the original

dataset. In the data exploration section it was pointed out that the dataset contained annotations for classes which are not the points of interest. These classes can contain text just like the text blocks and are likely to prove difficult for the original model. For example, headers are similar to title text blocks, even though they belong to the backround, which may cause missclassification issues. The purpose of the balanced test is to determine if a more balanced training set can help improve the efficiency of the models in such cases. Finally, it also serves as a stress test for the original model since it is trained using a dataset in which the majority of the annotations belong to the "text block" class, which is not the case in the more balanced test set. That being the case though, it needs to be pointed out that this balanced test set, does not represent a viable scenario under any circumstances.

4.3.2 Text block classification

For the text block classification as either paragraph or title, since there were no data available, 255 cases were manually annotated.[9] Out of the 255 cases, 198 were annotated as paragraphs and 57 as titles. The efficiency of the rule-based approach will be determined on these cases, using a macro f1 score.

4.3.3 Text zone classification

Following the same principles as the text block detection, a subset of the dataset was withheld from the baseline, the RoBERTa and the hierarchical model. This subset features 40.000 text zones spanning over 810 documents. Out of the 40.000 zones, the models will predict the class of 22.741 as the rest will automatically be labeled as "Main body". The reason behind this decision, as explained earlier, is because of the structure of the documents. More specifically, the "Introduction","Recitals", "Cover page" and "Table of contents" labels never appear in the end of the documents, so the last zones can be safely considered as being part of the "Main body" class. RoBERTa and the hierarchical model have different architectures and input and output formats. As a result the structure of their test sets is different but they include the same documents regardless. For the evaluation of the models an f1 score will be calculated per class along with a macro-average f1 score taking into account all the classes.

[9]These cases were annotated by the author, who is not a specialist on the specific field, and as a result few of them may be wrongly labeled.

5 Results and Error Analysis

5.1 Text block detection

As previously mentioned, there are two distinct tasks, namely the text block detection and classification and the text zone classification. Since the text block detection and classification task is accomplished using two separate approaches, there are two separate test case. The results of these test cases along with the case of the text zone classification task are presented in this section. To satisfy the curiosity of performance on a more balanced dataset an extra test case is presented featuring the YOLOv5medium model.

5.1.1 Model results

In Table 5.1 the results of each model featuring the metrics mentioned in Section 4.2 are depicted. Starting with the baseline, the mAP@0.5 score can be regarded as being satisfactory, considering that the region proposals stem from a purely rule-based approach. On the other hand, for the rest of the mAP values, the difference is considerable. Keeping in mind that mAP is calculated by taking into account the predicted bounding box position, an issue arises, as the region proposal algorithm of this case is not able to be trained on golden bounding boxes coordinates. This means that, it will never be able to adapt to the coordinates of the golden bounding boxes the same way that a trainable model would. This is attributed to the regions the model is trained to predict as positive. Looking back at Subsection 3.1.3, only the text block regions with IoU higher than 0.6, with their respective golden bounding boxes, are considered positive. The majority of the resulting positive cases though have an IoU of about 0.65 with their respective ground truth. In the case of mAP@0.5 where the IoU level is 0.5, the majority of the predictions were correct. On the other hand, as the IoU level rises higher that 0.7, the correct predictions are rapidly declining, leading to the case of IoU 0.9 where no prediction was considered correct, as no prediction managed to surpass an IoU threshold of 0.9 with the golden bounding boxes. This results in a very low mAP@0.75 and mAP@0.5:0.95 as seen in the table. In terms of inference, since this is a case of a two-stage detector, the performance was expected to be considerably slower than the one-stage ones.

Looking at the YOLO models, they all feature similar performance. YOLOv5medium surprisingly managed to outperform the YOLOv5large model for both mAP@0.5 and

Model	mAP@0.5	mAP@0.75	mAP@0.5:0.95	inference time
Baseline	0.888	0.790	0.775	28389 ms
YOLOv5small	0.969	0.965	0.920	2.3 ms
YOLOv5medium	0.978	0.974	0.931	5.9 ms
YOLOv5large	0.971	0.968	0.931	10.9 ms
RetinaNet	0.97	0.964	0.922	73.6 ms
LP+CNN	0.791	0.746	0.703	44187 ms

Tab. 5.1: Text block detection results.

mAP@0.75 but they both have the same mAP0.5:0.95 value. This means that the large model, even though it makes slightly more mistakes than the medium one in terms of predictions, managed to capture more closely the golden boxes' coordinates in higher IoUs, leading to the increase of the correct predictions in these IoUs. In such cases some predictions of the medium model were considered wrong due to the insufficient IoU value of its predictions. In terms of speed, YOLOv5small managed to score the lowest inference time at 2.3 milliseconds. In comparison the medium and large model feature two and four times the inference time respectively. As expected, all of the models come easily on top compared to the baseline.

RetinaNet managed to slightly outperform YOLOv5small which is surprising considering that YOLOv5small was trained using three times the amount of data. This may hint to the fact that with more training data and better anchor configuration (if a better configuration does indeed exist), RetinaNet could have performed a lot better, probably surpassing the top YOLOv5 model. Regardless, it wasn't able to match the performance of the other two models. Finally, there is a huge difference between the YOLO models and RetinaNet's inference time which is unsurprising since YOLO models are famed for their speed.

Lastly, the combination of LayoutParser and CNN (depicted as LP+CNN in Table 5.1) performed the worst in every case, outperformed by the baseline model, even though they featured the same CNN architecture. This underlines the importance of training the object detector using the images provided by the region proposal architecture as was the case with the baseline. If that was the case, most likely, the model would have outperformed the baseline, considering that the region proposal step is done through a pretrained model rather than a rule-based approach.

5.1.2 Special test case

The class imbalance of the dataset has been mentioned several times. In order to compare the results of one of the best performing models given a more balanced dataset for training and the original one, two scenarios are further tested. These scenarios revolve around the use of a balanced test set and the original one. The training, development and test sets

Model	mAP@0.5	mAP@0.75	mAP0.5:0.95
YOLOv5medium	0.978	0.974	0.931
balanced YOLOv5medium	0.863	0.859	0.835

Tab. 5.2: Balanced and original model comparison on the original test set.

Model	mAP@0.5	mAP@0.75	mAP0.5:0.95
YOLOv5medium	0.745	0.710	0.679
balanced YOLOv5medium	0.78	0.734	0.691

Tab. 5.3: Balanced and original model comparison on the balanced test set.

used for the balanced model, are all balanced subsets of the balanced dataset. For the sake of comparison, the models used are the medium YOLOv5 and the same model trained on the balanced training set. In tables 5.2 and 5.3 the two scenarios are depicted.

In the case of the original test set, the balanced YOLOv5 model performed poorly. Having been trained on a more balanced set and thus predicting more cases as backround compared to the original model, it is not able to detect many text blocks, which are more dominant on the specific set, making the balanced model unsuitable for such a case.

When comparing the models on a more balanced yet non-realistic test case, the balanced model outperforms the original YOLO model. One of the main obstacles for the model regarding the task is making a distinction between bounding boxes that contain text and are considered text blocks and bounding boxes with text that are considered background. The balanced model seems to be able to make such a distinction easier, compared to the original model, as it was given a more balanced training set. Nevertheless, no model performed to a satisfactory level.

In conclusion, the results of the models in the balanced case are not far off. Furthermore, it has to be noted that the balanced test set scenario is most unlikely as the number of text blocks in an actual complete document is much larger. When it comes to the original and realistic test case, it is the original YOLO model that performs better, meaning that a balanced training set does not necessarily imply a better perfromance. As a result, the use of the balanced model is not recommended.

