Satakunnan ammattikorkeakoulu
Satakunta University of Applied Sciences

ANDREA KOVALOVA

Deep Learning-based Table Detection in Documents

DEGREE PROGRAMME IN ARTIFICIAL INTELLIGENCE
2023

Author Kovalova, Andrea	Type of Publication Bachelor’s thesis	Date January 2023
	Number of pages 44	Language of publication: English
Title of publication Deep Learning-based Table Detection in Documents		
Degree Programme Artificial Intelligence		

Abstract

Extracting content from tables in documents was identified as problematic, especially in the cases of tables without clear borders. The proposed solution was to apply document layout analysis, particularly using the LayoutParser library, to identify tables in company documents. The hypothesis was that fine-tuning a model available in the Model Zoo of LayoutParser on the custom dataset would significantly improve the table detections, at least for the documents with borderless tables.

As the custom dataset, 302 company documents were annotated and used for experiments with different data augmentations and their evaluations. A Faster R-CNN model trained on the TableBank dataset was selected as the pre-trained model for fine-tuning. The results of experiments showed that fine-tuning the model with augmentations where only resizing of the input images was applied was the best approach and it substantially improved table detection in company documents, especially the detection of borderless tables. In accordance with the results, the thesis hypothesis was accepted.

Keywords
tables, document layout analysis, object detection, LayoutParser, convolutional neural network, deep learning

FOREWORD

I would like to thank my thesis supervisor at SAMK, Petteri Pulkkinen, for providing feedback and prompt responses to my numerous inquiries. I also want to express my appreciation to my manager at Wärtsilä, Tal Katzav, for initiating and supporting this thesis project, and to my colleagues Juho-Pekka Koponen and Teemu Sormunen for their unwavering support throughout the project. Finally, I am deeply grateful to my thesis supervisor at Wärtsilä, Tony Savander, for the productive discussions, constructive feedback, and timely guidance on deadlines that he provided throughout the process.

CONTENTS

LIST OF SYMBOLS AND TERMS

AP	Average Precision
AR	Average Recall
CNN	Convolutional Neural Networks
CPU	Central Processing Unit
DLA	Document Layout Analysis
FN	False Negatives
FP	False Positives
IoU	Intersection over Union
JSON	JavaScript Object Notation
mAP	Mean Average Precision
OCR	Optical Character Recognition
PDF	Portable Document Format
PNG	Portable Network Graphics
R-CNN	Region-based Convolutional Neural Network
RoI	Region of Interest
RPN	Region Proposal Network
TN	True Negatives
TP	True Positives

1 INTRODUCTION

Documents, in paper or digitalized form, are an inseparable part of every company. Wärtsilä, as an international corporation with about 17 000 employees operating in marine and energy markets, naturally needs to process an immense number of different kinds of documents in various departments (About, n.d.). Some departments use software for intelligent automatic processing of their documents. However, information from some documents is not retrieved well enough using the current software. The team members, therefore, need to allocate their working time to manual data entry that could have been allocated elsewhere. Additionally, when manually extracting information from documents, the focus is on data required for the current business processes due to the high costs of manual extraction. Nevertheless, company documents may contain other data which if analyzed, might provide invaluable information and lead to higher business intelligence and smart and sustainable decisions.

Since manual data entry is not a sustainable solution, programmatic alternatives were explored. Various optical character recognition (OCR) tools were applied to the documents and different problems have been identified. One of the problems is information extraction from tables. There are several options for extracting tables from documents. For example, Camelot (Camelot, 2021) and tabula-py (Tabula-py, 2022) are both Python libraries that enable extracting tables from PDF documents. Azure Form Recognizer (Form Recognizer, n.d.) and Amazon Textract (Tables, n.d.), on the other hand, are paid solutions provided by companies that belong among the world's major ones in artificial intelligence. Apart from regular tables with clearly outlined borders, however, company documents also contain tables without some or any clear borders, and many tools are not able to detect tables without marked borders. For this problem, a tool for document layout analysis can be applied which recognizes elements of the documents, such as a figure, heading, table, and text, and can be customized for own datasets.

The proposed tool for document layout analysis is a Python library called LayoutParser, which uses deep learning for layout analysis. The library allows the use of already pre-trained models to identify tables (and other layout elements) and provides tools for fine-tuning the existing models on custom data (Shen et al., 2021). Identifying tables in documents will then provide borders for the location of interest for extracting the information using OCR. In this thesis, however, only fine-tuning of a selected pre-trained model will be described and evaluated. The hypothesis is that fine-tuning an appropriate model available in the Model Zoo of LayoutParser on the custom dataset of documents provided by Wärtsilä will significantly improve the table detections, at least for the documents with borderless tables.

The rest of the thesis is organized as follows. The second chapter, Theoretical framework, explains document layout analysis and describes the LayoutParser library in detail. Furthermore, the chapter describes the type of task and important topics and terms for fine-tuning the selected pre-trained model. Next, the third chapter, Data, provides a detailed description of the documents used in this thesis and the process required for using the data for fine-tuning. In the fourth chapter, Experiments, training of models and their results are described and subsequently, explained in the fifth chapter, Discussion. The last chapter, Conclusion, naturally summarizes the findings of the thesis.

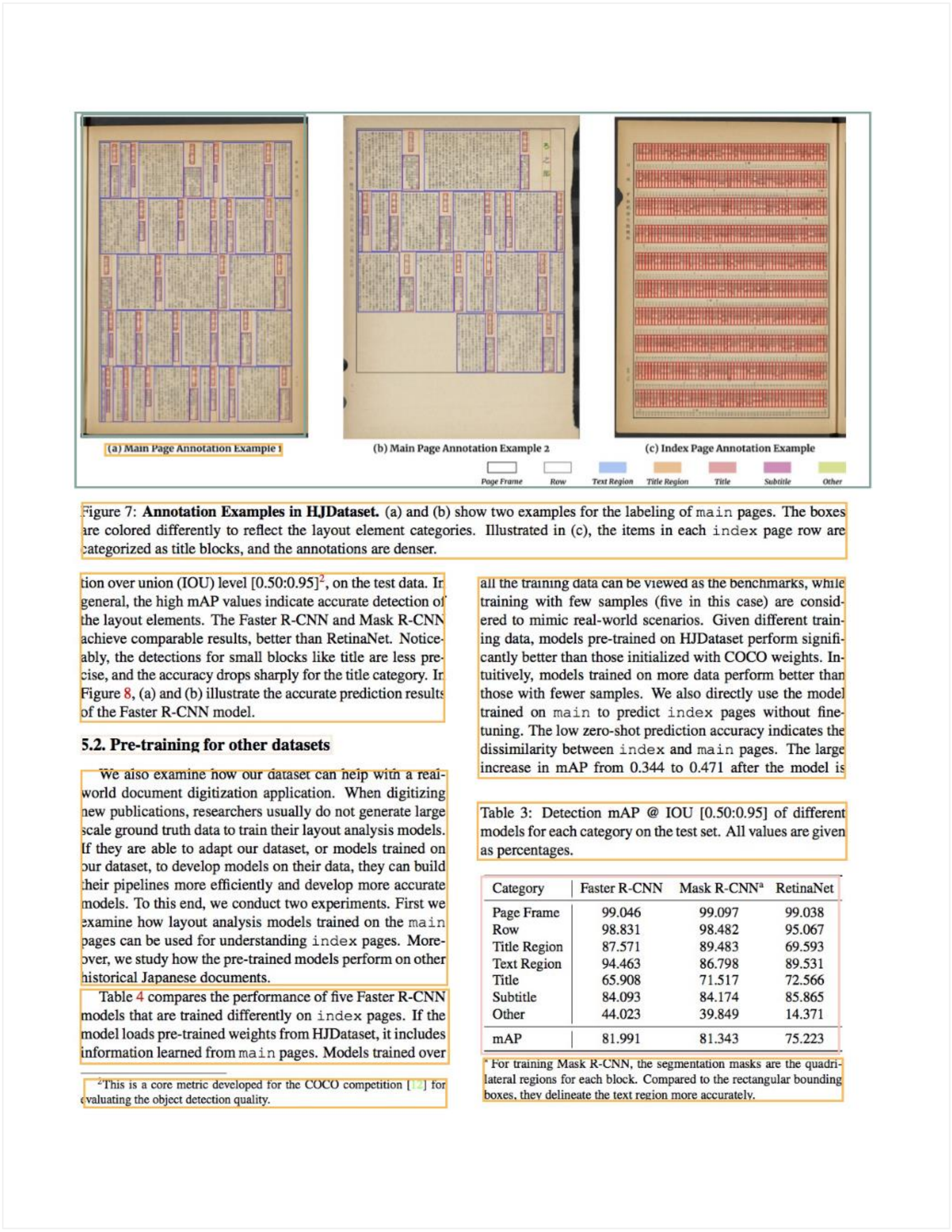
2 THEORETICAL FRAMEWORK

This section will describe important topics and terms for fine-tuning a model for table detection with LayoutParser. The first and second parts will introduce document layout analysis of image documents and one of its tools called LayoutParser. Then, the object detection and object recognition tasks will be described in the third part. The last part is devoted to a type of neural network called convolutional neural networks. This part has further subparts introducing the used pre-trained model, Faster R-CNN, transfer learning and large datasets for object detection, and finally, metrics for measuring the performance of a trained model.

2.1 Document Layout Analysis

Document layout analysis (DLA) is one of the tasks in computer vision and its goal is to identify and categorize different content regions of a document image (Wu et al., 2022). These regions can be figures, tables, and texts and one image usually represent one page (an example is illustrated in Figure 1). Among other applications, DLA contributes to content understanding and knowledge extraction from an image document (Wu et al., 2021). The result of DLA allows targeting information extraction from regions of the document rather than extracting text from the whole image document, which is typical for extracting process applying solely an OCR tool. Layout analysis is a part of the document image analysis and recognition process, in which image documents typically first undergo pre-processing operations such as clearing up noisy images and then, layout analysis and character and symbol recognition follow (Lombardi and Marinai, 2020).

Wu et al. (2022) divide the DLA task processing method into two groups – traditional and deep learning methods. In traditional methods, image segmentation algorithms are typically used to divide the layout. The increasing complexity of document layouts made the processing challenging for traditional methods. However, the continuous advances in the field of machine learning have had a great impact on the research in DLA as well, especially the use of deep learning to process the DLA tasks (e.g. Xu et al., 2020; Shen et al., 2021). Yet despite the progress in research, DLA still faces challenges in the real environment in terms of specificities of different languages, fonts, handwritten text, or document types (Wu et al., 2022).



(a) Main Page Annotation Example 1 (b) Main Page Annotation Example 2 (c) Index Page Annotation Example

Figure 7: **Annotation Examples in HJDataset.** (a) and (b) show two examples for the labeling of `main` pages. The boxes are colored differently to reflect the layout element categories. Illustrated in (c), the items in each `index` page row are categorized as title blocks, and the annotations are denser.

tion over union (IOU) level [0.50:0.95][2], on the test data. In general, the high mAP values indicate accurate detection of the layout elements. The Faster R-CNN and Mask R-CNN achieve comparable results, better than RetinaNet. Noticeably, the detections for small blocks like title are less precise, and the accuracy drops sharply for the title category. In Figure 8, (a) and (b) illustrate the accurate prediction results of the Faster R-CNN model.

5.2. Pre-training for other datasets

We also examine how our dataset can help with a real-world document digitization application. When digitizing new publications, researchers usually do not generate large scale ground truth data to train their layout analysis models. If they are able to adapt our dataset, or models trained on our dataset, to develop models on their data, they can build their pipelines more efficiently and develop more accurate models. To this end, we conduct two experiments. First we examine how layout analysis models trained on the `main` pages can be used for understanding `index` pages. Moreover, we study how the pre-trained models perform on other historical Japanese documents.

Table 4 compares the performance of five Faster R-CNN models that are trained differently on `index` pages. If the model loads pre-trained weights from HJDataset, it includes information learned from `main` pages. Models trained over all the training data can be viewed as the benchmarks, while training with few samples (five in this case) are considered to mimic real-world scenarios. Given different training data, models pre-trained on HJDataset perform significantly better than those initialized with COCO weights. Intuitively, models trained on more data perform better than those with fewer samples. We also directly use the model trained on `main` to predict `index` pages without fine-tuning. The low zero-shot prediction accuracy indicates the dissimilarity between `index` and `main` pages. The large increase in mAP from 0.344 to 0.471 after the model is

[2]This is a core metric developed for the COCO competition [12] for evaluating the object detection quality.

Table 3: Detection mAP @ IOU [0.50:0.95] of different models for each category on the test set. All values are given as percentages.

Category	Faster R-CNN	Mask R-CNN[a]	RetinaNet
Page Frame	99.046	99.097	99.038
Row	98.831	98.482	95.067
Title Region	87.571	89.483	69.593
Text Region	94.463	86.798	89.531
Title	65.908	71.517	72.566
Subtitle	84.093	84.174	85.865
Other	44.023	39.849	14.371
mAP	81.991	81.343	75.223

[a] For training Mask R-CNN, the segmentation masks are the quadrilateral regions for each block. Compared to the rectangular bounding boxes, they delineate the text region more accurately.

Figure 1. Example of layout detection with a figure, title, table, and texts as detected elements (Deep Layout Parsing, n.d.).

