Master's thesis
Master's Programme in Data Science

# Image Segmentation methods for fine-grained OCR Document Layout Analysis

Ari Vesalainen

September 5, 2022

Supervisor(s): Professor Mikko Tolonen

Professor Eetu Mäkelä

computer vision, image segmentation, deep learning

University of Helsinki
Faculty of Science

P. O. Box 68 (Pietari Kalmin katu 5)
00014 University of Helsinki

HELSINGIN YLIOPISTO — HELSINGFORS UNIVERSITET — UNIVERSITY OF HELSINKI

| Tiedekunta — Fakultet — Faculty | Koulutusohjelma — Utbildningsprogram — Degree programme |
|---|---|
| Faculty of Science | Master's Programme in Data Science |

Tekijä — Författare — Author

Ari Vesalainen

Työn nimi — Arbetets titel — Title

Image Segmentation methods for fine-grained OCR Document Layout Analysis



| Työn laji — Arbetets art — Level | Aika — Datum — Month and year | Sivumäärä — Sidantal — Number of pages |
|---|---|---|
| Master's thesis | September 5, 2022 | 59 |



Tiivistelmä — Referat — Abstract

Digitization has changed history research. The materials are available, and online archives make it easier to find the correct information and speed up the search for information. The remaining challenge is how to use modern digital methods to analyze the text of historical documents in more detail. This is an active research topic in digital humanities and computer science areas.

Document layout analysis is where computer vision object detection methods can be applied to historical documents to identify the document pages' present objects (i.e., page elements). The recent development in deep learning based computer vision provides excellent tools for this purpose. However, most reviewed systems focus on coarse-grained methods, where only the high-level page elements are detected (e.g., text, figures, tables). Fine-grained detection methods are required to be able to analyze texts on a more detailed level; for example, footnotes and marginalia are distinguished from the body text to enable proper analysis.

The thesis studies how image segmentation techniques can be used for fine-grained OCR document layout analysis. How to implement fine-grained page segmentation and region classification systems in practice, and what are the accuracy and the main challenges of such a system? The thesis includes implementing a layout analysis model that uses the instance segmentation method (Mask R-CNN). This implementation is compared against another existing layout analysis using the semantic segmentation method (U-net based P2PaLA implementation).

ACM Computing Classification System (CCS):
Computing methodologies → Image processing and computer vision → Segmentation
Computing methodologies → Artificial intelligence → Vision and scene understanding

Avainsanat — Nyckelord — Keywords

Säilytyspaikka — Förvaringsställe — Where deposited

Muita tietoja — Övriga uppgifter — Additional information

# Contents

# 1. Introduction

Digitization has revolutionized history research. At the turn of the millennium, studying historical materials always required visiting an archive or a library. With the latest development in digitization, online archives make it easier to find the correct information and speed up the search for information. Furthermore, everyone who wants can access this digitized data from their computer, regardless of time and place. For example, the National Archives of Finland has more than 150 million pages publicly available through its digital archive. The National Library of Finland has more than 24 million pages available [27] [28].

Although the materials are more readily available, using traditional methods for digital collections ( e.g., close reading) is difficult and time-consuming. Instead, one should use modern digital methods to analyze the text of historical documents in more detail, i.e., transform digital images of documents into text format and utilize text mining and other analysis tools for the research. Several techniques have been developed for this purpose and are still actively researched in the computer science and digital humanities domains.

Working on historical documents requires close cooperation between computer scientists and humanities researchers. Computer scientists are working on algorithms, methods, and tools to access the information while humanists are interested in collecting, organizing, and preserving data to discover information and knowledge [30].

This thesis focuses on printed materials, especially on different editions of David Hume's History of England written in the mid-1700, and contributes to the Critical Edition of David Hume's History of England project by providing tools and methods for the task. The project is currently working on providing full text of Hume's books, also showing all textual changes between editions. This work requires the ability to identify, detect and present different page elements separately, e.g., marginalia and body text, page headers and headings, and footnotes and catchwords should not be mixed.

One of the supporting techniques is optical character recognition (OCR), a technology that automatically converts printed text images into machine-encoded text. The image is a scanned document or a photo. The first computer-based OCR systems

were introduced during the 1970s [7]; however, this area of research has taken significant steps forward due to the latest development in computer vision and deep learning specifically.

Figure 1.1 describes the typical phases of the workflow, which utilize OCR methods [7], and hand-written text recognition (HTR) uses similar techniques. Raw material covers the materials in archives or libraries; materials can be historical hand-written documents or printed books. Therefore, the processing always starts with digitization: materials are scanned or photographed and stored in digital format. During the pre-processing phase, the system transforms image files to a uniform format and improves the quality of images for the remaining steps. Pre-processing includes image deskewing, resizing, normalization, noise reduction, and binarization methods [26] [29].

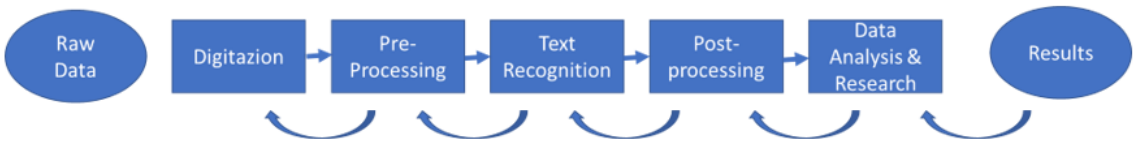



**Figure 1.1:** Typical phases of OCR or HTR process.

Text recognition covers two main steps: document layout analysis and recognition. The idea of layout analysis is to break the whole image into subparts (e.g., text regions and lines) that can be processed further. Actual character and word recognition happen in the recognition phase. First, the system identifies the characters during this phase. Then, an XML (extensible markup language) file or similar is created, including a reference to object location in the original image file and recognized text.

The researcher prepares the text material for the actual research during the post-processing phase. Depending on the research questions, a researcher can use different techniques. For example, the phase can include steps to improve OCR accuracy by comparing detected words against a given dictionary to find interpretation errors. The data analysis & research phase uses data mining and other techniques to extrapolate patterns and new knowledge from the existing data.

Several open-source OCR implementations are available, e.g., Tesseract, OCRopus, Kraken, and Calamari [51] [32] [18] [2]. The functionality of these implementations covers the entire OCR pipeline, i.e., document layout analysis and character recognition. Most implementations use recurrent neural networks with the Long Short Term Memory (LSTM) method for character recognition. Text recognition models are available for many languages [51], and users can train a new model if needed [51] [32] [18] [2]. The document layout analysis components of open-source implementation are simple and, in many cases, include only text region and text line detection [51] [32] [18].

OCRopus material indicates the development team is working on deep learning-based layout analysis; however, no further details are available [33].

Although several OCR systems are available, using them for a specific project is still challenging. Implementors of most systems have designed these for some particular purpose, and usage in a different environment is complicated. Systems are also poorly documented. Integration and adaptation to other environments would be a difficult task. Solutions working in one domain might need fine-tuning if applied to another; this can cover layout analysis and character recognition.

Modular OCR platforms have been introduced to overcome this challenge, e.g., OCR-D, LayoutParser, and Transkribus [29] [26] [47]. These platforms offer a full OCR pipeline that is modular and extendable. These platforms integrate several open-source OCR tools under one information architecture. The objective is to create workflows for mass-digitization of printed or hand-written historical material to produce texts of good quality for different purposes [29] [47].

As mentioned earlier, one of the main functions of OCR platforms is document layout analysis which consists of three tasks: line segmentation, page segmentation, and region classification. A document image is an input for all these three tasks. In line segmentation, the process tries identifying all individual text lines on a page. In page segmentation, the system segments image into various regions depending on the content of each region. Region classification labels each region. Page segmentation can be coarse-grained or fine-grained. Coarse-grained page segmentation detects and labels higher-level elements (e.g., text, image, table). In contrast, fine-grained page segmentation detects and labels regions on a detailed level, i.e., if the text region is body text, marginalia, footnote, or heading. OCR platforms can utilize pre-trained deep learning models for layout analysis tasks [29] [26] [47]. Figure 3.12 shows how the line and page segmentations differ. Region classification is an extension of page segmentation, which is not visible in the figure. Regions are polygons (red), and text lines are underlined (dark lilac).

The accuracy of OCR output is critical, and low accuracy can prevent future use of the data. OCR accuracy is measured using the character error rate (CER). CER calculates the edit distance between two strings (i.e., how many add-, delete- or change operations are required to transform one string into another).

$$CER = [(a + d + c)/n] * 100,$$

where n is the number of characters including spaces, a, d, and c are the number of add-, delete-, and change-operations, respectively.

Character recognition processes text lines one at a time, the document layout analysis does not impact the results, and as an outcome, CER of 1% is easy to reach

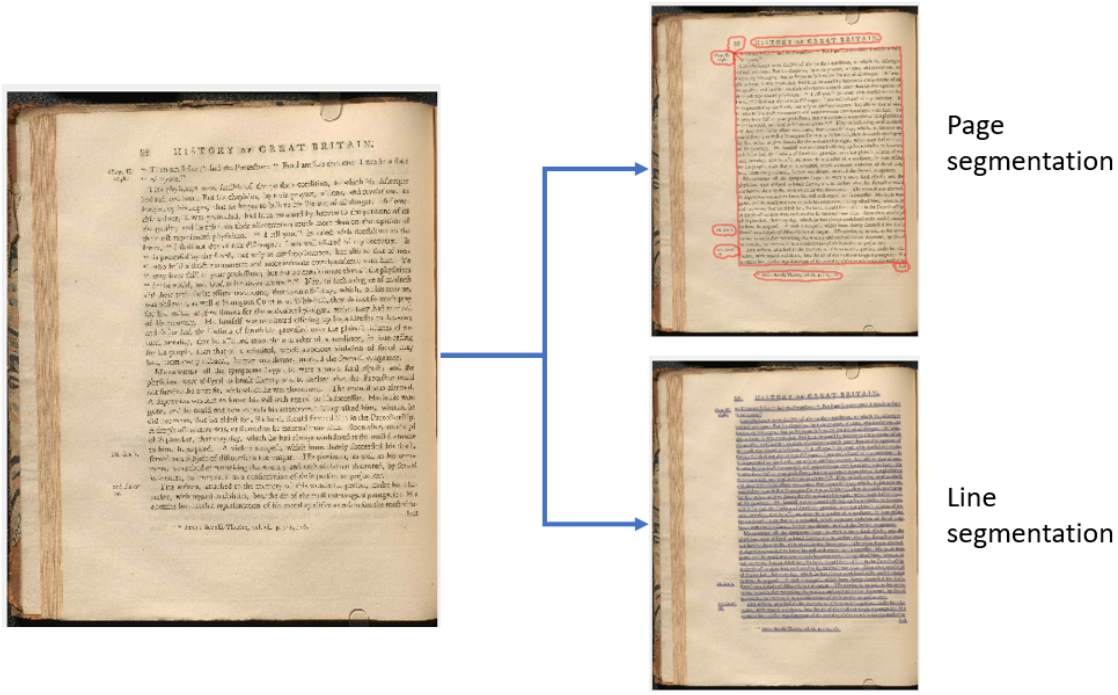



**Figure 1.2:** Main tasks of document layout analysis.

[52]. The primary sources for character interpretation errors are skewed text lines and noise on images. The proper pre-processing of images enables the elimination of this kind of error. However, line-level accuracy is not meaningful in practice as the researchers are more interested in the content of the whole page (i.e., words, sentences, and paragraphs) rather than characters on an individual line.

Practical tests for page-level accuracy show much worse performance for CER as it drops significantly, even as low as to 19% [52]. The analysis indicated poor page segmentation was the most significant factor for low performance. When CER was applied to each correctly segmented text region separately, the average CER was about 3%. These results highlight the importance of proper page segmentation as part of the end-to-end OCR pipeline.

As OCR processes digitized materials, it is natural to use computer vision methods and tools. Computer vision is a field of computer science that uses computers to understand digital images and videos automatically. Object detection and especially image segmentation are research areas of computer vision that aim to identify objects and their locations in digital images and videos. Computer vision and object detection have significantly progressed during the last decade due to machine learning and deep learning development. As a result, object detection methods have been successfully applied also to OCR.

Although the OCR process covers several phases, the primary purpose of this thesis is to focus on the document layout analysis phase. Specifically, how page segmentation and region classification tasks can provide fine-grained layout analysis of printed pages to detect different document elements. In the literature, document segmentation sometimes refers to the same function; however, this thesis systematically uses the document layout analysis term.

Hand-printing was the printing method during the $18^{th}$ century. Gutenberg developed this technique in the mid-1400, based on mechanical movable types. Before the year 1500, it was established and remained somewhat similar until the invention of machine printing around 1800 [10]. In this technique, a large sheet of paper covering several pages on each side (e.g., 2, 4, 8, 12, 16, 18, 24 pages) is printed at a time [10]. After printing, sheets were folded and cut to make a group of leaves, and one or several such groups were combined to form a book. Printed sheets include some unique markings (e.g., catchwords (first word of the next page), signature marks (location of the page in printed sheet)) to identify the correct place for each page in the book [10]. Additionally, each page can include irrelevant text, and the OCR recognition process should detect these (e.g., page headers or decorations). For detailed text analysis, it would be essential that all different text elements (e.g., body texts, paragraphs, headings, footnotes, marginalia) on a page are identified and can be processed separately.

Table 1.1 lists the page elements that are relevant for $18^{th}$-century books. The body text (e.g., paragraphs and headings) is the most critical text element from the content point of view. However, some elements are used for printing purposes only and do not include content relevance (e.g., signature-mark, catchword). Also, the significance of each element depends on the research question. For example, scholars focusing on linguistics might only be interested in body text, while scholars focusing on old printing techniques could be interested in all elements.

Figure 1.3 gives an example of a typical opening of a book printed in the $18^{th}$ century. It shows two pages. Each page has body text, footnotes, and headings (1). At the top of each page is a headline (2). In addition, it includes a running title on the left-hand page and a title of the chapter on the right-hand page. The page number is at the outer ends of the headline on each page. The direction line (3) at the bottom of each page includes a catchword and signature marks. The margins around the text are called head (4), tail (5), outer (6), and inner (7) margins.

Some elements are on every page (e.g., page number, body text, header, catchword). The appearance of footnotes and marginalia varies based on the book in question.

As mentioned above, this thesis focuses on printed materials printed in England during the $18^{th}$ century, especially on the works of David Hume. This thesis tries to

**Table 1.1:** Page element types in $18^{th}$ century books.

| Page element | Description |
|---|---|
| Body text | main content of the page |
| Paragraph | paragraphs of text inside the body text. In this thesis, paragraphs are not segmented separately, and these are part of the body text due to the annotation effort required |
| Heading | title located at the beginning of chapter or section |
| Header | text line printed at the top of each page of a book |
| Footer | text line printed at the bottom of each page of a book |
| Page number | number of the specific page |
| Drop capital | the first character of the chapter is usually printed much larger than the rest of the characters. In this thesis, drop capitals are not segmented separately |
| Decoration | text separator |
| Figure | picture or other illustrations |
| Marginalia-top | note(s) on margin, typically very close to the first line of body text and can include horizontal arch brackets |
| Marginalia | note(s) on margin |
| Footnote | note(s) at the bottom of the page |
| Endnote | note(s) collected and located at the end of the chapter |
| TOC-entry | Table of Content entry |
| Credit | list of credits |
| Signature mark | special printing marks to identify page or leaf location in a book |
| Catchword | a word of the first word of the following page helps to find the right order of the printed pages |
| Index | a list of words or phrases and associated locations in the book |

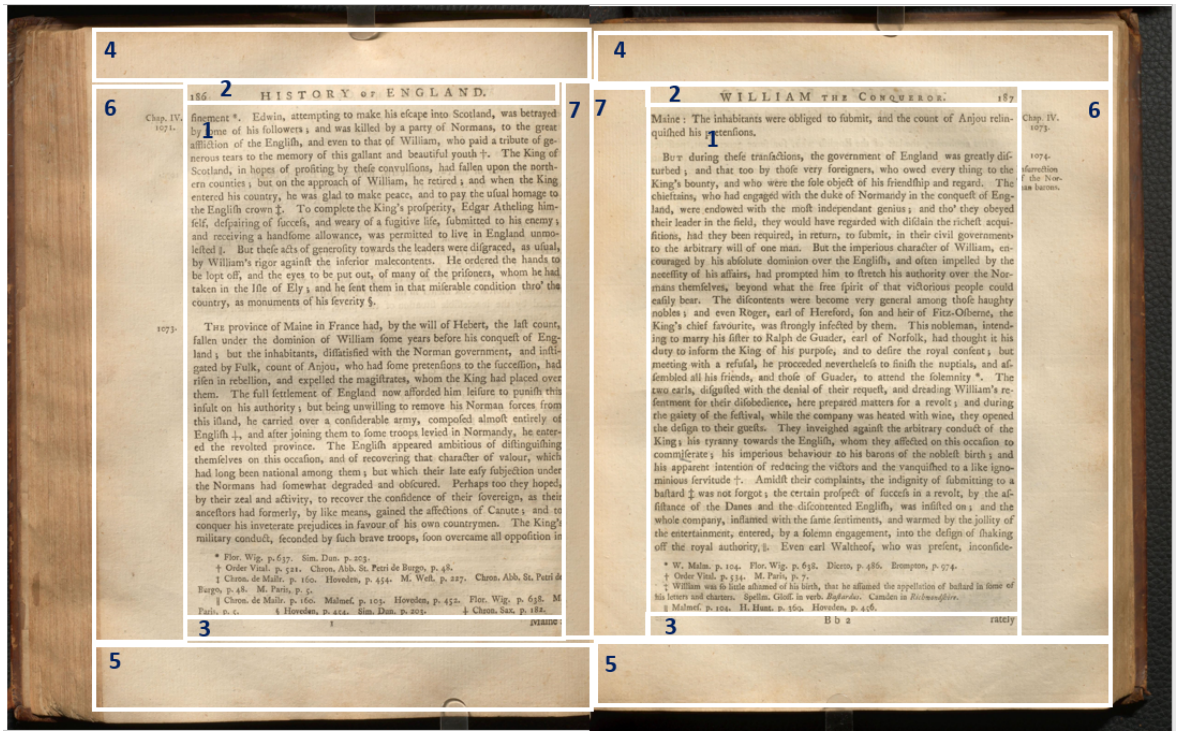



**Figure 1.3:** Typical opening of the $18^{th}$ century book.

answer the following questions:

- How to implement fine-grained page segmentation and region classification system using image segmentation methods?
- What is the accuracy and the main challenges of such a system?
- Is the proposed approach adequate for large-scale digitization projects?