5.1.3 Error Analysis

Every Computer Vision model developed is only able to distinguish between text blocks and background. The number of correct and false text block and backround predictions of the models featured in the error analysis will be presented through a confusion matrix. Nonetheless, since the test data have annotations of their own, it is possible to further

analyze the positive cases predicted by the models, and pinpoint the annotation label of each one. These annotations include the "text", "anonymity" and "form" labels which are treated as text blocks and "handwriting", "footnote", "column", "signature", "caption", "table", "header", "image" and "footer" labels that are treated as backround. Every class includes a form of text regardless. In order to determine regions in which the models confuse backround text as being text blocks, each prediction was labeled according to its golden bounding box. This was done by comparing each prediction with the golden bounding boxes, given an IoU threshold. If the IoU between a prediction and a golden bounding box was higher than the threshold, the predicted box was given the golden box's label (backround classes included). For the purpose of comparison, the IoU threshold was set to 0.5. Based on the results, a barplot per model will also be presented, along with the respective confusion matrix, showing the number of prediction labels compared to the number of the golden ones. It is important to note that a prediction that is associated with a golden bounding box by means of IoU, shares the same label with its golden counterpart. As a result, there are no misclassifications on that part.

In the barplots mentioned, the total count of each golden label is depicted in orange, while the predicted, in grey. Left of the dashed red line on each barplot, the text block labels are presented, where the grey bars represent the true positive cases. Similarly, the difference between the number of the golden and predicted cases in the text block classes, equals to the false negative cases of the corresponding confusion matrix. It should be noted that there are no true negative cases considered, as the backround consists of anything that is not a text block (not just the dataset's backround classes) and thus the number of such cases cannot be accurately calculated. On top of that, true negative instances are not taken into consideration when calculating mAP scores per class. As a result the number of these cases are depicted as 0 in every confusion matrix, on purpose. Moreover, the false positive cases that were mapped to a golden annotation are depicted as grey bars on the right of the dashed lines. Their total count is only a subset of the total number of false positive cases, as there were more falsely predicted bounding boxes that were not mapped to any dataset bounding box.

Finally, the error analysis will revolve around the YOLOv5 and RetinaNet models (the models that the company requested), in order to shed some light on usual errors made, the exploration of which may lead to improvements in the future. An error analysis regarding the special test case scenario will also take place for comparison purposes between the two YOLOv5 models involved.

YOLOv5medium

In Figure 5.1 the results of the error analysis regarding YOLOv5medium are depicted. The confusion matrix of the figure, shows that the YOLOv5 model has predicted correctly the vast majority of text block cases with minimal false negative predictions, making it an effective model. Only a few text block cases weren't predicted correctly, most likely

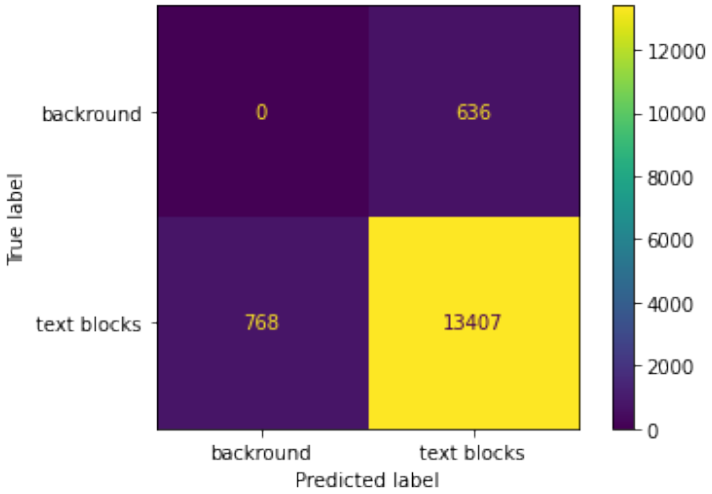
belonging to irregular cases such as the one depicted in Figure 5.2 where the annotated text blocks start with a number in a similar format as a footer while being inside a table. The header seems to be the backround class that the model struggles with the most, as seen from Figure 5.1b. Misclassifications such as these though, are to be expected since title text blocks and headers are very similar, with their distinguishable feature being mostly their position in the document.

RetinaNet

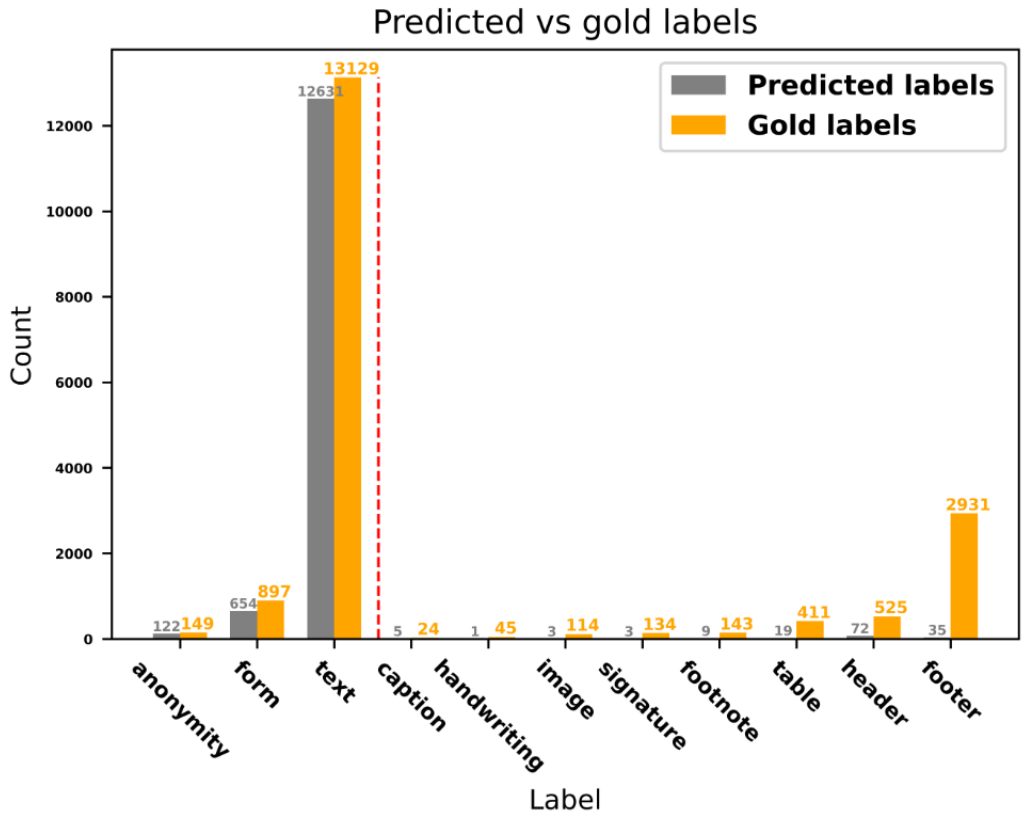
RetinaNet managed to match the results of YOLOv5small, as was pointed in Subsection 5.1.1. RetinaNet was trained at 1/3 of the data due to the prohibitively long training time, which may be one possible explanation. Nevertheless, the results as seen from Figure 5.3 are similar to those of the YOLOv5medium. There are many backround labels which RetinaNet was able to differentiate from the text block labels. More specifically, there are less "caption", "footer", "header" and "image" bounding boxes but only for a small margin, compared to YOLO, which was not able to sway the results. RetinaNet also managed to predict more accurately the "form" label compared to YOLOv5medium, although the same cannot be said for the "anonymity" and "text" classes. Regarding Figure 5.2, RetinaNet was not able to detect any text blocks, as was the case with YOLOv5medium, probably hinting once again to the existence of some rare irregular cases, that the models have not encountered before during training. Overall, RetinaNet came close to the YOLOv5 model's number of true positive case, but made more false negative errors. Surprisingly enough though, RetinaNet's made less false positive mistakes, compared to the YOLOv5's case. These are promising results, possibly suggesting that with more training instances RetinaNet may be able to outperform YOLOv5.