2.2 LayoutParser

According to Shen et al. (2021), LayoutParser is a Python library allowing document image analysis and processing using deep convolutional neural networks. Among its components are a toolkit for applying deep learning models for layout detection, character recognition and other document image analysis tasks, and an archive of pre-trained neural network models. In LayoutParser, layout analysis is formulated as an object detection problem. An image document is taken as an input of a layout model which produces a list of rectangular boxes of the content regions. The tool is built upon Detectron2 and uses state-of-the-art models like Faster R-CNN and Mask R-CNN for layout detection. While Mask R-CNN is typically used for image segmentation tasks, Faster R-CNN is used for object detection and will be used in experiments in this thesis. Detectron2 is a second-generation Python library implemented in PyTorch providing state-of-the-art detection and segmentation algorithms (Wu et al., 2019).

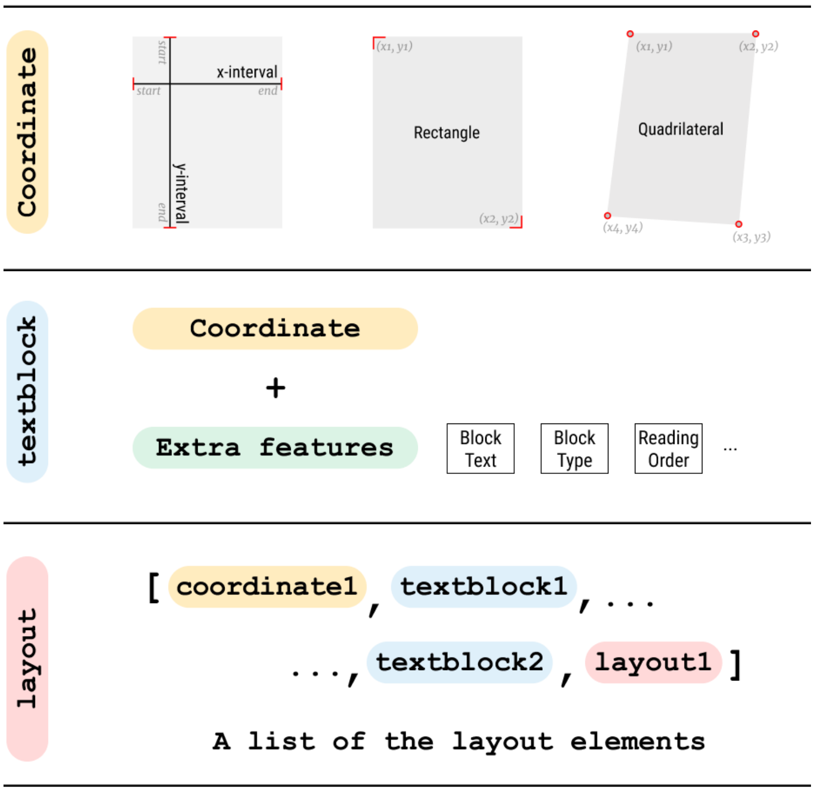


Figure 2. The relationship between the key layout data structures – the coordinate, textblocks and layout (Shen et al., 2021).

Shen et al. (2021) describe the implemented series of data structures and operations as being a critical attribute of the tool because traditionally DLA required other post-processing steps to get the final outputs. As shown in Figure 2, the three key parts are the Coordinate system, the Text Block, and the Layout. The coordinates of layout elements can be defined by an interval (class Interval), two points forming a rectangle (class Rectangle) or four points forming a four-sided polygon (class Quadrilateral) allowing the capture of regions in skewed or distorted image documents. Coordinates are part of the TextBlock class which stores, besides the location of the layout element within the image and the type of coordinate, other content-related information about the element such as the type of layout element (e.g., table) and the prediction confidence of the element. Lastly, the Layout class stores a list of the blocks, i.e., layout elements, including other layout elements.

Target image documents may substantially differ from the datasets that the existing layout models were trained on causing low detection accuracy. As Shen, et al. (2021) state, LayoutParser allows customization by providing tools for document image annotation and model tuning. The toolkit for the annotation of document layouts is using object-level active learning. Then only the most important regions in an image need to be annotated. The rest of the layout objects labelling is performed automatically using predictions from the layout detection model.

Additionally, LayoutParser provides a model hub and community platform to promote sharing and discussion of pre-trained models as well as practices (Shen, et al., 2021).

2.3 Object Detection and Recognition

DLA aims to detect different regions in an image document. As stated above, DLA is a computer vision task, particularly an object recognition task as the regions to be detected on a page are objects. Therefore, common object recognition models can also be applied to this problem. One of the most prevalent types of neural networks for object recognition is a convolutional neural network. The approach to solving table detection in this thesis will include solely the usage of deep learning. However, before the popularization of artificial neural networks, models and algorithms such as Histogram of Oriented Gradients Feature Extractor and Support Vector Machine model (e.g., Mizuno et al., 2012), Bag of features model (e.g., Delaitre et al., 2010) or Viola-Jones algorithm (e.g., Castrillón et al., 2011) were widely used.

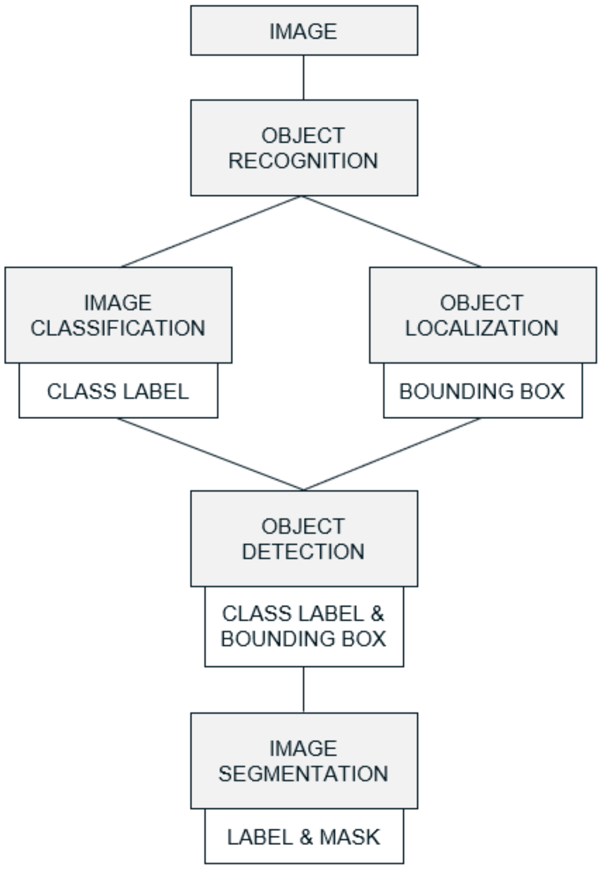


Figure 3. Object recognition and related tasks.

Object recognition is a task that identifies different types of objects in an image and video. It is one of the most widespread applications of machine learning. As illustrated in Figure 3, other tasks related to object recognition are image classification, object localization, object detection, and image segmentation (Szeliski, 2010, pp. 657-721). Image classification identifies the type or class of the object and outputs class labels. Object localization identifies the presence of objects and their location in an image, and outputs bounding boxes that are drawn around these objects. Both tasks are illustrated in Figure 4 (in the left image). Object detection is a combination of both preceding tasks. It identifies the position in an image and the class of the objects as visualized in Figure 4 (in the middle image). Lastly, a further extension of object detection is image segmentation where the objects are identified by highlighting specific pixels of the objects and the output can be a mask. Unlike bounding box outputs, segmentation by pixels allows recognition shape of objects as depicted in Figure 4 on instance segmentation. According to Cyganek (2013, p. 408), some systems consider object recognition and object detection as two separate classifiers that cooperate. In other systems, object detection also means the recognition of what object is in the image. In this thesis, the main task is to identify the location of one class, i.e., tables, in a document image and the task will be referred to as object detection.

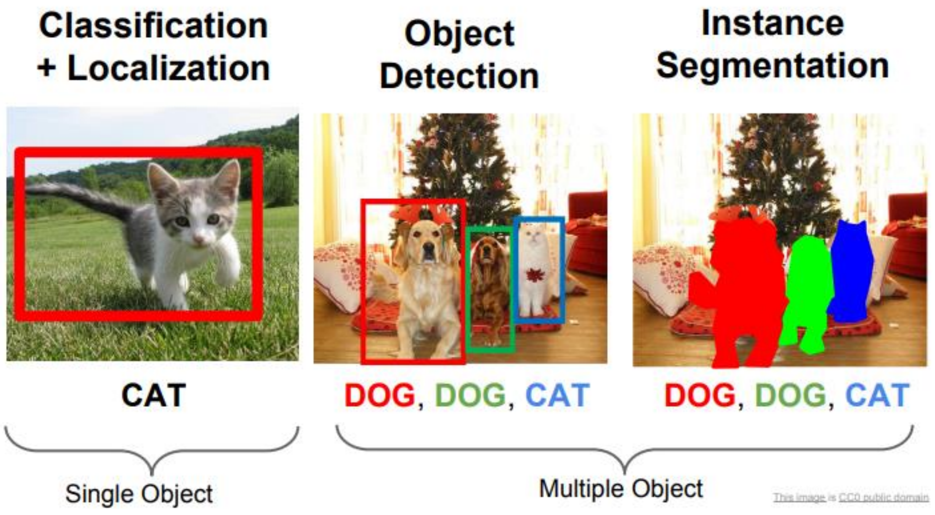


Figure 4. Comparison of object recognition tasks (Fei-Fei et al., 2018).

Humans and animals perform the detection and recognition of objects naturally in daily life based on their learned and inherited experiences. These processes are not so straightforward for machines. Nonetheless, biological systems have provided inspiration for some machine realizations. Among these are artificial neural networks and neurons, their basic building block.

2.4 Convolutional Neural Networks

Deep learning is a field of machine learning that is based on the training of artificial neural networks and has had several applications in computer vision. A convolutional neural network (CNN) is a type of neural network that has been widely used for processing images. An image is a two-dimensional grid of pixels where each pixel defines the color and brightness. Each pixel is an input node in a traditional neural network. Attempting to solve an object detection task by training a traditional neural network, where every input node interacts with every output node, with images would therefore require thousands of weights per node for even small images (e.g., images of size 100×100 would require 10 000 weights per node). Optimization would likely be unmanageable.[1] (Goodfellow et al., 2016, pp. 330-372; Burkov, 2019, Section 6)

Classification of complex images can be solved by a CNN because CNN significantly reduces the number of parameters in the neural network without much compromise in the quality of the model. Unlike traditional neural networks, CNN utilizes the correlation among pixels. For example, in the classification of dog images, any brown pixel is likely located next to a brown pixel, and it is likely that pixels further away would be less correlated. (Goodfellow et al., 2016, pp. 330-372; Burkov, 2019, Section 6)

There are three types of layers in CNN: convolutional, pooling and fully connected layers. In the convolutional layers, CNN uses filters, i.e., kernels, to transform data before it is passed to another layer. Kernels are usually very small (e.g., 3×3 or 5×5) and they represent weights that are given to pixels in the image. By computing the dot product between two two-dimensional arrays, i.e., the input and the filter, the filter is convolved with the input. As depicted in Figure 5, the filter slides over the input from left to right starting at the top left corner and ending at the bottom right corner. A stride defines the number of pixels of the filter's movement. In the pictured example, the stride is one and the filter, therefore, moves by one pixel at a time. Furthermore, there is no padding in the picture which would prevent the decrease of the input's special dimensions. For a more precise analysis of an image, padding can be added around the image to keep the dimensions the same. After the addition of a bias, the result of the convolution operation is mapped to a feature map which summarizes the features of the input image. Each cell in the feature map corresponds to a group of neighbouring pixels and hence utilizes correlations among the pixels. Then, an activation function is typically applied, for instance, ReLU[2]. (Goodfellow et al., 2016, pp. 330-372; Burkov, 2019, Section 6)

[1] If the reader is not familiar with the basics of neural networks, please review the topic in Goodfellow et al. (2016) and/or Burkov (2019).

[2] ReLU is defined as $f(x) = x^+ = \max(0, x)$ and after application of the function all negative values are set to 0.

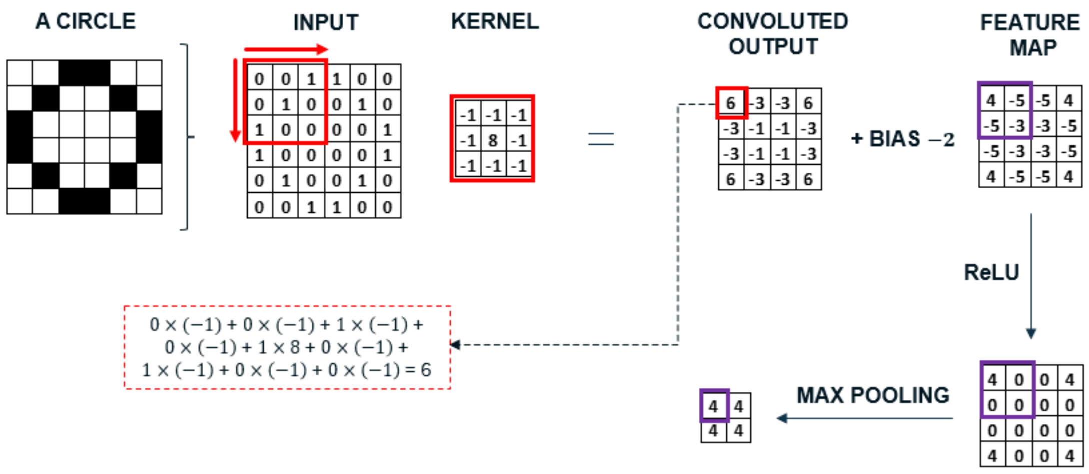


Figure 5. An example of convolution operation between a 6×6 matrix representing an image and a 3×3 matrix representing an edge detection filter, and further application of ReLU activation function and max pooling operation.

