8 A machine learning pipeline for digitalising historical printed materials – from data collection to a searchable database

Dalia Ortiz Pablo
Uppsala University

Sushruth Badri
Uppsala University

Gijs Aangenendt
Uppsala University

Mo von Bychelberg
Uppsala University

Matts Lindström
Uppsala University

Recent developments in the fields of machine learning and computer vision have created new opportunities for the digitalisation of printed historical materials. However, successful integration of machine learning models requires interdisciplinary collaboration between computer- and data scientists, researchers, librarians and/or archivists, and digitisation experts. This chapter describes a comprehensive pipeline designed to address the challenges of digitalising printed historical materials, from document-scanning best practices to incorporating state-of-the-art machine learning techniques. It aims to streamline the management and processing of historical data, making the digitalised materials accessible and searchable through the application of machine learning techniques. The content of this chapter encompasses scanning best practices, annotation approaches, model training, and deployment. This chapter presents a collection of useful tools for each stage of building a machine learning model, step-by-step instructions and example notebooks designed to be easily adapted to other cases.

1 *Introduction*

Availability and accessibility of historical data has always been a condition for Humanistic research, long before the term "Digital Humanities" was coined. When European Renaissance humanism emerged in the 1400s, this also meant the establishment of a new mode of knowledge production that was focused on the uncovering and reconstitution of historical (classical) remnants and records – thus forming the basis for the scholarly endeavour that would eventually, over the centuries and across various specialised

HUM INFRA Dalia Ortiz Pablo, Sushruth Badri, Gijs Aangenendt, Mo von Bychelberg & Matts Lindström. 2025. A machine learning pipeline for digitalising historical printed materials – from data collection to a searchable database. In Gerlof Bouma, Dana Dannélls, Dimitrios Kokkinakis & Elena Volodina (eds.), *Huminfra handbook: Empowering digital and experimental humanities* (NEALT Proceedings Series 59), 207–250. University of Tartu Library. DOI: 10.58009/aere-perennius0177

university disciplines, become configured as modern "Humanities". In the early- to mid-1800s, at the same time as History was formed as a professional discipline with new ambitions of rigour and objectivity modelled on the natural sciences, this impulse was further stimulated by a growing abundance of archival sources that became available in post-revolutionary (1789) Europe and its emerging nation states. The archives of Kings and other dynastic rulers suddenly became properties of nations and institutions such as national archives and national libraries were established. In the wake of this new archival and documentary landscape, influential historiographers such as Leopold von Ranke expressed a belief that History, finally, would be truthfully uncovered – "wie es eigentlich gewesen" (eng. "how it really was"; Von Ranke [1824] 1885: VIII).

Today, as the archival landscape is yet again changing through digitalisation, Digital Humanities similarly promises new modes of Humanistic knowledge production, often in conjunction with the application of new machine learning and artificial intelligence techniques. However – whether researchers are looking to use such new possibilities to further glean from posterity unnoticed clues and details (Ginzburg 1979), or to discover patterns at a macro-scale that would otherwise be impossible to observe (Jockers 2013, Moretti 2005) – any such endeavour presupposes the basic operation of converting analog, historical documents to digital representations that can be further processed and analysed.

Integrating state-of-the-art machine learning requires technical skills and knowledge about how to construct a digitalisation pipeline and implement machine learning models. These skills are not typically associated with the average humanist, but rather with computer- and data scientists. At the same time, successful digitalisation requires intimate knowledge of the physical collections, usually held by archivists or librarians working in cultural heritage institutions, with support from digitisation experts (Smith & Whearty 2023). The possibilities and challenges presented by artificial intelligence applied to digitalisation thus extends to and includes the cultural heritage institutions that hold the historical collections. Without scanning and digitalisation of the physical sources, there can be no sets of historical data to peruse in the process of training machine learning models.

Interdisciplinary collaboration is therefore a necessary component in any successful digitalisation project, ideally involving professional experts from all disciplines relevant to the entire digitalisation pipeline. Awareness of this need to crossing traditional and disciplinary boundaries is growing; the archival discipline, for instance, is currently facing major theoretical and practical changes due to digitalisation and the introduction of artificial intelligence (Colavizza et al. 2022), leading to calls for "a blend of computational

and archival thinking" (Marciano et al. 2018: 181), including participation of archival professionals in the creation of datasets for training, testing, and controlling the output of algorithms (Jaillant & Caputo 2022).

By drawing on practical experience from a specific real-world case that involves such interdisciplinary collaboration and researcher-driven digitalisation, this chapter provides hands-on advice and perspectives, more broadly applicable to the problem of digitalising printed historical documents and applying machine learning techniques. We describe a general comprehensive methodology and pipeline for digitalising printed historical materials using machine learning, including data acquisition (scanning of printed text). Most of this description is based on the authors' experience with the digital infrastructure project *Communicating Medicine: Digitalisation of Swedish Medical Periodicals, 1781–2011* (SweMPer) hosted by the Department of History of Science and Ideas at Uppsala University in collaboration with the Centre for Digital Humanities and Social Sciences (CDHU) and Uppsala University Library. SweMPer aims to build a searchable database of a wide range of Swedish medical periodicals created by medical professionals, patients, and medical students. The pipeline used in SweMPer utilises machine learning models for object detection to perform layout recognition and optical character recognition (OCR) to extract detailed metadata for increased searchability. At the end of the chapter, notebooks for specific parts of the pipeline are provided.

2 *Research context*

As digitised materials are increasingly being used by historical researchers for Natural Language Processing (NLP) and text mining tasks, the demands on digitalisation quality are rising. High quality digitalisation – including accurate character and word recognition, proper layout recognition, and correct reading order – is needed for successfully performing popular NLP tasks such as topic modelling, named entity recognition, and training word embeddings (Hill & Hengchen 2019, Neudecker et al. 2021, van Strien et al. 2020). Historical print often includes heterogeneous and complex layouts, as well as font- and spelling variations, causing standard layout detection and OCR software to fall short (Jarlbrink & Snickars 2017). Application of machine learning methods can offer significant advantages for digitalisation by improving quality, and it may also open up new research venues and searchability possibilities that are not available with more basic procedures.

With a digitalisation pipeline that produces rich and detailed metadata, materials can be filtered on a more granular level based on the type of

content. For example, in the case of newspapers and periodicals it could be valuable to separate main article text from other text-types that occur on the printed pages such as advertisements, tables, lists, and crossword puzzles. In addition to the textual layer, researchers may also want to explore visual elements – such as illustrations, photographs, diagrams, etc. – something that can be facilitated by machine learning-based computer vision tasks, such as image classification and clustering (Wevers & Smits 2020) and the automatic generation of image tags and descriptions (Thomas & Testini 2024, Villaespesa & Crider 2021).

Designing a customised digitisation pipeline, in comparison to off-the-shelf solutions, often provides greater flexibility for integrating and adapting components that are optimally tailored to the material of interest. Machine learning models for computer vision offer new opportunities for addressing the quality of the digitisation, in terms of the accuracy of the layout detection, optical character recognition (OCR), and metadata extraction (Thomas & Testini 2024). Achieving higher quality and better, enriched metadata through machine learning, however, requires annotating training data for fine-tuning layout recognition and OCR models. This is particularly the case with historical materials where pre-existing annotated data usually does not exist.

While there is great potential in the application of machine learning to digitalisation, ethical issues also need to be taken into account in relation to the annotation and training process. One well-known problem is the issue of bias in machine learning models. Here, we refer to bias in the broader, humanities-centered sense of misrepresentation in various forms and at various levels of Artificial intelligence (AI) usage, for example under- and misrepresenting certain groups of people such as women and minorities (Jaillant & Caputo 2022, Nicoletti & Bass 2023). Machine learning models are trained based on datasets that have been constructed by humans; therefore, they also pick up any implicit and explicit biases contained in the data (Ntoutsi et al. 2020). Due to institutionalised racism, sexism, and various cultural stereotypes, certain kinds of people may also be overrepresented in the data and subsequently also in the AI algorithm. As Jaillant & Caputo (2022: pp. 824-5) have put it: "for example, since White men are over-represented at the most senior levels of government and other sectors, documents by women or ethnic minorities could be flagged as less important". While using OCR when digitising historical documents is a technical exercise that does not in itself necessarily introduce or perpetuate bias, it can still be error-prone and it is important to be aware of the affordances of written language and limits of any particular technology; for example, Swedish or Spanish has certain letters that OCR models trained on American

English may not recognize. Choices made in subsequent efforts of "cleaning" and "sanitising" problematic OCR may, yet again, potentially introduce bias (Arroyo-Ramirez 2016). At later stages, during computational analysis and post-processing, further biases may also be introduced – for instance, textual bias may occur when producing word embeddings (Bolukbasi et al. 2016). Being aware of and addressing such problems is important to all phases of the construction and implementation of a digitalisation pipeline.