Original case vs. balanced case

In the case of the tests depicted in Figure 5.4, a comparison between the original YOLOv5 model and a more balanced version is depicted on a more balanced test case. The results of the balanced version are slightly better but still there is a number of undetected golden bounding boxes, as seen from the difference between the numbers of predicted and golden bounding boxes of the text block labels, that cannot be disregarded. These undetected golden bounding boxes are the false negative cases depicted in each confusion matrix in Figure 5.5. After the examination on a subset of the results, it appears that their difference stems mostly from mistakes made in text block predictions belonging to the "text" class that are present in irregular positions like the top or bottom of the image, as is the case with Figure 5.6. It seems, in the training set, the text blocks of such type are scarce, compared to the number of text block instances overall. As such the original YOLOv5 model, most likely, ended up disregarding the upper right text block presented on this example as a header, judging from its position. On the other hand, the balanced model managed to make a distinction between text blocks in irregular positions, thus predicting the case correctly. The confusion matrices in Figure 5.5, show the reason why the mAP values of Table 5.3 are so low. Both models seem to misclassify many backround regions as being



(a) YOLOv5medium confusion matrix.



(b) YOLOv5medium label prediction frequency.

Fig. 5.1: YOLOv5 result plots.

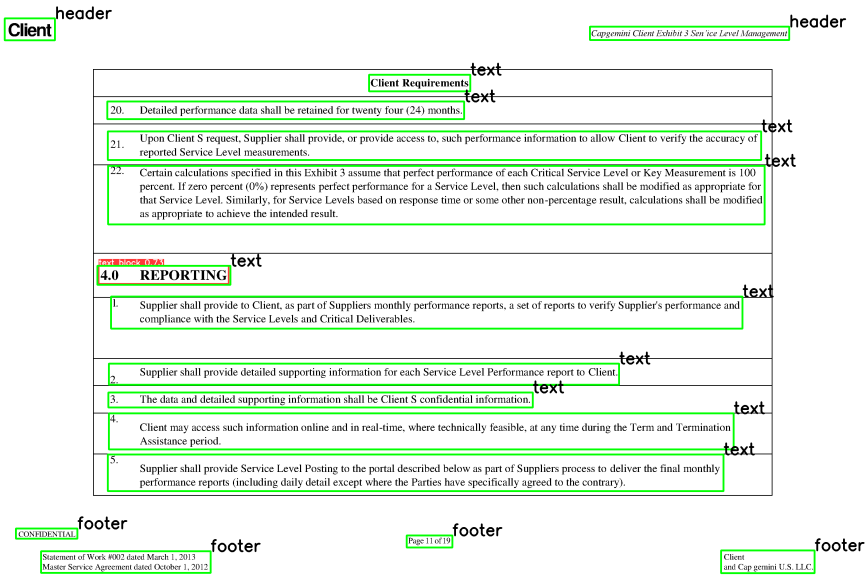


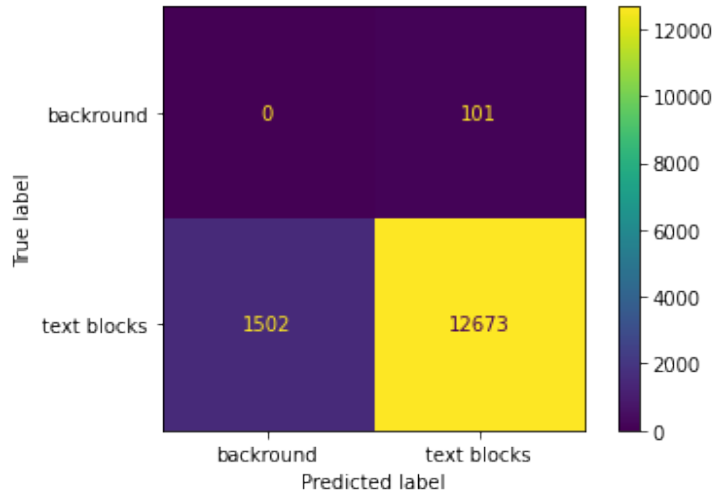
Fig. 5.2: An irregular case of YOLOv5medium predictions versus the golden boxes (predictions are in red). Every "text"-labeled bounding box should have had a corresponding red bounding box.

text blocks, probably hinting that some backround and foreground texts are in reality very similar. Overall, given the current test scenario, the balanced model performed better, but nevertheless, as it has been stated several times, this test is not to be taken as a possible scenario and thus the use of the balanced model is not recommended.

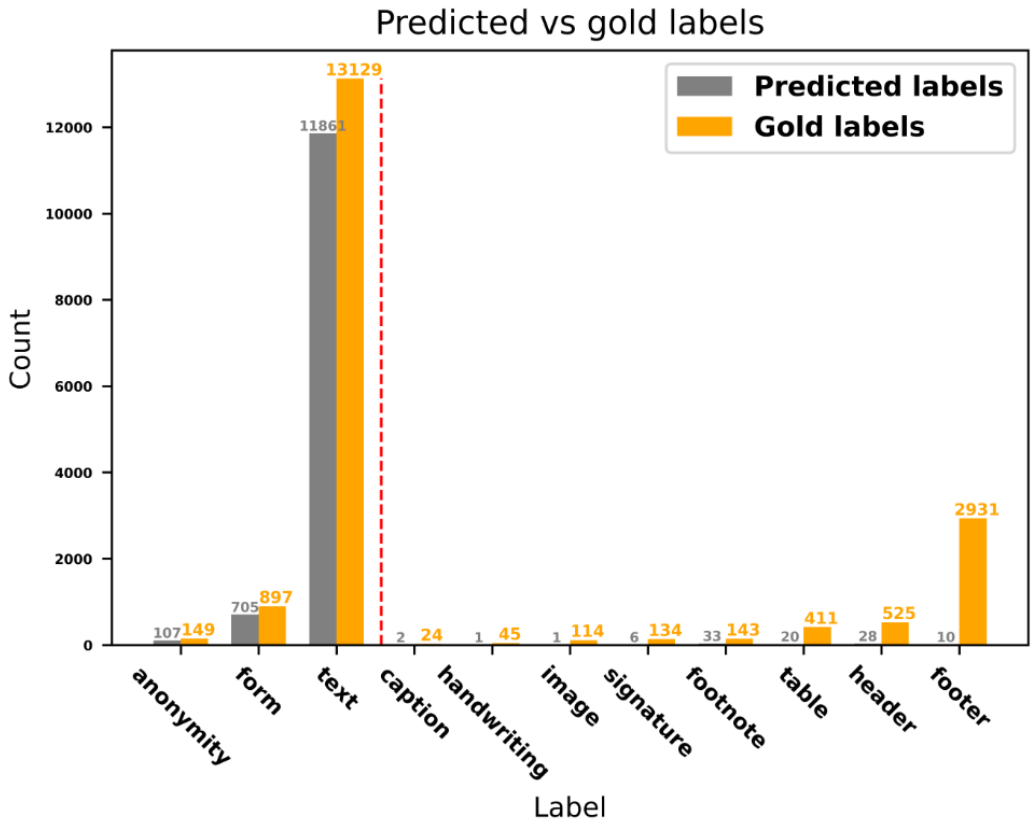
5.2 Text block classification

Regarding the text block classification test case, since the results are produced by the application of rules, there are no prediction probabilities. As a result, the performance overview is based solely on a macro f1 score. Applying these rules to 255 cases yielded a macro f1-score of 0.92. A confusion matrix is presented in Figure 5.7 showing the number of correctly classified cases and mistakes. Label 0 represents titles while label 1 represents paragraphs.