The problem with the featured map is its sensitivity to the position of features. Even minor image augmentation would result in a different feature map. This problem can be addressed by downsampling which further summarizes the features in an image and reduces the size of the feature map. A common approach is to use a pooling layer where a pooling operation, e.g., max pooling, is applied to the feature map. A filter is again systematically moved over the feature map. This filter is typically 2×2 and does not overlap the areas of the feature map. As illustrated in Figure 5, the max pooling operation selects the maximum value in each area and the input size is further reduced by 75%. After these operations, the max pooled output is passed to a fully connected neural network in the fully connected layer. (Goodfellow et al., 2016, pp. 330-372)

2.4.1 Faster R-CNN

R-CNN stands for Region-based Convolutional Neural Networks. As Ren et al. (2015) state, region proposal algorithms are used to hypothesize the locations of objects in an image. Faster R-CNN is an object detection model and was introduced with improved architecture and performance and reduced training and detection time as the successor of Fast R-CNN. Unlike its predecessor, it is designed as a single model. Yet, the model consists of two modules. Because of the two-module architecture, Faster R-CNN belongs to the group of two-stage detectors, in opposition to one-stage detectors such as YOLO (You only look once; Redmon et al., 2016) or SSD (Single Shot Detector; Liu et al, 2016).

As illustrated in Figure 6, the first module is the Region Proposal Network (RPN), a CNN that proposes regions and types of objects in the regions. R-CNN and Fast R-CNN both used CPU-based algorithms, for instance, Selective Search, which had a significant negative impact on the region proposal time. The second module is the already mentioned Fast R-CNN, a CNN for extracting features, which is told by RPN where to look, and for classifying the image which then outputs class labels and bounding boxes. As depicted in Figure 6, region of interest (RoI) pooling is applied in the second stage which is essentially max pooling on the selected areas of interest. (Ren et al., 2015)

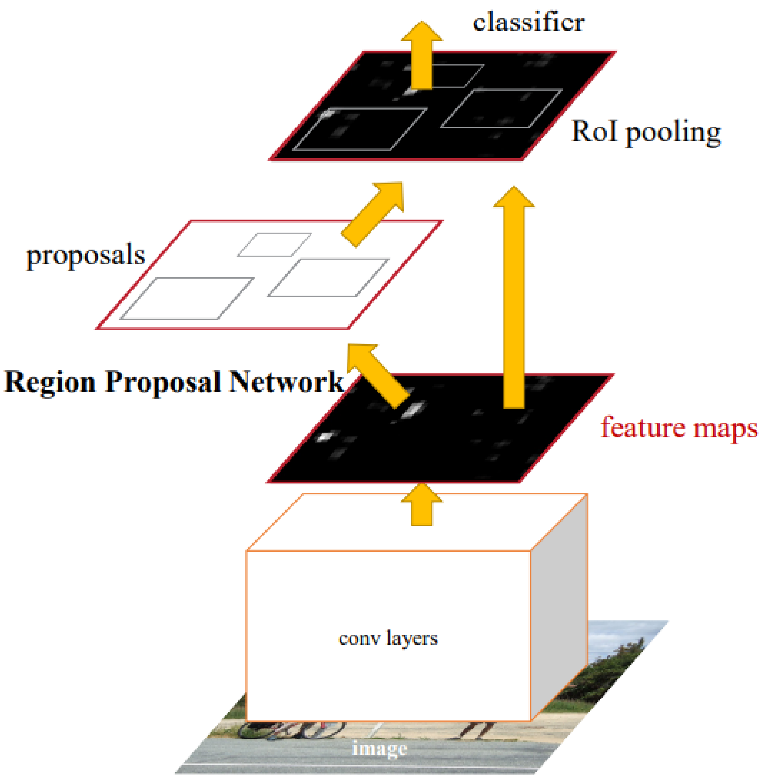


Figure 6. Summary of Faster R-CNN architecture (Ren et al., n.d.).

Many algorithms attempt to optimize a loss function. Such a function maps non-negative errors made during training, the differences between the ground truth and the prediction, and aims to minimize the average loss. In object detection, models optimize a multitask loss function which is a combination of a classification loss and a bounding box regression loss (Equation 1; Elgendy, 2020, p. 307).

Equation 1

$$L = L_{cls} + L_{loc}$$

Faster R-CNN consists of two CNNs, an RPN and a classification network. The model performs multitask learning and both of its networks are trained with multitask loss. Therefore, the model jointly trains altogether with four losses: RPN classification loss, RPN localization loss, bounding box regression loss and classification loss. RPN classification loss is a binary classification loss, and it represents the error of whether there is an object of interest or not in the proposed region. RPN localization loss regresses bounding box coordinates for each of the proposals. Bounding box regression loss indicates how tight the predictions are in comparison to ground truth, correcting errors coming from the RPN. And classification loss indicates the inaccuracy of predictions of object classes. (Ren et al., 2015)

2.4.2 Transfer Learning and Datasets

Transfer learning is a method in machine learning when knowledge from an existing model trained on some dataset is utilized in training another model on a different dataset which might have a different distribution (Goodfellow et al., 2016, p. 536). Goodfellow et al. (2016, p. 20) suggest that to achieve acceptable performance in supervised deep learning, an algorithm needs to be trained with around 5 000 labeled samples for each class and to achieve performance on the level of human or better, 10 million labeled samples would be required. Not only gathering such an amount of data but also labeling would be considerably costly. This problem has been continuously tackled by researchers and companies that have been sharing their datasets of different kinds for the public to use in their machine learning problems. With the use of these datasets and transfer learning, it is possible to train a neural network with significantly fewer samples in the dataset, for instance, a few hundred, and achieve satisfactory results in a new setting.

Extensive datasets of labeled images have been created for computer vision research. ImageNet is an ongoing project to provide data for training large-scale object recognition models and it currently includes over 14 million images which were annotated and belong to over 20 thousand categories (About ImageNet, n.d.; Deng et al., 2009). MS COCO (Common Objects in Context) is a dataset for object detection and segmentation that consists of more than 330 thousand images, out of which over 200 thousand are labeled (Lin et al., 2014). In the dataset, there are 80 object categories and 1.5 million object instances (Lin et al., 2014).

In recent years, there have been efforts in publishing datasets consisting of image documents. As Zhong et al. (2019) state, PubLayNet is a dataset developed for document layout analysis. It contains over 360 thousand document images with annotations of typical layout elements including tables. The documents in this dataset are scientific papers. They also provided some pre-trained models in their GitHub repository (zhxgj, 2020).

Based on Li et al. (2020), another large dataset designed for document layout analysis is DocBank, which includes 500 thousand of document pages. In this dataset, the labels include various types of elements, including tables and the documents are also scientific papers. Pre-trained models on the DocBank dataset are available in their GitHub repository as well (Li and wolfshow, 2021).

A more specific dataset is TableBank that is specializing in images with just one layout element. According to Li et al. (2020), it is a table detection and recognition dataset, consisting of image documents with 417 thousand labeled tables. Documents were sourced from the Internet and includes Word and Latex documents in English, Chinese, Japanese, Arabic and other languages. Like with the previous document datasets, pre-trained models are shared in their GitHub repository (Li, 2021). The PubLayNet and TableBanks datasets are among the datasets on which the available pre-trained models for LayoutParser were trained (lolipopshock, 2021).

Furthermore, to address a problem with small datasets of images, data augmentation techniques can be employed. This method is one of the regularization techniques to improve the performance of a model and prevent overfitting (Burkov, 2019, Section 8.4). The labeled dataset for training is expanded by adding synthetically transformed samples of the original samples. There are many different options for how images can be augmented, for instance rotating, flipping, cropping, or changing brightness. However, the original labels of objects in the images remain the same.

2.4.3 Metrics for Evaluation

To understand if the model achieved the desired accuracy and compare it to others, different methods are used for the evaluation of performance. Typically used evaluation methods for object detection tasks are precision, recall, and intersection over union.

To calculate precision and recall, the components of the confusion matrix need to be described. As shown in Figure 7, a common confusion matrix for binary classification problems (e.g., is a table, not a table prediction) has a 2×2 size and shows actual values (according to labels) on one axis and predicted values by model on the other axis. The components are defined as follows: True positive (TP) is the number of correct positive predictions. False positive (FP) is the number of incorrect positive predictions. False negative (FN) is the number of incorrect negative predictions (e.g., missed predictions of tables). True negative (TN) is the number of correct negative predictions. In object detection, this would naturally be all possible bounding boxes that were correctly not predicting the labeled class and thus, the number is not used for evaluation.

		Actual values	
		Positive	Negative
Predicted values	Positive	True positive (TP)	False positive (FP)
	Negative	False negative (FN)	True negative (TN)

Figure 7. Confusion matrix for binary classification.

Before defining precision and recall, it is important to understand how accurate the prediction of the bounding box was. Intersection over union (IoU) measures the overlap between the ground truth bounding box and predicted bounding box of the same object by diving their intersection area by their union area (see Figure 8). In an ideal situation, both areas would be the same and the result then equals 1. By setting a threshold to determine a correct or incorrect prediction and applying this evaluation to each prediction, true and false positives are identified. TP are predictions with IoU results equal to or bigger than the threshold. FP are predictions with IoU result smaller than the threshold.

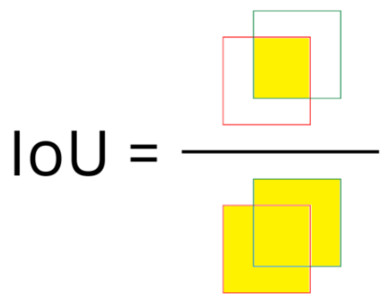


Figure 8. Intersection over Union.

As defined in the equations below, precision is indicating what proportion of positive predictions are correct (Equation 2). Recall (or sensitivity) is indicating what proportion of actual positive values was correctly identified, i.e., how sensitive is the model to detect true cases (Equation 3). A perfect model would have zero FP and FN and both precision and recall would thus be 1.[3] The precision and recall results are sensitive to the change of the threshold as the number of TP and FP changes with a different threshold.

Equation 2

$$Precision = \frac{TP}{TP + FP} = \frac{TP}{all\,predictions}$$

Equation 3

$$Recall = \frac{TP}{TP + FN} = \frac{TP}{all\;ground\;truths}$$

Average Precision (AP) and Average Recall (AR) are defined as the weighted means of precision and recall at each threshold. They are calculated for each class and the average of AP for each class is called Mean Average Precision (mAP). In this thesis, only one class will be predicted, i.e., table, and mAP will therefore not be calculated.

Table 1 shows that AP in COCO detection evaluation metrics is calculated over 10 different IoU thresholds. Two more APs are calculated for the IoU thresholds 0.5 and 0.75. AR is the maximum recall of a determined number of detection per image, averaged over classes and IoUs. Furthermore, COCO metrics provide evaluations for different sizes of objects – small, medium, and large. However, this distinction is

[3] In this thesis, the average precision and averarage recall will be reported as a franction of 100, not 1. Thus, the closer the results will be to 100, the better.

irrelevant to the prediction of tables as they all fall under large objects. The result of AP^{large} is hence identical to AP.

Table 1. COCO metrics (Detection Evaluation, n.d.).

Average Precision (AP):	
AP	% AP at IoU=.50:.05:.95 (primary metric)
$AP^{IoU=.50}$	% AP at IoU=.50 (PASCAL VOC metric)
$AP^{IoU=.75}$	% AP at IoU=.75 (strict metric)
AP Across Scales:	
AP^{small}	% AP for small objects: area < 32^2
AP^{medium}	% AP for medium objects: 32^2 < area < 96^2
AP^{large}	% AP for large objects: area > 96^2
Average Recall (AR):	
$AR^{max=1}$	% AR given 1 detection per image
$AR^{max=10}$	% AR given 10 detections per image
$AR^{max=100}$	% AR given 100 detections per image
AR Across Scales:	
AR^{small}	% AR for small objects: area < 32^2
AR^{medium}	% AR for medium objects: 32^2 < area < 96^2
AR^{large}	% AR for large objects: area > 96^2

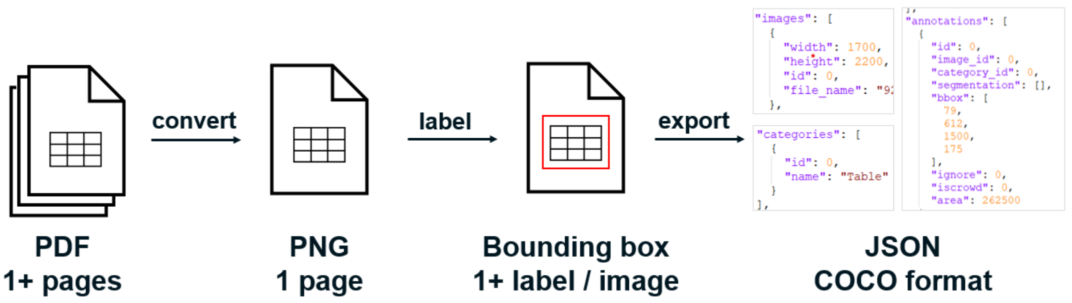
Nevertheless, having two metrics might cause difficulties when comparing different models. Thus, a single-number evaluation metric F1 score is typically used. F1 score is defined as the harmonic mean between precision and recall and is calculated as in the equation below (Equation 4).