Another important aspect to take into consideration is the priorities made and selection of sources when digitalising. Digitalisation efforts are commonly driven by institutional actors and custodians of historical collections, i.e. libraries and archives. One recent example is a digitisation program initiated at the The National Library of Scotland (NLS) that is expected to provide digital versions of as much as a third of its collection by 2025 (Jaillant & Caputo 2022). Many larger scale and well-known projects have been concluded in the past decades, most prominently perhaps the Google Books Scanning project, primarily in collaboration with many of the major American university libraries[1]. In the Swedish context one impactful effort concerns the The Swedish Newspapers digitalisation project (Svenska dagstidningar[2]). This project, initiated and hosted by the National Library of Sweden, makes it possible to search and read digital historical newspapers from 1645 up to today, and has in many ways proven critical to research.

While there is undeniable value in such general, institutional initiatives, there may be a further benefit in researcher-driven efforts tailored to the needs of individual projects and specific research questions. This is important in order for researchers to get access to sources that reflect their particular research interests, making them less dependent on what is available and what is being prioritised by cultural heritage institutions. From the perspective of preservation, for instance, sources from the twentieth century might be considered too 'modern' to be prioritised for digitisation over rare objects, or not consulted enough by users to warrant inclusion. Furthermore, they might simply be excluded because of legal problems concerning copyright status. In the case of The Swedish Newspapers mentioned above, the database currently limits access to anything published after 1924 for copyright reasons (unless physically situated at the National Library or certain university and public libraries) – and there are no options for bulk downloading of data, required for further and more advanced text mining or computational analysis. In other cases, metadata might be lacking or the data could be organised in ways that make it less conducive to efficient research.

1 https://books.google.com; see also, https://www.hathitrust.org/
2 http://tidningar.kb.se/

Researcher-driven digitalisation allows for control over the selection of sources and produces high quality data and metadata adapted to specific aims, priorities and research questions. The SweMPer-project provides one example of this approach to digitalisation, but other projects can also be mentioned to further exemplify a growing trend of research- and researcher-driven digitalisation efforts. One such project is ActDisease[3], which aims to study the history of twentieth century patient organisations. The study includes patient organisations from four European countries (Sweden, United Kingdom, France and Germany). ActDisease incorporates digitalisation workflows customised to and driven directly by the project's aims. To meet the project's research goals, member newsletters and magazines have been digitalised, creating a corpus of sources unlikely to be digitised without any research initiative (Aangenendt et al. 2024). Another example is the research infrastructure project Swedish Riksdag 1867-2022: An Ecosystem of Linked Open Data (SWERIK)[4] which illustrates the need for high quality digitalisation and descriptive metadata. The Swedish parliamentary proceedings, motions, and reports have previously been digitised by the National Library of Sweden, but the quality has proven insufficient for many research demands. The aim of SWERIK is to enrich the Swedish parliamentary documents by correcting digitisation errors and providing linked metadata for parliamentary documents and members of parliament, opening up new ways for researchers to explore the materials both quantitatively and qualitatively.

In conclusion, digitalisation efforts initiated and driven directly by researchers and their perspectives are crucial to create conditions for high quality historical research. By interdisciplinary collaboration that involves not only humanities researchers, but also machine learning experts and professionals at relevant libraries and archives, digitalisation workflows that are highly customised and tailored to the researchers' needs can be achieved within the framework of a specific project. The general and project-specific digitalisation pipeline presented below can hopefully help facilitate future researcher-driven digitalisation projects.

3 *Description of methods and tools*

Training a machine learning (ML) model—such as one designed to assist in the digitization of historical materials—typically involves four key stages: data acquisition, annotation and processing, model training, and deployment. Each stage benefits from collaboration among archivists, librarians,

3 https://www.actdisease.org, see also chapter NN in this book
4 https://swerik-project.github.io/

researchers, and engineers, ensuring that issues are addressed, project quality is enhanced, and transitions between stages are seamless. In the context of digitization, and specifically for the case study presented in this chapter, the data acquisition stage focuses primarily on scanning. Accordingly, data acquisition will be addressed in Section 3.1, annotation and processing in Section 3.2, model training in Section 3.3, and deployment in Section 3.4.

3.1 *Scanning and data cleaning preparation*

The first step in developing a machine learning based digitalisation pipeline is acquiring the data, which in the context of this handbook chapter means acquiring digital copies of the materials of interest. This can be done in several ways depending on the availability of the data. This section focuses in particular on scanning, which is the methodology employed in the SweMPer-project and presented in section 4. The scanning of historical materials is best done by digitisation experts at libraries or archives. However, if you want to digitalise material on your own there are some general guidelines that are important to follow. These guidelines can also be helpful when discussing the details of digitisation with libraries and archives that are often necessary collaborating partners in such undertakings.

Before initiating the scanning process, it is crucial to establish a consistent and descriptive file-naming convention and folder structure that captures key metadata. This decision is usually made by the lead researcher, ideally in collaboration with the engineer who will process the data in subsequent steps. One example of such an organization is incorporating metadata such as year of publication, ascending and "zero-padded" file sequence numbering (i.e. 0001, 0002, 0003 ... 0999), and any other information that may be of relevance. This helps keep the digitised materials organised at the file system level. Alternatively, a unique identifier can be used for each publication or collection that will be digitised, which can later be replaced with a more descriptive name. Common practice here is to use underscore, dashes, or to capitalise the first letter of each section, as computers struggle handling files with blank spaces [5].

Next, the specifications of the digitisation need to be decided upon, often in collaboration with the digitisation experts at cultural heritage institutions. This includes deciding on a suitable output format (tiff, jpg, or pdf), required image resolution (300 ppi or higher for printed materials with small font), colour settings, and what should be captured on each image (single page, double page).

5 https://datamanagement.hms.harvard.edu/plan-design/file-naming-conventions

When performing the scanning, it is crucial to keep the page and the text as straight as possible. Try to avoid scanning pages warped or skewed, as this can negatively affect the recognition of layout and text elements in later stages of the pipeline. Another recommendation is to avoid adding additional 'noise' on the image, in the form of spots, marks, and lines that are not part of the original document. This 'noise' could later on be interpreted as characters or lead to misinterpreted characters in the OCR process. A regular quality control can help identify any issues. Lastly, having a high contrast makes it easier to segment between the background and page elements. There are post-processing methods to deskew the page, straighten text lines, and remove noise but in general the better the image quality is from the outset, the better the final outcome.

3.2 *Data annotation and preparation for training*

Once the scans are available in digital image format, following the steps outlined in section 3.1, the next step is data annotation. This process involves several key considerations, including selecting the elements to annotate, determining appropriate labels or names, choosing an annotation tool, and deciding on the output format. Ideally, these decisions are made collaboratively by the lead researcher, the engineer, and the annotators to ensure consistency, efficiency, and to identify and address any potential biases. In this section, we aim to clarify the following questions:

- What is data annotation in the context of image-based collection digitalisation? What does an annotation look like?
- What annotation format is best for your task?
- What are the best practices for data annotation?
- What are the best tools for easy, efficient, and secure data annotation?
- How do we prepare the data to feed it into our ML model?