The paragraph misclassification cases are mostly attributed to the default rule case, meaning that more rules for titles may be needed. In terms of title misclassifications the usual cases are presented in Figure 5.8. In case 5.8a the "form" text blocks were labeled as paragraphs, but through the use of rules they were classified as titles. This means that new rules need to be made to specifically target the "form" regions. Nevertheless, the way each company creates their forms is different and a common characteristic, that can be detected by pytesseract, is difficult to be found. As a result, the use of rules targeting the "form"

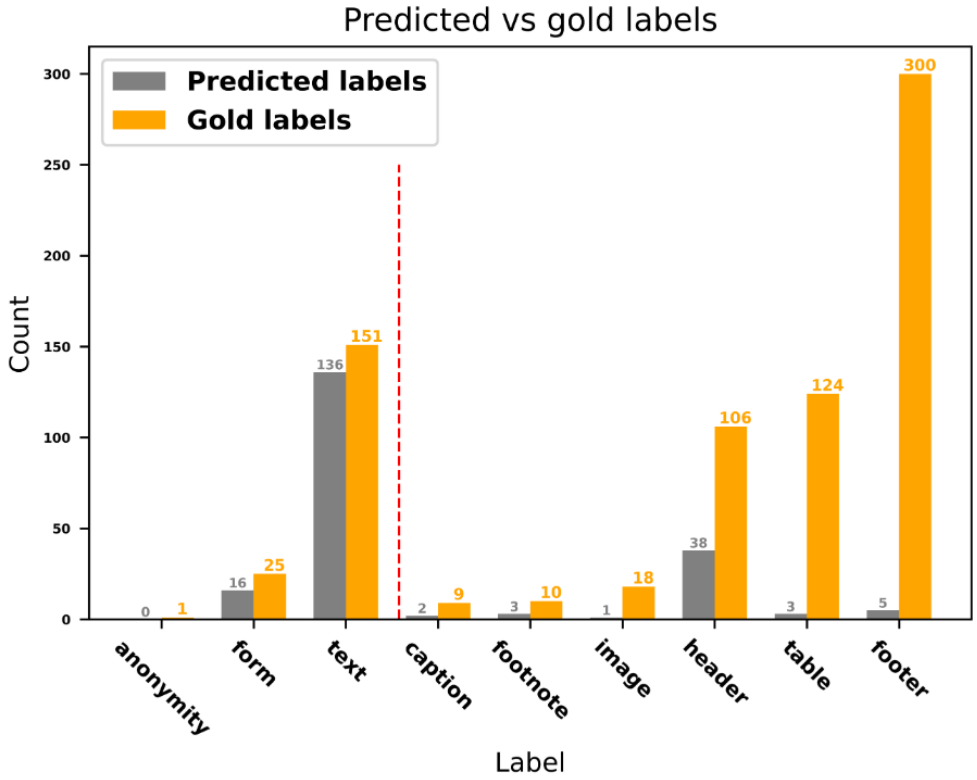


(a) RetinaNet confusion matrix.

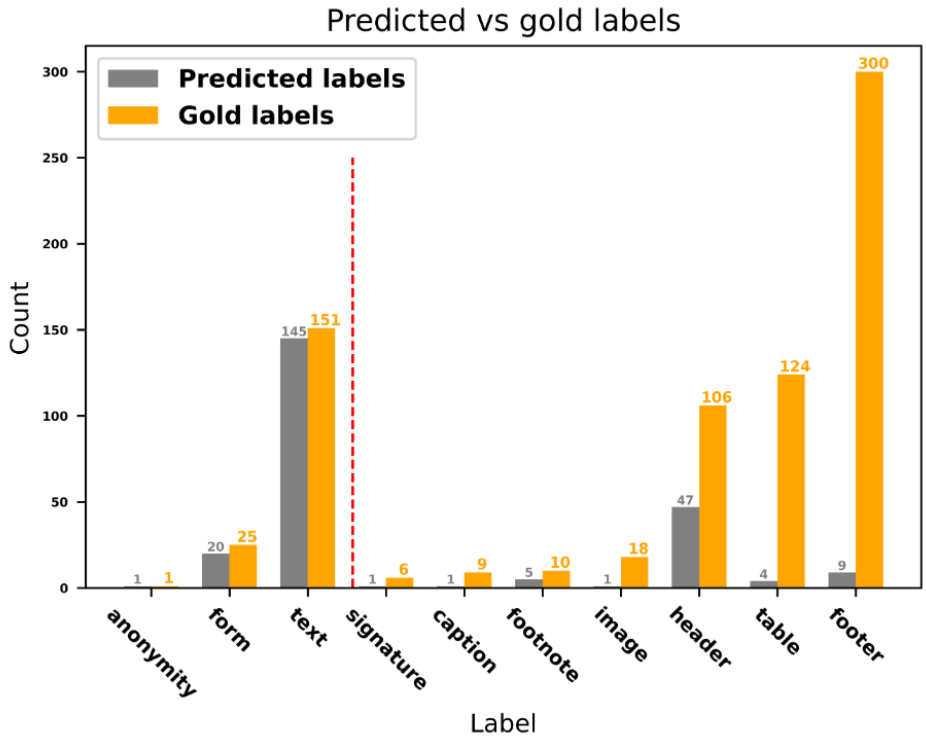


(b) RetinaNet label prediction frequency.

Fig. 5.3: RetinaNet result plots.

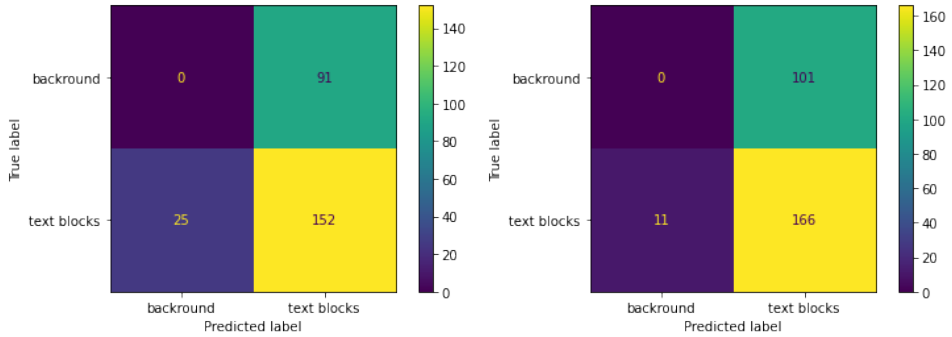


(a) Original YOLOv5medium label prediction frequency.



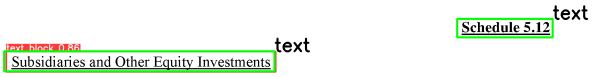
(b) Balanced YOLOv5medium label prediction frequency.

Fig. 5.4: Comparison of the balanced and original YOLOv5 models on a more balanced test case.

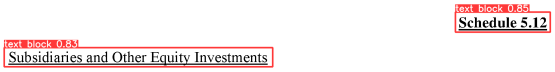


(a) Original YOLOv5medium confusion matrix. **(b)** Balanced YOLOv5medium confusion matrix.

Fig. 5.5: Confusion matrix comparison of the balanced and original YOLO models on a more balanced test case.