This thesis is organized as follows. First, chapter 2 discussed the related work on text document segmentation. Then, chapter 3 describes computer vision's different object recognition tasks and how deep learning and convolutional neural networks (CNN) can resolve the problem. Next, chapter 4 describes the implementation of the Mask R-CNN layout analysis module and benchmarks it against another existing layout analysis module (P2PaLA). Finally, chapter 5 concludes the thesis.

# 2. Related works

Document segmentation or document layout analysis is a process of dividing the scanned image of a text document into different regions and identifying these regions [8]. Regions can be text lines, headers, paragraphs, graphics, illustrations, math symbols, and tables. Detection and labeling of the different regions are called geometric layout analysis [8]. The text regions inside the document (e.g., headers, footnotes, marginalia) can have a different semantic meaning, and therefore it is called logical layout analysis [8].

Document segmentation for text documents is an active research topic. According to the textual document segmentation algorithm survey between 2008 and 2016, researchers published more than 110 scientific articles [8]. Eskenazi et al. split the solutions into three groups: layout constrained by the algorithm (group 1), layout constrained by the parameters (group 2), and layout potentially unconstrained (group 3) [8].

Group 1 covers solutions that segment a specific, predefined layout. Grammar or a set of rules (e.g., Manhattan) defines the layout as mainly vertical and horizontal segments. Other types in this group are algorithms that use filters to make regions appear or classic computer vision algorithms that detect straight lines or square borders (e.g., Hough transform) [8].

Group 2 algorithms can adapt to local variations, enabling the segmentation of a broader range of layouts. The main types are clustering algorithms trying to identify regions based on geometry or texture, functional-analysis methods trying to identify region boundaries, and classification algorithms trying to identify regions based on a given set of features [8].

Group 3 is the latest addition, and it covers two main sub-groups: hybrid algorithms and neural network-based algorithms. Hybrid algorithms combine different solutions and try to overcome the limitations of other methods. On the other hand, neural network based algorithms use deep learning to identify correct segments [8].

The survey covers research articles from 2008 to 2016, and the dominance of deep learning-based solutions is already visible [8]. This trend continues, and deep learning is currently the primary document layout analysis technique [22]; therefore, this thesis

focuses on deep learning based techniques.

As described in the introduction, document layout analysis can be divided into three tasks: line segmentation, page segmentation, and region classification. In line segmentation, text lines are extracted from the input image containing only a single text line. Page segmentation aims to identify single text regions, an essential pre-processing step for text line detection. Finally, although the region classification focuses on separating different regions from each other, in its simplest form, it only separates text from non-text regions (e.g., figures). Most current OCR systems focus only on detecting text regions and extracting text lines. They are not capable of separating different element types [51] [32] [18].

This thesis focuses on page segmentation and region classification as part of the layout analysis of historical documents, it assumes that the document is a set of images, and each image represents a page or double page of the book. Although the focus is on printed materials, the same methods can be used for other historical documents, e.g., hand-written documents.

Document layout analysis can be coarse-grained or fine-grained. Coarse-grained detects different elements (e.g., text, separator, image, table), and fine-grained can also detect the semantic meaning of the page element (e.g., paragraph, marginalia, footnote). Most of the approaches presented in scientific papers are coarse-grained, and only few articles claim that these can be used for fine-grained detection. Unfortunately, no such systems were available for the evaluation.

In the OCR-D, the document layout analysis is called Optical Layout Recognition, and it includes all three tasks: line segmentation, page segmentation, and region classification. Page segmentation and region classification can be coarse-grained, or fine-grained [29].

OCR-D processing is based on OCR processors and each OCR processor is running a sub-task of full OCR pipeline. Several OCR processors are available based on open-source OCR packages. OCR-D project has built a framework to ensure these sub-tasks can be combined to run a full pipeline. OCR-D can utilize pre-trained deep learning models for character recognition and layout analysis. These models are called resources in OCR-D, and OCR-D has a uniform way of managing these resources [29].

Figure 2.1 shows three examples of layout analysis of OCR-D. The example on the left is based on the Tesseract layout analysis [51]. The page segmentation is minimal, and only two regions (i.e., paragraphs) are detected. The first paragraph includes the first text line and the header of the page, page number, and part of the marginalia. The second paragraph includes the rest of the text lines and part of the marginalia, catchword, and signature marks. The example in the middle represents an OCRopus layout analysis [32] and is even more limited; only one region (e.g., paragraph) is de-

tected. The paragraph includes the main body of the page; however, the page's header, catchword, and signature mark are also included. For some reason, the marginalia and page number are ignored. The example on the right is based on Eynollah segmentation engine [31]. The page segmentation detects separate page elements; however, text elements close to each other are combined and not separately presented as expected.

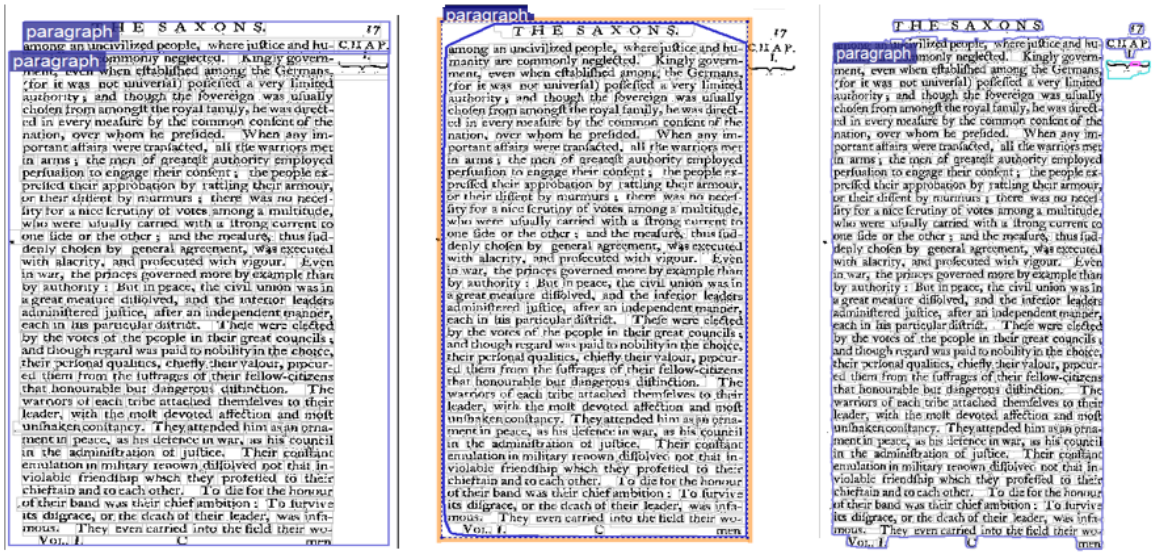



**Figure 2.1:** Example of OCR-D layout analysis results.

The Transkribus document layout analysis uses the CITlab model by default [15]. CITlab model can detect complex layouts like tables and text spanning multiple columns. However, it does not support region classification.

Figure 2.2 shows two examples of Transkribus' default page and line segmentation. Both pages are very similar; however, page segmentation detects one region including all different page elements (on the right) whereas the page on the left detects two regions, one region for marginalia's and one region for the rest of the page elements.

ParserLayout has a more advanced layout analysis than OCR-D or Transkribus, and it has several pre-trained deep learning models [47] for this. However, these models support coarse-grained detection only or can detect only specific items on images (e.g., table or equation).

ParserLayout capabilities were tested using different models for a page from a scientific article. The results are presented in figure 2.3. Figure a) shows an extract from table detection where orange bounding boxes are drawn for detected tables. As it can be seen, results are almost perfect; all tables are detected. The bounding boxes are accurately predicted with only one exception; the bounding box for the bottom left table is a little too small for the table. This seems to be fully in alignment with the accuracy metrics provided [20]. Figure b) is an example of coarse-grained detection

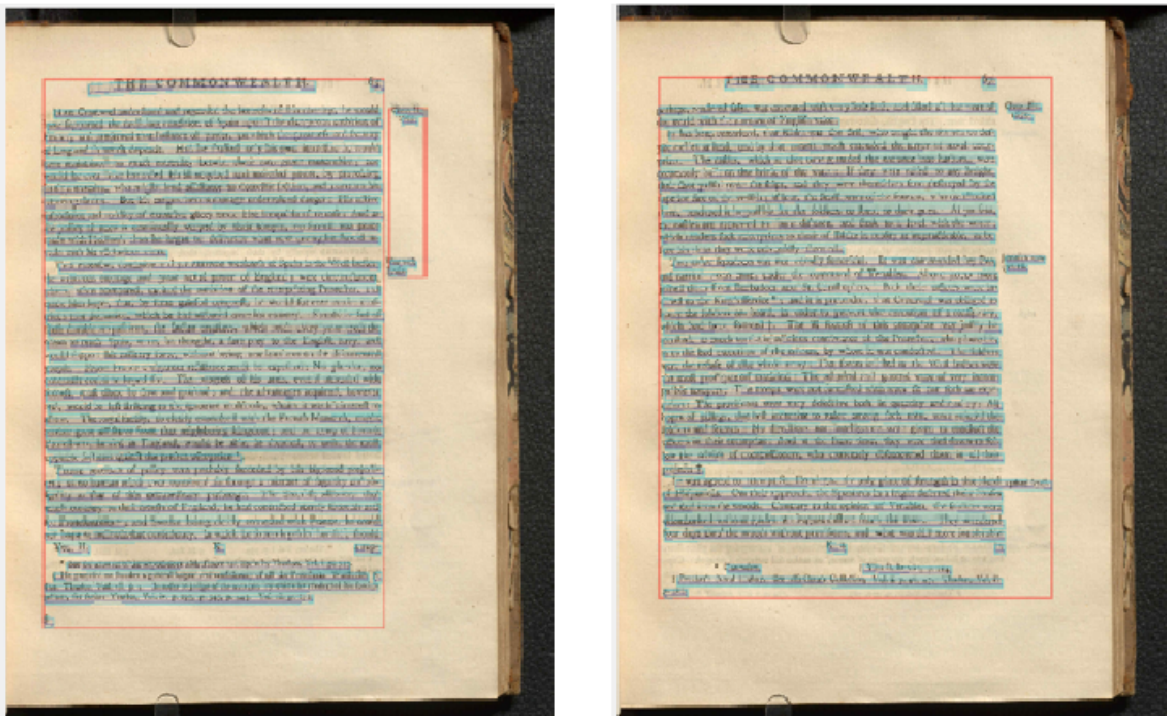


**Figure 2.2:** Example of Transkribus default layout analysis results.

(orange bounding boxes for text regions and ping bounding boxes for tables). It shows that the model has difficulties detecting tables from the image, and only two out of five tables are correctly detected. However, all text regions are detected although the bounding box for some regions seems slightly too small (e.g., text region on top-center).

dhSegment is a standalone layout analysis solution designed for historical documents, and it includes page extraction, baseline extraction, and layout analysis[34]. All these tasks can be done simultaneously using a semantic segmentation-based approach. dhSegment was trained for three different manuscript types. Each training experiment included 20 images for training, ten for evaluation, and ten for testing. DhSegment implements coarse-grained layout analysis as it can detect the following classes: text region, decoration, comments, and background. However, the solution is generic, and the fine-grained model can be trained if suitable training data is available. According to the authors, the achieved performance is good, and when compared to other similar solutions (e.g., P2PaLA), it was among the best performers [34].

Abuelwafa et al. present a method to detect footnotes from the printed books from the $18^{th}$ century [1]. They are using the Eighteenth-Century Collections Online (ECCO) database. ECCO database consists of page images and texts of material printed between 1700 and 1800 [14]. ECCO database covers approximately 32 million pages representing almost 185 thousand different book titles. Focusing only on footnotes provides a method to detect page elements at the detailed level. The solution uses

a)

| net-depth-features | AP | $AP_{50}$ | $AP_{75}$ |
|---|---|---|---|
| ResNet-50-C4 | 30.3 | 51.2 | 31.5 |
| ResNet-101-C4 | 32.7 | 54.2 | 34.3 |
| ResNet-50-FPN | 33.6 | 55.2 | 35.3 |
| ResNet-101-FPN | 35.4 | 57.3 | 37.5 |
| ResNeXt-101-FPN | **36.7** | **59.5** | **38.9** |

(a) **Backbone Architecture**: Better backbones bring expected gains: deeper networks do better, FPN outperforms C4 features, and ResNeXt improves on ResNet.

| | AP | $AP_{50}$ | $AP_{75}$ |
|---|---|---|---|
| *softmax* | 24.8 | 44.1 | 25.1 |
| *sigmoid* | **30.3** | **51.2** | **31.5** |
| | *+5.5* | *+7.1* | *+6.4* |

(b) **Multinomial *vs.* Independent Masks** (ResNet-50-C4): *Decoupling* via per-class binary masks (sigmoid) gives large gains over multinomial masks (softmax).

| | align? | bilinear? | agg. | AP | $AP_{50}$ | $AP_{75}$ |
|---|---|---|---|---|---|---|
| *RoIPool* [9] | | | max | 26.9 | 48.8 | 26.4 |
| *RoIWarp* [7] | | ✓ | max | 27.2 | 49.2 | 27.1 |
| | | ✓ | ave | 27.1 | 48.9 | 27.1 |
| *RoIAlign* | ✓ | ✓ | max | **30.2** | **51.0** | **31.8** |
| | ✓ | ✓ | ave | **30.3** | **51.2** | **31.5** |

(c) **RoIAlign** (ResNet-50-C4): Mask results with various RoI layers. Our RoIAlign layer improves AP by ~3 points and $AP_{75}$ by ~5 points. Using proper alignment is the only factor that contributes to the large gap between RoI layers.

| | AP | $AP_{50}$ | $AP_{75}$ | $AP^{bb}$ | $AP^{bb}_{50}$ | $AP^{bb}_{75}$ |
|---|---|---|---|---|---|---|
| *RoIPool* | 23.6 | 46.5 | 21.6 | 28.2 | 52.7 | 26.9 |
| *RoIAlign* | **30.9** | **51.8** | **32.1** | **34.0** | **55.3** | **36.4** |
| | *+7.3* | *+5.3* | *+10.5* | *+5.8* | *+2.6* | *+9.5* |

(d) **RoIAlign** (ResNet-50-**C5**, *stride 32*): Mask-level and box-level AP using *large-stride* features. Misalignments are more severe than with stride-16 features (Table 2c), resulting in massive accuracy gaps.

| | mask branch | AP | $AP_{50}$ | $AP_{75}$ |
|---|---|---|---|---|
| MLP | fc: 1024→1024→80·$28^2$ | 31.5 | 53.7 | 32.8 |
| MLP | fc: 1024→1024→1024→80·$28^2$ | 31.5 | 54.0 | 32.6 |
| **FCN** | conv: 256→256→256→256→256→80 | **33.6** | **55.2** | **35.3** |

(e) **Mask Branch** (ResNet-50-FPN): Fully convolutional networks (FCN) *vs.* multi-layer perceptrons (MLP, fully-connected) for mask prediction. FCNs improve results as they take advantage of explicitly encoding spatial layout.

Table 2. **Ablations** for Mask R-CNN. We train on `trainval35k`, test on `minival`, and report *mask* AP unless otherwise noted.

b)

| net-depth-features | AP | $AP_{50}$ | $AP_{75}$ |
|---|---|---|---|
| ResNet-50-C4 | 30.3 | 51.2 | 31.5 |
| ResNet-101-C4 | 32.7 | 54.2 | 34.3 |
| ResNet-50-FPN | 33.6 | 55.2 | 35.3 |
| ResNet-101-FPN | 35.4 | 57.3 | 37.5 |
| ResNeXt-101-FPN | **36.7** | **59.5** | **38.9** |

| | AP | $AP_{50}$ | $AP_{75}$ |
|---|---|---|---|
| *softmax* | 24.8 | 44.1 | 25.1 |
| *sigmoid* | **30.3** | **51.2** | **31.5** |
| | *+5.5* | *+7.1* | *+6.4* |

| | align? | bilinear? | agg. | AP | $AP_{50}$ | $AP_{75}$ |
|---|---|---|---|---|---|---|
| *RoIPool* [9] | | | max | 26.9 | 48.8 | 26.4 |
| *RoIWarp* [7] | | ✓ | max | 27.2 | 49.2 | 27.1 |
| | | ✓ | ave | 27.1 | 48.9 | 27.1 |
| *RoIAlign* | ✓ | ✓ | max | **30.2** | **51.0** | **31.8** |
| | ✓ | ✓ | ave | **30.3** | **51.2** | **31.5** |

| | AP | $AP_{50}$ | $AP_{75}$ | $AP^{bb}$ | $AP^{bb}_{50}$ | $AP^{bb}_{75}$ |
|---|---|---|---|---|---|---|
| *RoIPool* | 23.6 | 46.5 | 21.6 | 28.2 | 52.7 | 26.9 |
| *RoIAlign* | **30.9** | **51.8** | **32.1** | **34.0** | **55.3** | **36.4** |
| | *+7.3* | *+5.3* | *+10.5* | *+5.8* | *+2.6* | *+9.5* |

| | mask branch | AP | $AP_{50}$ | $AP_{75}$ |
|---|---|---|---|---|
| MLP | fc: 1024→1024→80·$28^2$ | 31.5 | 53.7 | 32.8 |
| MLP | fc: 1024→1024→1024→80·$28^2$ | 31.5 | 54.0 | 32.6 |
| **FCN** | conv: 256→256→256→256→256→80 | **33.6** | **55.2** | **35.3** |

**Figure 2.3:** Example of LayoutParser layout analysis results: table detection (a) and region detection (b).

four different methods, where two are traditional computer vision methods (rule-based and layout-based), and two are CNN-based methods. The final method is an ensemble method where the final prediction is based on voting. Authors state that the precision is 96.2% and recall 67.9%, i.e., two out of three footnotes are correctly detected with very high confidence.