3.2.1 *Data annotation and how it looks*

Before annotating any data, it is essential to understand what data annotation entails in the context of computer vision. *Data annotation* involves identifying and labelling elements of interest within images, videos, or other forms of visual media. During this process, the annotations are converted into tags and coordinates that a computer can interpret and utilise (OpenCV 2024). There are different forms of annotation in computer vision, including:

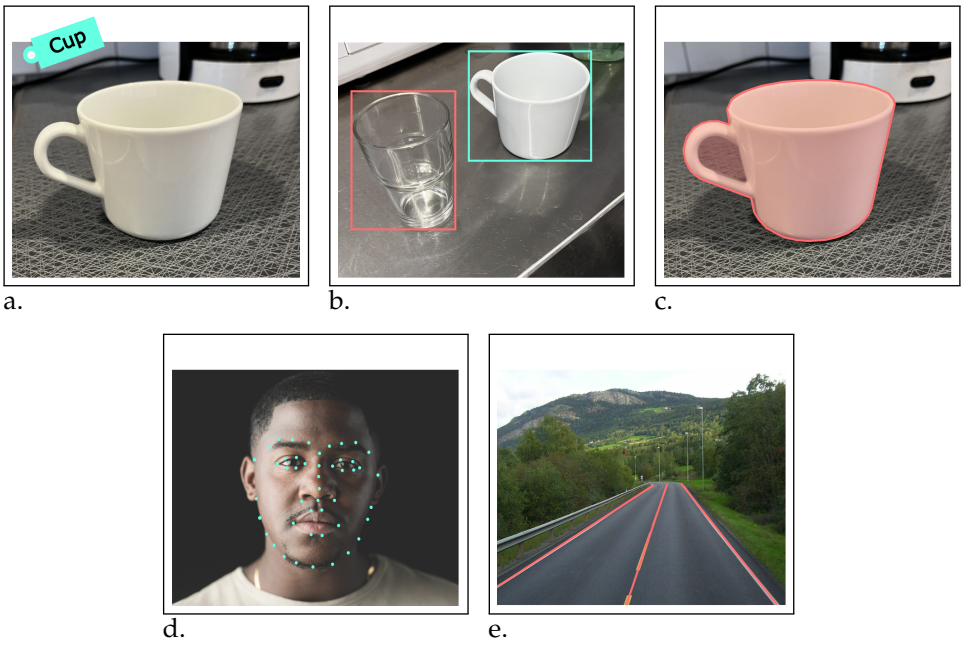


Figure 1: Types of data annotation in computer vision: (a) full image classification, (b) 2D bounding boxes, (c) segmentation, (d) key point and landmark, (e) lines and splines (image sourced from Małecki 2006).

- *Full image tagging*, such as labelling an image as "cup" or "glass" (Figure 1a), which is primarily used in classification tasks.
- *2D bounding boxes* (Figure 1b), which are rectangles used to identify and determine the location of objects, commonly utilised in object detection and localisation tasks.
- *Polygonal segmentation* (Figure 1c), which goes beyond rectangles by surrounding objects with more complex polygons to obtain a more precise shape and localisation.
- *Key point and landmark* (Figure 1d) involves drawing points on an object within an image, which is useful for pose detection and facial recognition.
- *Lines and splines* annotation (Figure 1e) is created by using lines to mark clear paths, such as roads, which is particularly useful in navigation tasks (OpenCV 2024, Xailient 2024).

Linking the types of data annotations to the use case presented in this

chapter, we will focus on 2D bounding box annotations. Bounding boxes are one of the most common types of annotation for object detection, their aim is to determine the location and boundaries of target objects by surrounding them with tight rectangles. The annotations provide metadata for each box in an image, such as the coordinates of the corners or centres, the widths and heights relative to the image, and the class of each box (Xailient 2024).

3.2.2 *Bounding boxes label formats*

As mentioned earlier, annotating data with 2D bounding boxes generates information that can be fed into a ML model to teach it to detect elements in an image. How these data are organised and stored varies based on the object detection ML model one plans to use and the data annotation tool employed. It is possible to convert between different formats, as will be demonstrated in the last section "Hands-on Practical Introduction".

COCO and *YOLO* are conventional annotation formats used by the most popular object detection ML models in computer vision. The choice between the two relies on the specific requirement of the ML model. The COCO format stores information for all images and all bounding boxes in a single *JSON* file. Each bounding box is formatted as $[x,\ y,\ width,\ height]$, where the tuple (x, y) corresponds to the coordinates of the upper-left corner, and *width* and *height* are the dimensions of the box from (x, y). The file also contains the IDs for each class and each image (Medium 2024). Figure 2 (left side) illustrates an example of this. The single JSON file associates an integer ID with each category, image, and box per image. The details of each box are stored in the "annotations" section with a *bbox* entry per bounding box. In the *bbox* entry, the values 0 and 1 correspond to x and y, respectively, and 2 and 3 correspond to the width and height. In contrast, YOLO uses *TXT* files, generating one file per image. If an image has no target objects, a TXT file is not needed. Each TXT file contains one row per annotated object, formatted as: *class x y width height* (see Figure 2 right side bottom rectangle), where (x, y) correspond to the center of the box (Ultralytics 2023), as opposed to the COCO format, where (x, y) correspond to the upper-left corner.

Two important differences to highlight are image re-scaling and the way information is stored. In the COCO format, the image size information is not re-scaled, so the size of the bounding box matches the image's scale. This means that if the image has dimensions (W, H), then the box widths and heights will fall within the intervals $[0, W]$ and $[0, H]$, respectively. In contrast, the YOLO format re-scales the image, resulting in width and height values that are always between the interval $[0, 1]$ (Medium 2024,