Equation 4

$$F1 = 2 \times \frac{precision \times recall}{precision + recall} = \frac{2 \times TP}{2 \times TP + FP + FN}$$

3 DATA

The data used in experiments described in this thesis consisted of business documents provided by Wärtsilä.[4] The documents were provided by two departments from Wärtsilä, and they were all in PDF (Portable Document Format). The majority of the documents were in English, some were in German and French. As described in the Introduction, the main aim of this thesis is to fine-tune a model for detecting tables in an image document. Thus, each document used for forming the datasets had to include at least one table. Some of the documents had an identical layout of the document, including the layout of the tables, and they only differed in the content of their tables.



```
"images": [
  {
    "width": 1700,
    "height": 2200,
    "id": 0,
    "file_name": "9
  },
```

```
"categories": [
  {
    "id": 0,
    "name": "Table"
  }
],
```

```
"annotations": [
  {
    "id": 0,
    "image_id": 0,
    "category_id": 0,
    "segmentation": [],
    "bbox": [
      79,
      612,
      1500,
      175
    ],
    "ignore": 0,
    "iscrowd": 0,
    "area": 262500
```

Figure 9. Data processing.

The whole data processing is illustrated in Figure 9. There were initially 280 documents in PDF available for experiments. Most of the PDF documents consisted of a single page. Some documents could not be included in experiments due to their outlier nature, particularly photographed documents with heavily skewed tables and documents with tables overlapped by another piece of paper. After removing the outliers, the majority of documents were not scanned or photographed documents or showing any discoloration that can appear in scanned or photographed documents, and none of them was significantly rotated or skewed. Subsequently, the PDF documents were converted into images in PNG (Portable Network Graphics) format, so they could be labeled. After the conversion, there were 302 image documents available for experiments, where each image represented one page of a document.

[4] The documents provided by Wärtsilä showed in this thesis to illustrate examples of tables and their layouts have all text and information redacted by the author. Experiments were performed with the original documents.

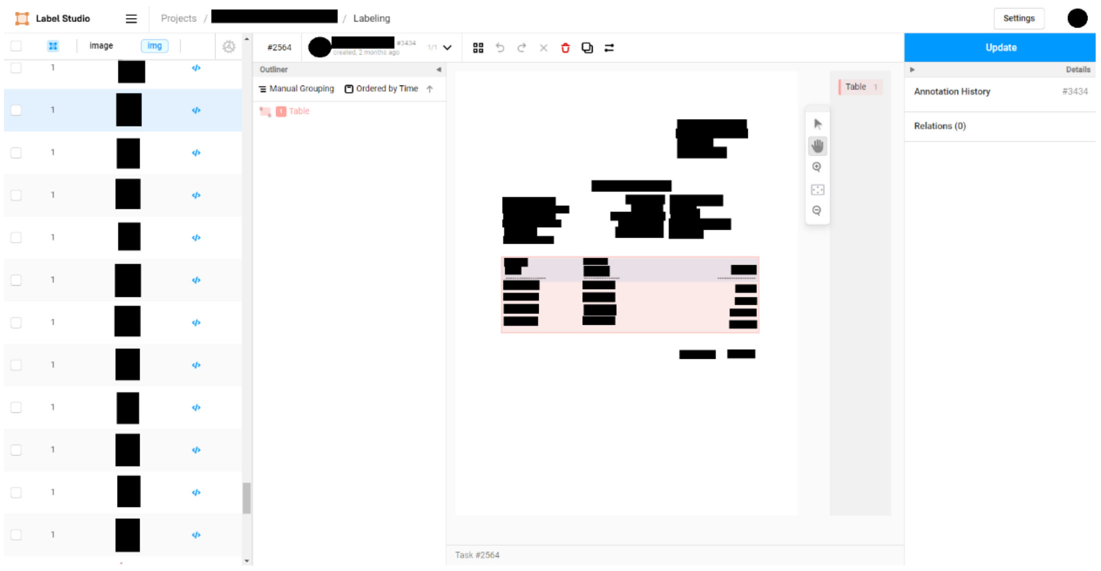


Figure 10. Example of an annotated document in Label Studio.

Tables in the image documents were labeled using an open-source data labeling tool called Label Studio as illustrated in Figure 10.[5] Labeling (also known as data annotation) is an important part of object detection as the quality of annotation has an impact on the performance of the trained model. Before starting, it was necessary to define classes of objects that would be labelled in the images. In this case, it was also necessary to approximately determine what is considered a table. A table was defined as a layout element that contains two or more column headings in the first row[6] and one or more rows containing table items. Some image documents contained bordered layout elements that consisted of a heading and one row with an item. These elements resembled a table. However, since the rows with items were not expected to increase with different documents, these layout elements were not labeled as a table (see Figure 11 a). Furthermore, some tables included a sum of table items at the bottom. This sum was sometimes located further away from the table. In these cases, the sum was not included in the table label (see Figure 11 c). Nevertheless, if the sum was located in a way that it appeared to belong to the layout of the table, it was included in the table label (see Figure 11 b).

[5] Label Studio is available in GitHub at https://github.com/heartexlabs/label-studio.
[6] In some image documents, the heading was split across the first two rows. The first column items were then located on the third row.

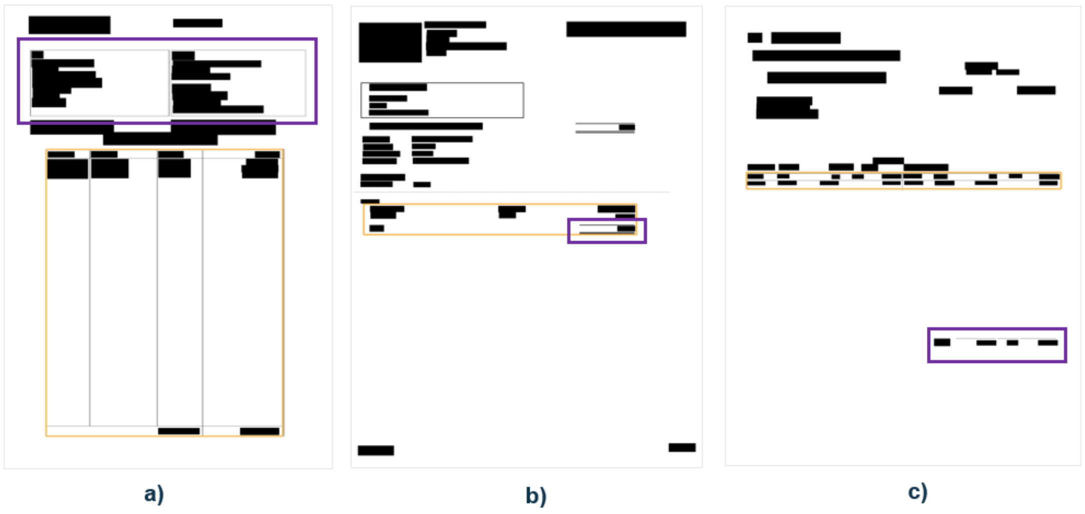


Figure 11. Example a) shows a document with a bordered table (yellow rectangle) and a bordered layout element (purple rectangle). Examples b) and c) show documents with tables (yellow rectangles) with sums (purple rectangles) relating to the content of the tables.

Afterwards, rectangle bounding boxes were drawn closely around each table in each image document. In this thesis, images were labeled only with one class – a table. Most image documents had one table per image (page). Then, the labeled dataset was exported from Label Studio in COCO format and saved in a JSON (JavaScript Object Notation) file. COCO format is a JSON structure determining the way of storing annotations and metadata for an image dataset. It is commonly used in object detection tasks, and it is an acceptable format of the dataset for training using the detectron2 library. An exported dataset consists of a JSON file and all images that include at least one labeled object.[7] Images that do not contain an object of interest are, therefore, not included in the dataset.

After image annotation, the dataset was ready for splitting into datasets for training, validation, and testing. However, before the splitting, the image documents were divided into two groups – documents with so-called bordered tables (125 documents) and borderless tables (177 documents). A borderless table was a table without any lines on its sides or between rows and columns, or a table with some lines on the side, usually top and bottom, and/or between the rows (see Figure 12 a and b). A bordered table was defined as a table delimited by a line on each side (see Figure 12 c).

[7] The size of the images does not change after the export from Label Studio.

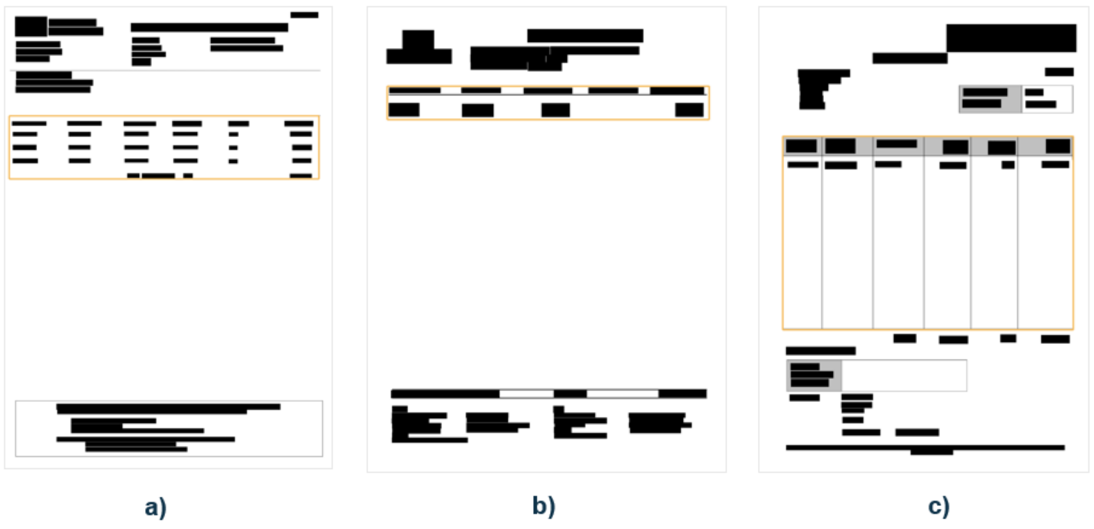


Figure 12. Examples of image documents with borderless (a and b) and bordered tables (c).

In this thesis, the split was in approximately 60-20-20% ratio for the respective sets. To create a balanced dataset, the documents were picked randomly from the two groups in the same ratio. Therefore, as shown in Table 2, 59% of the image documents in each dataset were documents with borderless tables. The train dataset consisted of 180 image documents. The validation and test datasets both had the same size, and each included 61 image documents.

Table 2. Division of images into train, validation, and test datasets.

	Train set	**Validation set**	**Test set**	**Total**	
All	180	61	61	302	100%
Bordered tables	74	25	25	125	41%
Borderless tables	106	36	36	177	59%

Because some image documents included more than one table on the page and this was not taken into account during the dataset split, the number of bordered and borderless table instances was not as balanced as the number of documents belonging to each group per each set. However, as shown in Table 3, the ratio was still relatively close to 41% of bordered and 59% of borderless table instances per set. If the instances ratio was greatly disproportional, it could have produced a biased comparison of the average precision results of the sets. For example, if the validation set had a proportionally much lower ratio of borderless instances than the test set, it could have resulted in better average precision results as borderless tables are more difficult to predict than bordered tables. Thus, the results in the next chapter will be also presented per each table group – bordered and borderless tables – to learn if the sets were balanced.

Table 3. Division of table instances across train, validation, and test datasets.

	Train set	**Validation set**	**Test set**	**Total**
All	275	91	105	471
Instances of bordered tables	103 37%	39 43%	42 40%	184 39%
Instances of borderless tables	172 63%	52 57%	63 60%	287 61%

4 EXPERIMENTS

In the first part of this chapter, the setting of training of models is described, including the choice of the pre-trained model for fine-tuning, selected hyperparameters, and data augmentation. The second part describes four experiments from which the best model was chosen for further evaluation.

4.1 Experiments setting

The thesis hypothesis is that fine-tuning a pre-trained model available in Model Zoo of LayoutParser will significantly improve at least the detection of borderless tables using the LayoutParser library.[8] All experiments to prove the hypothesis were conducted on an EC2 instance in AWS with GPU Tesla T4. All files required for training, datasets and configuration file, and outputs of training and evaluation runs were stored in an S3 bucket in AWS which was mounted to the EC2 instance. Training a model took from 6 to 7 hours when evaluation on the validation dataset was performed every 10 iterations of the training.

Python library Detectron2, on top of which is LayoutParser built, has an implementation for R-CNN models. In the Model Zoo of LayoutParser, there are 3 types of models trained on 6 different datasets available. The datasets relevant for a table-detecting task are PubLayNet and TableBank because they both contain the table layout element in their label maps. PubLayNet is an image document dataset that is trained to recognize text, title, list, and figure layout elements besides a table. Nevertheless, a visual quality check of inferences on a validation dataset with a confidence threshold of 0.8 showed that most tables were not recognized, especially the borderless tables, and some tables were misidentified as other elements, e.g., figures (see Table 4). Additionally, other elements, e.g., text or parts of text or titles, were often misidentified as tables. That happened less often in inferences done by models trained on the TableBank dataset. There are two Faster R-CNN models trained

[8] LayoutParser is available in GitHub at https://github.com/Layout-Parser/layout-parser.

on the TableBank dataset available in the Model Zoo. Their table predictions on the validation dataset were both more precise than the predictions of the model trained on the PubLayNet dataset and less text around the tables was included in the predicted table bounding box. Again, the majority of misrecognized tables were borderless tables. Faster R-CNN R 50 FPN 3x was more successful in detecting the tables than Faster R-CNN R 101 FPN 3x (see Table 4).