The examples above demonstrates the shortcomings of the current layout analysis systems. OCR systems focus on detecting text lines and can only detect a limited number of different types of elements from an image of a document. This kind of coarse-grained detection works for simple layouts, but for more complex layouts fine-grained method is required. Some references to fine-grained capabilities were mentioned in articles; however, no results or further details were presented.

If document layout analysis is not capable of separating different page elements correctly, these can be mixed during the process and result may include undesired results, e.g., paragraph can include fractions of marginalia in the middle of the text and output is not understandable. Another problem is that layout analysis should

provide consistent output otherwise the results can not be reliably processed later during the process.

# 3. Object recognition and CNNs

As described earlier, document layout analysis focuses on detecting different page elements (i.e., objects) from digital images representing the book's pages. This task is a typical object recognition problem seen in computer vision, and the same tools, methods, and algorithms can be used here as well. This chapter briefly introduces object recognition. As the deep learning based methods are the most commonly used algorithm for object recognition [22], this chapter introduces how convolutional neural networks are used for this purpose.

## 3.1 Object recognition

Object recognition is essential to computer vision, describing the tasks that identify different objects in digital images. Object recognition can be divided into image classification, object localization, object detection, and image segmentation [49]. First, image classification assigns a class label to an image. Object localization detects the physical coordinates of an object and indicates these by drawing a bounding box around the object. Object detection combines two previous tasks and provides locations and labels for objects in an image. Finally, object or image segmentation classifies every pixel according to its semantic label [49].

Image segmentation covers three types: semantic, instance, and panoptic [49]. Semantic segmentation classifies every pixel in the image into a class according to its context, i.e., these classes could be person, cat, dog, text, graphics, or page number in the case of OCR. Instance segmentation classifies every pixel in the image into a class, and each pixel is assigned to an instance of an object. The difference between semantic segmentation and instance segmentation is that if semantic segmentation identifies a group of people in the image, then instance segmentation identifies each person separately. Finally, panoptic segmentation combines two other segmentation methods: classifying all pixels in the image as belonging to a class (semantic segmentation) and identifying what instance of this class these pixels belong to (instance segmentation). It also labels so-called stuff (e.g., grass, sky).

Figure 3.1 illustrates the differences between different object recognition tasks.

The image classification process provides labels for the object in the image (i.e., bottle, cup, and cube). Object localization provides bounding boxes for each identified object (coordinates of top left and bottom right corners). Semantic segmentation provides four binary masks, one for each class (lilac for cubes, light green for cup, and blue for a bottle) and one for background (yellow). Instance segmentation provides five binary masks for each object instance (e.g., one for the bottle (blue), one for the cup (light green), and three for cube instances (red, dark green, and light blue)). Background mask is typically ignored in instance segmentation.

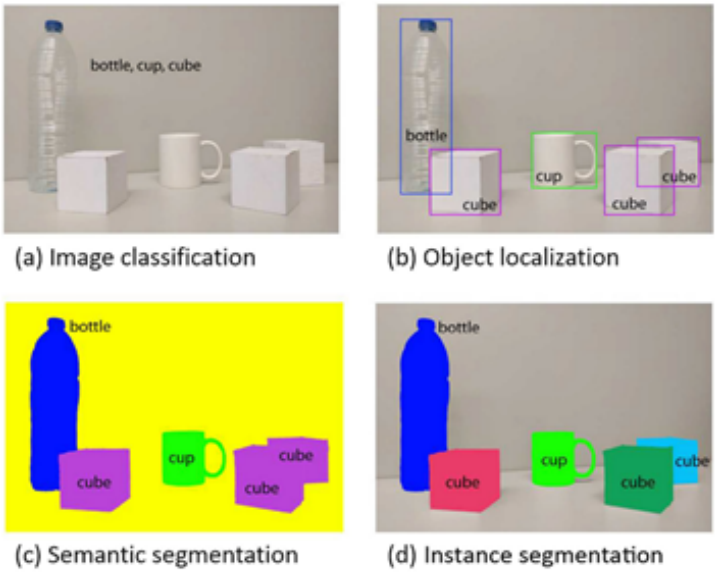



**Figure 3.1:** Examples of differences between object recognition types [9]

## 3.2 CNN key concepts

Deep learning is a machine learning method based on neural networks and has been applied to computer vision. The convolutional neural network (CNN) is a particular neural network model designed for working with two-dimensional data, e.g., images. A digital image represents visual data in a grid-like format containing a series of pixels that define the color and brightness of each point in an image. This thesis assumes that the reader is familiar with the neural network's functionality and concepts; if not, please, study this from [13].

### 3.2.1 CNN building blocks

A CNN typically has three layers: convolutional, pooling and fully connected layers [13]. The core part of the CNN is the convolutional layer as it performs an operation called a 'convolution'. Convolution is a dot product between two matrices, input data, and a two-dimensional array of weights called a filter or a kernel. As the kernel is spatially smaller than the input image, the element-wise multiplication between the filter and filter-sized patch of the input is applied systematically from left to right and from top to bottom. The two-dimensional output array is called a feature map. It can be passed through a nonlinear activation function (e.g., Rectified Linear Unit (ReLU)) to be used for the following layers.

Figure 3.2 shows an example of the convolution operation applied for $3 \times 3$ matrix with $2 \times 2$ filter. Figure 3.3 shows a practical example of how convolution can help to detect features. Example shows how a $3 \times 3$ edge detection filter is applied to a gray-scale image, and the output image shows the detected edges, i.e., points where the original image's color changes.

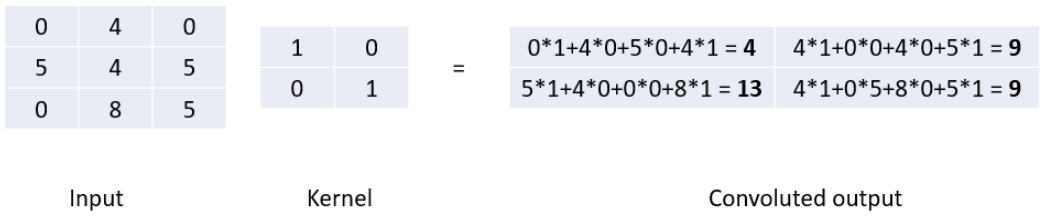



**Figure 3.2:** Example of convolution operation for $3 \times 3$ input data and $2 \times 2$ filter.

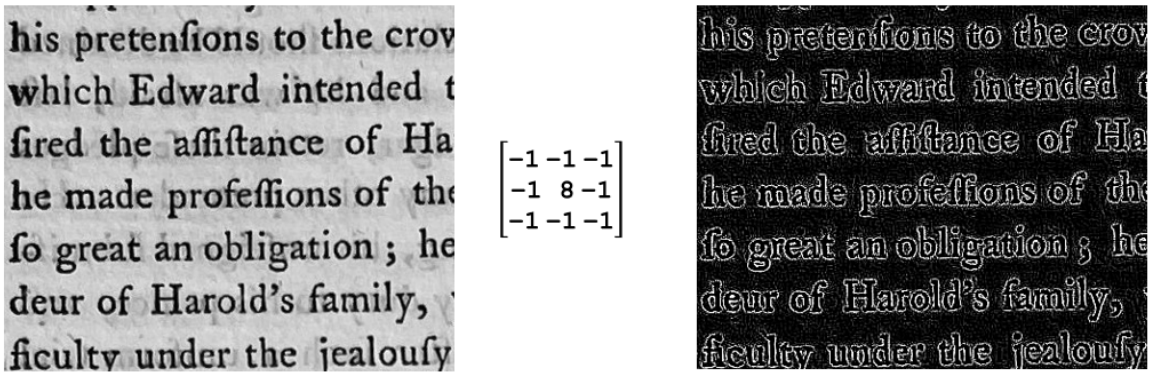



**Figure 3.3:** Example of applying edge detection filter for an image.

The role of the convolutional layer is to reduce the size of the image without losing critical features for a good prediction. The standard neural network assumes that every output unit interacts with every input unit. However, this is not a valid assumption for images as the interaction between closely located pixels is much more

probable than for pixels further away in the image. Furthermore, as the same filter is applied systematically across the whole image, it can detect features anywhere in the image.

Three parameters control the convolution operation, the filter size, the stride, and the padding. The filter is usually a tiny odd-sized (e.g., $3 \times 3$), and it has a nice feature where pixels from the input layer would move symmetrically around the output pixel on the output layer. The stride defines the movement over the image when the filter slides over all locations, starting from the top left corner and going to the bottom right corner. If stride is one, the movement is one pixel at a time. Finally, the padding controls the number of pixels added to an image when the convolution operation is done. The reason for padding is that the spatial dimensions of input decrease in the convolution operation, and adding extra pixels for the border enables keeping the image's dimensions the same. It enables a more accurate analysis of the image.

In CNN, convolutional layer filters are used to create feature maps to summarize the presence of those features. Several convolutional layers can be connected. This approach enables layers close to the input to learn low-level features (e.g., lines or edges) and layers deeper in the model to learn more abstract features (e.g., shapes or specific objects). However, one problem remains as the feature maps are sensitive to where the features are in the image. This problem can be addressed using down-sampling, where the presence of a feature in a particular area (i.e., patch) is summarized and used for the down-sampled feature map. Max pooling is a commonly used operation where the maximum value of the patch is for the down-sampled feature map. The patch size is often $2 \times 2$; in this case, the input size is reduced by 75%.

The spatial dimensions are reduced when the input images are passed through convolutional layers. This size reduction is not a problem for classification tasks, where the idea is to predict if a particular object is in the image or not. However, this is a problem for object detection as the spatial dimensions are required for correctly predicting the object's location. The problem can be addressed using the transposed convolution. It is a backward version of convolution where Transposed Convolution layers are used for upsampling the input feature map to create an output feature map where the spatial dimensions are greater. This operation is done by broadcasting via the filter.

### 3.2.2 Transfer Learning

Transfer Learning is a machine learning method where knowledge gained from one problem is applied to resolve a different but related problem. The technique is used

primarily in cases where there is insufficient training data. Tan et al. describe this especially from deep leaning's point of view [50]. They classify deep transfer learning into four categories: instance-based, mapping-based, network-based, and adversarial-based deep transfer learning. Instance-based deep transfer learning assumes that although the source and target domains are different, similar instances from the source domains can be used with appropriate weights, and other instances are excluded. In this way, a dataset of equal distribution is created, enabling more efficient model training for the target domain [50].

Mapping-based deep transfer learning maps instances from source and target domains into new data space, which is then used for training a new model. Transfer component analysis (TCA) has been widely used for machine learning and can also be extended for deep learning networks [50].

Network-based deep learning is a method where the model is trained in the source domain, and part of the network structure and parameters are transferred to the target domain [50]. Figure 3.4 shows an example of this method. First, the network is trained with a large dataset (e.g., Resnet) [13]. Then, these trained parameters are used for initializing the parameters for the new network. In most cases, only the parameters for feature extraction are used, and parameters for fully connected layers are ignored. In other words, the new model reuses the initial and middle layers of the pre-trained model, and only the final layers are trained for the new task. The effectiveness of transfer learning is based on the intuition that low-level spatial features are learned better if the image dataset is large enough.

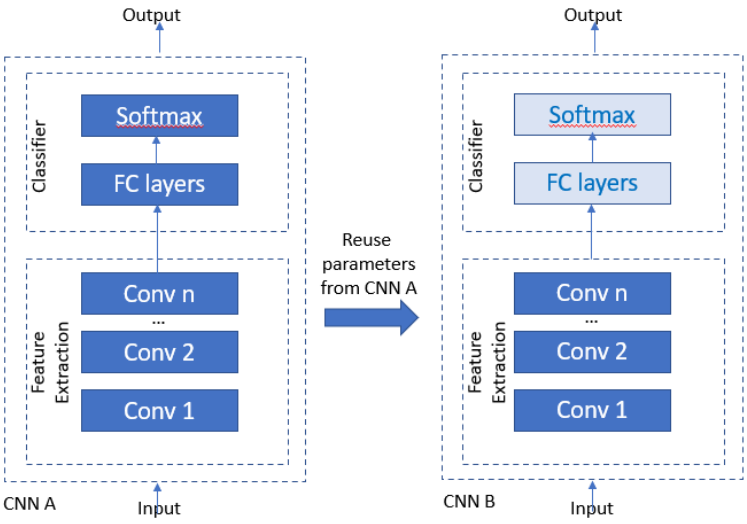



**Figure 3.4:** Principle of transfer learning.

Adversarial-based deep transfer learning uses technology developed for generative adversarial networks (GAN) [50]. The features are extracted from source and target

domains during training and fed to another adversarial layer. The adversarial layer tries to discriminate the origin of the features. If it performs poorly, features of these two different domains are very similar, and transferability is good. On the other hand, if it performs well, features are different, and transferability is poor.

### 3.2.3 Large datasets

Deep Learning networks require a lot of data to be trained, and creating a training data set is challenging. To address this, several research groups have created large publicly available datasets that could be used to resolve computer vision problems.

ImageNet [5] is an extensive database designed for object recognition research, and initially, it was used for image classification. It consists of more than 14 million images, and each image has been annotated to describe what objects are in the images, and there are more than 20 thousand image categories. Additionally, bounding boxes for objects are provided for more than one million images.

Common Objects in Context (COCO) dataset [21] is designed for object detection and segmentation. The dataset includes more than 330 thousand images, and about 200 thousand are labeled, covering 80 object categories and 1.5 million object instances.

PubLayNet dataset [55] is designed for document layout analysis. The dataset contains over 360 thousand images, including journal articles and preprints. The layouts are annotated with bounding boxes and polygonal segmentations. In addition, several CNN models trained for PubLayNet are publicly available for researchers.

### 3.2.4 Data augmentation

Data augmentation is a method where the number of images in the training dataset is artificially extended using the existing images [4]. The primary purpose of the operation is to decrease the overfitting and improve the generalization of the trained model.

Data augmentation for images can be split between two main categories: image manipulation and deep learning methods [4]. Image manipulation includes classic computer vision methods and operations like kernel filters to blur or sharpen the image, flipping or rotating images, adding random noise, and transforming images' color space. Most image manipulation methods are straightforward and do not require many computing resources. On the other hand, deep learning methods have become widespread recently when new deep learning techniques have developed. This category includes the generative adversarial network (GAN) method, where GAN creates artificial instances and retains similar characteristics to the original dataset's images.

## 3.3 Object recognition using CNNs

Usage of CNN for object recognition has become a standard as deep learning methods and algorithms are developed further [22]. As all object recognition methods are tightly related, the same techniques are used, and more advanced methods are built on top of the other methods.

### 3.3.1 Image Classification

The breakthrough for deep learning happened in 2012 when Krizhevsky et al. introduced Alexnet CNN [19]. Alexnet was originally built for the image classification task. Alexnet architecture is shown in figure 3.5. The architecture consists of five convolutional layers (e.g., $3 \times 3$ convolution and related activation functions) followed by pooling (e.g., max pooling). This part of the CNN is called feature extraction. Then, two fully connected layers follow feature extraction, and the final layer is a softmax layer to predict the class label for the image.

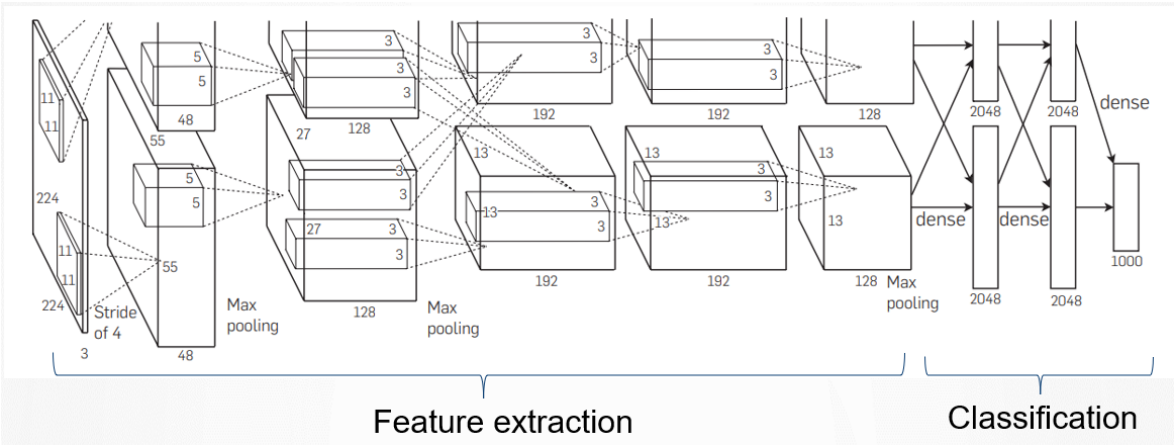



**Figure 3.5:** Alexnet architecture [19].

Alexnet was a very successful CNN, and it worked well on image classification tasks. Further development of deep learning methods led to a belief that really deep models would resolve complex computer vision problems. As a result, more than 100 layer models were tested to provide better accuracy. In theory, this is true, and deeper networks should be more accurate. However, experiments were not supporting this behavior; on the contrary, training accuracy dropped after a certain point. The difference between theory and results obtained from experiments was due to vanishing/exploding gradients and made the model unstable and unable to learn accurately [16].