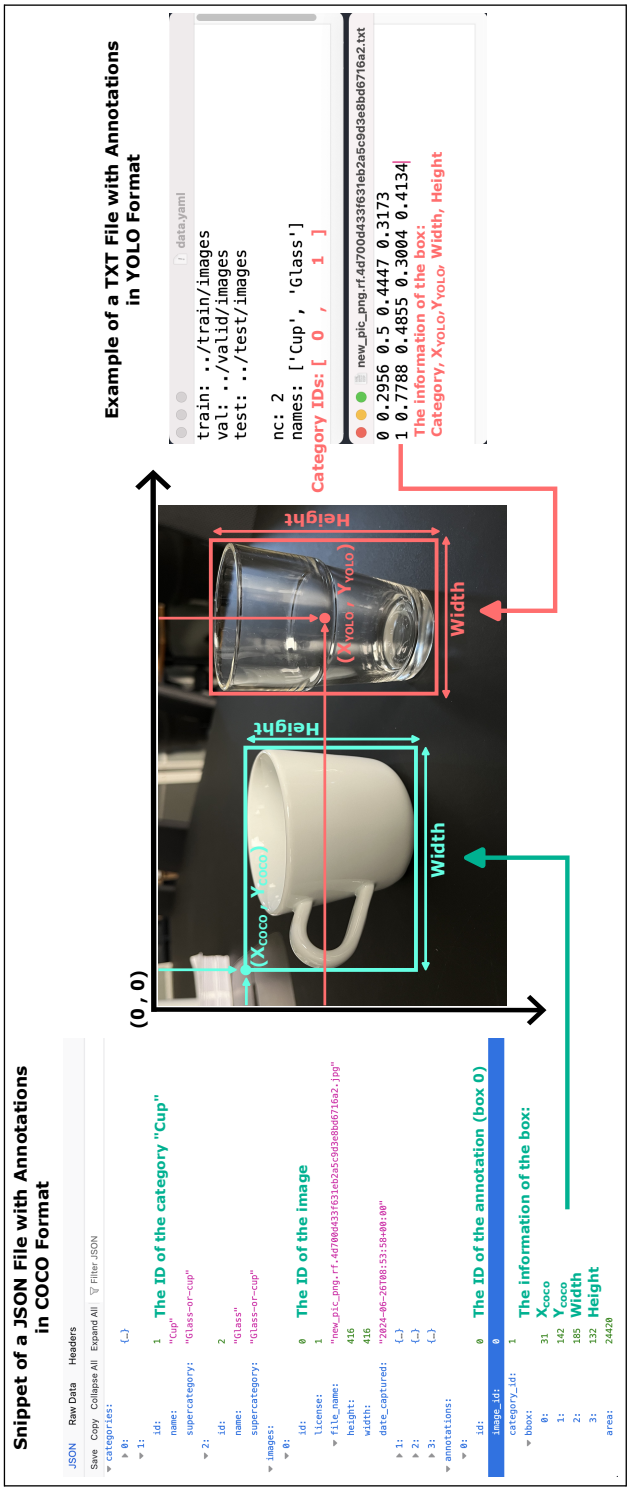


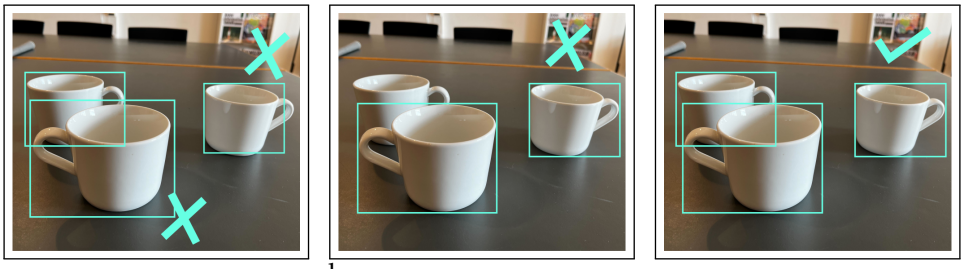
Figure 2: Comparison of COCO and YOLO annotation formats. The left side shows a COCO JSON snippet with bounding boxes in image scale. The right side shows a YOLO TXT snippet with normalised bounding boxes.

Ultralytics 2023). This re-scaling process involves normalising the bounding box coordinates by the dimensions of the image, so the box coordinates are relative to the image size. For instance, if an image has a width of 500 pixels and a bounding box is 250 pixels wide, the YOLO format will store this width as 0.5. Regarding file organisation, the COCO format stores all the information for a set of images in a single file (such as the one in Figure 2). Conversely, the YOLO format creates one file for each image, along with a *YAML* file (Figure 2, top rectangle on the right side) that contains links to the folders for each subset of images (we discuss these subsets in more detail in Section 3.2.5).

3.2.3 *Best practices for data annotation*

While various best practices for data annotation have been proposed, there is no universal consensus. The approach to annotating data largely depends on the specific task at hand. In some instances, it is essential to annotate every object within the category of interest in an image, whereas in other cases, this level of detail may not be necessary. Regardless of the annotation process, consistency and clear guidelines are crucial, especially when collaborating with others or using crowd-sourced labelling. Consistency in naming conventions is particularly important to prevent future confusion (Labellerr 2023). For instance, in the cup-or-glass scenario, it is essential to use ”cup” uniformly rather than allowing variations such as ”mug” or ”Cup.” Such variations could lead to inconsistencies, as the label for the same object would shift from one—”cup”—to three—”Cup,” ”mug,” and ”cup.” Inconsistent labels could confuse a ML model when trying to distinguish between objects in these classes. Similarly, it is crucial to agree on the number of classes of interest. For example, if one is annotating images of pets, one should decide whether to classify them into just two categories—dogs and cats—or to include additional classes, such as specific breeds (Nanonets n.d., Roboflow 2024b).

Tight bounding boxes are another key practice in data annotation. It is recommended to draw boxes that tightly encompass the entirety of the object without cutting off any portions (Nanonets n.d., Roboflow 2024b). In Figure 3a, an example is shown where the front most cup is annotated with a box that is too large, which is not recommended, sub-figure (c) shows a box that fits the cup correctly. Related to this, it is advisable to ensure that the entire object is covered without missing any borders. This practice is crucial for helping the ML model accurately learn the contours of the object (Roboflow 2024b). Figure 3a shows a box that is too small to fit the object (rightmost cup), which goes against best practices, and a correct annotation



a. b. c.

Figure 3: Examples of the implementation of good and bad practices in data annotation: (a) This is an incorrect annotation because two of the boxes do not properly fit the objects. (b) This is incorrect because one of the objects is not annotated. (c) This image is correctly annotated.

is shown in sub-figure (c). It has also been noted to be beneficial to label occluded objects if they are identifiable, even if this results in overlapping boxes. Additionally, these objects should be labelled as if they were in full view (Roboflow 2024b). In Figure 3c, we can see that the cup at the left-back is annotated as it was in full view.

Finally, it is recommended to label every instance of the object in each image. Failing to do so can introduce false negative predictions, i.e. incorrect or no detection when in fact there was an object of interest, which would negatively impact the ML model's performance (Nanonets n.d., Roboflow 2024b). Figure 3b shows an example of an incorrect annotation where the left-back cup is not surrounded by a rectangle, which goes against the recommendations. However, in some situations, it might not be necessary to recognize all instances of an object, even within the same class. For instance, when developing a ML model specifically for close-up stop signs, it might be valuable to consider whether to annotate distant stop signs. Annotating every stop sign would train the ML model to recognize all stop signs, regardless of their distance. On the other hand, choosing not to annotate distant objects could emphasize the importance of proximity, though this approach may have its own drawbacks such as introducing false negatives, as mentioned previously.

3.2.4 *Tools for data annotation*

There are several tools for data annotation. These range from cloud based services to locally hosted ones and paid or open source. In this section, we will mention some of the tools that we have used and found accessible.

```
{
  "version": "5.4.1",
  "flags": {},
  "shapes": [
    {
      "label": "cup",
      "points": [
        [
          368.0952380952381,
          178.88888888888889
        ],
        [
          779.2063492063493,
          675.7142857142857
        ]
      ],
      "group_id": null,
      "description": "",
      "shape_type": "rectangle",
      "flags": {},
      "mask": null
    }
  ],
  "imagePath": "../photo_2023-07-06 14.58.47.jpeg",
  "imageData": "/9j/4AAQSkZJRgABAQAAAQABAAD/
```


Figure 4: Example of the LabelMe annotation format for an image containing a single rectangle.

LabelMe

In 2007, MIT released a dataset alongside a data annotation tool called *LabelMe*. This tool was designed to be web-based, allowing a large number of users to access and annotate the material. The emphasis was on simplicity and ease of use, with the added benefit of being open-source (Russell et al. 2008). Although access to the original web-based platform has since been closed, as it was specifically developed for annotating their dataset, the popularity of LabelMe led to the creation of a new open-source version. This updated version can be installed on most common operating systems and provides an interface for image annotation using polygons, rectangles, circles, lines, points, line strips, and more[6].

Annotations generated with this tool are typically stored on the same computer where the tool is run and follow a specific annotation format. LabelMe produces a JSON file for each image, which includes key information for each shape, such as labels and points (for example, the top-left and bottom-right corners of a rectangle, see Figure 4). Thus, when using LabelMe for annotation, a set of JSON files —one per image— with annotations in the LabelMe format will be generated. Then, converting this format into one that better suits the requirements of the ML model intended to be used

6 https://github.com/wkentaro/labelme

may be needed. An example of how to perform this conversion is provided the Hands-on Practical Introduction section.

The biggest advantage we have found with this tool are that it is open-source and hosted on a local server, providing a secure way to annotate sensitive data. However, one disadvantage is that it is a very simple and lightweight tool, limited to certain types of image data annotation and not suitable for other data types. Moreover, this tool does not facilitate easy collaboration, and it is difficult to track what has been annotated and what has not; thus, it is recommended for small datasets where one main person is performing the annotation. Finally, while this tool is simple, its interface may not be the most user-friendly for all users.

Roboflow

Roboflow[7] is another image annotation tool primarily focused on computer vision. It provides a platform where users can easily train ML models, although with limited customisation options. Further, it can be used for tasks such as segmentation, image captioning, and keypoint detection, among others. It offers a web-based free version, as well as premium versions that include features like access to premium GPUs for training, additional credits for ML model training and inference, and, most importantly, private datasets and trained ML models. This tool supports working with images in the most common formats, as well as videos. It also allows users to upload annotations generated with other tools. Roboflow offers various output formats, which can be further specified according to the ML model version, such as YOLO and COCO.

Roboflow has two main advantages. First, its simplicity and user-friendly interface make it easy to navigate, which is especially suitable for beginners in data annotation. Second, the ability to train ML models on the same platform is valuable for testing purposes. The main disadvantage of Roboflow is that, in the free version, datasets and ML models created on the platform are open to the community, making it unsuitable for sensitive data or data with copyright concerns.