Table 4. Visual quality check of pre-trained models' predictions with confidence threshold 0.8 on the validation dataset.

	Table missed	Wrong element classification	Detected	Partly detected
Faster R-CNN R 101 FPN 3x PubLayNet	29	6	22	4
Faster R-CNN R 50 FPN 3x TableBank	25	--	34	2
Faster R-CNN R 101 FPN 3x TableBank	28	--	29	4
Note: If a substantial area around a table is included in the predicted bounding box or if a group of tables is detected as one table, also classified as a "missed table".				

However, as Table 5 shows, when the F1 score was calculated to compare the models, Faster R-CNN R 101 FPN 3x trained on the TableBank dataset had a better score than Faster R-CNN R 50 FPN 3x. On the other hand, Faster R-CNN R 50 FPN 3x trained on the PubLayNet dataset scored close to 0. Therefore, Faster R-CNN R 101 FPN 3x trained on the TableBank dataset was used for fine-tuning the table detector.

Table 5. Average precision and average recall results and F1 scores for pre-trained models.

	AP	AP50	AP75	AR (Max Dets = 1)	AR (10)	AR (100)	**F1 Score**
Faster R-CNN R 101 FPN 3x PubLayNet	0.47	1.73	0.01	0.10	4.50	28.20	**0.93**
Faster R-CNN R 50 FPN 3x TableBank	40.29	53.94	42.25	38.50	54.60	54.60	**46.36**
Faster R-CNN R 101 FPN 3x TableBank	41.77	54.95	40.83	37.70	58.00	58.00	**48.57**

Appendix 1 includes the configuration file of Faster R-CNN R 101 FPN 3x trained on the TableBank dataset (lolipopshock, 2021). Most of the hyperparameters were kept as defaults. Maximum iterations were lowered to 10 000 iterations in training. According to the default configuration, one iteration trains on 2 images. To reflect the change in maximum iterations, iteration numbers when the learning rate decreases by 0.1 were changed to iterations 6 000 and 8 000. As mentioned earlier, the evaluation period on the test dataset was set to every 10 iterations.

Based on the default configuration file, the input images are augmented by resizing the image. The minimum size of the smaller side of the image during training is sampled from 640, 672, 704, 736, 768, and 800. The maximum size of the longer side of the image is 1333. The aspect ratio is kept unchanged. In addition to the size transformation, random flip and random rotation of the image are added by the default augmentation setting in the training script of LayoutParser. The image is flipped horizontally with 0.5 probability and rotated at an angle between -90 and 0 degrees.

Moreover, two more image augmentations were gradually added – random brightness and random contrast. Brightness intensity is uniformly sampled from values between 0.8 and 1.2. Equally, contrast intensity is uniformly sampled from values between 0.8 and 1.2. The intensity of brightness as well as contrast decreases with values below 1 and increases with values above 1.

In inference, the transformation of image size is applied. The minimum size of the smaller side of the image is 800 and the maximum size of the longer side of the image is 1333. The aspect ratio is kept unchanged.

4.2 Results

Initially, four experiments were done using different data augmentation approaches. In Experiment I, the default augmentation setting from the training script of LayoutParser was applied, i.e., resizing, random flipping and random rotation of the images. In Experiment II, the data augmentation was reduced to image resizing. In Experiment III, random flipping and random brightness were added next to the image resizing. In Experiment IV, random contrast was added to the three types of augmentation from Experiment III. The evaluation of these experiments was done on the validation dataset.

During training, inference and subsequent evaluation are performed on unseen images from the validation dataset, and after training, evaluation is performed on the testing dataset. Training of a model outputs several files. The most important is the configuration file, which includes the changed hyperparameters, final weights[9], and the metrics file which contains the progress of the training.

The figures below depict the comparison of total training loss and validation loss. For a better understanding of the learning progress, the moving average of the losses was plotted over the plotted loss curves. Experiment I had a slightly different learning experience than Experiments II, III, and IV. The losses continued to decrease considerably until approximately iteration 6500, then remained stable with a gap bigger than in Experiments II, III, and IV (see Figure 13). Training loss stabilized around 0.6 and validation loss around 0.16, however, the variation remained rather high. On the other hand, total training and validation losses in Experiments II, III, and IV considerably decreased until approximately iteration 2000, then continued slightly decreasing until around iteration 6500 and remained stable afterwards with a smaller gap than in Experiment I (see Figure 14, Figure 15, and Figure 16). In Experiment II,

[9] It is possible to choose the iteration number x and save weights every x iteration.

training loss stabilized around 0.03 and validation loss around 0.08. In Experiment III, training loss stabilized around 0.03 and validation loss around 0.06. In Experiment IV, training loss stabilized around 0.03 and validation loss around 0.06. There was still a variation of the losses present at the end of the training, nevertheless, it was smaller.

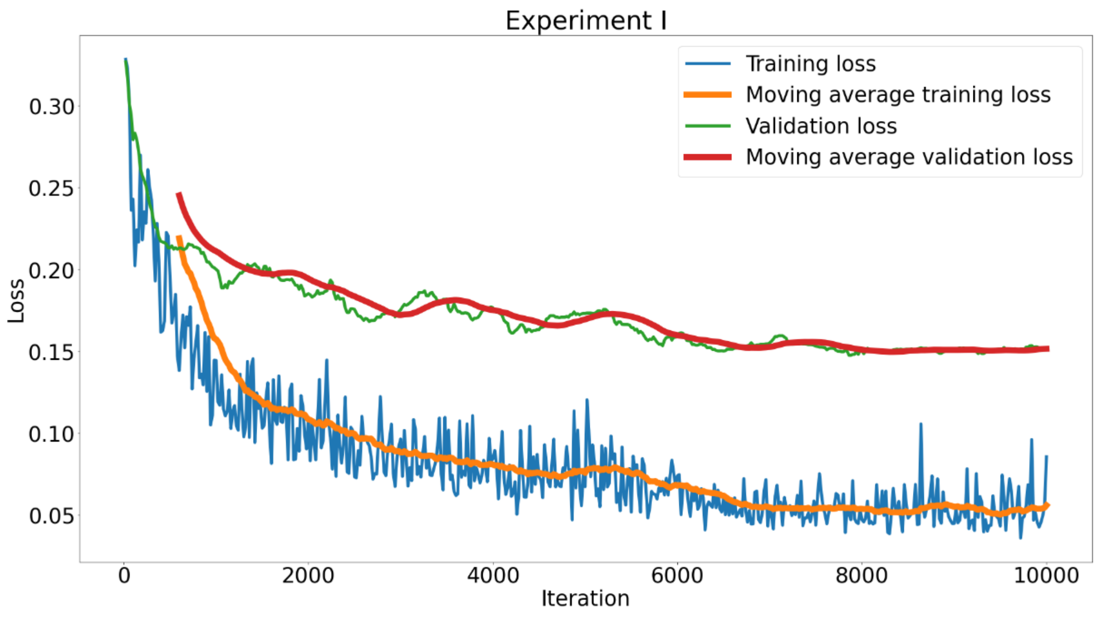


Figure 13. Comparison of training losses from Experiment I and validation dataset over 10 000 iterations.

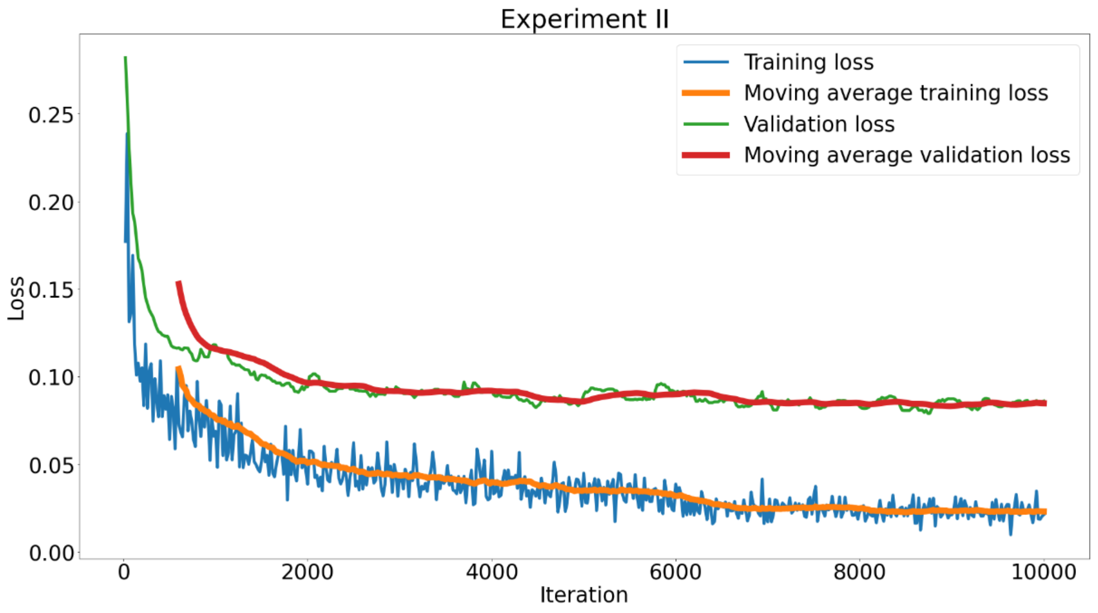


Figure 14. Comparison of training losses from Experiment II and validation dataset over 10 000 iterations.

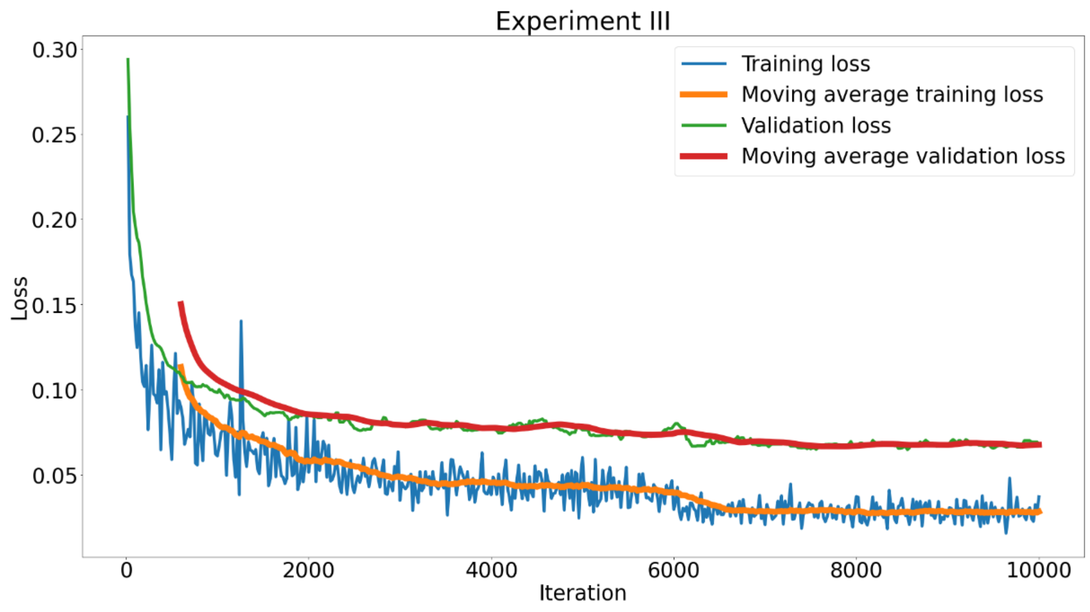


Figure 15. Comparison of training losses from Experiment III and validation dataset over 10 000 iterations.

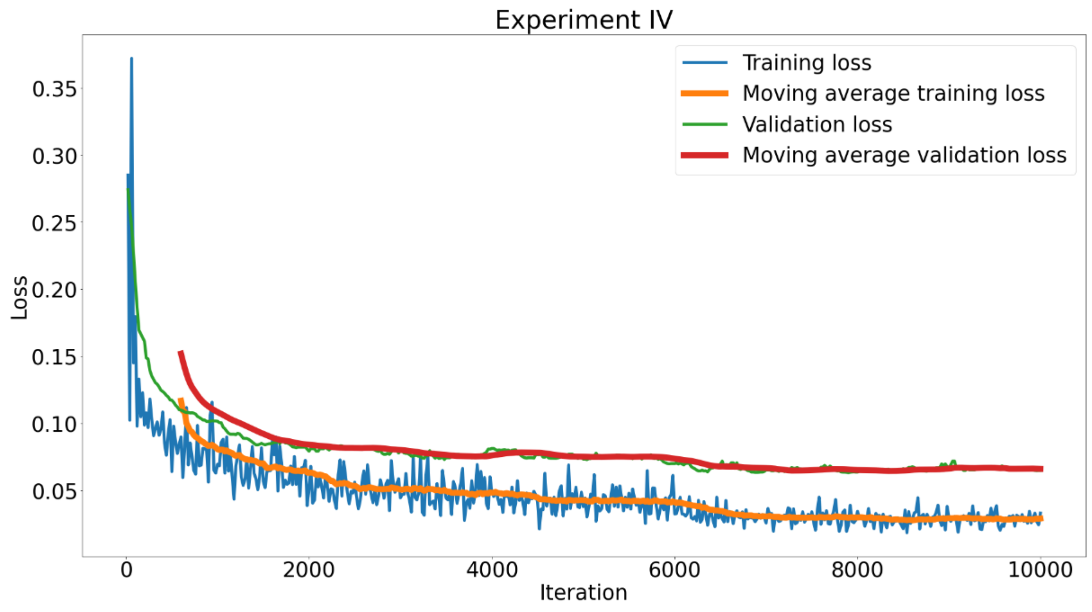


Figure 16. Comparison of training losses from Experiment IV and validation dataset over 10 000 iterations.