He et al. introduced a deep residual learning framework (ResNet) to address this problem [16], and this was further extended (ResNeXt) [54]. Figure 3.6 shows how

ResNet implements this using skip (a.k.a. shortcut) connections. In skip connection, data bypasses one or more layers and is fed deeper into the network, enabling training in very deep networks. The author hypothesized that optimizing the residual mapping would be easier than optimizing the original stacked mapping. The hypothesis was proven by testing in reality, and results showed that accuracy improved, and it was possible to train deeper networks.

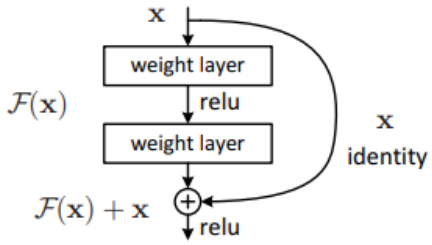



**Figure 3.6:** Residual connection[16].

For ResNet network architecture, the critical building block is called a residual block. Two different residual blocks are defined: a pair of $3 \times 3$ convolution layers, a skip connection, three convolutions layers ($1 \times 1$, $3 \times 3$, and $1 \times 1$ convolutions), and a skip connection. In the case of three-layer variant $1 \times 1$ convolutional layers are responsible for reducing/increasing dimensions.

ResNeXt architecture is very similar compared to Resnet [54] using shortcuts. Instead of sequential layers like in Resnet, ResNeXt uses parallel stacking layers. Figure 3.7 shows the man difference between ResNet and ResNeXt. The number of parallel blocks defines (i.e., cardinality) the size of the set of transformations. The architecture in the picture includes 32 blocks, and therefore the cardinality is 32. The tests demonstrated that the approach improves accuracy while reducing network complexity and the number of parameters.

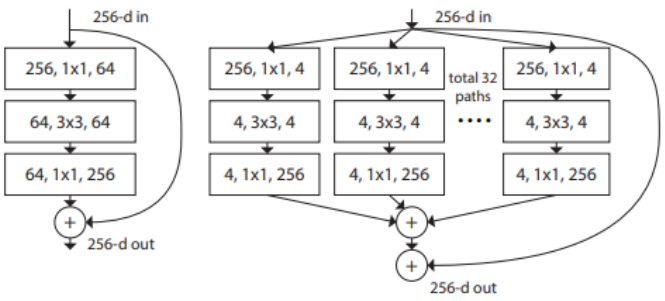



**Figure 3.7:** ResNet and ResNeXt building blocks [54].

Figure 3.8 shows architectures for Resnet-50 and ResNeXt-101 networks. As ResNet and ResNeXt are designed for classification tasks, these can be used as feature

extractor backbone networks for other object detection networks, e.g., image segmentation tasks.

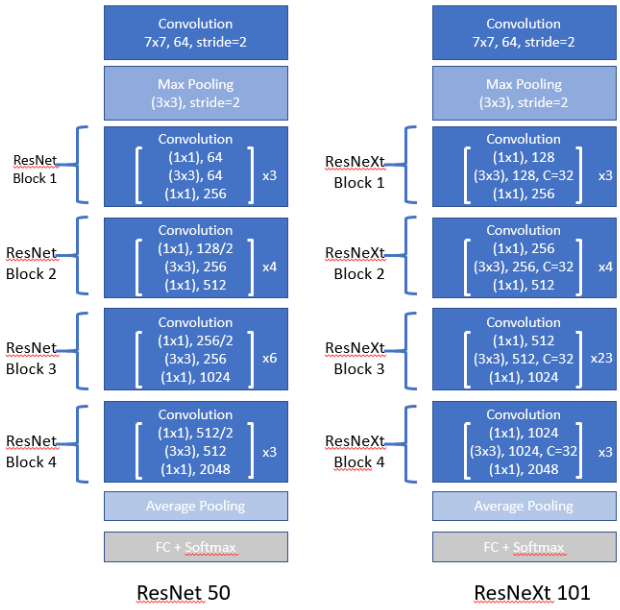



**Figure 3.8:** ResNet-50 and ResNeXt-101 architecture [54].

### 3.3.2 Semantic Segmentation

Long et al. introduced fully convolutional networks (FCN) for semantic segmentation [23]. FCN has only convolutional layers and pooling operations. FCN is a modification of existing classification networks where fully connected layers are removed, and the final layer is replaced with upsampling.

U-net is another semantic segmentation method [46], and it enables handling arbitrary large images. It was originally designed for biomedical segmentation problems. However, it can be applied to other domains also. The CNN model is called U-net as its U-shape consists of two paths: contracting and expanding. The contracting path includes traditional CNN to capture different high-resolution features from the image, and the symmetrical expanding path supplements the model enabling precise localization.

The contracting path starts with two $3 \times 3$ convolutions and continues with four groups, where each group consists of a max pool $(2 \times 2)$ operation and two $3 \times 3$ convolutions. The symmetrically expanding path is very similar; however, the max pool operations are replaced by $2 \times 2$ transpose convolutions, and each transpose convolution increases the resolution of the output.

High-level features from the contracting path are combined with the counterparts on expanding side . Therefore, the output includes more precise data from the original

image. Additionally, because of many feature channels, it is possible to propagate detailed information to a higher layer. Figure 3.9 describes the U-net architecture.

During the training, U-net uses a loss function that computes a pixel-wise softmax for the final feature map followed by the cross-entropy loss function, i.e., if the pixel deviates from the valid label in a specific position, then it is penalized.

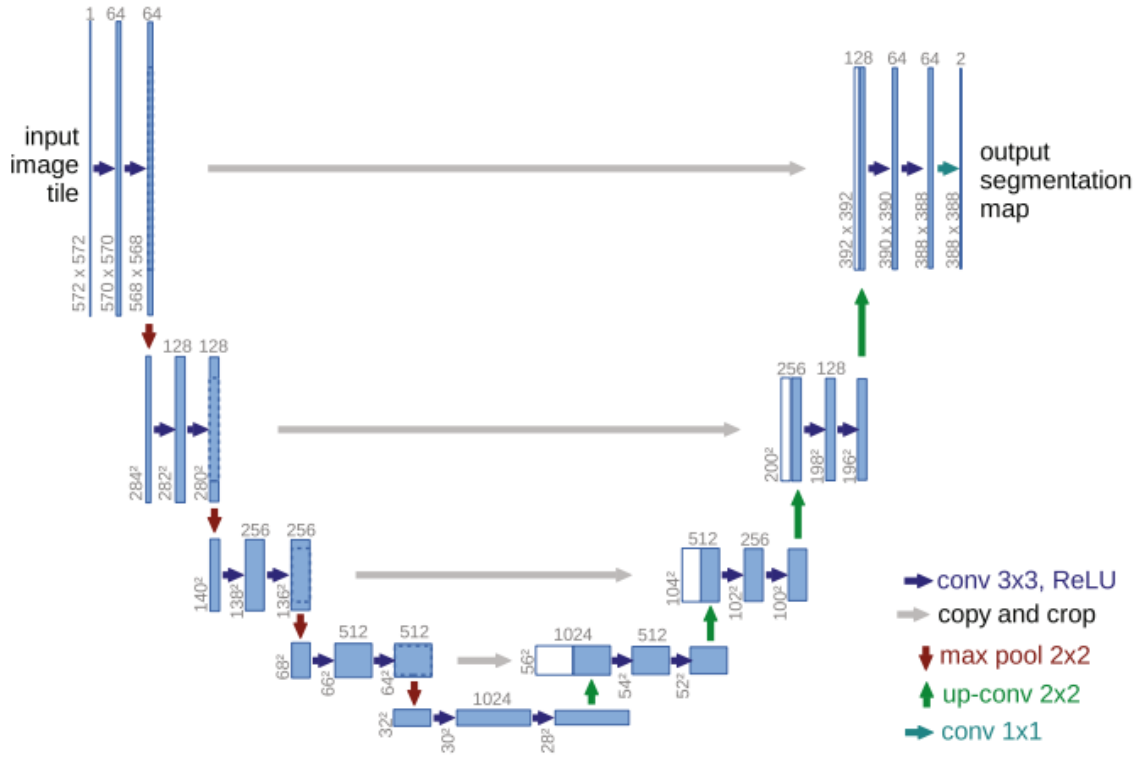



**Figure 3.9:** U-net architecture [46].

### 3.3.3 Object Detection

Region-based CNN (R-CNN) was developed for object detection in 2014 [11]. The basic functionality of R-CNN has three stages. During the first stage selected search is used to create a set of region proposals (ROI, region of interest) (about 2000 regions). Next, each ROI is warped to a fixed size ($224 \times 224$ pixels) and forwarded through CNN to create a feature vector. Finally, two outputs are based on the feature vector: support machine vector (SVM) classifier to predict object class or regressor for bounding box for the region. R-CNN can detect 80 classes.

The R-CNN model has several independent stages and cannot be trained end-to-end. It also uses selective search to detect region proposal candidates; it is inefficient as the search happens outside the model.

Due to these limitations, a revised version, Fast R-CNN, was developed [12]. Figure 3.10 describes how the Fast R-CNN works. Similarly, as for R-CNN, Fast R-CNN uses the same selected search algorithm to create a set of region proposals. The original image is forwarded through CNN to create a feature map of the image. The ROIs are mapped to the feature map, cropped, resized, and fed to the ROI Pooling layer. ROI Pooling layer splits each ROI into a grid of cells ($7 \times 7$), and max pooling is applied to create a feature vector where each grid cell is represented by one value. The size of region features is always the same even if input regions' size varies. Finally, the output of the ROI Pooling layer is forwarded to some fully connected (FC) layers. Two predictions are provided, one for the class label and one for the bounding box of the detected object.

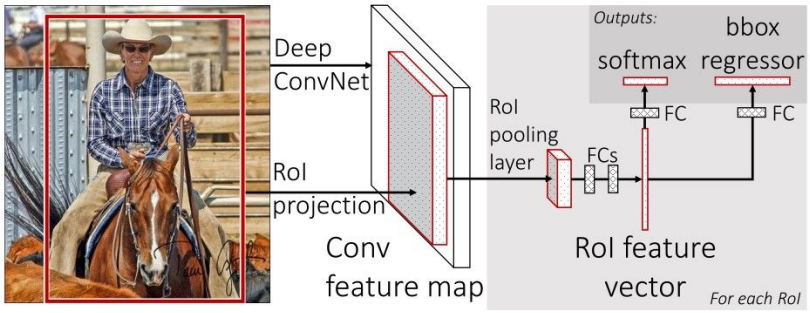



**Figure 3.10:** Fast R-CNN architecture [12].

Fast R-CNN is 25 times faster than R-CNN; it has only one stage and can share computation across region proposals. Also, accuracy is significantly improved. Although Fast R-CNN is much faster than R-CNN, its accuracy and speed depend on the selective search method and cannot be customized.

Limitations of Fast R-CNN implementation led to enhancing and creating a Faster R-CNN model where CNN is also used to create the region proposals [45]. Faster R-CNN introduces a new component, Region Proposal Network (RPN), used to generate region proposals. Otherwise, the Faster R-CNN follows the same logic as Fast R-CNN.

RPN implementation uses anchors. Anchor is a sliding window overall points of a feature map and predicts if it contains an object or not. In parallel, it also predicts bounding box corrections. The method uses K (K=9) different anchor boxes for each point where the size and scale vary (three scales and three sizes). The size of the feature map is $Weight \times Height$. Therefore, there are $W \times H \times K$ anchors. However, all anchors are not used; instead, the anchors are sorted based on the 'objectness ' score and only the top 300 anchors are selected. Non-max suppression (NMS) method is used to ignore overlapping proposals.

The Faster R-CNN model can be trained as one entity based on four losses (classification loss for RPN, bounding box regression loss for RPN, classification loss of the

whole model, and bounding box regression loss of the entire model).

### 3.3.4 Instance Segmentation

Mask R-CNN is an object recognition method capable of handling several recognition tasks in parallel [17]. It is an extension of Faster R-CNN, and its output includes labels (classifications), bounding boxes (object detection), and binary masks (instance segmentation).

Mask R-CNN consists of two stages; it uses Faster R-CNN for the first stage, including a backbone network to extract features and RPN to create a set of region proposals for each image. Instead of using ROI Pooling, it provides a more advanced ROI Align method with better accuracy [17].

The comparison between ROI Pooling and ROI Align is described in Figure 3.11. ROI Pooling method uses quantized coordinates for feature maps, and some information is lost as ROI Pooling uses integers as coordinates. In the example, the size of the original region in the image is $655 \times 655$ pixels, and in the feature map, the size of the region is $20 \times 20 (\lfloor (655/32) \rfloor = 20)$. The same kind of loss of information is repeated when another convolution operation is done for the pooling phase. ROI Align uses a Non-quantized method, which can preserve the information in the process. In the example, ROI Align uses floating points instead of integers as feature map coordinates. The pooling phase uses a bilinear interpolation technique to prevent information loss.

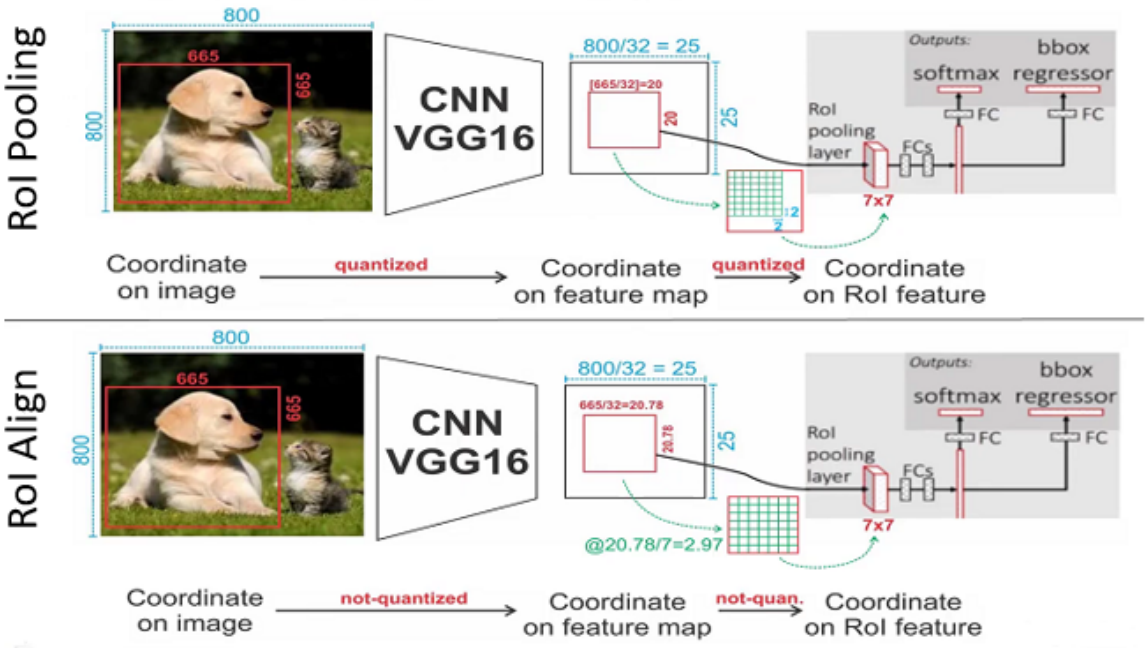



**Figure 3.11:** ROI Pooling vs. ROI Align.

The second stage is a slightly modified fully convolutional network (FCN), where the network predicts a binary mask for each ROI. The whole Mask R-CNN can be trained as one end-to-end model. The equation below defines the combined loss function for the end-to-end training:

$$L = L_{cls} + L_{bbox} + L_{mask}$$

The $L_{cls}$ defines the classification loss, i.e., how close the predictions are to the ground truth class, and $L_{bbox}$ defines the bounding box loss, i.e., how well the model predicts the localization of the bounding boxes. $L_{mask}$ is the loss for mask prediction and is calculated as a binary cross-entropy between the predicted mask and the ground truth mask.

Mask R-CNN provides a binary mask per class for each of the ROIs. As the class and mask predictions are made independently, the mask loss calculation for specific ROI is based only on the true class mask.

## 3.4 Metrics for object detection

As described earlier, object detection covers the classification and localization of the object. These two folded tasks affect how to measure the method's performance in question. For traditional classification tasks, the metrics evaluate the probability of the class object appearing in the image. It is straightforward to calculate the accuracy based on correct and incorrect predictions. However, a different metric is required for the localization tasks where associated bounding boxes are presented. Three commonly used methods are used for measuring the performance of object detection:

- Accuracy, Precision, and Recall,
- Intersection over Union (IoU), and
- Average Precision (AP) and mean Average Precision (mAP).

Average, Precision and Recall are often used to measure the accuracy of machine learning algorithms. The components of these measurements are:

- True Positive (TP): the number of correct detections
- False Positive (FP): the number of incorrect detections
- False Negative (FN): the number of missed ground-truth objects, i.e., not detected
- True Negative (TN): the number of correctly predicted negative classes. Not used for object detection as TN would be all possible bounding boxes that were correctly not detected.

Accuracy is calculated as the ratio between the number of correct predictions to the total number of predictions. Precision measures the fraction of correctly detected instances among all detected instances and can be seen as a measure of quality. The recall is the fraction of correctly detected instances among retrieved relevant instances and is seen as a measure of quantity. Both precision and recall are therefore based on relevance.

As the True Negatives are not used for object detection, the definitions are slightly changed and can be presented as:

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

$$precision = \frac{TP}{TP + FP} = \frac{TP}{all\,detections}$$

$$recall = \frac{TP}{TP + FN} = \frac{TP}{all\,ground\,truths}$$

In layout analysis, precision and recall are not meaningful as standalone metrics and should be used together. For example, the recall value of one can be achieved by selecting every item. Also, good precision can be reached by picking only a few highly likely items. One approach is to use the F1-score, which combines precision and recall by calculating the harmonic mean of the values. As both are weighed equal, F1 gives a good indication of the accuracy.

$$F1 = 2 * \frac{precision * recall}{precision + recall}$$

Additional measures are required because the precision and recall of detected objects do not tell anything about the localization's accuracy. For this purpose, Intersection over Union (IoU) can be used, and it is a standard metric for image segmentation tasks. IoU measures overlap between two bounding boxes or masks.