(a) YOLOv5medium predictions (in red) and golden boxes(in green).



(b) Balanced YOLOv5medium predictions.

Fig. 5.6: Comparison of the balanced and original YOLO models on a test case. The top right block was regarded as backround by the original model, but was correctly predicted by the balanced one.

label may not be optimal. In case 5.8b, after examining the text that pytesseract detected, it was apparent that pytesseract considered it to be a series of the characters "o" and "e". This means that, not having satisfied the previous rules, the detected text reached the POS rule, that checks whether the string contains verbs, as explained in Subsection 3.1.7. The string, not containing a verb, was falsely regarded as a title. Even if a text was not detected it would still be misclassified as according to the first rule, text blocks whose text is not detectable are classified as titles. As a result, such cases of anonymity, although rare, may also pose a problem. Overall though, the majority of titles and paragraphs were detected by using the token counter rules. It would seem that classifying texts just by checking whether they contain more that 20 tokens (paragraphs), or fewer than 5 tokens (titles), the majority of the text block were correctly classified, which greatly contributed to the increase of the f1 score.

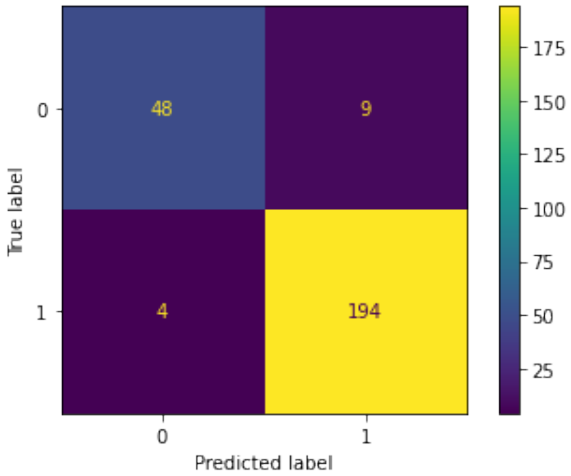
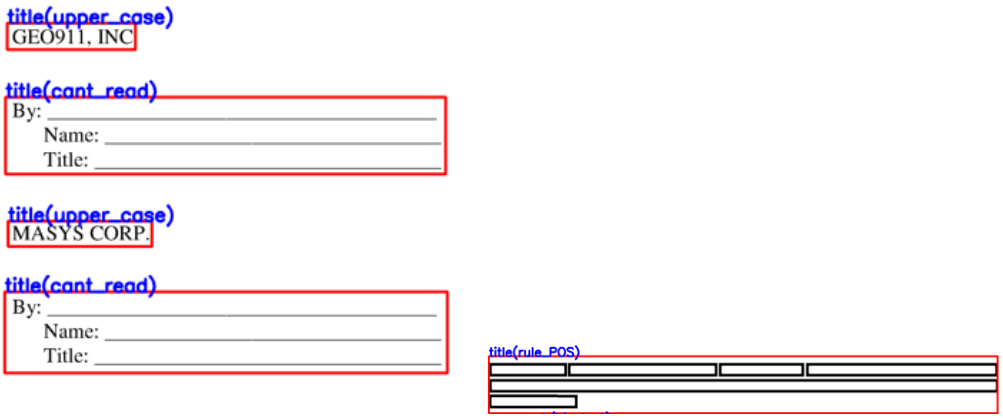


Fig. 5.7: Rule-based predictions confusion matrix.



(a) Title misclassification of a form text block.

(b) Title misclassification of an anonymity text block.

Fig. 5.8: Rule-based approach title misclassifications.

Model	macro f1	Cover Page f1	Introduction f1	Recitals f1	Contents f1	Body f1
Baseline	0.866	0.858	0.806	0.814	0.867	0.987
RoBERTa	0.955	0.940	0.924	0.971	0.944	0.995
Hierarchical	0.967	0.951	0.953	0.980	0.955	0.997

Tab. 5.4: Text zone classification baseline, RoBERTa and Hierarchical model comparison.

5.3 Text zone classification

For the purpose of the text zone classification task a hierarchical model and a RoBERTa, alongside a simple baseline were tested. Due to the input of each model (text zones vs. documents) the test sets used for each one revolve around the exact same documents but are different in terms of format, as was the case with their training sets. For example, as the RoBERTa accepts and classifies text zones directly, the test set remains untouched. The same goes for the baseline model. On the other hand the hierarchical model requires a document as input with a fixed number of text zones. For that to happen some text zones were cut or padded for both the hierarchical model's training, validation and test set. All the excluded text zones represent "Main body" cases which are easy to infer, given their position and as a result do not degrade the performance of the model.

5.3.1 Model results

Both RoBERTa and the hierarchical model performed to a satisfactory level, outperforming the text zone classification baseline with ease, but the hierarchical is the one that came on top, as presented in Table 5.4. Not only did the hierarchical model achieve the best macro-averaged f1 score, it also managed to surpass RoBERTa in every class. Of course this is to be expected, since the hierarchical model builds upon the RoBERTa model, and utilizes its context-aware representations of the text zones in each document to make more accurate predictions.

5.3.2 Error Analysis

Similar to the error analysis of the text block detection task, the error analysis will revolve around the models which the company is interested in. Once again an effort will be made to determine and possibly give an explanation for the most frequent errors.

RoBERTa

In Figure 5.9, the RoBERTa predictions' confusion matrices are depicted. The false negative cases are the most frequent type of mistake in all classes but the "Main body" for this model. Careful examination of errors suggests that most of the mistakes were cases of

"Main body" misclassification. This is to be expected since the number of "Main body" instances is greater, but there are more cases of concern than that. There were cases that included very small texts that cannot be distinguished easily or at all. Such cases include the phrases "dated september 10 2007", "americredit financial services inc. americredit", "definitive purchase agreement" and more, which are all part of the "Main body" class, but such cases for other class do exist, making them very hard to classify correctly. On top of that after testing, possible cases of mislabeled data were found. An example includes a whole table of content being labeled as "Main body". Such cases count towards the number of mistakes made by the model. Per the mislabeled cases, even though their number cannot be easily estimated, judging from the f1 scores the model performed to a satisfactory level regardless.