Average precisions and average recalls were obtained as outputs of the training of models and were used to calculate F1 scores for all four models. The F1 scores were used as the indicator of the best model. The model with the highest F1 score was then selected as the final model for evaluation on the test dataset. As seen in Table 6, the less strict AP50 metrics were very similar across all four models (95.4 for Experiment I, 96.0 for Experiment II, 95.9 for Experiment III, and 96.0 for Experiment IV).

However, the rest of the metrics were substantially smaller for the model from Experiment I with resizing, flipping and random rotation applied to the images (for example, AP was 67.5 for Experiment I, 89 for Experiment II, 86.6 for Experiment III, and 88.2 for Experiment IV; and AR (Max Dets = 100) was 72.7 for Experiment I, 91.1 for Experiment II, 89.2 for Experiment III, and 90.7 for Experiment IV). The models from Experiments II, III, and IV had very small differences in their results. The close results of average precisions and average recalls were imprinted in the close scores of F1 scores for Experiments II, III, and IV. The worst model was the model from Experient I with an F1 score of 70. The best model was the model from Experiment II with an F1 score of 90. Thus, the model with just resizing applied as an augmentation to the input images was selected as the final model for evaluation on the test dataset. However, the difference in the F1 score between the best model (Experiment II) and the second-best model (Experiment IV) was only 0.6.

Table 6. Average precision and average recall results and F1 scores for Experiments I, II, III, and IV.

	AP	AP50	AP75	AR (Max Dets = 1)	AR (Max Dets = 10)	AR (Max Dets = 100)	**F1 Score**
Experiment I: **Resize Shortest Edge of Image + Horizontal Flip + Random Rotation**	67.5	95.4	80.2	50.0	72.5	72.7	**70.0**
Experiment II: **Resize Shortest Edge of Image**	89.0	96.0	94.9	61.0	91.1	91.1	**90.0**
Experiment III: **Resize Shortest Edge of Image + Horizontal Flip + Random Brightness**	86.6	95.9	94.6	60.1	89.2	89.2	**87.9**
Experiment IV: **Resize Shortest Edge of Image + Horizontal Flip + Random Brightness + Random Contrast**	88.2	96.0	94.9	60.8	90.7	90.7	**89.4**

The below tables show the evaluation of the final model not only on the test dataset but also on the train and validation datasets to understand whether the split of the dataset into training, testing and validation datasets was balanced. Moreover, a comparison of the pre-trained model, Faster R-CNN R 101 FPN 3x trained on the TableBank dataset, and the final fine-tuned model from Experiment II was performed to find out whether an improvement was achieved. As shown in Table 7, the average precisions of the fine-tuned model for the training dataset, on which the fine-tuned model was trained, are higher than for the testing and validation datasets. Nevertheless, the results were consistent across the datasets. The average precision results were also consistent across the datasets for the pre-trained model. For the majority of the results (AP and AP75), average precisions for the fine-tuned model were more than double the average precisions of the pre-trained model. For example, AP for the pre-trained model on the validation was 41.77 set and 89 for the fine-tuned model which was a 2.1 times better result. Then, on the test set, APs were 39.09 and 86.48 respectively which was a 2.2 times better result for the fine-tuned model.

Table 7. Average precision evaluation on datasets including documents with bordered and borderless tables (mixed dataset) for the pre-trained and fine-tuned model.

Bordered and borderless		**AP**	**AP50**	**AP75**
Train	Pre-trained model	41.01	56.12	39.61
	Fine-tuned model	96.88	98.96	98.96
Validation	Pre-trained model	41.77	54.95	40.83
	Fine-tuned model	89.00	96.01	94.87
Test	Pre-trained model	39.09	49.44	43.10
	Fine-tuned model	86.48	93.96	92.89

Additionally, as seen in Table 8 and Table 9, the evaluation was performed on images with bordered and borderless tables separately to learn whether the improvement was different for each group and also to see whether the split was balanced with respect to the table groups. As in the previous table, the results were consistent across the datasets for the pre-trained model as well as the fine-tuned model, even though the results for the training dataset were consistently higher for the fine-tuned model and reached average precisions equal to or close to 100.

Whereas the pre-trained model scored significantly better on the datasets with only bordered tables than on the mixed datasets, the results decreased by more than half in some cases of average precisions on the datasets with only borderless tables in comparison to the results on the mixed dataset. For example, AP for the pre-trained model on the mixed test dataset was 39.09 (as shown in Table 7). Then, on the split test set according to the table groups, AP was 71.69 for the documents with bordered tables (as shown in Table 8) and 15.9 for documents with borderless tables (as shown in Table 9).

The results for average precisions for the fine-tuned model were substantially higher than for the pre-trained model for both, images with bordered as well as borderless tables. For example, on the test set for the documents with bordered tables, AP was 71.69 for the pre-trained model and 87.57 for the fine-tuned model (as shown in Table 8). Then, for the documents with borderless tables, APs were 15.9 and 85.51 respectively (as shown in Table 9).

As shown in Table 9, for the majority of the results (AP and AP75) on the validation and test sets for the documents with borderless tables, average precisions for the fine-tuned model were about five times better than the average precisions of the pre-trained model. For example, AP for the pre-trained model on the validation set was 16 and 87.62 for the fine-tuned model which is a 5.5 times improved result. Then, on the test set, APs were 15.9 and 85.51 respectively which is a 5.4 times better result for the fine-tuned model.

Table 8. Average precision evaluation on datasets including documents with bordered tables for pre-trained and fine-tuned models.

Bordered		**AP**	**AP50**	**AP75**
Train	Pre-trained model	74.90	84.31	78.00
	Fine-tuned model	99.20	100.00	100.00
Validation	Pre-trained model	74.34	78.75	78.75
	Fine-tuned model	91.50	96.88	96.88
Test	Pre-trained model	71.69	78.17	78.17
	Fine-tuned model	87.57	95.05	91.98

Table 9. Average precision evaluation on datasets including documents with borderless tables for pre-trained and fine-tuned models.

Borderless		**AP**	**AP50**	**AP75**
Train	Pre-trained model	19.11	37.41	14.02
	Fine-tuned model	95.94	98.91	98.91
Validation	Pre-trained model	16.00	33.88	10.30
	Fine-tuned model	87.62	96.04	93.90
Test	Pre-trained model	15.90	27.81	16.72
	Fine-tuned model	85.51	92.81	92.81

Lastly, a visual quality check was performed to evaluate predictions of the fine-tuned model on the testing and validation datasets with a confidence threshold of 0.8, and classify them as detected, fully or partly, or missed, fully and partly. As fully detected are considered predictions when a table is fully within a bounding box. Partly detected are detections missing part of the table such as half of the column. Fully missed were tables without any predictions and partly missed were image documents having only some tables on their page detected. Visual quality evaluation can not only evaluate the model's predictions with respect to the practical use of the results after training the model, but also reveal what cases were problematic during the training.

As shown in Table 10, in both datasets, only 1 document did not have any table detected in the image. In both cases, it was a borderless table, and it was a unique sample of the borderless table layout across the datasets. In the rest of the documents with the borderless tables, all tables or at least some of them were detected in each image. 83% of the image documents in the testing dataset (30 out of 36) and 92% of the image documents in the validation dataset (33 out of 36) had all tables correctly detected.

The bordered tables had the same results for the testing as well as the validation dataset. In all image documents, bordered tables were detected at least partly and 92% of all bordered tables were correctly detected (23 out of 25).

Table 10. Visual quality evaluation of fine-tuned model's predictions of bordered and borderless tables with confidence threshold 0.8 on validation and test datasets.

		Detected fully	Detected partly	Missed fully	Missed partly
Test set	Bordered	23	2	0	0
	Borderless	30	5	1	0
Validation set	Bordered	23	2	0	0
	Borderless	33	2	1	0

Moreover, the comparison of predictions of the pre-trained and fine-tuned models has shown two more improvements. Firstly, as mentioned earlier, most of the tables that were missed by the pre-trained model were borderless tables. Nevertheless, as illustrated in Figure 17, the predictions of bordered tables became more precise, i.e., closer to the ground truth and included the whole content of the table. Even though in this case the fine-tuned model incorrectly predicted the three small bordered layout elements, for the practical work with the final fine-tuned model, it is not an issue.

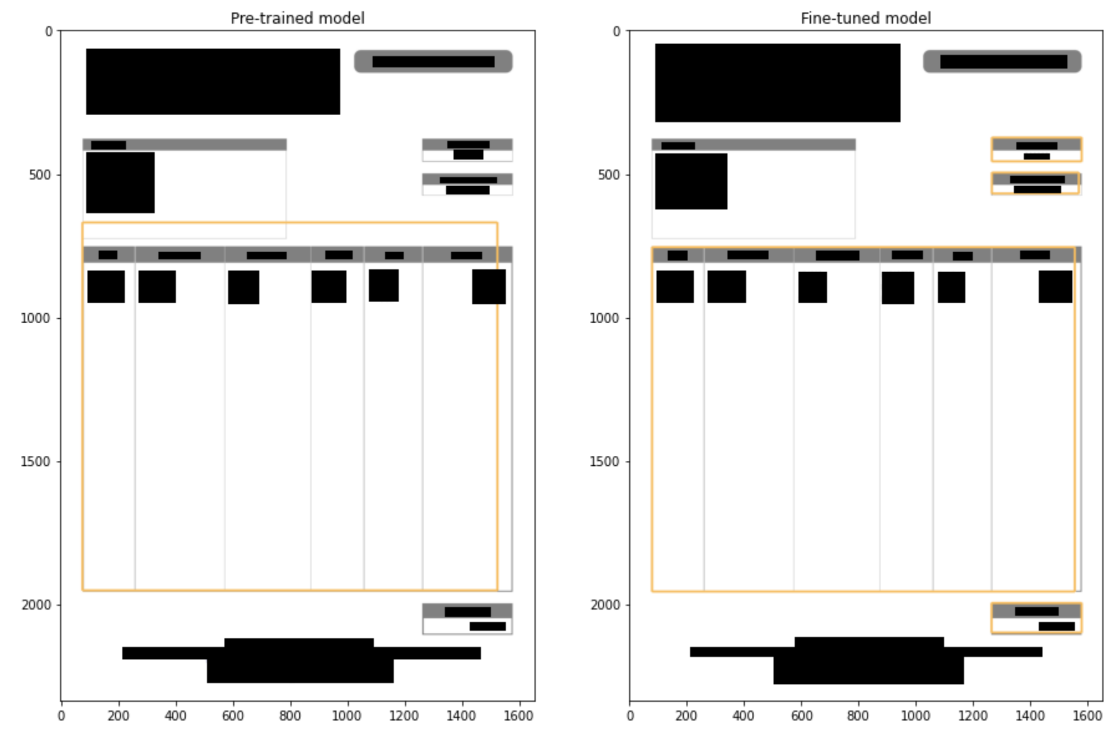


Figure 17. Example of improved prediction of a bordered table with confidence threshold at least 0.8 by the fine-tuned model (right) in comparison to the pre-trained model (left).

Secondly, despite the previous example, the detection of the correct layout elements has improved with the fine-tuned model. As illustrated in Figure 18, whereas the pre-trained model detected the bordered element as a table and missed the borderless table, the fine-tuned model correctly detected the borderless table in the image document and correctly did not detect the bordered element above it.

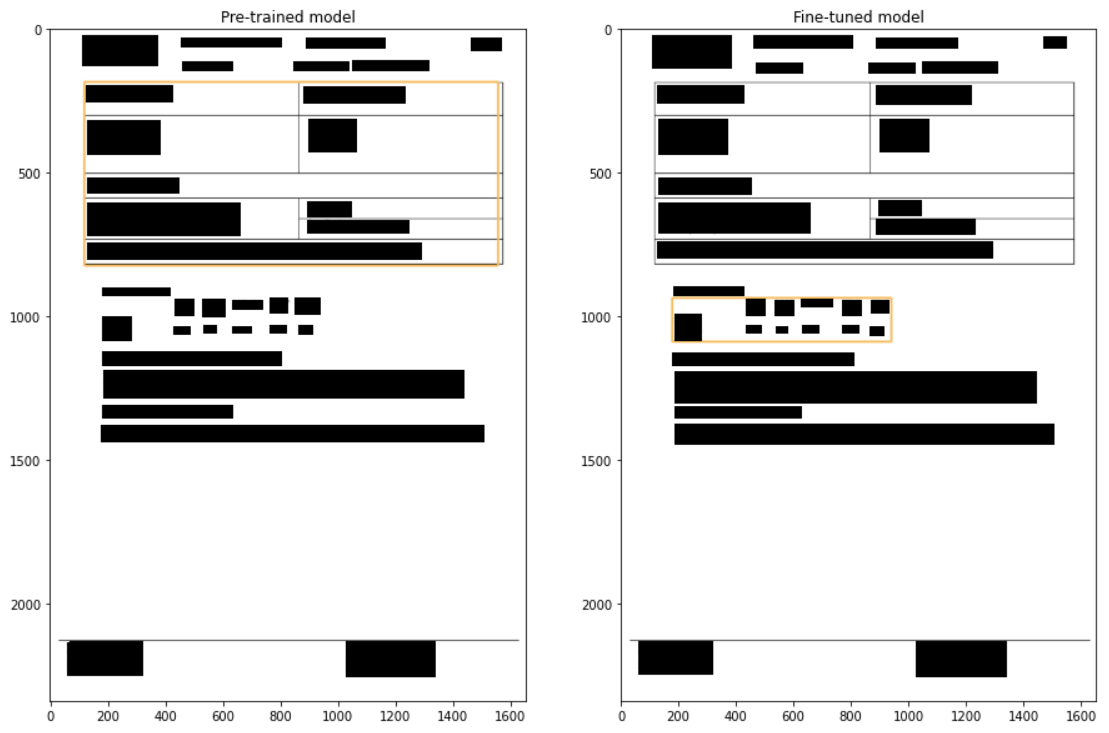


Figure 18. Example of improved prediction of a borderless table with confidence threshold at least 0.8 by the fine-tuned model (right) in comparison to the pre-trained model (left).