Label Studio

Label Studio is a versatile data labelling and annotation tool that supports multiple projects and users. It offers an open-source version that can be hosted on a local computer, as well as premium versions that provide additional

7 https://roboflow.com/

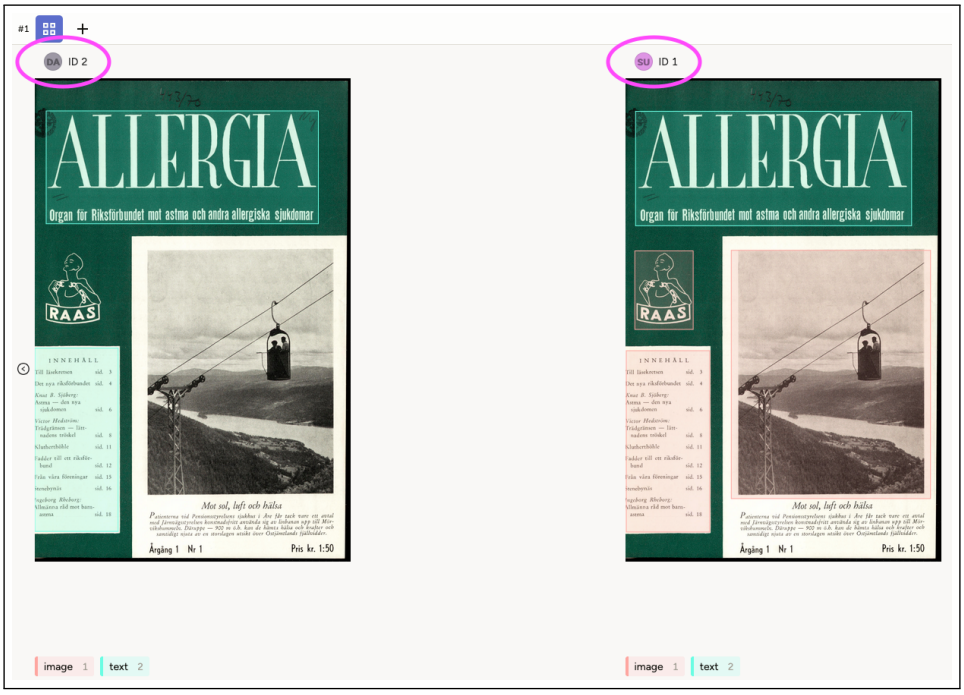


Figure 5: Example of collaborative annotation in Label Studio. The pink ovals highlight two different collaborators who have annotated separate copies of the same image, producing distinct annotation versions for comparison.

features such as cloud-based usage and data storage, user management, activity logs, and AI-assisted labelling, among other capabilities.[8] Label Studio focuses on three main areas: simplicity, configuration, and integration. The tool features a user-friendly interface that is easy to navigate and set up. Additionally, users can configure it to suit their specific annotation needs, styles, and data types. This flexibility is made possible by its multi-type data capabilities, which allow users to annotate, for example, both images and text simultaneously. Finally, Label Studio can be integrated with a variety of ML frameworks and models (Medium 2020).

Based on our experience, some key advantages of this tool include: (1) a highly capable open-source option that allows multiple users to collaborate and annotate the same dataset. This can be done in two ways: either in parallel, where users add annotations to the same file without seeing who created each annotation, or by annotating separate copies of the same file, generating multiple versions of the annotation. The latter approach, for

8 https://docs.humansignal.com/guide/get_started

example, enables comparison to detect discrepancies in labels (see Figure 5); (2) the ability to host the tool locally, ensuring that both the service and data remain on a local computer, shared only through a secure protocol; and (3) support for annotating data in various formats beyond traditional ones, enabling tasks such as visual question answering, image captioning, text summarisation, speaker segmentation, response generation, and more—all within a simple and user-friendly interface. Regarding output formats, Label Studio allows users to export annotations in different formats, including COCO, and YOLO.

3.2.5 *Getting the data ready for training*

Proper data organisation and processing are essential, though sometimes cumbersome, steps in the machine learning pipeline. This process typically involves modifying the dataset and splitting it into three subsets: training, validation, and testing. In some cases, only two splits—training and testing—are used. The choice of splits and data processing if any is ultimately up to the engineer responsible for training the machine learning model.

The *training dataset*, which contains the largest number of images among the three subsets, is used by the ML model to learn features and patterns. This dataset plays a pivotal role in the training process as it provides the raw data from which the ML model extracts and learns to recognise various features (Goodfellow 2016: 129). To enhance the effectiveness of this learning process, various transformations, also know as *data augmentation*, can be applied to the training dataset, including operations such as rotation, scaling, cropping, and colour adjustments (Mikołajczyk & Grochowski 2018, Shorten & Khoshgoftaar 2019). By introducing such variations, the dataset is artificially increased in size and diversity, helping the ML model to improve its ability to generalise to new, unseen data, and preventing it from learning the data too wall (this phenomenon is also known as *overfitting*). This approach ensures that the ML model does not become overly specialised to the specific examples it was trained on but rather learns to recognise patterns and features that are more broadly applicable (Obeso et al. 2017, Bahrami & Albadvi 2024). Additionally, these transformations can be used to ensure other important aspects of the training data, such as balance and representativeness. Data augmentation can expand certain classes, potentially preventing biased results (Pastaltzidis et al. 2022, Zhang et al. 2022). The *validation dataset* is used to evaluate the ML model's performance during training (Goodfellow 2016: 129). Finally, the *test dataset* is employed to evaluate the fully trained ML model "in the wild" using images the ML model has never encountered before (Google n.d., Xie et al. 2011), which is a crucial step for assessing how

the ML model will perform on new, unseen data. Usually, the ML model is evaluated on the validation data several times before it is evaluated on the test data.

As previously discussed, the choice of a ML model, which is intrinsically linked to the annotation format, along with the tool employed for data annotation, plays a critical role in how data must be organised for training. The choice of ML model is often dictated by the specific requirements of the task, which in turn determines the appropriate annotation format and the necessary tools for data preparation. For instance, the YOLO family of ML models (see Section 3.3.2 for more details) requires the YOLO annotation format, a specific file structure, and a .YAML configuration file. To prepare data for input into a YOLO ML model, it needs to be divided into three directories—train, validation, and test—each containing two subdirectories: images and labels. A configuration file is also required to specify the locations of these subdirectories, as well as the number and names of the classes (see Figure 2). Another example is the Layout Parser family of ML models (see Section 3.3.2 for more details), which utilises the COCO annotation format and a specific file structure during the training process. In this case, the data must be organised into three directories—train, validation, and test—with each containing a JSON file that includes annotations in COCO format.

The process of preparing and organising data for training can be cumbersome, which underscores the importance of the data annotation tool. In Roboflow, for example, a dataset can be exported in a format that is ready for training. This includes automatically dividing the data into the required file structure for popular ML model families and generating any necessary configuration files. In contrast, tools like Label Studio and Labelme do not provide automated file organization capabilities, requiring this step to be performed manually or through alternative methods. Manual data organization can be error-prone and time-consuming, requiring careful attention to file structure and format to ensure compatibility with the model. This can introduce additional complexity, especially for large datasets or when working with multiple ML models.

When discussing data splitting, creating the training, validation, and testing subsets is a process in itself. The method described in this section for splitting the data is known as *hold-out cross-validation* and has become popular due to its simplicity and effectiveness (Reitermanova 2010). Further, the specific ratio in which to split the data can heavily affect a ML model's performance. For this reason, alternative methods have been proposed in the literature that depend on the type and quantity of data at hand (Reitermanova 2010, Xu & Goodacre 2018).

In conclusion, the process of data preparation, including careful consid-

eration of splitting, annotation formats, and tools, is vital for successful ML model training. Choosing the right methods and tools as well as documenting these choices can streamline this process, reduce potential errors, and improve the overall performance of the ML model.