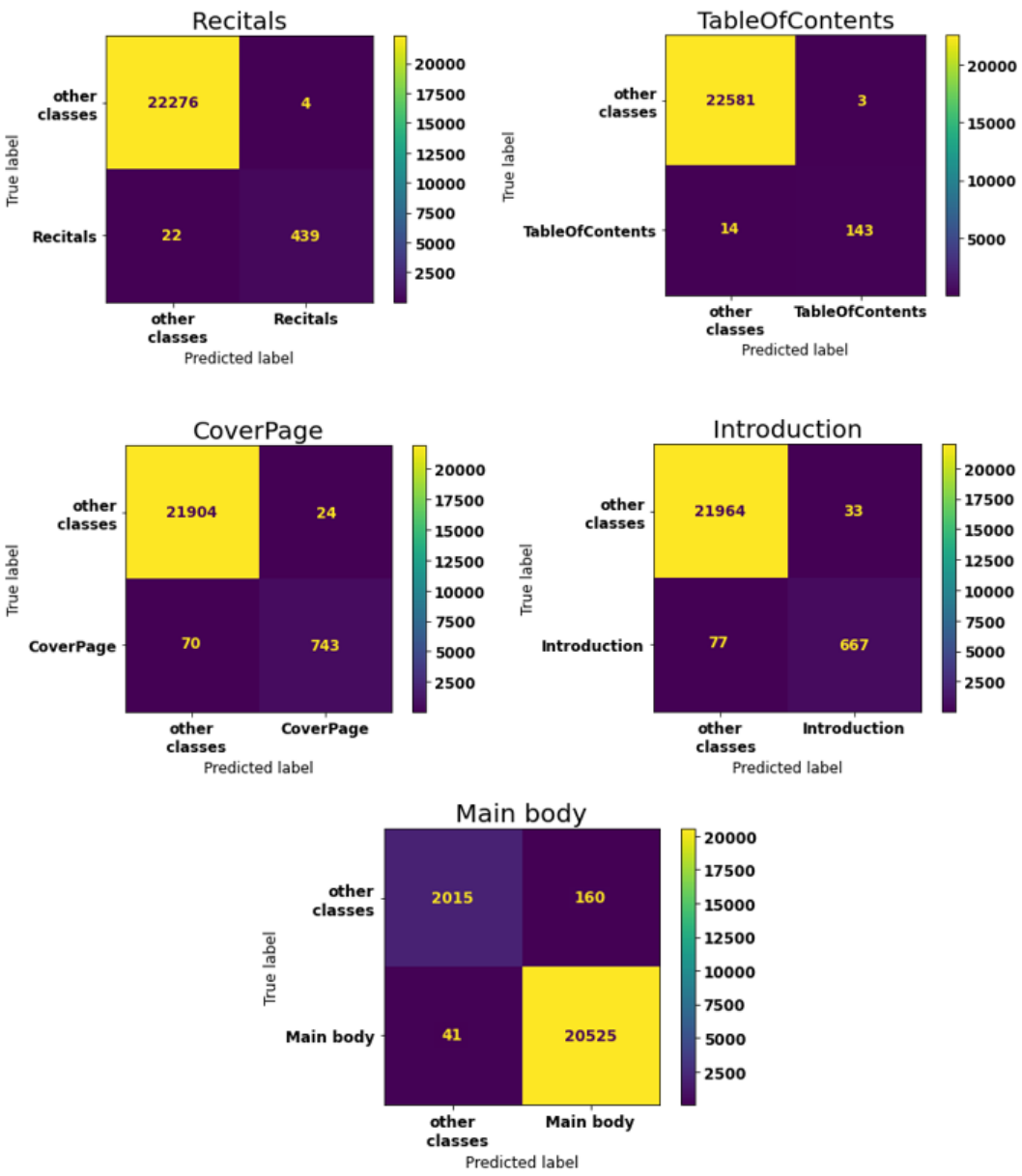


Fig. 5.9: RoBERTa model confusion matrices.

Hierarchical model

The hierarchical model results of Figure 5.10 seem even more promising, compared to the other text zone classification models, as was the case with the Table 5.4. Of course again there is a possibility that some cases were incorrectly annotated, the extent of which cannot be easily estimated, especially in instances where their labels cannot be defined by the author, which adds to the number of mistakes. After examining the misclassified predictions, the model seems to make fewer mistakes in terms of "Main body" misclasifications, compared to RoBERTa, but the misclassification cases for the other cases remain almost identical. Taking the "Cover page" zone as an example, both models falsely predicted 6 cover page cases as introduction.

Excluding the increased inference time and memory size that comes with the use of the hierarchical model, it is the preferable choice, mostly because of the results but also due to the convenient input format of the model. The target being a whole document, the hierarchical model can easily classify the input document's text zones in context, rather than the RoBERTa approach which classifies each text zone input independently.

Overall, both models perform quite well in the given test. For the use of each one, it all boils down to the performance and properties of each. The RoBERTa model requires less memory and offers faster inference, while the hierarchical one, more precise output. Regardless their differences are not far off and as such both can be used with minimal drawbacks.

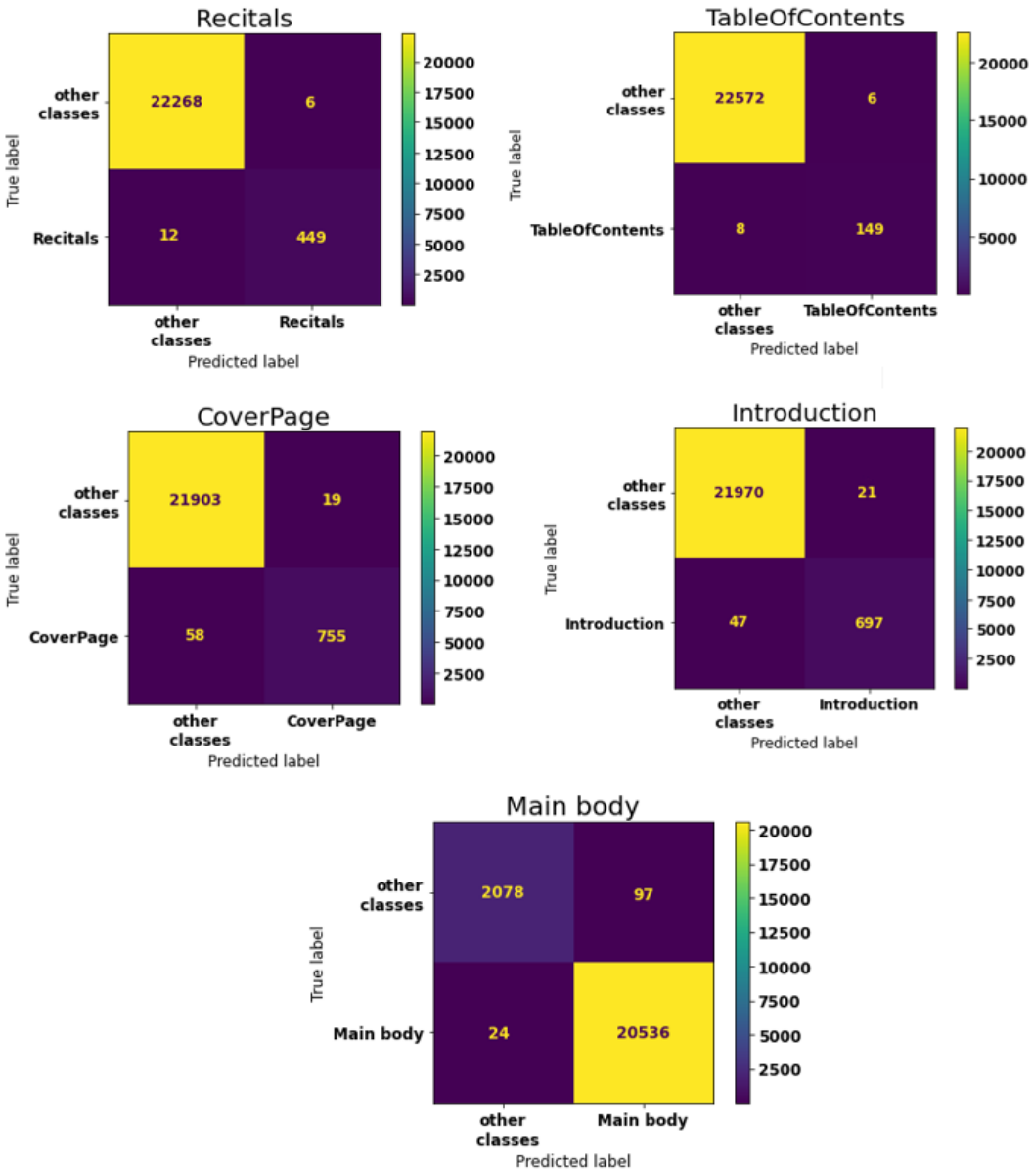


Fig. 5.10: Hierarchical model confusion matrices.