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

where A is the predicted bounding box and B is the ground truth bounding box. If the prediction is completely correct, IoU = 1, and low IoU means poor prediction. Typically IoU value of 0.9 is considered almost perfect.

IoU metric is sensitive to the object's size, which is illustrated in figure 3.12. It shows how the IoU value changes when the size of the object increases and the margin for the predicted bounding box remains the same. In the example, the size of the first object is $100 \times 100$ pixels, and the predicted object is 10 pixels larger to every direction (i.e., the size of the predicted box is $120 \times 120$ pixels). In this case, the IoU value is 69.4%, and increases to 92.5% when object size increases to $500 \times 500$. In other words,

the IoU is better for larger objects as the relative error decreases. This is clearly visible in layout analysis, where it is easier to achieve good IoU for larger objects (e.g., body text) than for smaller objects (e.g., page numbers).

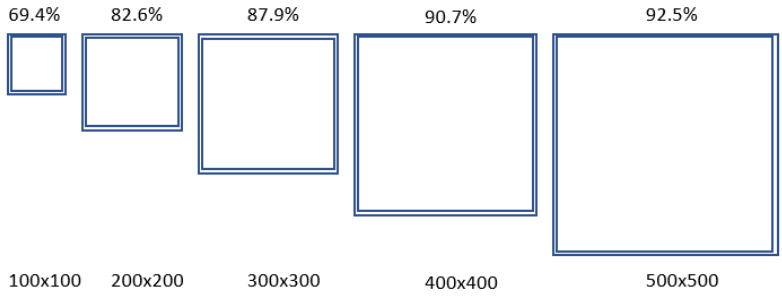



**Figure 3.12:** Impact of margin size for IoU metric.

IoU is a good measure for calculating the accuracy between different bounding boxes. Therefore it works well for layout analysis, where bounding boxes are a natural way to represent text regions that are often rectangles and show an approximation of the area used for the text regions. All image segmentation also provides binary mask predictions; however, these were seen as too detailed, and, in many cases, some critical information would be easily lost.

As described earlier, object detection model prediction includes objects and bounding boxes. Additionally, prediction includes a confidence score indicating the prediction's confidence. Finally, a threshold is a parameter that could be used to control the predictions. On the one hand, if the threshold is high more objects are missed leading to low recall and high precision, and on the other hand, if the threshold is low, more false positives are detected, leading to low precision and high recall. Therefore, precision and recall should stay high for a good model even if the confidence score varies.

Average Precision (AP) and Average Recall (AR) are defined as weighted means of precision and recall at each threshold. AP and AR are calculated for individual objects. Mean Average Precision (mAP) is the average AP of each class. Table 3.1 gives an example of how the AP is used for COCO detection evaluation metrics [3]. The most important metric is the Average Precision as it is calculated over all IoU thresholds. The COCO metric distinguishes different size categories (e.g., small, medium, large), and AP and AR are calculated separately for all three categories.

Although Average Precision and Average Recall are commonly used metrics for image segmentation, the experiments with layout analysis indicate these are too complex metrics and are not practical. One reason is that the number of different classes and calculating a mean over different objects is not meaningful. Another reason is the

**Table 3.1:** COCO metric.

| | |
|---|---|
| Average Precision (AP): | |
| AP | % AP at IoU=.50:.05:.95 Average precision over IoU tresholds |
| APIoU=.50 | % AP at IoU=.50 (PASCAL VOC metric) |
| APIoU=.75 | % AP at IoU=.75 (strict metric) |
| AP Across Scales: | |
| APsmall | % AP for small objects: $area < 32^2$ |
| APmedium | % AP for medium objects: $32^2 < area < 96^2$ |
| APlarge | % AP for large objects: $area > 96^2$ |
| Average Recall (AR): | |
| ARmax=1 | % AR given 1 detection per image |
| ARmax=10 | % AR given 10 detections per image |
| ARmax=100 | % AR given 100 detections per image |
| AR Across Scales: | |
| ARsmall | % AR for small objects: $area < 32^2$ |
| ARmedium | % AR for medium objects: $32^2 < area < 96^2$ |
| ARlarge | % AR for large objects: $area > 96^2$ |

fixed size of objects: COCO metric defines three size categories (i.e., small, medium, and large) where the sizes are fixed. However, experiments indicate that almost all objects were in the large size category, and the size categorization did not reveal any additional insight.

Based on the observations above, the comparison between different approaches is using a combined metric: precision, recall, and F1 metrics for classification and per class IoU for overall accuracy.

# 4. Experiments

As mentioned earlier, document layout analysis uses image segmentation techniques. Semantic segmentation classifies every pixel into a class according to its context (i.e., page element). On the other hand, instance segmentation classifies every pixel into a class, and each pixel is assigned to an instance of a page element. For example, different instances of marginalia are separated in this way. All research articles identified during the thesis have been using coarse-grained layout analysis; however, this thesis uses a fine-grained layout analysis. It would provide valuable information if this approach is feasible in reality.

Most research articles focus on semantic segmentation, although instance segmentation provides more detailed information than semantic segmentation as it enables a separation between different instances of similar objects. A semantic segmentation method, P2PaLA, is presented in more detail [44]. An instance segmentation method is based on the Mask R-CNN technique and implemented as part of the thesis work using PyTorch and Torchvision libraries. Both methods use the same training data, and the results are directly comparable.

## 4.1 Data and Data pre-processing

As described in the introduction, this thesis focuses on hand-printed materials from the $18^{th}$ century and specifically on David Hume's The History of England editions. It consists of six volumes (Table 4.1) and was published in 1754, 1756, 1759, and 1762. There are several editions of the History of England as Hume continued to change the text. The final version was published after his death in 1778. Detailed layout analysis is required to detect textual changes between different editions.

Digitized versions of books printed in 1754, 1759, 1762, and 1778 are available through cooperation in Critical Edition of David Hume's History of England project, and selected pages are used for this work.

Currently, the digitized dataset for Hume's books covers roughly 5000 color images. Most of the images are from McGill University Library's collection [24]. The rest are photographed during winter 2022 in the National Library of Scotland using an

**Table 4.1:** Volumes of David Hume's History of England.

| Volume(s) | Title | Publishing year |
|---|---|---|
| 1-2 | The history of England from the invasion of Julius Caesar to the accession of Henry VII. | 1762 |
| 3-4 | The history of England under the House of Tudor | 1759 |
| 5 | The history of Great Britain, containing the reigns of James I and Charles I. | 1754 |
| 6 | The history of Great Britain, containing the Commonwealth, and the reigns of Charles II and James II | 1757 |

**Table 4.2:** Training dataset distribution for different editions.

| Edition | Pages |
|---|---|
| 1754 | 162 |
| 1759 | 170 |
| 1762 | 175 |
| 1778 | 155 |

iPhone. The images cover 1754, 1759, 1762, and 1778 editions. For training purposes, 702 images were annotated manually using Transkribus tools. Annotated images were split between training, evaluation, and testing datasets of 602, 60, and 40 images. Table 4.2 describes the split between different editions for the training and evaluation dataset (90/10 split between training and evaluation datasets). The testing dataset was constructed separately, and for each edition, ten images were selected representing typical content pages.

Pages follow the basic layout structure of the hand-press era. Pages for all editions have headlines where left-hand pages have the running title of the book, and right-hand pages have the running title of the chapter. All editions have a smaller font size for footnote and marginalia than for the rest of the text. Drop-capital is used for all editions. The first word of each paragraph is written in capital. Also, the direction line is always used.

Differences between editions are minor; however, some differences may complicate the implementation of deep learning models for layout detection. Volume number is included as part of the direction lines for 1754, 1762, and 1778 editions, but for the 1759 edition, it is not used. The location of the direction line varies; its location for 1754, 1762, and 1778 is always the lowest line. However, for the 1759 edition, its location is between text and footnote. For the 1778 edition, the page number for the chapter title page is in the center of the headline. In other cases, it is at the headline's beginning (left-hand page) or the end (right-hand side). The spacing between different elements

**Table 4.3:** Occurrence of page elements in the training dataset.

| Page element | 1754 | 1759 | 1762 | 1778 |
|---|---|---|---|---|
| text body | 159 | 170 | 182 | 168 |
| header | 159 | 160 | 143 | 137 |
| heading | 44 | 129 | 327 | 137 |
| footnote | 44 | 113 | 120 | 97 |
| marginalia | 133 | 183 | 140 | 105 |
| marginalia-top | 129 | 120 | 150 | 135 |
| page number | 161 | 157 | 142 | 139 |
| catchword | 167 | 174 | 175 | 155 |
| signature mark | 58 | 90 | 112 | 76 |
| toc-entry | 22 | 38 | 82 | 17 |
| separator | 0 | 0 | 29 | 17 |

also varies. For example, the spacing between paragraphs is remarkably constant for the 1778 edition when there is more variation with earlier editions. The space between marginalia and the rest of the text is very narrow for 1778 editions, and there is more space for other editions.

Data was also explored to analyze how balanced the training set is. Table 4.3 shows the instances of different elements in the training set. As it can be seen, the balance is good for some elements (e.g., text body, headers, marginalia, marginalia-tops, page numbers, and catchwords). In some cases, the balance is not perfect (e.g., footnotes, headings); here, the reason is that those earlier editions had fewer instances of these elements. The overall number of instances is small for some elements, and detection of these elements is expected to be challenging. However, these are irrelevant from the OCR process point of view (e.g., separators, decorations, and figures). In addition, the dataset includes some unknown elements due to the elements identified later during the annotation phase and was not included in labeling.

Figure 4.1 shows a distribution of image size, which confirms that the 1778 edition uses a smaller page size, although the number of lines per page is approximately the same. The width of the pages is smaller than in earlier editions, which is why there is less space between the text and margins.

## 4.2 P2PaLA - Semantic Segmentation method

Page to PAGE Layout Analysis (P2PaLA) is an implementation of Document Layout Analysis software [44]. P2PaLA covers text line segmentation, page segmentation, and region classification functionality.

P2PaLA architecture follows generative adversarial networks (GAN) architecture

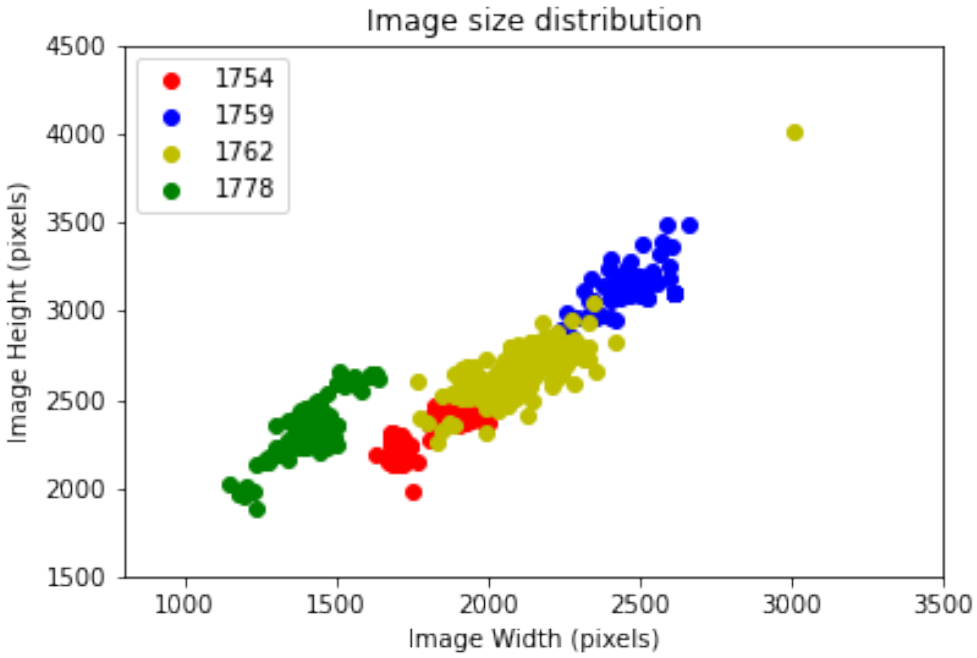



**Figure 4.1:** Training dataset image size distribution.

with two components, A-net and M-net. However, the architecture performs discriminative rather than generative processing, as commonly in GAN. A-net is a simple six-layer CNN, and it is trained to distinguish between ground truth labels and predicted labels. On the other hand, M-net follows the typical U-net architecture as presented in Chapter 3.

Loss of A-net is a classical cross entropy loss and is defined as

$$\mathcal{L}_A(X,Y) = \frac{1}{2}\{\mathcal{L}_A^1(X,Y) + \mathcal{L}_A^0(X,Y)\}$$

where $\mathcal{L}_A^1(X,Y)$ is calculated for labels generated by M-net and $\mathcal{L}_A^0(X,Y)$ for ground truth labels. Loss of M-net is a combination of two lost functions:

$$\mathcal{L}_M(X,Y) = \frac{-1}{N}\sum_{n=1}^{N}\mathcal{L}(x_n,y_n) + \lambda\mathcal{L}_A^0(x_n,y_n)$$

where $\mathcal{L}(x_n,y_n)$ is cross entropy loss to learn from the training data and $\mathcal{L}_A^0()$ tries to mislead A-net.

P2PaLA is available in Transkribus as an optional module, and all tests were done using the same training and test data for other experiments. Figure 4.2 shows two examples of P2PaLa predictions. In the figure, all page elements are shown; for binary masks, randomly selected colors are used to show the area of the mask. The same illustration principle is used throughout the whole document.

The prediction accuracy for the page on the left is almost perfect: IoU for the

text region is almost 100% (98.35%) and for other elements also very good: header 92.63%, marginalia-top 88.47%, and page number 93.03%. All elements are correctly detected. Although all elements on the right-hand page are correctly detected, the overall accuracy is worse. Text body 95.45%, header 91.02%, marginalia-top 87.65%, and page number 86.15% are correctly classified. However, a footnote is detected only partially, and part of it has been combined with the catchword.

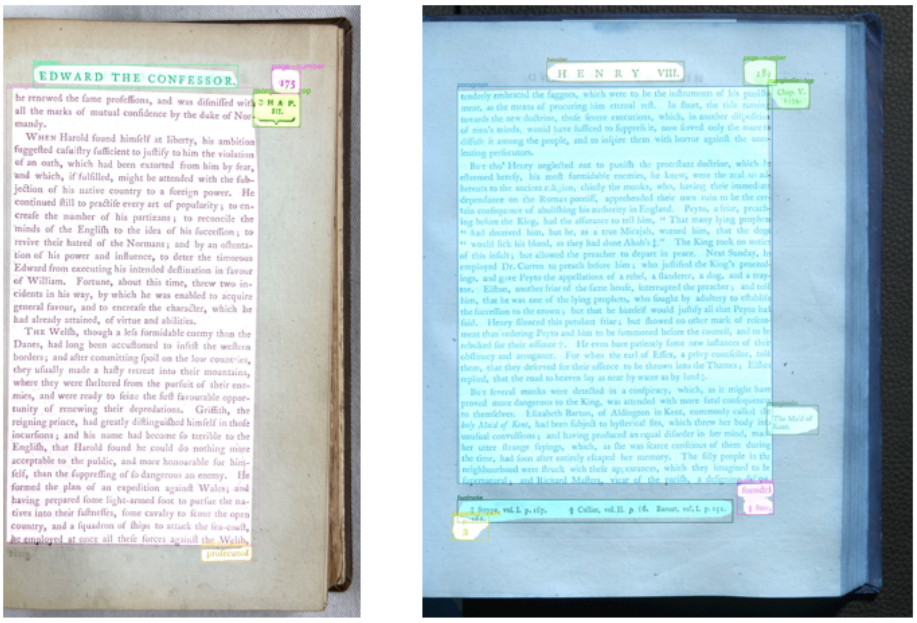



**Figure 4.2:** Examples of P2PaLA predictions.

Overall, P2PaLA detects objects accurately, and only very few objects are mislabeled. Typical errors seen with testing dataset are:

- Marginalia-top is sometimes only partially detected or combined with text body (Figure 4.3).
- Phantom object (labeled as decoration) is sometimes detected close to header (Figure 4.3).
- Footnote is sometimes broken, and part of it is labeled as a catchword or border between footnote and text body is misplaced.