3.3 *Model training*

3.3.1 *What "training a model" really means*

In the preceding sections, the concept of training a machine learning model has been mentioned multiple times. It is therefore pertinent to examine what this process entails. However, before embarking on this, it is relevant to consider two related concepts: *mathematical model* and *machine learning model*. A mathematical model is an abstract mathematical representation of a system's dynamics (Bender 2000). Such models can be derived using differential equations, graphs theory, boolean relations, among other methods (Bunge & Bunge 1973, Pavlopoulos et al. 2011, Wang et al. 2012). In contrast, a machine learning model is a type of mathematical model that integrates this mathematical framework with an algorithm designed to identify patterns in a training dataset. These patterns are then encoded into the mathematical model's parameters. In the upcoming sections, the terms model and machine learning model will be used indistinguishable to refer to machine learning models.

Training a machine learning model involves feeding it a training dataset, performing computations, and adjusting the model's parameters to better fit the data. During this training process, a validation dataset is used to assess the model's performance (Goodfellow 2016: 110). This iterative process continues until a predefined stopping criterion is met. There are two main criteria for stopping model training: a fixed number of iterations, known as *epochs*, or the use of *early stopping*. In the first approach, a predetermined number of epochs, such as 300, is specified. This means that the process of learning from the entire dataset, adjusting the parameters, and evaluating on the validation dataset is repeated for, for example, 300 times. While this method is straightforward, it is not highly effective, as the model may overfit to the training dataset. To address this issue, an alternative approach known as early stopping has been proposed. In early stopping, the learning process is terminated when the model's parameters show no improvement, based on the validation metric, for a pre-specified number of iterations (Goodfellow 2016: 238).

A model can be trained in two ways: *from scratch* or using *transfer learning*. A model is said to be trained from scratch when its parameters, before learn-

ing anything from the training data, are initialised randomly. In contrast, transfer learning involves using a pre-trained network that has been fully trained on a different dataset, similar to the target dataset, and task. The parameters learned from this initial task can then be repurposed or transferred to a second network to be trained on the target dataset and task (Yosinski et al. 2014). In this context, all the parameters learned from the initial task can be used to initialise the new machine learning model, or only a subset of them may be utilised.

3.3.2 *Common machine learning models and tools in document digitalisation*

In document digitalisation, the choice of machine learning models varies depending on the task, and the assesment of which model is best is primaritly done by the engineer. In the context of digitalising large-scale printed collections, object detection, together with layout analysis, and optical character recognition (tasks in the realm of computer vision) are often involved (Aangenendt et al. 2024, Gruber et al. 2020, Matschak et al. 2022). Moreover, these models can be used in a chain to obtain and analyse different components in an image. In section 4.1, an example of how such models are used in a digitalisation project is presented.

Object detection models

Object detection is fundamental to understanding the structure and layout of a document. Machine learning models for object detection can be trained to locate and classify specific elements within a page, such as text blocks, tables, and figures. These models can be further refined for more specific tasks, such as distinguishing between photographs and advertisements or separating body text from titles. Currently, several pre-trained machine learning models for object detection are openly available, each offering varying levels of performance depending on the task. These models do not need to be trained from scratch, as transfer learning can be applied. Furthermore, these convolutional neural network-based models can be modified and combined with others to enhance functionality.

According to the HuggingFace leaderboard, some of the best-performing models or model families are (Rafael Padilla & the Hugging Face Team 2023):

- The YOLO Family: This family of models, developed by Redmon et al. (2016), is designed for real-time object detection and include a series of models from YOLOv2 to YOLOv11. These models have been proven to be both fast and accurate (Kaur & Singh 2023) and have been applied

in various tasks, such as automatic person detection in thermal images (Krišto et al. 2020) and layout analysis (Naiman et al. 2023).

- Deformable DETR (Deformable detection transformer): DETR, introduced by Zhu et al. (2020), is a recent object detection model that eliminates the need for anchor boxes or points (Zou et al. 2023). As a transformer-based model, it requires different data annotation and training processes compared to traditional object detection models. Deformable DETR builds on DETR, offering improvements in speed and performance, particularly for detecting small objects (Zhu et al. 2020).

In relation to the content of this chapter, as mentioned earlier, object detection models can also be employed to learn page layouts. For instance, by training models on labeled page layouts, the system can infer and categorise different structural elements across various document types. This is particularly useful when working with historical documents, which often exhibit complex and inconsistent layouts.

Layout analysis toolkits

Layout ML models are a specialized subset of object detection models that focus on learning the arrangement of objects on a page. One such toolkit is Layout Parser, which simplifies the application of layout detection models. It includes a model zoo with pre-trained models, ssuch as frcnn and mask-rcnn networks trained on PubLayNet (Zhong et al. 2019), a modern scientific documents dataset and newspaper navigator (Lee et al. 2020), a U.S. newspaper dataset from the 20th century, among others (Shen et al. 2021).

Optical character recognition toolkits

Another crucial aspect of document digitalisation is converting scanned images of text into machine-readable form, which is achieved through optical character recognition (OCR) models. Some of the most popular OCR models include:

- EasyOCR: It is known for providing cost-efficient results while maintaining a competitive accuracy (Roboflow 2024a). EasyOCR is a lightweight and user-friendly OCR model that supports over 80 languages. It excels in handling complex documents and works well with various fonts

and handwriting, making it a versatile tool for a wide range of OCR tasks[9].

- Tesseract: Developed by Google, Tesseract is an open-source OCR engine extensively used for converting printed text into digital form. It supports up to 127 languages, depending on the version, and can process various text formats (e.g., single or multiple columns) and fonts. A list of supported fonts is available in the tool's documentation[10]). Tesseract is particularly popular for its integration with different tools and its ability to be customized for specific tasks (Smith 2007).

3.4 *Model deployment*

Once a model (or a series of models) is fully trained, the next step to make predictions or extract results on the complete dataset; this process is also called *inference*. This step involves extracting metadata from the objects of interest and text from digital images, if an OCR tool was utilised. The results can be stores in files, such as one file per image processed, for future export to a database. The output files may contain different attributes depending on the model's objective. For instance, a metadata file might include information about the locations of titles, images, subtitles, and other elements of a digital image of a newspaper page. These files are typically stored in formats like JSON or XML, which allow for easy integration with databases and other systems. Thus, these files can be stored in a database, which, in turn, can be connected to a web portal with a searchable system [11].

4 *A use case scenario*

4.1 *Introduction to SweMPer*

The Swedish Medical Periodicals (SweMPer) project aims to digitize, curate, and make accessible a vast collection of printed Swedish medical journals spanning over two centuries. These periodicals, which have been crucial in documenting medical discourse from the 18th century to the present day, provide an invaluable resource for historians, researchers, and the general public. By the end of the project around 550 000 pages (28 titles, ca 1100 physical volumes) will have been scanned, digitalised and collected in a

9 https://github.com/JaidedAI/EasyOCR
10 https://github.com/tesseract-ocr/tesseract/blob/4.1/src/training/language-specific.sh
11 See for example https://books.google.com; see also, https://www.hathitrust.org/.

database. This data will be made available through a web interface (The Swedish Medical History Portal), also under development by the project. However, the complexity of these historical documents ranging from varied layouts and formats to different languages and degraded print quality poses significant challenges for digital processing and analysis. While similar difficulties exist in other large-scale digitization efforts, historical medical periodicals present additional hurdles due to their specialized terminology, evolving typographic conventions, and domain-specific elements, making automated processing more challenging. To address these obstacles, the SweMPer project employs advanced machine learning and computer vision techniques, with a particular focus on object detection, and optical character recognition (OCR).

The primary objective of SweMPer is to create a comprehensive digital archive of these journals, enabling researchers to access and study them in a structured and searchable format. To allow for long-term historical perspectives, the SweMPer-database includes medical journals published in Sweden in the period 1781–2011. The starting point is the year when the earliest periodical was first published; the end point marks the year when the availability of existing digital editions makes further digitisation redundant. During this long era of emerging and changing media and technologies, including print, photography and digital media, scientific journals became an inextricable part of medical culture (Podolsky et al. 2012) and evolved into an extensive knowledge infrastructure that has enabled and redistributed medical authority, influence and power across different media.