5 DISCUSSION

The experiments in the previous chapter showed that the default combination of image augmentation setting of LayoutParser, i.e., application of resizing of the image's shortest side, horizontal flipping and random rotation, was not the best approach for the custom dataset used in this thesis. The results were substantially better when the random rotation of the image was not applied to the training dataset. As the majority of image documents were not scanned documents and even the scanned documents did not appear to be significantly rotated or skewed, only slight changes in brightness and contrast were tested in combination with the resizing of images and horizontal flipping. The experiments proved that other tested combinations of augmentations do not make much difference in the results of the compared models. It is assumed that if the custom dataset contained more scanned or photographed image documents, the models with data augmentations reflecting the nature of these documents, such as random brightness, contrast, or even rotation and skewness, might have performed better. In previous studies of DLA on historical documents, random distortion, skewness and rotation did improve the performance of models (Baloun et al., 2021). Nonetheless, the results of the model with random rotation applied to the training images were also showing substantial improvement in the table detections, and would, therefore, still be a satisfactory improvement of the pre-trained model.

The maximum number of iterations chosen for the training of the models seemed to be optimal for learning the custom dataset. Even though the model was only slightly improving its loss approximately from 2000 iterations until 6500 iterations, the losses remained stable with a constant gap which suggests no overfitting or underfitting and that the model would not learn more with more time. As mentioned in the previous chapter, the learning rate was decreased by 0.1 after 6000 and 8000 iterations which perhaps affected the stabilization of the learning. A gap between training and validation loss that is too big could mean that the model does not have a suitable capacity for the complexity of the custom dataset. Nevertheless, the difference between the training loss and validation loss after 6500 iterations was approximately 0.05 which is not considered to be a big gap.

The evaluation of the final model on the testing and validation datasets showed that there was a very small difference in their results. The datasets were therefore balanced. The results were higher for the training datasets which had been expected because that was the data seen during training.

Furthermore, the comparison of performances of the final model and the pre-trained model proved that fine-tuning the model on the custom dataset substantially improved the results for detecting tables in the image documents used in this thesis. When comparing the performance separately on the image documents with bordered and borderless tables, the improvement was much bigger for the documents with borderless tables. Figure 19 illustrates the improved prediction on the comparison of predictions for the pre-trained and fine-tuned model on a document with a borderless table. The reason for this is probably a small representation of borderless tables in the TableBank dataset. In the performance of the final fine-tuned model, the results for borderless tables were slightly lower than for the bordered tables. This had been expected since it is generally easier to detect borders of tables with clear lines around all sides. IoU for the borderless tables could have been smaller and hence resulted in the smaller performance numbers. However, visually it could have still been a completely good prediction of the borderless table. That is why also visual quality check was performed.

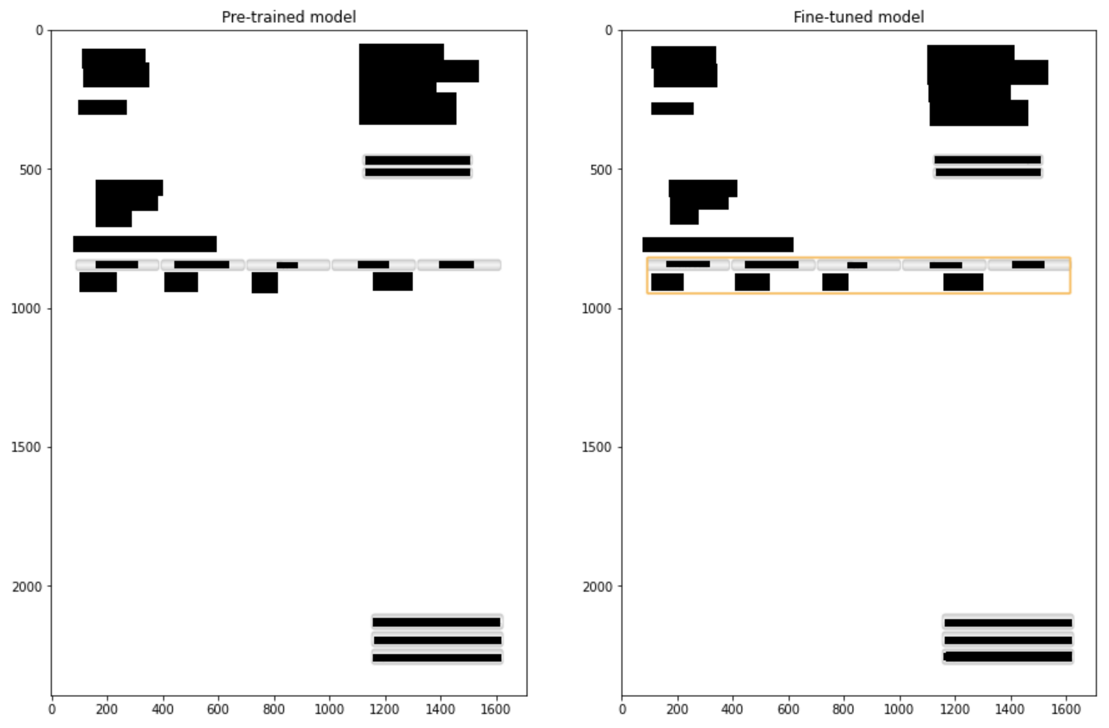


Figure 19. Example of improved prediction of a borderless table with confidence threshold at least 0.8 by the fine-tuned model (right) in comparison to the pre-trained model (left).

The visual quality evaluation confirmed the significant improvement of the table detections after fine-tuning the model. Almost all tables were correctly detected or at least detected in a way that the border could be adjusted to its correct position when working with the results (60 out of 61 in the validation as well as in the test dataset). The pre-trained model missed almost half of the tables in the validation dataset (28 out of 61). Whereas the fine-tuned model missed a table only in one document in the validation as well as in the testing dataset. In both cases, the table was a unique table layout in the whole custom dataset. This might suggest that the model would have problems with generalizing if applied to new image documents with unique table layouts. However, in other cases with unique tables, where the layout was more "traditional", the model generalized well (see examples in Figure 20).

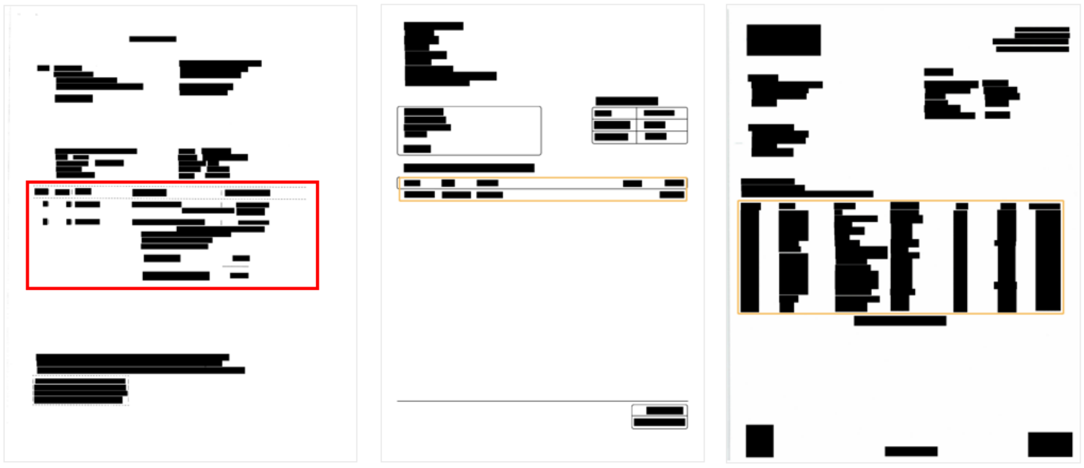


Figure 20. The three documents are from the validation dataset and the training dataset does not include a document with the same table layout. On the left is the document that did not have a table prediction with a confidence threshold of 0.8 and its layout is rather complex. The documents in the middle and on the right did have a correct table prediction with a confidence threshold of 0.8 and their layouts are rather "traditional" based on the position of text in rows and columns.

6 CONCLUSION

Big companies like Wärtsilä process an immense number of documents. Even if documents are digitalized, there might still occur an obstacle to extracting some information automatically. Apart from costs caused by extracting the information manually, documents may contain information that does not have the prime focus now but potentially is a source for higher business intelligence and better decisions in the future.

One of the problematic areas of information extraction from documents is tables. The proposed solution was table detection using LayoutParser, a tool for document layout analysis which enables the detection of various page elements in a document including tables. LayoutParser also allows using and fine-tuning of the existing pre-trained models available in its Model Zoo on the custom dataset. The main goal of this thesis was to accept or reject the hypothesis that fine-tuning a selected pre-trained model would significantly improve table detection in documents provided by Wärtsilä, at least in documents with borderless tables.

The model chosen for fine-tuning was Faster R-CNN R 101 FPN 3x trained on the TableBank dataset. After training with different data augmentations on input image documents, the model with the best results was from the experiment where only resizing of the shortest image side was applied. However, all the experiments resulted in improvement in comparison to the pre-trained model.

Further evaluation of the best model presented in this thesis, showed that the correctness of predictions has doubled on some metrics in comparison to the pre-trained model. Moreover, when looking at the results only for documents with borderless tables, the correctness of predictions has improved five times on some metrics. In addition to the evaluation metrics, visual quality evaluation of the predictions showed that the majority of the tables were detected well enough for further practical use. Thus, the hypothesis was accepted as fine-tuning improved the table detection of borderless as well as bordered tables on the custom dataset provided by Wärtsilä.

The model seemed to generalize well on the documents available for training. Nevertheless, the image documents used for training of models in this thesis were mostly containing a simple table layout and they were not significantly rotated, skewed or containing discoloration as a result of scanning or photographing the document. Further testing on table layouts unique to the dataset might show the need for additional fine-tuning. Further work could, therefore, improve the results on documents with more complex table layouts or documents that were poorly scanned or photographed.

REFERENCES

About Wärtsilä (n.d.). Wärtsilä. Retrieved January 8, 2023 from https://www.wartsila.com/about

About ImageNet (n.d.). ImageNet. Retrieved November 27, 2022 from https://www.image-net.org/about.php

Baloun, J., Král, P., & Lenc, L. (2021). ChronSeg: Novel Dataset for Segmentation of Handwritten Historical Chronicles. In *ICAART (2)* (pp. 314-322).

Burkov, A. (2019). The hundred-page machine learning book (Vol. 1, p. 32). Quebec City, QC, Canada: Andriy Burkov.

Camelot: PDF Table Extraction for Humans (2021, July 11). Camelot in GitHub. Retrieved January 13, 2023 from https://github.com/camelot-dev/camelot

Castrillón, M., Déniz, O., Hernández, D., & Lorenzo, J. (2011). A comparison of face and facial feature detectors based on the Viola–Jones general object detection framework. Machine Vision and Applications, 22(3), 481-494.

Cyganek, B. (2013). Object detection and recognition in digital images: theory and practice. John Wiley & Sons.

Deep Layout Parsing (n.d.). Layout Parser documentation. Retrieved December 15, 2022 from https://layout-parser.readthedocs.io/en/latest/example/deep_layout_parsing/index.html

Delaitre, V., Laptev, I., & Sivic, J. (2010). Recognizing human actions in still images: a study of bag-of-features and part-based representations. In BMVC 2010-21st British Machine Vision Conference.

Deng, J., Dong, W., Socher, R., Li, L. J., Li, K., & Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In 2009 IEEE conference on computer vision and pattern recognition (pp. 248-255). Ieee.

Detection Evaluation (n.d.). COCO - Common Objects in Context. Retrieved December 11, 2022 from https://cocodataset.org/#detection-eval

Elgendy, M. (2020). Deep learning for vision systems. Simon and Schuster.

Fei-Fei, L., Johnson, J., & Yeung, S. (2018, May). Lecture 11: Detection and Segmentation. CS231n: Deep Learning for Computer Vision. Retrieved November 27, 2022 from http://cs231n.stanford.edu/slides/2018/cs231n_2018_lecture11.pdf

Form Recognizer – Automated Data Processing Systems (n.d.). Microsoft Azure. Retrieved January 13, 2023 from https://azure.microsoft.com/en-us/products/form-recognizer/

Goodfellow, I., Bengio, Y., & Courville, A. (2016). Deep learning. MIT press.