## 4.3 Mask R-CNN Implementation

The hypothesis is that instance segmentation would provide more accurate predictions than semantic segmentation. This hypothesis was tested by implementing

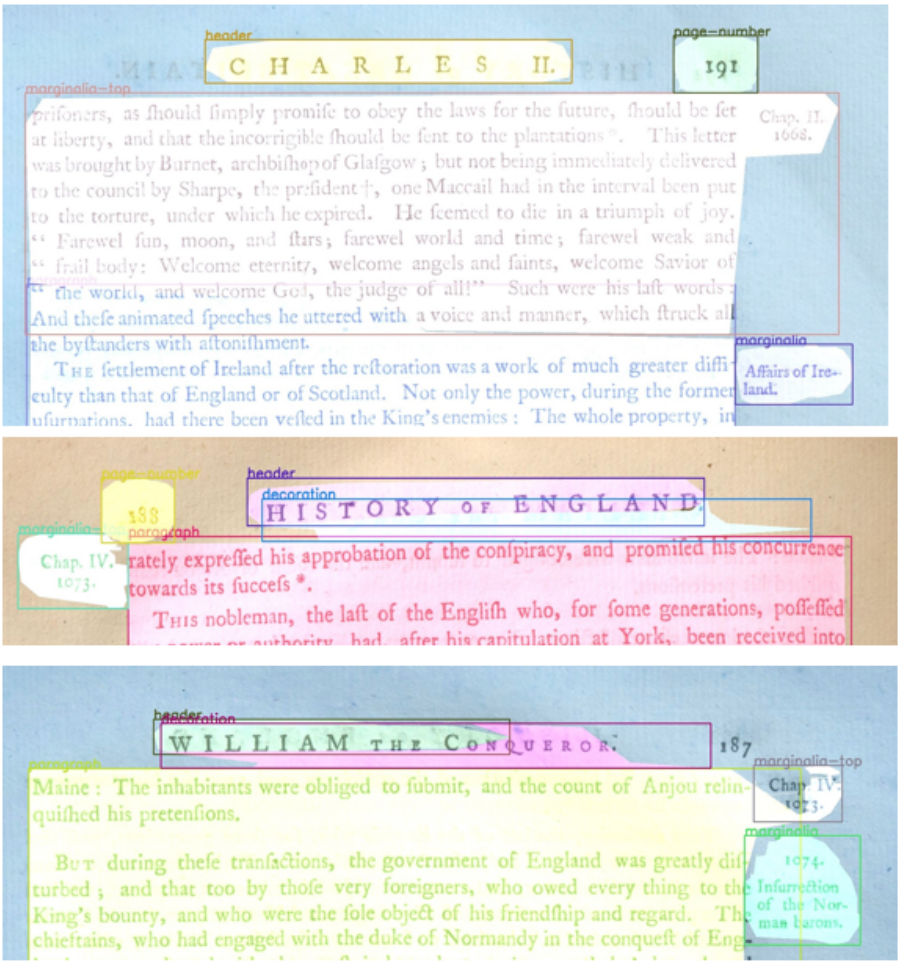



**Figure 4.3:** Examples of P2PaLA issues with marginalia-top.

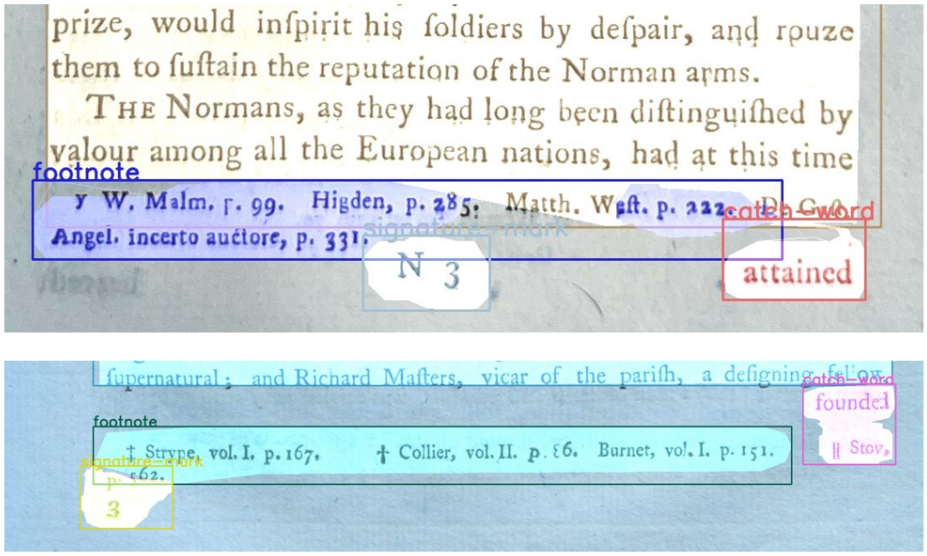



**Figure 4.4:** Examples of P2PaLA issues with a footnote.

training and prediction modules using the Mask R-CNN approach. The experimentation setup is described in figure 4.5. It consists of two separate phases: training and prediction. During the training phase, a Mask R-CNN model is created to detect different page elements on an image. During the prediction phase, the instance segmentation is applied for new unseen images, and PAGE XML files are created that could be used in the later phases of the OCR pipeline. Approach uses Torchvision-library [42]. The output of PAGE XML is uploaded to the Transkribus system and tested to see if it can be used for further OCR processing and if it complies with PAGE XML definitions. All programming is done

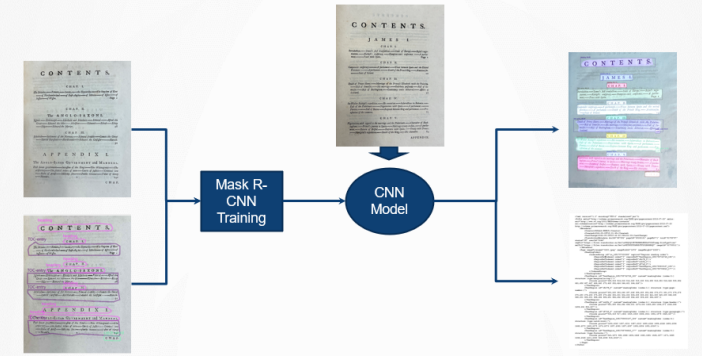



**Figure 4.5:** Test setup.

using Python and PyTorch. PyTorch is an open-source Deep Learning framework developed by Meta AI and commonly used in Deep Learning projects [40]. PyTorch uses tensors (multidimensional rectangular arrays of numbers ) for storing and manipulating data. PyTorch includes several modules supporting different deep learning tasks, e.g., optim implementing different optimization algorithms, utils.data including different functions for data loading and nn building different neural networks. PyTorch supports using graphical processing units (GPU) to accelerate computing.

Torchvision is an additional library built on top of PyTorch to support Deep Learning tasks for Computer Vision [42]. It consists of datasets, model architectures, and typical image transformation functions. Additionally, it includes an implementation of the Mask R-CNN algorithm.

Additionally, another test was done using Detectron2 -library to examine the impact of different feature extraction backbone networks for the accuracy of the approach. Detectron2 is a library specially built for object detection tasks and developed by Meta AI [6]. It is also built on top of PyTorch. The abstraction level of Detectron2 is higher than Torchvision, and it is easy to implement complex object detection models. Detectron2 includes implementations of state-of-the-

art object detection algorithms, including Mask R-CNN. Detectron2 is highly configurable, and usage of transfer learning is easy, i.e., changing the feature extraction backbone network takes only a few lines of code.

The annotated images and PAGE XML were downloaded from Transkribus, and special conversion programs were created to transform them to a format required by Torchvision [42]. The code snippet below describes the simplified example of the input format for the solution. Torchvision-based Mask R-CNN implementation expects to receive images, bounding boxes, masks, and labels.

```
<Image in PIL format>,
{'boxes': list of box coordinates
          [x1, y1, x2, y2] for each label,
 'labels': list of labels,
 'masks': list of binary masks for each label,
 'image_id': unique id for image,
 'area': list of size of areas for each masks,
 'iscrowd': list of 0 # not used,
```

For the output, the data is very similar; the model provides predictions and includes binary masks and bounding boxes for each predicted label. The output also includes a confidence score for predictions. Again, a threshold should be set, and only those predictions where the confidence score threshold is exceeded should be selected for further processing.

```
[{'boxes': list of bounding boxes,
'labels': list of predicted labels,
'scores': list of confidence scores
          for each predicted label,
'masks': list of binary masks for each
         predicted label]
```

### 4.3.1 Model architectures

The high-level architecture of Torchvision Mask R-CNN implementation is described in figure 4.6. Backbone network act as the feature extractor of Mask R-CNN. The selected backbone for Torchvision is ResNet-50. Region Proposal Network (RPN) scans the feature map created by the backbone and creates a proposal of regions of interest (ROI), i.e., regions that may have objects. ROIAlign

splits each ROI into a grid of cells and creates a feature map for each ROI. These ROIs are then processed by two parallel heads, the object detection head and the mask generation head.

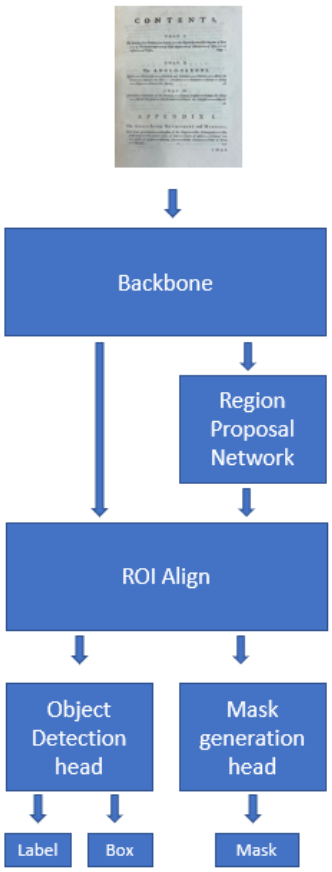



**Figure 4.6:** Architecture building blocks of model implementation

### 4.3.2 Training of models

Implementation of Torchvision is done according to the standard PyTorch training approach, and the simplified steps are described in Algorithm 1. Training of model was done in CSC - IT Center for Science 's Puhti system using Graphical Processing Units (GPU).

As described earlier, Mask R-CNN provides the prediction for objects and scores indicating how confident the model is and how realistic the prediction is. An F1 score was calculated to identify an optimal threshold for different object score values. Figure 4.7 shows that the optimal object score value is close to 0.5, i.e., predictions where the score is less than 0.5 are automatically rejected.

**Algorithm 1** Pseudocode for training of Torchvision model

```
num_classes = len(OCRclasses) + 1
full_dataset = read pre − processed data
train_size = int(TRAIN_RATIO ∗ len(full_dataset))
test_size = len(full_dataset) − train_size
dataset_train, dataset_test = random_split(full_dataset, [train_size, test_size])
data_loader = create DataLoader(dataset_train, batch_size = 2)
data_loader_test = create DataLoader(dataset_test, batch_size = 1)
model = get_model_instance_segmentation(num_classes)
params = [p for p in model.parameters() if p.requires_grad]
optimizer = create SGD optimizer
lr_scheduler = adjust lr_scheduler
for epoch in range(num_epochs) do
    train_one_epoch(model, optimizer, data_loader, device, epoch)
    lr_scheduler.step()
    evaluate(model, data_loader_test)
end for
savemodel(model, save_path)
```

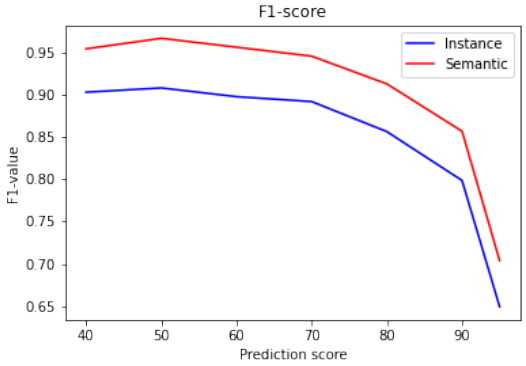



**Figure 4.7:** Training loss of a final model.

Mask R-CNN uses three different losses (e.g., $L_{cls}$,$L_{bbox}$, $L_{mask}$) and additionally it reports RPN losses. The loss development during training can be seen in figure 4.7. The figure demonstrates that the loss plateau was reached relatively quickly, training can be stopped, and the model can be saved. Therefore, it was concluded that the optimal training time was ten epochs for the Torchvision implementation. The training time for the Torchvision model was roughly 84 minutes in CSC 's Puhti GPU environment.

The standard rule of thumb for Deep Learning training is that more data is better; the impact of training dataset size was also analyzed. The test was conducted by running training for randomly selected images where the number of images varies from 100 to 600 images. Figure 4.8 shows the impact on mAP and mAR, and as can be seen, the graphs show there is still potential to improve the accuracy by

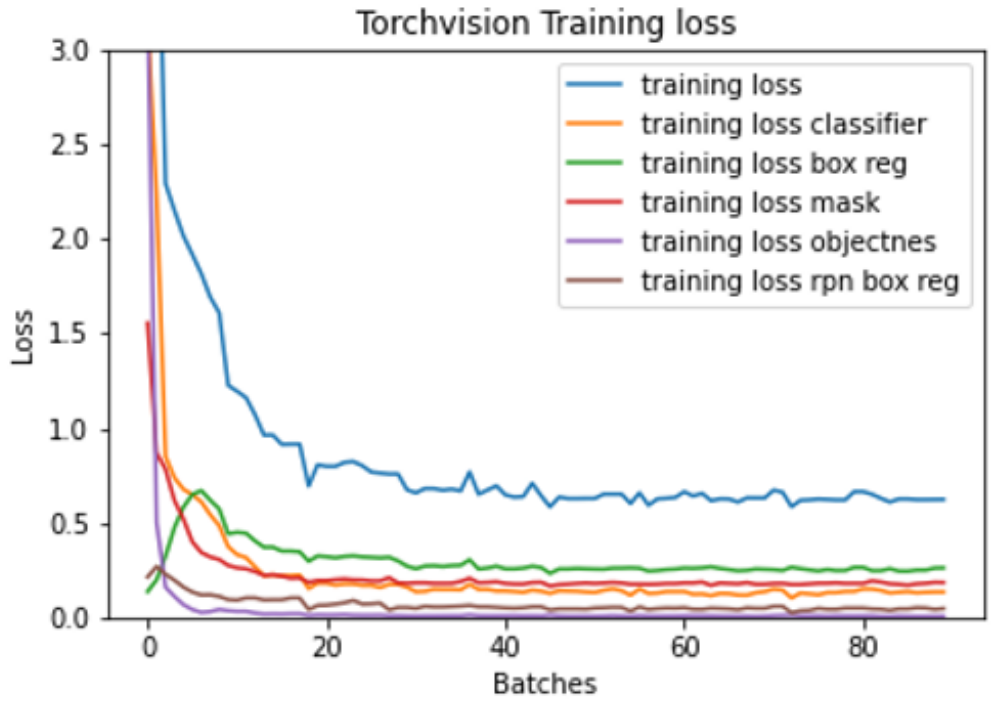



**Figure 4.8:** Training loss of a final model.

increasing the training size. However, this was left for further study due to the required work. In addition, to make the model more generic, it might be a good idea to extend the training dataset size and consider how to ensure the different types of pages are included.

Data augmentation was considered at the beginning to extend the existing training dataset. However, the idea was dropped quite soon as it was not seen as an appropriate method for page layout problems. For example, horizontal or vertical flipping does not work as the page structure is not symmetrical, color changes are not applicable, as in many cases, pages are binarized, and color information is lost.

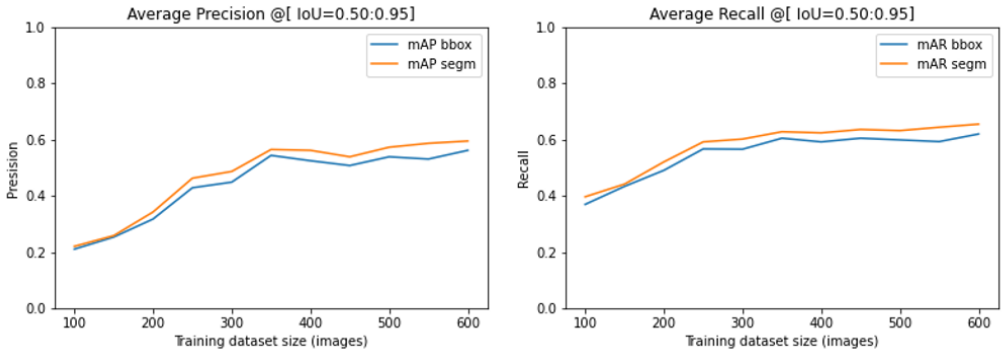



**Figure 4.9:** mAP and mAR by training dataset size

No hyper-parameter tuning was done for the models, and mainly the default parameters were used. For all implementations, stochastic gradient descent (SGD) optimizer with a learning rate value of 0.005 and momentum of 0.9 was used.

## 4.4 Predictions and test results

Analysis of models is based on visual inspections and accuracy metrics. In visual inspection for each tested page image, a particular image was created, showing the predicted bounding boxes and binary masks on top of the original image. Visual inspection helps to detect typical detection problems.

Accuracy metrics cover two different measurements: accuracy, precision, recall & F1 - and IoU metrics. IoU measurements were calculated for each class and for the whole test dataset. The mean Average Precision/Recall was calculated only for the Torchvision model.

P2PaLA results were used for the analysis to understand how well the models are performing against an existing approach. As described earlier, P2PaLA is a semantic segmentation method, and to make results comparable, both semantic and instance segmentation results were calculated for the Torchvision model.

### 4.4.1 Visual inspection

Figure 4.10 gives an example of prediction results with Ground Truth for Torchvision and P2PaLA. As can be seen, both results are good and very close to each other. All elements are detected correctly, and masks and bounding boxes are approximately the correct sizes.

The visual inspection was done for all images in the test dataset, and the following observations were made:

- The most frequent problem with the Torchvision Mask R-CNN model is that the mask for the text region is too small, i.e., part of the bottom or top part of the text region is not included in the mask (figure 4.10).
- the Torchvision model sometimes detects two overlapping text regions, although, in reality, these are only one region. In some cases, the other text region is labeled as a footnote. This seems to happen quite often if there is much space between paragraphs on the page (figure 8.4). Another example of difficulties separating text regions and footnotes is presented in figure 8.4, where the different regions overlap.

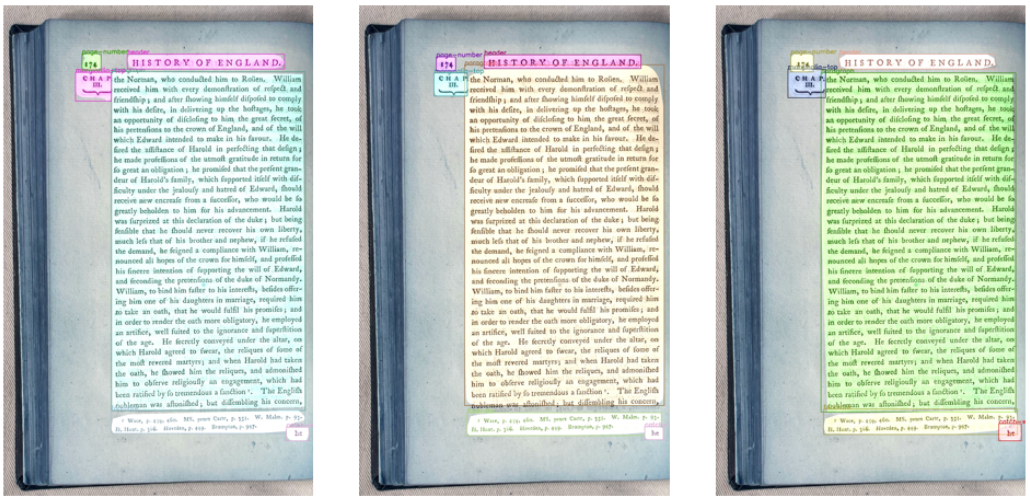



**Figure 4.10:** Model prediction results compared to the ground truth. From left to right: GT, Torchvision Mask R-CNN, and P2PaLA.