In Sweden and elsewhere the earliest collegial publications, produced by and for doctors, were soon followed by commercial periodicals more broadly aimed at communicating medical and pharmaceutical knowledge to the public, nationally as well as locally. By the 1930s self-governed patient organisations began publishing independent journals, enabling patients and laymen to collect their own data and articulate knowledge claims, while making demands on the professional agents of medical care. Medical students also started to issue journals that from the 1960s increasingly participated in the political turmoil of the day.

Thus, to ensure broad representativity, the full SweMPer-database includes the major Swedish journals produced by doctors, public health and health care professionals, by student and patient organisations, as well as publishers in the commercial market. This enables study of themes ranging from professional debates to popular discourse and the experiences of patients.

Crucially, the journals included in SweMPer are heterogeneous in terms of content and genres, including editorials, professional debates, readers'

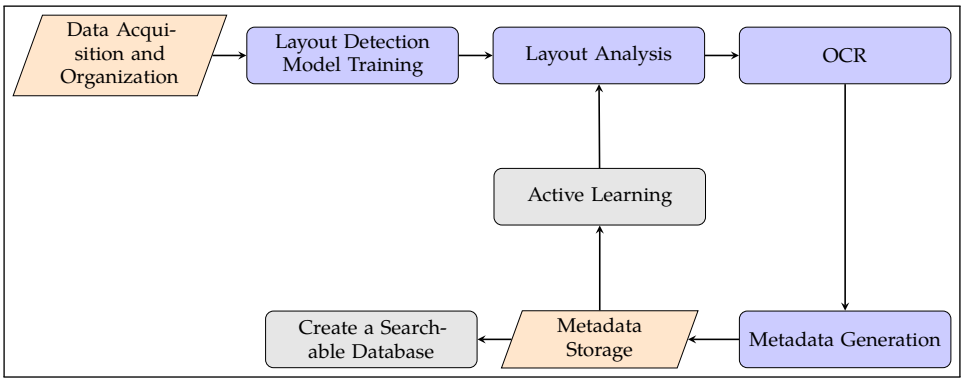


Figure 6: SweMPer pipeline overview

letters and advertisements for drugs and treatments. They also contain large amounts of non-textual media such as photographs, tables, x-ray images and illustrations, mirroring the fact that medical history is as richly visual as it is textual. While this variety is challenging in terms of automated metadata extraction, it also means that the finished SweMPer-database will support not only rich textual but also visual analyses of medical history.

By automating the extraction of textual and visual content from these documents, SweMPer not only preserves important historical records but also enhances their accessibility and usability for a wider audience. This use case scenario outlines the development, implementation, and outcomes of the current SweMPer digitisation pipeline.

4.2 *SweMPer pipeline overview*

The SweMPer pipeline is designed to handle the entire process of digitizing historical documents, from image acquisition to the creation of a searchable platform to access the images and metadata generated in the SweMPer project. The pipeline begins with the collection of high-resolution images of the journals, which are then processed using machine learning models for layout detection and OCR tools, and create a searchable platform. Consequently, the extracted data is organised and stored, ensuring that each document is easily retrievable and its content is searchable.

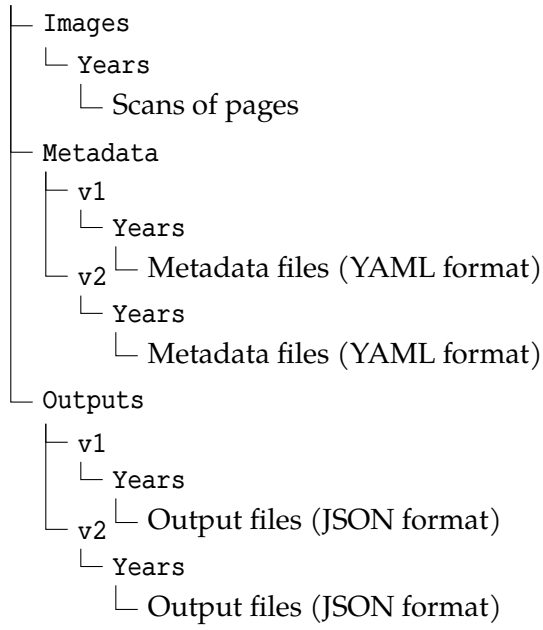
A visual abstract of the pipeline in Figure 6 illustrates its key components and the flow of data through the system. The components in gray will be implemented in the future. A detailed explanation of each part in the pipeline is presented in the following sections.

4.3 *Explanation of pipeline components*

4.3.1 *Data acquisition and organization*

The first step in the SweMPer pipeline is the acquisition of images from the historical journals. The data for this step was obtained by scanning physical copies of journals in-house at the Center for Digital Humanities and Social Sciences at Uppsala University, and from the Uppsala University Library. The scans are stored in .jpg format and organized into a structured folder hierarchy to facilitate efficient processing in later stages.

Before continuing with any other processing, a plan for organising the data, intermediate and final results was also necessary. A thorough organisation not only simplifies the storage and retrieval of images, metadata, and output files, but also plays a critical role in the metadata extraction process in later stages. By embedding key information into the filenames and organising the scans and associated files in a structured manner, it is ensured that metadata and layout data can be accurately and efficiently extracted and used during subsequent processing. This level of detail is particularly important when working with large datasets where manual intervention would be impractical. In this manner, the folder structure was designed as follows:



In this structure:

- `Images` directory: Contains scans of historical journal pages, organized by year, with each scan renamed to include essential metadata such as the journal name, year, volume, and scan number. For example:

  `journalname_year_numberofthescan.jpg`

  Additional information such as the volume and issue numbers may be included depending on availability, for example:

  `status_1947_vol010_nr006_0017.jpg`

- `Metadata` directory: This directory holds metadata files generated during the layout analysis and processing stages. Each version of the metadata extraction is stored in separate folders (e.g., `v1`, `v2`), representing different iterations or improvements in the metadata extraction process. The files are organized by year and named according to the corresponding image, with the same naming convention as the images but in YAML format:

  `status_1947_vol010_nr006_0017.yaml`

- `Outputs` directory: This directory stores the results of the layout detection inference, with the output files saved in JSON format. As with the metadata directory, the output files are organized by version (e.g., v1, v2) and year, with each file representing the layout detected in the corresponding image:

  `status_1947_vol010_nr006_0017_layout.json`

Each scan and its associated metadata and output files are named consistently, allowing for easy cross-referencing between the image files, the extracted metadata, and the layout detection results. For example:

- A typical file in the `Images` directory might be named: `allergia_1957_0054.jpg`.
- If the volume and issue numbers are available, the filename might be: `status_1947_vol010_nr006_0017.jpg`.
- The corresponding metadata file: `status_1947_vol010_nr006_0017.yaml`.
- The corresponding output file: `status_1947_vol010_nr006_0017_layout.json`.

4.3.2 *Layout detection model training*

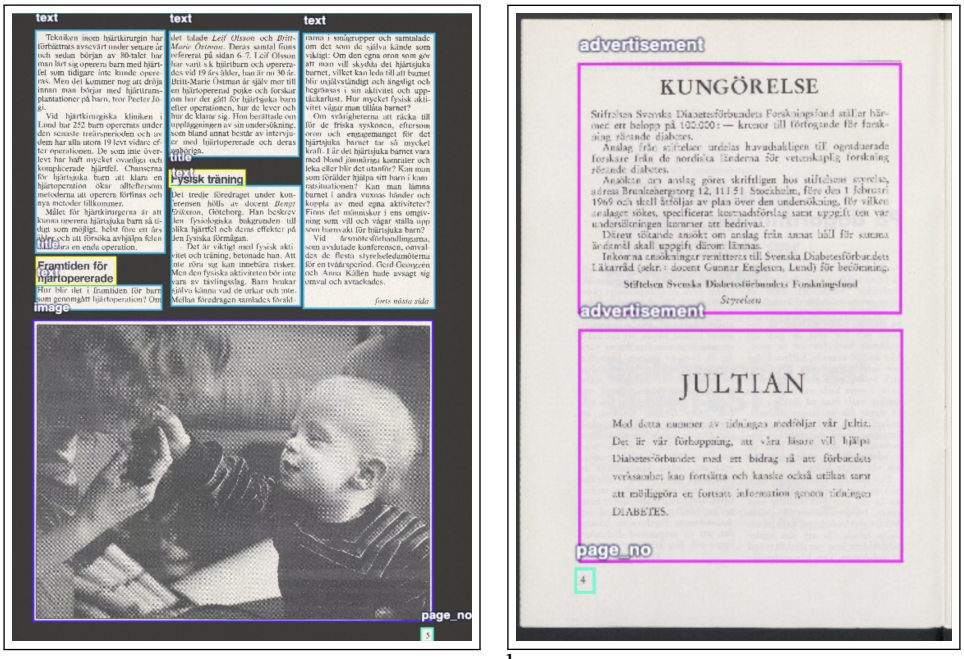
After obtaining and organising the scans of each journal, the next step in the SweMPer pipeline involves detecting the layout of historical journal pages. Detecting layout elements such as authors, tables, lists, and titles is crucial because it allows us to create a structured, searchable database from the digitized journals. This structure enables users to easily locate specific sections, such as an article's title or a table of medical data, thus improving both the research experience and the accessibility of the documents. To do this, we first selected a subset of data for manual data annotation, followed by model training using the annotated data.