6 Conclusions and Future Work

6.1 Conclusions

In this thesis several approaches to two tasks were presented. The first task revolved around the detection of text blocks in legal documents and their classification as either paragraphs or titles. The data available included images and coordinates pointing towards the desired output. Even though the dataset was imbalanced, the core models, being trained to perform under such circumstances had no problem dealing with it. Keeping in line with the company's requests, RetinaNet and YOLOv5 models were tested. A custom two-stage approach was also implemented featuring two different region proposal techniques. The first provided surprisingly accurate results, thanks to the region proposal rules than were able to capture the essence of a paragraph. Nevertheless, they were not on par with the pretrained models.

Due to the low training requirements of the YOLOv5 model several alternatives were also created featuring a small, large and a medium YOLOv5 approach. The YOLOv5 models proved almost equal in the original test scenario, with every one offering different advantages and drawbacks. The fact that the small, medium and large YOLOv5 are the only models that were trained using the whole training set involving more that 19000 images, can be attributed to the training speed as well. In general, the YOLOv5 models proved most efficient both in terms of accuracy and memory and time consumption. Finally, it was deduced that the balanced YOLOv5 model was not able to outperform the original one in a realistic scenario.

The final model tested regarding the text block detection task, was RetinaNet promising strict training by introducing the Focal loss function, which focuses on improving training instances with low prediction probability. Its effect was obvious from the results, in which, having only been trained on 1/3 of the training images, it managed to match the YOLOv5small model and almost reached the medium one. RetinaNet featuring the highest amount of time per epoch, made it quite time consuming to train in to its full extent, given the project's time restrictions.

After the detection of the text blocks a set of rules were applied in order to distinguish between titles and paragraphs. For a rule-based approach it performed quite well, even though there were some mistakes made, due to the nature of the rules and the misleading

results of pytesseract in some cases. As a result, its performance may not be on par with approaches involving annotated data and trainable models.

Moving on to the second task, a model was needed able to classify text zones stemming from English documents. To that extent, a RoBERTa model was fine-tuned using a different dataset than the one used in the first task. This dataset was also imbalanced, but nevertheless RoBERTa performed admirably. A more advanced approach was also attempted, using the trained RoBERTa's last state to encode each text zone. The hierarchical model created, incorporated RoBERTa in order to create an embedding (a vector of length 768) for each text zone of the input documents. Afterwards, by utilizing stacked bi-directional LSTMs, the new representations were made context aware. The idea was that, given the context of each text zone, they would be easier to classify. The implementation of this idea, did manage to improve the results by a non-negligible amount, but introduced the need of a new model to keep track of the context and make predictions using the RoBERTa's embeddings. This in turn introduced more inference time and memory usage. Regardless, both models easily outperformed the simple logistic regression model used as a baseline.

6.2 Future Work

Regarding the text block detection task, RetinaNet cannot be easily disregarded. The results imply that with more training data there is a chance for RetinaNet to outperform even the medium and large YOLOv5 models. This of course would add to the inference time and memory consumption which is not in the interest of the company but the results may prove the investment worthy. More YOLO oriented alternatives also exist. For instance, a new official YOLO model was introduced in the summer of 2022, promising more accurate results and even faster inference. It would be possible to even train the model to detect both text and tables, thus combining the two models of the company to one. Finally, if the results are found to be unsatisfactory in some cases, datasets such as PubLayNet offer many annotated data, that could be added to the existing ones.[10] Being trained on more data, the model's performance is expected to increase.

For the text block classification as either title or paragraph, the rule-based approach is far from optimal, but it is the only option given the lack of annotations. On the other hand, with a set of annotations an NLP-oriented Deep Learning approach can be implemented. These annotations could enable the use of pretrained models even on a multilingual level and thus drastically improve the results.

Finally, regarding the text zone classification task, some problems regarding the dataset were avoided. Such cases involved empty cases of text or possible misplaced labels (e.g.

[10]PubLayNet dataset can be found in https://developer.ibm.com/exchanges/data/all/publaynet/.

cover page after the main body). But cases such as label missclassifications that were noticed could not be handled and other issues may still be present. That being the case, a dataset with less gold annotation errors may be needed. As for the models themselves, there are many alternatives to be explored. A promising example is the use of Transformers in combination with a hierarchical model. The current hierarchical model implementation revolves around the use of RoBERTa which is indeed a case of Transformers, and a top model implemented by means of RNN layers. Instead, a Transformer based approach can be used for the top model as well. Chalkidis et al. [Cha+21] proposed the hierBERT model which uses Transformers in both levels of the hierarchical architecture. Cases such as this promise better performance, even though the results of the current models did not suggest they were necessary.

Bibliography

[ADF12] B. Alexe, T. Deselaers, and V. Ferrari. "Measuring the Objectness of Image Windows". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 34.11 (2012), pp. 2189–2202.

[Ala20] J. Alammar. *The Illustrated Transformer*. 2020. URL: https://jalammar.github.io/illustrated-transformer/.

[AMA17] S. Albawi, T. Abed Mohammed, and S. Al-Zawi. "Understanding of a convolutional neural network". In: *2017 International Conference on Engineering and Technology (ICET)*. Antalya, Turkey, 2017, pp. 1–6.

[AWL20] A.Bochkovskiy, C.Y. Wang, and H.Y. M. Liao. "YOLOv4: Optimal Speed and Accuracy of Object Detection". In: *CoRR* abs/2004.10934 (2020). arXiv: 2004.10934.

[CAM18] I. Chalkidis, I. Androutsopoulos, and A. Michos. "Obligation and Prohibition Extraction Using Hierarchical RNNs". In: *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*. Melbourne, Australia, 2018, pp. 254–259.

[Cha+21] I. Chalkidis, A. Jana, D. Hartung, et al. "LexGLUE: A Benchmark Dataset for Legal Language Understanding in English". In: *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. Dublin, Ireland, 2021, pp. 4310–4330.

[Dev+19] J. Devlin, M. W. Chang, K. Lee, and K. Toutanova. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding". In: *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*. Minneapolis, Minnesota, USA, 2019, pp. 4171–4186.

[Gir+14] R. Girshick, J. Donahue, T. Darrell, and J. Malik. "Rich Feature Hierarchies for Accurate Object Detection and Semantic Segmentation". In: *2014 IEEE Conference on Computer Vision and Pattern Recognition*. Columbus, Ohio, USA, 2014, pp. 580–587.

[HBS16] J. Hosang, R. Benenson, and B. Schiele. "A Convnet for Non-maximum Suppression". In: *Pattern Recognition*. Springer, Cham, 2016, pp. 192–204.

[HBS17] J. Hosang, R. Benenson, and B. Schiele. “Learning Non-maximum Suppression”. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Los Alamitos, California, USA, 2017, pp. 6469–6477.

[He+14] K. He, X. Zhang, S. Ren, and J. Sun. “Spatial Pyramid Pooling in Deep Convolutional Networks for Visual Recognition”. In: *Computer Vision - ECCV 2014*. Zurich, Switzerland, 2014, pp. 346–361.

[He+17] K. He, G. Gkioxari, P. Dollár, and R. Girshick. “Mask R-CNN”. In: *2017 IEEE International Conference on Computer Vision (ICCV)*. Venice, Italy, 2017, pp. 2980–2988.

[He+19] Y. He, C. Zhu, J. Wang, M. Savvides, and X. Zhang. “Bounding Box Regression With Uncertainty for Accurate Object Detection”. In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*. Los Alamitos, California, USA, 2019, pp. 2883–2892.

[HF17] P. Henderson and V. Ferrari. “End-to-end training of object class detectors for mean average precision”. In: *Computer Vision - ACCV 2016*. Taipei, Taiwan, 2017, pp. 198–213.