– Small elements are detected quite well. However, in some cases, these are detected more than once (figure 4.11) and labeled as different elements (marginalia vs. marginalia-top) or as the same element (e.g., signature mark) and should be removed as part of the post-processing phase.

– Footnote is a challenging element as, in some cases, the space between different parts of the footnote can be detected as separate or overlapping elements (figure 4.11).

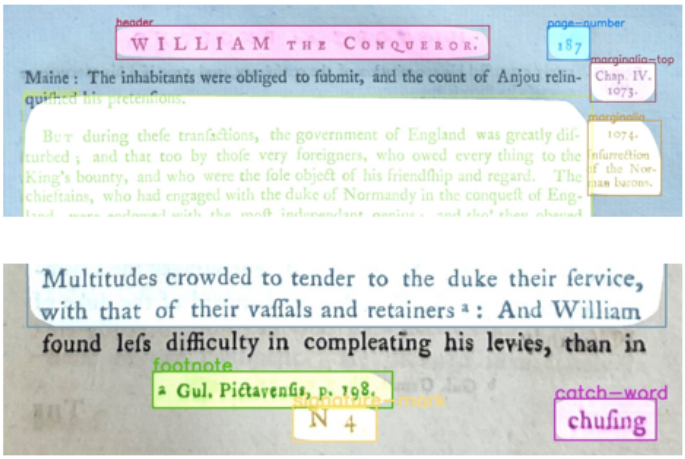



**Figure 4.11:** Mask does not cover the text region fully.

The case when mask does not cover the text region entirely is a severe problem. This can lead to a situation where some critical information is lost and unavailable for further steps in the process. Partially this can be avoided using

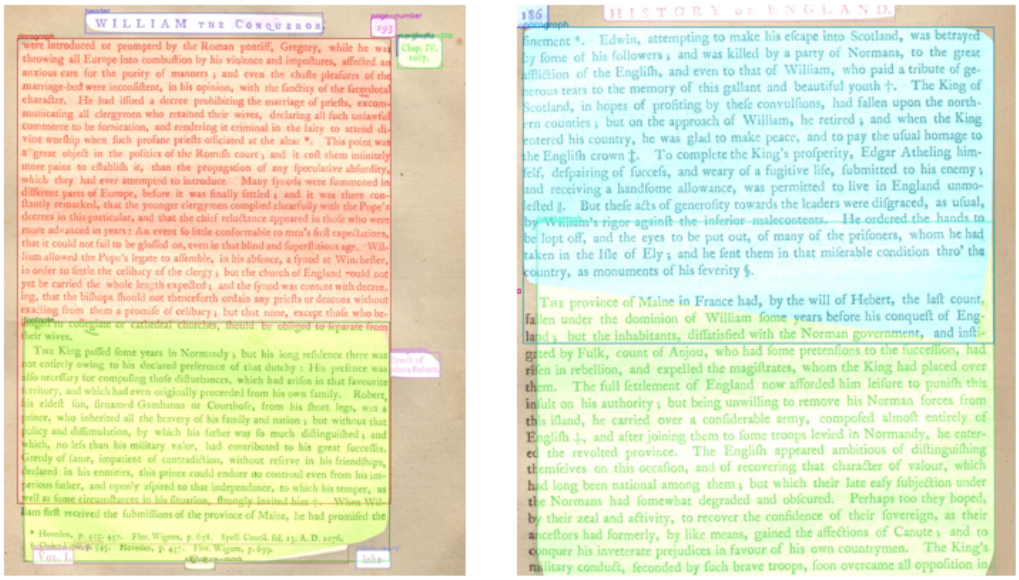


**Figure 4.12:** Large overlapping regions.

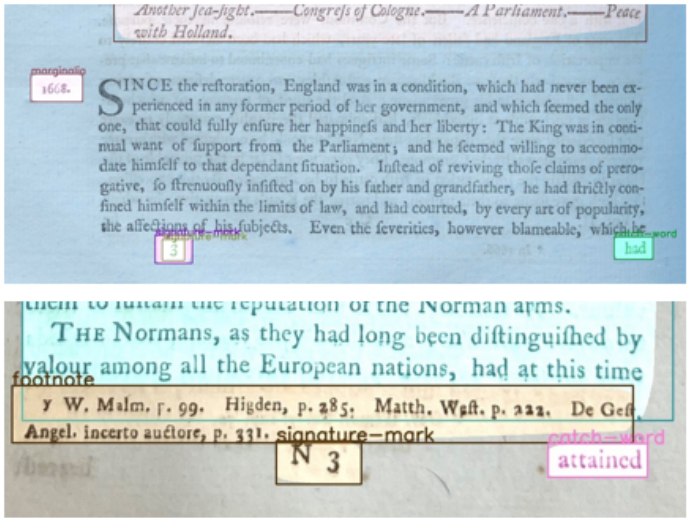



**Figure 4.13:** Overlapping footnote and text regions.

bounding boxes rather than exact masks. As the data is text, these regions are typically rectangles and, with bounding boxes, can better estimate the actual region than the predicted masks. However, this does not work in all cases, so another approach is required to resolve this issue.

Handling overlapping regions is much easier and can be recovered using the Non-Max Suppression (NMS) method. In this method, the highest-scoring bounding box is first selected, and then lower-scoring boxes with $IoU > threshold$ (e.g., 0.7). The steps are repeated if any boxes remain. This method can be applied as a post-processing task, and overlapping regions can be removed. The same

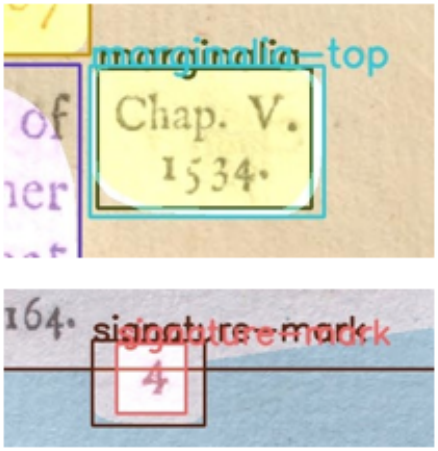



**Figure 4.14:** Same region is detected twice.

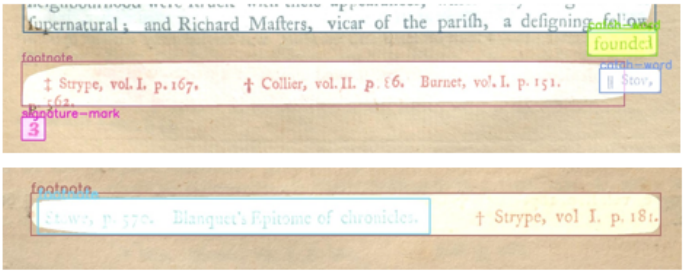



**Figure 4.15:** Split or overlapping footnote.

mechanism can be applied for regions detected twice.

### 4.4.2 Prediction accuracy

As mentioned earlier, the accuracy is measured using two metrics: accuracy, precision, recall & F1 - and IoU metrics. The first set of metrics measures how well the model can predict the label of objects on a page. However, it evaluates only the classification part of the model as it does not indicate how accurately the masks or bounding boxes are attached to the label. For this purpose, the IoU is a better metric.

Table 4.4 shows the accuracy, precision, recall & F1 -values for semantic segmentation, and table 4.5 for instance segmentation. P2PaLA results are only available for semantic segmentation. The results show that P2PaLA performs slightly better than Torchvision on the test dataset in the case of semantic segmentation and predicts the correct labels more accurately. The results with instance segmentation are lower, and the differences between models are minor. Lower results are expected as in instance segmentation, each instance of an object

**Table 4.4:** Precision and Recall results (Semantic Segmentation).

| Metric | P2PaLA | Torchvision |
|---|---|---|
| Accuracy | 94.33 | 92.09 |
| Precision | 0.989 | 0.952 |
| Recall | 0.953 | 0.966 |
| F1 | 0.971 | 0.959 |

**Table 4.5:** Precision and Recall results (Instance Segmentation).

| Metric | Torchvision | DetX101 | DetR50 |
|---|---|---|---|
| Accuracy | 83.10 | 81.17 | 82.95 |
| Precision | 0.858 | 0.880 | 0.888 |
| Recall | 0.964 | 0.912 | 0. 927 |
| F1 | 0.908 | 0.896 | 0.907 |

is labeled separately, which increases the possibility of mislabeling. Further analysis shows that the most significant cause for mislabeling for Torchvision Mask R-CNN was mixing marginalia and marginalia-top, causing incorrect detection. For the P2PaLA, some small elements were not detected (e.g., page number, catchword, and signature mark); however, this can be controlled by a parameter.

The IoU metric's prediction bounding boxes were compared against ground truth bounding boxes. The total IoU is calculated for all objects detected and element-specific IoU for each class separately, and no weighting is used. Due to the nature of IoU, the accuracy is easily better for larger objects, as the margin of error is smaller than for smaller objects. However, for some objects, the number of instances is small, and the impact of an individual object's accuracy is more significant.

Table 4.6 show the IoU results for semantic segmentation, and table 4.7 for instance segmentation. The results show that P2PaLA performs slightly better than the Torchvision model. However, IoU accuracy is almost perfect for large objects (e.g., text body) where the error caused by rounding margins is minor. As the difference between models is relatively small, the reason might be that Ground Truth's annotation is based on P2PaLA predictions that are manually corrected.

Similar results are measured for instance segmentation. When the two Detectron2-based models are compared together, it can be noticed that DetX101 (i.e., ResneXt-101) performs almost equally to DetR50 (i.e., ResNet-50). This indicates that using a more complex feature extractor is not improving the results.

Average Precision and Average Recall metrics were calculated for the Torchvision

**Table 4.6:** Intersection over union results (Semantic segmentation) .

| Element | P2PaLA | Torchvision |
|---|---|---|
| text body | 96.45 | 93.55 |
| header | 89.72 | 87.73 |
| heading | 89.71 | 68.15 |
| footnote | 83.73 | 73.82 |
| marginalia | 41.87 | 50.10 |
| marginalia-top | 69.47 | 79.96 |
| page number | 80.49 | 62.74 |
| catchword | 62.45 | 63.94 |
| signature mark | 56.19 | 66.82 |
| Overall | 76.41 | 74.26 |

**Table 4.7:** Intersection over union results (Instance Segmentation).

| | Torchvision | DetX101 | DetR50 |
|---|---|---|---|
| text body | 93.42 | 97.03 | 97.16 |
| header | 87.73 | 84.88 | 81.37 |
| heading | 91.09 | 83.38 | 87.52 |
| footnote | 85.03 | 87.35 | 83.03 |
| marginalia | 69.66 | 64.54 | 63.89 |
| marginalia-top | 79.96 | 72.76 | 72.06 |
| page number | 62.74 | 63.83 | 52.02 |
| catchword | 68.20 | 60.12 | 66.81 |
| signature mark | 65.09 | 62.09 | 67.41 |
| Overall | 77.36 | 75.10 | 73.79 |

**Table 4.8:** COCO metric: segm.

| | | | | |
|---|---|---|---|---|
| Average Precision (AP) | @[ IoU=0.50:0.95 | \| area= all | \| maxDets=100 | ] = 0.594 |
| Average Precision (AP) | @[ IoU=0.50 | \| area= all | \| maxDets=100 | ] = 0.877 |
| Average Precision (AP) | @[ IoU=0.75 | \| area= all | \| maxDets=100 | ] = 0.627 |
| Average Precision (AP) | @[ IoU=0.50:0.95 | \| area=medium | \| maxDets=100 | ] = 0.465 |
| Average Precision (AP) | @[ IoU=0.50:0.95 | \| area= large | \| maxDets=100 | ] = 0.601 |
| Average Recall (AR) | @[ IoU=0.50:0.95 | \| area= all | \| maxDets= 1 | ] = 0.608 |
| Average Recall (AR) | @[ IoU=0.50:0.95 | \| area= all | \| maxDets= 10 | ] = 0.645 |
| Average Recall (AR) | @[ IoU=0.50:0.95 | \| area= all | \| maxDets=100 | ] = 0.645 |
| Average Recall (AR) | @[ IoU=0.50:0.95 | \| area=medium | \| maxDets=100 | ] = 0.512 |
| Average Recall (AR) | @[ IoU=0.50:0.95 | \| area= large | \| maxDets=100 | ] = 0.654 |

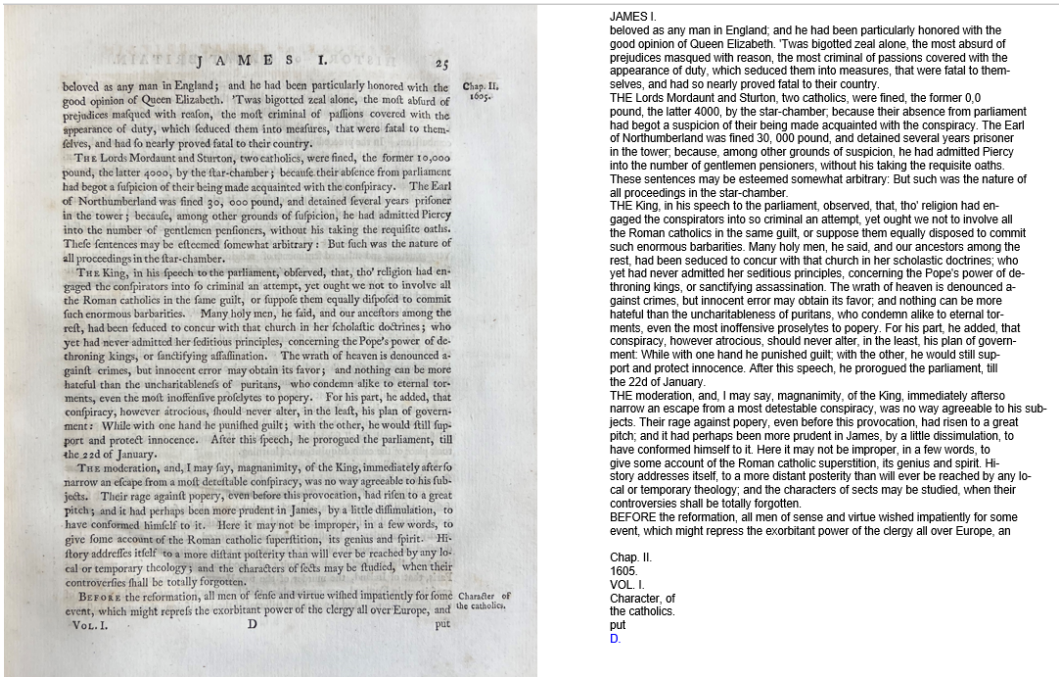
model. Unfortunately, as there is no reference point, it is impossible to say how well the model performs.

### 4.4.3 End-to-end pipeline

Although the focus of this thesis is on page layout analysis, it was necessary to test the solution also as part of the full OCR pipeline, and the following sequence of tasks was performed:

1. Store images of David Hume's 1754 edition of History of England in a folder
2. Run Page layout detection program. This produces as an output a PAGE XML file for each image under the subfolder page
3. Upload images and XML files to Transkribus
4. Execute 'Find Lines in Text Regions' task for all images
5. Execute 'Text recognition'task for all images
6. Download results from Transkribus

The end-to-end test demonstrates that Mask R-CNN-based Page Layout method can be used as a step in the OCR pipeline. The method is compatible with other steps in the process, and the output is acceptable. Few problems are detected, e.g., if the predicted mask or bounding box does not fully cover the text region, some characters may be missed, and further process steps will not be able to use these.



JAMES I. 25

beloved as any man in England; and he had been particularly honored with the good opinion of Queen Elizabeth. 'Twas bigotted zeal alone, the most absurd of prejudices masqued with reason, the most criminal of passions covered with the appearance of duty, which seduced them into measures, that were fatal to themselves, and had so nearly proved fatal to their country. Chap. II. 1605.

THE Lords Mordaunt and Sturton, two catholics, were fined, the former 10,000 pound, the latter 4000, by the star-chamber; because their absence from parliament had begot a suspicion of their being made acquainted with the conspiracy. The Earl of Northumberland was fined 30, 000 pound, and detained several years prisoner in the tower; because, among other grounds of suspicion, he had admitted Piercy into the number of gentlemen pensioners, without his taking the requisite oaths. These sentences may be esteemed somewhat arbitrary: But such was the nature of all proceedings in the star-chamber.

THE King, in his speech to the parliament, observed, that, tho' religion had engaged the conspirators into so criminal an attempt, yet ought we not to involve all the Roman catholics in the same guilt, or suppose them equally disposed to commit such enormous barbarities. Many holy men, he said, and our ancestors among the rest, had been seduced to concur with that church in her scholastic doctrines; who yet had never admitted her seditious principles, concerning the Pope's power of dethroning kings, or sanctifying assassination. The wrath of heaven is denounced against crimes, but innocent error may obtain its favor; and nothing can be more hateful than the uncharitableness of puritans, who condemn alike to eternal torments, even the most inoffensive proselytes to popery. For his part, he added, that conspiracy, however atrocious, should never alter, in the least, his plan of government: While with one hand he punished guilt; with the other, he would still support and protect innocence. After this speech, he prorogued the parliament, till the 22d of January.