Li, M. [liminghao1630] (2021, September 28). Model Zoo. TableBank in GitHub. Retrieved November 27, 2022 https://github.com/doc-analysis/TableBank/blob/master/MODEL_ZOO.md

Li, M. [liminghao1630], & wolfshow (2021, September 28). Model Zoo. DocBank in GitHub. Retrieved November 27, 2022 from https://github.com/doc-analysis/DocBank/blob/master/MODEL_ZOO.md

Li, M., Cui, L., Huang, S., Wei, F., Zhou, M., & Li, Z. (2020). Tablebank: Table benchmark for image-based table detection and recognition. In Proceedings of The 12th language resources and evaluation conference (pp. 1918-1925).

Li, M., Xu, Y., Cui, L., Huang, S., Wei, F., Li, Z., & Zhou, M. (2020). DocBank: A benchmark dataset for document layout analysis. arXiv preprint arXiv:2006.01038.

Lin, T. Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., ... & Zitnick, C. L. (2014). Microsoft coco: Common objects in context. In European conference on computer vision (pp. 740-755). Springer, Cham.

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C. Y., & Berg, A. C. (2016). Ssd: Single shot multibox detector. In European conference on computer vision (pp. 21-37). Springer, Cham.

lolipopshock (2021, August 21). Model Zoo. LayoutParser in GitHub. Retrieved November 27, 2022 from https://github.com/Layout-Parser/layout-parser/blob/main/docs/notes/modelzoo.md

Lombardi, F., & Marinai, S. (2020). Deep learning for historical document analysis and recognition—A survey. Journal of Imaging, 6(10), 110.

Mizuno, K., Terachi, Y., Takagi, K., Izumi, S., Kawaguchi, H., & Yoshimoto, M. (2012). Architectural study of HOG feature extraction processor for real-time object detection. In 2012 IEEE Workshop on Signal Processing Systems (pp. 197-202). IEEE.

Nagy, G., & Seth, S. C. (1984). Hierarchical representation of optically scanned documents. International Conference on Pattern Recognition, IEEE, 347–349.

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. In Proceedings of the IEEE conference on computer vision and pattern recognition (pp. 779-788).

Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. Advances in neural information processing systems, 28.

Ren, S., He, K., Girshick, R., & Sun, J. (n.d.) Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. Retrieved November 27, 2022 from https://asset-pdf.scinapse.io/prod/639708223/639708223.pdf

Shen, Z., Zhang, R., Dell, M., Lee, B. C. G., Carlson, J., & Li, W. (2021). LayoutParser: A unified toolkit for deep learning based document image analysis. In International Conference on Document Analysis and Recognition (pp. 131-146). Springer, Cham.

Szeliski, R. (2010). Computer Vision: Algorithms and Applications.

Tables - Amazon Textract (n.d.). Amazon Web Services. Retrieved January 13, 2023 from https://docs.aws.amazon.com/textract/latest/dg/how-it-works-tables.html

Tabula-py (2022, November 24). Tabula-py in GitHub. Retrieved January 13, 2023 from https://github.com/chezou/tabula-py

Wu, X., Xiao, L., Du, X., Zheng, Y., Li, X., Ma, T., & He, L. (2022). Cross-Domain Document Layout Analysis via Unsupervised Document Style Guide. arXiv preprint arXiv:2201.09407.

Wu, X., Zheng, Y., Ma, T., Ye, H., & He, L. (2021). Document image layout analysis via explicit edge embedding network. Information Sciences, 577, 436-448.

Wu, Y., Kirillov, A., Massa, F., Lo, W. Y., & Girshick, R. (2019, October 10). Detectron2: A PyTorch-based modular object detection library. Meta AI. Retrieved December 4, 2022 from https://ai.facebook.com/blog/-detectron2-a-pytorch-based-modular-object-detection-library-/

Xu, Y., Li, M., Cui, L., Huang, S., Wei, F., & Zhou, M. (2020). 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 (pp. 1192-1200).

Zhong, X., Tang, J., & Yepes, A. J. (2019, September). Publaynet: largest dataset ever for document layout analysis. In 2019 International Conference on Document Analysis and Recognition (ICDAR) (pp. 1015-1022). IEEE.

zhxgj (2020, March 26). Pre-trained Faster-RCNN and Mask-RCNN models on PubLayNet. PubLayNet in GitHub. Retrieved November 27, 2022 from https://github.com/ibm-aur-nlp/PubLayNet/tree/master/pre-trained-models

```
CUDNN_BENCHMARK: false
DATALOADER:
  ASPECT_RATIO_GROUPING: true
  FILTER_EMPTY_ANNOTATIONS: true
  NUM_WORKERS: 4
  REPEAT_THRESHOLD: 0.0
  SAMPLER_TRAIN: TrainingSampler
DATASETS:
  PRECOMPUTED_PROPOSAL_TOPK_TEST: 1000
  PRECOMPUTED_PROPOSAL_TOPK_TRAIN: 2000
  PROPOSAL_FILES_TEST: []
  PROPOSAL_FILES_TRAIN: []
  TEST:
  - word-val
  - latex-val
  TRAIN:
  - word
  - latex
GLOBAL:
  HACK: 1.0
INPUT:
  CROP:
    ENABLED: false
    SIZE:
    - 0.9
    - 0.9
    TYPE: relative_range
  FORMAT: BGR
  MASK_FORMAT: polygon
  MAX_SIZE_TEST: 1333
  MAX_SIZE_TRAIN: 1333
  MIN_SIZE_TEST: 800
  MIN_SIZE_TRAIN:
  - 640
  - 672
  - 704
  - 736
  - 768
  - 800
  MIN_SIZE_TRAIN_SAMPLING: choice
MODEL:
  ANCHOR_GENERATOR:
    ANGLES:
    - - -90
      - 0
      - 90
    ASPECT_RATIOS:
    - - 0.5
      - 1.0
      - 2.0
    NAME: DefaultAnchorGenerator
    OFFSET: 0.0
    SIZES:
    - - 32
    - - 64
    - - 128
    - - 256
    - - 512
  BACKBONE:
    FREEZE_AT: 2
    NAME: build_resnet_fpn_backbone
  DEVICE: cuda
  FPN:
    FUSE_TYPE: sum
    IN_FEATURES:
```

```
  - res2
  - res3
  - res4
  - res5
  NORM: "
  OUT_CHANNELS: 256
KEYPOINT_ON: false
LOAD_PROPOSALS: false
MASK_ON: false
META_ARCHITECTURE: GeneralizedRCNN
PANOPTIC_FPN:
  COMBINE:
    ENABLED: true
    INSTANCES_CONFIDENCE_THRESH: 0.5
    OVERLAP_THRESH: 0.5
    STUFF_AREA_LIMIT: 4096
  INSTANCE_LOSS_WEIGHT: 1.0
PIXEL_MEAN:
- 103.53
- 116.28
- 123.675
PIXEL_STD:
- 1.0
- 1.0
- 1.0
PROPOSAL_GENERATOR:
  MIN_SIZE: 0
  NAME: RPN
RESNETS:
  DEFORM_MODULATED: false
  DEFORM_NUM_GROUPS: 1
  DEFORM_ON_PER_STAGE:
  - false
  - false
  - false
  - false
  DEPTH: 101
  NORM: FrozenBN
  NUM_GROUPS: 1
  OUT_FEATURES:
  - res2
  - res3
  - res4
  - res5
  RES2_OUT_CHANNELS: 256
  RES5_DILATION: 1
  STEM_OUT_CHANNELS: 64
  STRIDE_IN_1X1: true
  WIDTH_PER_GROUP: 64
RETINANET:
  BBOX_REG_WEIGHTS:
  - 1.0
  - 1.0
  - 1.0
  - 1.0
  FOCAL_LOSS_ALPHA: 0.25
  FOCAL_LOSS_GAMMA: 2.0
  IN_FEATURES:
  - p3
  - p4
  - p5
  - p6
  - p7
  IOU_LABELS:
  - 0
  - -1
  - 1
  IOU_THRESHOLDS:
  - 0.4
  - 0.5
  NMS_THRESH_TEST: 0.5
  NUM_CLASSES: 80
  NUM_CONVS: 4
  PRIOR_PROB: 0.01
  SCORE_THRESH_TEST: 0.05
  SMOOTH_L1_LOSS_BETA: 0.1
  TOPK_CANDIDATES_TEST: 1000
ROI_BOX_CASCADE_HEAD:
  BBOX_REG_WEIGHTS:
  - - 10.0
    - 10.0
    - 5.0
    - 5.0
  - - 20.0
    - 20.0
    - 10.0
    - 10.0
  - - 30.0
    - 30.0
    - 15.0
    - 15.0
  IOUS:
  - 0.5
  - 0.6
  - 0.7
ROI_BOX_HEAD:
  BBOX_REG_WEIGHTS:
  - 10.0
  - 10.0
  - 5.0
  - 5.0
  CLS_AGNOSTIC_BBOX_REG: false
  CONV_DIM: 256
  FC_DIM: 1024
  NAME: FastRCNNConvFCHead
  NORM: ''
  NUM_CONV: 0
  NUM_FC: 2
  POOLER_RESOLUTION: 7
  POOLER_SAMPLING_RATIO: 0
  POOLER_TYPE: ROIAlignV2
  SMOOTH_L1_BETA: 0.0
  TRAIN_ON_PRED_BOXES: false
ROI_HEADS:
  BATCH_SIZE_PER_IMAGE: 512
  IN_FEATURES:
  - p2
  - p3
  - p4
  - p5
  IOU_LABELS:
  - 0
  - 1
  IOU_THRESHOLDS:
  - 0.5
  NAME: StandardROIHeads
  NMS_THRESH_TEST: 0.5
  NUM_CLASSES: 1
  POSITIVE_FRACTION: 0.25
  PROPOSAL_APPEND_GT: true
  SCORE_THRESH_TEST: 0.05
ROI_KEYPOINT_HEAD:
  CONV_DIMS:
  - 512
  - 512
  - 512
  - 512
  - 512
  - 512
  - 512
  - 512
  LOSS_WEIGHT: 1.0
  MIN_KEYPOINTS_PER_IMAGE: 1
  NAME: KRCNNConvDeconvUpsampleHead
  NORMALIZE_LOSS_BY_VISIBLE_KEYPOINTS: true
  NUM_KEYPOINTS: 17
  POOLER_RESOLUTION: 14
  POOLER_SAMPLING_RATIO: 0
  POOLER_TYPE: ROIAlignV2
ROI_MASK_HEAD:
  CLS_AGNOSTIC_MASK: false
  CONV_DIM: 256
  NAME: MaskRCNNConvUpsampleHead
  NORM: ''
  NUM_CONV: 4
  POOLER_RESOLUTION: 14
  POOLER_SAMPLING_RATIO: 0
  POOLER_TYPE: ROIAlignV2
RPN:
  BATCH_SIZE_PER_IMAGE: 256
  BBOX_REG_WEIGHTS:
  - 1.0
  - 1.0
  - 1.0
  - 1.0
  BOUNDARY_THRESH: -1
  HEAD_NAME: StandardRPNHead
  IN_FEATURES:
  - p2
  - p3
  - p4
  - p5
  - p6
  IOU_LABELS:
  - 0
  - -1
  - 1
  IOU_THRESHOLDS:
  - 0.3
  - 0.7
  LOSS_WEIGHT: 1.0
  NMS_THRESH: 0.7
  POSITIVE_FRACTION: 0.5
  POST_NMS_TOPK_TEST: 1000
  POST_NMS_TOPK_TRAIN: 1000
  PRE_NMS_TOPK_TEST: 1000
  PRE_NMS_TOPK_TRAIN: 2000
  SMOOTH_L1_BETA: 0.0
SEM_SEG_HEAD:
  COMMON_STRIDE: 4
  CONVS_DIM: 128
  IGNORE_VALUE: 255
  IN_FEATURES:
  - p2
  - p3
  - p4
  - p5
  LOSS_WEIGHT: 1.0
```

```
    NAME: SemSegFPNHead
    NORM: GN
    NUM_CLASSES: 54
  WEIGHTS: https://www.dropbox.com/s/6vzfk8lk9xvyitg/model_final.pth?dl=1
OUTPUT_DIR: outputs/faster_rcnn_R_101_FPN_3x/
SEED: -1
SOLVER:
  BASE_LR: 0.0005
  BIAS_LR_FACTOR: 1.0
  CHECKPOINT_PERIOD: 30000
  CLIP_GRADIENTS:
    CLIP_TYPE: value
    CLIP_VALUE: 1.0
    ENABLED: false
    NORM_TYPE: 2.0
  GAMMA: 0.1
  IMS_PER_BATCH: 2
  LR_SCHEDULER_NAME: WarmupMultiStepLR
  MAX_ITER: 270000
  MOMENTUM: 0.9
  NESTEROV: false
  STEPS:
  - 210000
  - 250000
  WARMUP_FACTOR: 0.001
  WARMUP_ITERS: 1000
  WARMUP_METHOD: linear
  WEIGHT_DECAY: 0.0001
  WEIGHT_DECAY_BIAS: 0.0001
  WEIGHT_DECAY_NORM: 0.0
TEST:
  AUG:
    ENABLED: false
    FLIP: true
    MAX_SIZE: 4000
    MIN_SIZES:
    - 400
    - 500
    - 600
    - 700
    - 800
    - 900
    - 1000
    - 1100
    - 1200
  DETECTIONS_PER_IMAGE: 100
  EVAL_PERIOD: 0
  EXPECTED_RESULTS: []
  KEYPOINT_OKS_SIGMAS: []
  PRECISE_BN:
    ENABLED: false
    NUM_ITER: 200
VERSION: 2
VIS_PERIOD: 0
```