[Li+17] Z. Li, C. Peng, G. Yu, et al. “Light-Head R-CNN: In Defense of Two-Stage Object Detector”. In: *CoRR* abs/1711.07264 (2017). arXiv: 1711.07264.

[Lin+17] T. Y. Lin, P. Dollár, R. Girshick, et al. “Feature Pyramid Networks for Object Detection”. In: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Los Alamitos, California, USA, 2017, pp. 936–944.

[Lin+20] T. Y. Lin, P. Goyal, R. Girshick, K. He, and P. Dollár. “Focal Loss for Dense Object Detection”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 42.2 (2020), pp. 318–327.

[Liu+19] Y. Liu, M. Ott, N. Goyal, et al. “RoBERTa: A Robustly Optimized BERT Pretraining Approach”. In: *CoRR* abs/1907.11692 (2019). arXiv: 1907.11692.

[LKC19] S. Lee, S. Kwak, and M. Cho. “Universal Bounding Box Regression and Its Applications”. In: *Computer Vision – ACCV 2018*. Cham: Springer International Publishing, 2019, pp. 373–387.

[Loh+21] A. Lohia, K. Kadam, R. Joshi, and D. Bongale. “Bibliometric Analysis of One-stage and Two-stage Object Detection”. In: *Library Philosophy and Practice (e-journal)*. Nebraska, USA, 2021, p. 4910.

[Luc+19] D. R. Lucio, R. Laroca, L. A. Zanlorensi, G. Moreira, and D. Menotti. “Simultaneous Iris and Periocular Region Detection Using Coarse Annotations”. In: *2019 32nd SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)*. Rio de Janeiro, Brazil, 2019, pp. 178–185.

[NE22] U. Nepal and H. Eslamiat. “Comparing YOLOv3, YOLOv4 and YOLOv5 for Autonomous Landing Spot Detection in Faulty UAVs”. In: *Sensors* 22.2 (2022).

[Ope00a] OpenCV. *Image Thresholding*. 2000. URL: https://opencv24-python-tutorials.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_thresholding/py_thresholding.html#otsus-binarization.

[Ope00b] OpenCV. *Morphological Transformations*. 2000. URL: https://opencv24-python-tutorials.readthedocs.io/en/latest/py_tutorials/py_imgproc/py_morphological_ops/py_morphological_ops.html.

[Ope00c] OpenCV. *Structural Analysis and Shape Descriptors*. 2000. URL: https://docs.opencv.org/2.4/modules/imgproc/doc/structural_analysis_and_shape_descriptors.html.

[Red+16] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. "You Only Look Once: Unified, Real-Time Object Detection". In: *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. Los Alamitos, California, USA, 2016, pp. 779–788.

[Ren+15a] S. Ren, K. He, R. Girshick, and J. Sun. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks". In: *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*. Montreal, Canada, 2015, pp. 91–99.

[Ren+15b] S. Ren, K. He, R. Girshick, and J. Sun. "Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks". In: *Advances in Neural Information Processing Systems (NIPS)*. Montreal, Canada, 2015, pp. 91–99.

[Rez+19] H. Rezatofighi, N. Tsoi, J. Gwak, et al. "Generalized Intersection Over Union: A Metric and a Loss for Bounding Box Regression". In: Long Beach, California, USA, 2019, pp. 658–666.

[SA85] S. Suzuki and K. Abe. "Topological structural analysis of digitized binary images by border following". In: *Computer Vision, Graphics, and Image Processing* 30.1 (1985), pp. 32–46.

[She+21] Z. Shen, R. Zhang, M. Dell, et al. "LayoutParser: A Unified Toolkit for Deep Learning Based Document Image Analysis". In: *CoRR* abs/2103.15348 (2021). arXiv: 2103.15348.

[SI18] P. Soviany and R.T. Ionescu. "Optimizing the Trade-Off between Single-Stage and Two-Stage Deep Object Detectors using Image Difficulty Prediction". In: Timisoara, Romania, 2018, pp. 209–214.

[Sym+19] C. Symeonidis, I. Mademlis, N. Nikolaidis, and I. Pitas. "Improving Neural Non-Maximum Suppression for Object Detection by Exploiting Interest-Point Detectors". In: *2019 IEEE 29th International Workshop on Machine Learning for Signal Processing (MLSP)*. Pittsburgh, Pennsylvania, USA, 2019, pp. 1–6.

[SZ14] K. Simonyan and A. Zisserman. “Very Deep Convolutional Networks for Large-Scale Image Recognition”. In: *3rd International Conference on Learning Representations (ICLR 2015)*. San Diego, California, USA, 2014, pp. 1–14.

[Tam19] S. Tammina. “Transfer learning using VGG-16 with Deep Convolutional Neural Network for Classifying Images”. In: *International Journal of Scientific and Research Publications (IJSRP)* 9.10 (2019), p. 9420.

[Uij+13] J. R. R. Uijlings, K. E. A. van de Sande, T. Gevers, and A. W. M. Smeulders. “Selective Search for Object Recognition”. In: *International Journal of Computer Vision* 104.2 (2013), pp. 154–171.

[Ult21] Ultralytics. *The "fitness" function in train.py*. 2021. URL: `https://github.com/ultralytics/yolov5/issues/2303`.

[WBL22] C. Y. Wang, A. Bochkovskiy, and H. Y. M. Liao. “YOLOv7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors”. In: *CoRR* abs/2207.02696 (2022). arXiv: `2207.02696`.

[Yan+18] T. Yang, X. Zhang, Z. Li, W. Zhang, and J. Sun. “MetaAnchor: Learning to Detect Objects with Customized Anchors”. In: *NeurIPS*. Red Hook, New York, USA, 2018, pp. 318–328.

[Yan+22] C. Y. Yang, H. Samani, N. Ji, et al. “Deep Learning Based Real-Time Facial Mask Detection and Crowd Monitoring”. In: *COMPUTING AND INFORMATICS* 40.6 (2022), pp. 1263–1294.

[Yih+20] X. Yiheng, L. Minghao, C. Lei C, et al. “LayoutLM: Pre-Training of Text and Layout for Document Image Understanding”. In: *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery Data Mining*. New York, USA, 2020, pp. 1192–1200.

[Zho+20] Y. Zhong, J. Wang, J. Peng, and L. Zhang. “Anchor Box Optimization for Object Detection”. In: *2020 IEEE Winter Conference on Applications of Computer Vision (WACV)*. Los Alamitos, California, USA, 2020, pp. 1275–1283.

List of Acronyms

AI	Artificial Intelligence
AP	Average Precision
BERT	Bidirectional Encoder Representations from Transformers
Bi-LSTM	Bidirectional Long Sort Term Memory
CSP	Cross Stage Partial Network
DIA	Document Image Analysis
DP	Deep Learning
FCL	Fully Connected Layer
FCN	Fully Convolutional Network
FFNN	Feed Forward Neural Network
FN	False Negative
FP	False Positive
FPN	Feature Pyramid Network
IoU	Intersection over Union
mAP	mean Average Precision
NLP	Natural Language Processing

NMS Non Maximum Suppression

NSP Next Sentence Prediction

OCR Optical Character Recognition

PANet Path Aggregation Network

POS Part Of Speech

PR-curve Precision-Recall curve

ReLU Rectified Linear Unit

RNN Recursive Neural Network

RoBERTa Robustly optimized BERT Pretraining approach

ROI Region Of Interest

RPN Region Proposal Network

SPP Spatial Pyramid Pooling

TP True Positive

YOLO You Only Look Once

List of Figures

List of Tables

List of Algorithms