THE moderation, and, I may say, magnanimity, of the King, immediately after so narrow an escape from a most detestable conspiracy, was no way agreeable to his subjects. Their rage against popery, even before this provocation, had risen to a great pitch; and it had perhaps been more prudent in James, by a little dissimulation, to have conformed himself to it. Here it may not be improper, in a few words, to give some account of the Roman catholic superstition, its genius and spirit. History addresses itself to a more distant posterity than will ever be reached by any local or temporary theology; and the characters of sects may be studied, when their controverfies fhall be totally forgotten.

BEFORE the reformation, all men of fenfe and virtue wifhed impatiently for fome event, which might repreſs the exorbitant power of the clergy all over Europe, and Character of the catholics.

VOL. I. D put

JAMES I.
beloved as any man in England; and he had been particularly honored with the good opinion of Queen Elizabeth. 'Twas bigotted zeal alone, the most absurd of prejudices masqued with reason, the most criminal of passions covered with the appearance of duty, which seduced them into measures, that were fatal to themselves, and had so nearly proved fatal to their country.
THE Lords Mordaunt and Sturton, two catholics, were fined, the former 0,0 pound, the latter 4000, by the star-chamber; because their absence from parliament had begot a suspicion of their being made acquainted with the conspiracy. The Earl of Northumberland was fined 30, 000 pound, and detained several years prisoner in the tower; because, among other grounds of suspicion, he had admitted Piercy into the number of gentlemen pensioners, without his taking the requisite oaths. These sentences may be esteemed somewhat arbitrary: But such was the nature of all proceedings in the star-chamber.
THE King, in his speech to the parliament, observed, that, tho' religion had engaged the conspirators into so criminal an attempt, yet ought we not to involve all the Roman catholics in the same guilt, or suppose them equally disposed to commit such enormous barbarities. Many holy men, he said, and our ancestors among the rest, had been seduced to concur with that church in her scholastic doctrines; who yet had never admitted her seditious principles, concerning the Pope's power of dethroning kings, or sanctifying assassination. The wrath of heaven is denounced against crimes, but innocent error may obtain its favor; and nothing can be more hateful than the uncharitableness of puritans, who condemn alike to eternal torments, even the most inoffensive proselytes to popery. For his part, he added, that conspiracy, however atrocious, should never alter, in the least, his plan of government: While with one hand he punished guilt; with the other, he would still support and protect innocence. After this speech, he prorogued the parliament, till the 22d of January.
THE moderation, and, I may say, magnanimity, of the King, immediately afterso narrow an escape from a most detestable conspiracy, was no way agreeable to his subjects. Their rage against popery, even before this provocation, had risen to a great pitch; and it had perhaps been more prudent in James, by a little dissimulation, to have conformed himself to it. Here it may not be improper, in a few words, to give some account of the Roman catholic superstition, its genius and spirit. History addresses itself, to a more distant posterity than will ever be reached by any local or temporary theology; and the characters of sects may be studied, when their controversies shall be totally forgotten.
BEFORE the reformation, all men of sense and virtue wished impatiently for some event, which might repress the exorbitant power of the clergy all over Europe, an

Chap. II.
1605.
VOL. I.
Character, of
the catholics.
put
D.

**Figure 4.16:** Outcome of full OCR pipeline where Mask R-CNN segmentation model is used for Page Layout

### 4.4.4 Summary of experiments

The tests prove that image segmentation can be used for fine-grained document layout analysis. The accuracy is suitable especially for large objects; however, it is not necessarily enough for all objects. In some cases, the problem is insignificant; for example, if marginalia is labeled as marginalia-top, it does not cause any significant problems. Furthermore, some of the issues can be resolved in the post-processing phase. As described earlier, it is possible to use the Non-Max Suppression (NMS) method for overlapping regions.

The solutions are not ready for production use in their current form as all document layout predictions need to be checked manually. For the accuracy improvement and suitability for production use, two alternatives could be considered:

1. Develop a two-stage approach. First, the number of different elements could be reduced instead of using fine-grained detection, i.e., only one content region type covering text body, footnotes, and headings. The rest of the element types would remain the same. The modified list of elements will be used for the layout analysis phase. Then, fine-grained detection could be done as a rule-based task during the post-processing phase.
2. Building a more sophisticated detection model. P2PaLA successfully utilizes the GAN approach with U-net architecture. However, mask R-CNN solu-

tions use transfer learning. Therefore, a more sophisticated solution would be combining these two aspects and building a more advanced model to improve accuracy. In parallel, the training dataset size would be increased.

# 5. Conclusions

Digitization has changed history research. The materials are available, and online archives make it easier to find the correct information and speed up the search for information. The remaining challenge is how to use modern digital methods to analyze the text of historical documents in more detail. The area is an active research topic for computer scientists and digital humanities scholars, and deep learning-based computer vision solutions have taken significant steps forward during the past decade. As the methods, algorithms, software platforms, and computing environments have developed, more and more challenging tasks can be completed more efficiently.

Document layout analysis is where object detection methods can be applied directly to historical documents. In addition, new tools, e.g., LayoutParser, OCR-D, and Transkribus, are introduced that can benefit from the recent development. However, most systems reviewed focus on coarse-grained methods, where only the high-level page elements are detected (e.g., text, figures, tables), and fine-grained methods are not available.

This thesis focuses on fine-grained OCR document layout analysis using image segmentation techniques. The same image segmentation techniques developed for biomedical and other domains easily apply to historical document contexts. Furthermore, open-source programming libraries and software packages based on these techniques can be adapted for the fine-grained detection task.

One of the main challenges of using deep learning methods is the lack of annotated data. Annotating historical documents is time-consuming; many training datasets are relatively small. These can consist of only tens or a few hundred pages of data. This problem can be minimized by using transfer learning and data augmentation. Transfer learning enables the utilization of larger datasets with similar characteristics, especially the usage of PubLayNet is an important addition. Data augmentation is another approach to avoiding problems caused by small datasets as it provides an easy mechanism to increase the training set size. However, the user must know that all augmentation methods are not nec-

essarily suitable for historical documents due to specific needs (e.g., binarization of images and asymmetric page structure).

This thesis presents two methods, semantic segmentation with U-net and instance segmenting with Mask R-CNN. They represent today's state-of-the-art image segmentation techniques. The most often used page layout approach is based on semantic segmentation, which might be adequate for simple OCR tasks providing accurate granularity. However, for more detailed page analysis, instance segmentation provides much more detailed information by separating instances from each other and providing object boundaries at the pixel level. Detailed analysis is crucial for more complex layouts (e.g., newspapers) and cases with higher element granularity needs.

As part of the thesis work, an experimental system for Mask R-CNN was built using PyTorch libraries. This implementation provides a setup to provide a detailed analysis to understand the accuracy and limitations of using a real-life dataset. The dataset consists of more than 700 annotated pages from four different editions of David Hume's The History of England from the $18^{th}$ century.

The dataset was prepared manually as no fine-grained layout analysis datasets were available elsewhere. Unfortunately, there are no standards for defining old, printed document page elements, and every use case has its requirements. For example, PubLayNet uses only five different page elements (e.g., text, title, list, table, figure). This granularity is not enough to handle the variety of elements described in chapter 1. When the solution for more complex structures is required, Mask R-CNN offers a promising approach.

The experiments indicate that Mask R-CNN, instance segmentation implementation, and P2PaLA, semantic segmentation solution, perform well. Both systems can detect page elements accurately, which indicates high precision and recall values. However, the accuracy of predicted regions (e.g., size and borders) leaves some questions, especially for small regions. Due to these issues (e.g., overlapping regions and too small regions), further work for the post-processing phase would be required. Overall, both systems are performing equally well, but both methods have challenges, and it is impossible to declare the winner based on these tests.

One of the thesis research questions was whether the solutions were ready for the large project with thousands of pages. For this question, the answer would be 'no' now as too many manual corrections would be required in its current form, and therefore, a more accurate solution would be required. One idea would be to implement fine-grained detection using a two-stage approach. The first

phase would focus on detecting large elements on the pages using deep learning methods, and the second phase would use a rule-based traditional object detection approach to identify the more fine-grained elements.

# Bibliography


[1] Sherif Abuelwafa, Sara Zhalepour, Ehsan Arabnejad, Mohamed Mhiri, Emilienne Greenfield, James P. Ascher, Sofia Bach, Victoria Svaikovsky, Alayne Moody, Andrew Piper, Chad Wellmon, and Mohamed Cheriet. Detecting Footnotes in 32 million pages of ECCO. Journal of Cultural Analytics. December 3, 2018.

[2] Calamari OCR documentation. https://calamari-ocr.readthedocs.io/en/latest/. Accessed on 31.5.2022.

[3] Common Objects in Context (COCO). https://cocodataset.org/#detection-eval. Accessed on 31.5.2022.

[4] Connor Shorten and Taghi M. Khoshgoftaar. A survey on Image Data Augmentation for Deep Learning. Journal of Big Data (2019) 6:60. https://doi.org/10.1186/s40537-019-0197-0

[5] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei. Imagenet: A large-scale hierarchical image database. Computer Vision and Pattern Recognition. 2009.

[6] Detectron2. Docs. Tutorials. Use Custom Datasets. https://detectron2.readthedocs.io/en/latest/tutorials/datasets.html. Accessed on 2.5.2022.

[7] David Doermann, Karl Tombre (ed.). Handbook of Document Image Processing and Recognition. 2014. Springer.

[8] Sebastien Eskenazi, Petra Gomez-Krämer, Jean-Marc Ogier. A comprehensive survey of mostly textual document segmentation algorithms since 2008. Pattern Recognition 64 (2017).

[9] Alberto Garcia-Garcia, Sergio Orts-Escolano, Sergiu Oprea, Victor Villena-Martinez, Pablo Martinez-Gonzalez, Jose Garcia-Rodriguez. A survey on deep learning techniques for image and video semantic segmentation. Applied Soft Computing 70 (2018). 2018.

[10] Philip Gaskell. A new Introduction to Bibliography. Oak Knoll Press. Delaware. USA. 1995.

[11] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In CVPR, 2014.

[12] R. Girshick. Fast R-CNN. 2015 IEEE International Conference on Computer Vision. 2015.

[13] Ian Goodfellow, Yoshua Bengio, Aaron Courville: Deep learning. The MIT Press. 2016.

[14] Gregg, S. (2021). Old Books and Digital Publishing: Eighteenth-Century Collections Online (Elements in Publishing and Book Culture). Cambridge: Cambridge University Press. doi:10.1017/9781108767415

[15] Tobias Gruning, Gundram Leifert, Tobias Strauss, Roger Labahn. A Robust and Binarization-Free Approach for Text Line Detection in Historical Documents. 2017 14th IAPR International Conference on Document Analysis and Recognition. 2017.

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, Jian Sun. Deep Residual Learning for Image Recognition. Computer Vision and Pattern Recognition. 2016.

[17] Kaiming He, Georgia Gkioxari, Piotr Dollar. Ross Girshick. Mask R-CNN. 2017 IEEE International Conference on Computer Vision. 2017.

[18] Kraken documentation https://kraken.re/master/index.html. Accessed on 15.5.2022.

[19] Krizhevsky, Alex and Sutskever, Ilya and Hinton, Geoffrey E. ImageNet Classification with Deep Convolutional Neural Networks. In Advances in Neural Information Processing Systems, (eds) F. Pereira and C. J. C. Burges and L. Bottou and K. Q. Weinberger. Curran Associates, Inc. 2012

[20] Layout Parser Zoo documentation. https://layout-parser.readthedocs.io/en/latest/notes/modelzoo.html. Accessed on 15.5.2022.

[21] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ra manan, P. Dollar, and C. L. Zitnick. Microsoft COCO: Common objects in context. European Conference on Computer Vision. 2014.

[22] Liu, X., Deng, Z., Yang, Y. Recent progress in semantic image segmentation. Artificial Intelligence Review (2019) 52. 2019.

[23] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. Computer Vision and Pattern Recognition. 2015.

[24] McGill University Library. David Hume project. https://archive.org/search.php?query=subject%3A%22David+Hume+project%22. Accessed on 1.5.2022.

[25] Metadata Encoding & Transmission (METS): An Overview & Tutorial. https://www.loc.gov/standards/mets/METSOverview.v2.html. Accessed on 15.1.2022.

[26] Guenther Muehlberger et al., Transforming scholarship in the archives through handwritten text recognition: Transkribus as a case study. Journal of Documentation vol. 75 (5) 2019.

[27] National Archives of Finland: Annual report 2021. https://kansallisarkisto.fi. Accessed on 2.9.2022.

[28] National Library of Finland. https://digi.kansalliskirjasto.fi/etusivu. Accessed on 2.9.2022.

[29] Clemens Neudecker, Konstantin Baierer, Maria Federbusch, Matthias Boenig, Kay-Michael Wurzner, Volker Hartmann, Elisa Herrmann. OCR-D: An end-to-end open source OCR framework for historical printed documents. Proceedings of the 3rd International Conference on digital access to textual cultural heritage, 2019-05-08, p.53-58. In 3rd International Conference on Digital Access to Textual Cultural Heritage (DATeCH2019), May 8-10, 2019, Brussels, Belgium. ACM, New York, NY, USA

[30] Sarah Oberbichler, Emanuela Boros, Antoine Doucet, Jani Marjanen. Eva Pfanzelter, Juha Rautiainen, Hannu Toivonen, Mikko Tolonen. Integrated interdisciplinary workflows for research on historical newspapers: Perspectives from humanities scholars, computer scientists, and librarians. Journal of the Association for Information Science and Technology. Wiley. 2022.

[31] OCR-D wrapper for detectron2 based segmentation models. https://github.com/bertsky/ocrd_detectron2. Accessed on 15.1.2022.

[32] OCRopus. OCRopus in GitHub. https://github.com/ocropus. Accessed on 15.5.2022.

[33] OCRopus4. OCRopus4 in GitHub. https://github.com/ocropus/ocropus4. Accessed on 15.5.2022.

[34] Oliveira, S.A., Seguin, B., Kaplan, F.: Dhsegment: a generic deep-learning approach for document segmentation. In: 2018 16th International Conference on Frontiers in Handwriting Recognition (ICFHR), pp. 7-12. IEEE, (2018)

[35] The OpenCV Team. Opencv. https://opencv.org/. Accessed on 15.1.2022.

[36] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, Soumith Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library. 2019.

[37] Pillow software documentation. https://pillow.readthedocs.io/en/stable/. Accessed on 15.1.2022.

[38] S. Pletschacher, A. Antonacopoulos. The PAGE (Page Analysis and Ground-Truth Elements) Format Framework. Proceedings of the 20th International Conference on Pattern Recognition (ICPR2010), Istanbul, Turkey, August 23-26, 2010, IEEE-CS Press, pp. 257-260

[39] Python. Programming language. https://python.org. Accessed on 15.1.2022.

[40] Pytorch. Python library for Deep Learning. https://pytorch.org. Accessed on 15.1.2022.

[41] Transkribus P2PaLA Layout Analysis tool documentation. https://readcoop.eu/transkribus/docu/p2pala/. Accessed on 15.1.2022.

[42] TorchVision Object Detection Finetuning Tutorial. https://pytorch.org/tutorials/intermediate/torchvision_tutorial.html. Accessed on 15.1.2022.

[43] Quiros Lorenzo. P2PaLA: Page to PAGE Layout Analysis tookit. https://github.com/lquirosd/P2PaLA. 2017. Accessed on 15.1.2022.

[44] Quiros Lorenzo. Multi-Task Handwritten Document Layout Analysis. https://arxiv.org/abs/1806.08852. 2018. Accessed on 15.1.2022.

[45] Ren, S., He, K., Girshick, R., Sun, J.: Faster R-CNN: Towards real time object detection with region proposal networks. In: Cor tes, C.,Lawrence, N., Lee, D., Sugiyama, M., Garnett, R. (eds) Advances in Neural Information Processing Systems, Curran Associates, Inc, pp. 91-99 (2015)

[46] Ronneberger O., Fischer P., Brox T. (2015) U-Net: Convolutional Networks for Biomedical Image Segmentation. In: Navab N., Hornegger J., Wells W., Frangi A. (eds) Medical Image Computing and Computer-Assisted Intervention - MICCAI 2015. MICCAI 2015. Springer.

[47] Zejiang Shen, Ruochen Zhang, Melissa Dell,Benjamin Lee, Germain Charles, Jacob Carlson, Weining Li. LayoutParser: A Unified Toolkit for Deep Learn-

ing Based Document Image Analysis. Document Analysis and Recognition - ICDAR 2021, 2021-09-02, p.131-146.

[48] Smith Ray. An Overview of the Tesseract OCR Engine. Ninth International Conference on Document Analysis and Recognition (ICDAR 2007). IEEE. 2007.

[49] Richard Szeliski. Computer Vision: Algorithms and Applications. 2nd Edition. Springer. 2021.

[50] Tan C., Sun F., Kong T., Zhang W., Yang C., Liu C. (2018) A Survey on Deep Transfer Learning. In: Kurkova V., Manolopoulos Y., Hammer B., Iliadis L., Maglogiannis I. (eds) Artificial Neural Networks and Machine Learning - ICANN 2018. ICANN 2018. Lecture Notes in Computer Science, vol 11141. Springer.

[51] Tessdoc. Tesseract documentation. https://tesseract-ocr.github.io/tessdoc/Installation.html. Accessed on 15.1.2022.

[52] Vesalainen Ari. Re-OCR - Page layout analysis. Digital Humanities Project I. Unpublished project report. University of Helsinki. 2021.

[53] Yuxin Wu, Alexander Kirillov, Francisco Massa, Wan-Yen, Ross Girshick. Detectron2. https://github.com/facebookresearch/detectron2.2019.

[54] Xie Saining, Girshick Ross, Dollar Piotr, Tu Zhuowen, He Kaiming. Aggregated Residual Transformations for Deep Neural Networks. Computer Vision and Pattern Recognition. 2017.

[55] Xu Zhong, Jianbin Tang, Antonio Jimeno-Yepes. PubLayNet: Largest Dataset Ever for Document Layout Analysis. 2019 International Conference on Document Analysis and Recognition (ICDAR). 2019.