Data Selection and Annotation: For data selection, at first, we randomly selected an equal number of images from all the available journals expecting to obtain a balanced dataset and store the details like filenames, which journal they come from, etc., to track which images are selected for training. This helps us to avoid selecting duplicate images for training. At this step, we had a total of 3351 images, of which 2701 images were used for training and 650 images were reserved for testing. The chosen images were manually annotated using the Roboflow platform. This annotation provided bounding box information for each layout element. After the annotation process, we can easily export the annotations in the required format for model training. Some of the examples can be seen in Figure 7

Moreover, we chose to have the following classes which will allow us to maximise the metadata collection. These classes capture a wide range of elements commonly found in historical documents, maximizing the usability of the metadata for future analysis and search functionality:

- advertisement
- author
- header_or_footer
- image
- list
- page_no
- table
- text
- title

During the initial phase of annotation and data selection, we observed a significant underrepresentation of certain layout classes, such as tables and lists. This means that these types of layout elements appeared much less frequently compared to other classes, such as text or images, in the annotated

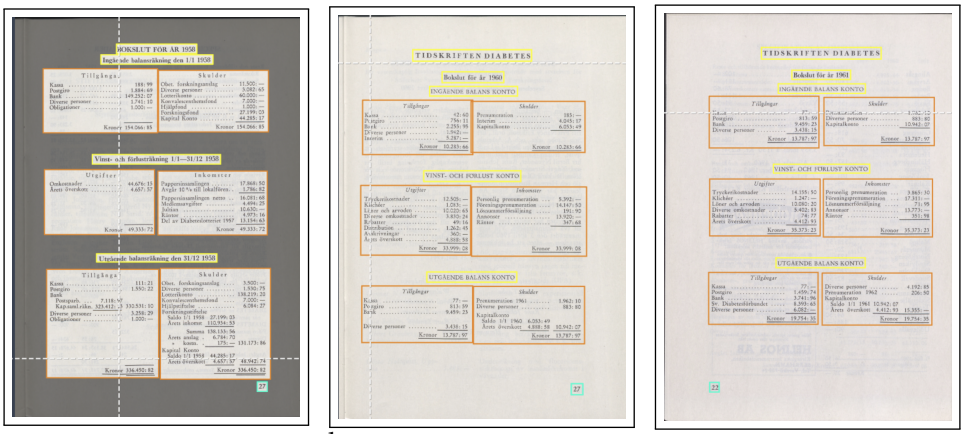


a. b.

Figure 7: Examples of annotations for the model training process.

data. This underrepresentation posed a challenge for training a robust model, as having too few examples of certain classes can limit the model's ability to learn and accurately predict those categories. As a result, we needed to take additional steps to balance the dataset to ensure that the model could learn to accurately detect these underrepresented elements. To overcome this imbalance, we employed a emperical approach using image feature vectors, which are numerical representations of the essential features of an image. These feature vectors encapsulate key visual information, such as shapes, textures, and patterns. By leveraging these feature vectors, we were able to identify images in the dataset that were visually similar to those containing the underrepresented classes. This allowed us to increase the presence of these underrepresented classes in our training data and improve the model's performance.

Using ResNet18 for Feature Extraction: To efficiently select more images with similar layouts (containing tables and lists), we used a ResNet18 (He et al. 2016) model as a feature extractor. ResNet18 is a pre-trained convolutional neural network (CNN) that is widely used for feature extraction due to its efficient architecture. The model generates a 512-dimensional vector (a



a. b. c.

Figure 8: Examples of selected images from the SweMPer dataset, which contain the underrepresented class of tables/lists. After calculating cosine similarity between the annotated images with underrepresented classes and the rest of the dataset, visually similar images were selected and added to the training set. These images helped increase the representation of tables in the training data and improving the model's ability to generalize across different layouts. Images (b) and (c) are examples of similar images retrieved using cosine similarity with (a).

feature vector) for each image, which essentially acts as a compact representation of that image's visual content. The 512-dimensional vectors that ResNet18 produces are rich enough to capture meaningful features of the images without being computationally expensive. These feature vectors can then be used to measure similarity between images.

Cosine Similarity for Similarity Search: After obtaining the 512-dimensional feature vectors for all images in the dataset, we needed a way to find images that were visually similar to those containing underrepresented classes (tables and lists). Examples of these can be seen in Figure 8. To do this, we used cosine similarity, which measures the cosine of the angle between two vectors in a high-dimensional space. Cosine similarity is a widely-used metric for comparing vectors because two images with similar visual content will have vectors pointing in similar directions, resulting in a high cosine similarity score (closer to 1).

Finding Similar Images to Expand Training Data: We focused on the images in the annotated set that contained the underrepresented classes. For each of these images, we extracted their 512-dimensional feature vectors using ResNet18. Then, we computed the cosine similarity between the selected

images (with underrepresented classes) and all other images in the dataset after which, for each image, we sorted all the similarity scores and selected the top 3 most similar images (excluding the image itself) based on this score. This process allowed us to find additional images that were visually similar to those containing the underrepresented classes. These extra images were then added to the training set, boosting the representation of those classes. By adding these similar images to our dataset, we were able to increase the total number of images in the dataset from 3351 to 3506 (2805 for training). This enriched dataset was more balanced, ensuring that the model would be exposed to a wider variety of tables and lists during training.

Selecting the Layout Detection Model: Once the data has been annotated and prepared, the next step in the SweMPer pipeline is training the machine learning model for layout detection. This process requires careful selection of the model, training data, and hyperparameters used during training. For layout detection, we experimented with two different models: the LayoutParser (Shen et al. 2021)'s Mask R-CNN model (He et al. 2017) (pre-trained on the PubLayNet dataset (Zhong et al. 2019)) and YOLO-NAS (AI 2023) (a more recent object detection model). The goal was to identify a model that not only provided high accuracy but also performed well on the specific task of detecting various layout elements in historical documents.

LayoutParser's Mask R-CNN: This model uses the Mask R-CNN architecture, which is well-regarded for its ability to perform both object detection (bounding boxes) and instance segmentation (pixel-level masks). LayoutParser provides pre-trained models specifically tailored for document analysis tasks, with the PubLayNet dataset serving as its training base. PubLayNet consists of over 360,000 document pages, making it highly suitable for tasks like detecting layout elements in historical journals. Mask R-CNN requires annotations in the COCO format.

YOLO-NAS: YOLO-NAS is a more recent advancement in the YOLO family of models, known for its speed, accuracy, and architectural improvements that leverage neural architecture search (NAS). YOLO-NAS is optimized for object detection and provides state-of-the-art performance in terms of both speed and accuracy. YOLO-NAS excels at detecting objects quickly, making it ideal for certain real-time layout analysis tasks. YOLO-NAS, like other YOLO models, uses a different annotation format. Each image is accompanied by a separate TXT file for its annotations called the YOLO format.

Early in the training process, we conducted test training runs on a subset of the dataset to evaluate the performance of both the Mask R-CNN model

and YOLO-NAS. During these tests, we closely monitored key performance metrics such as mean Average Precision (mAP), as well as the quality of the model's outputs, especially in complex layout scenarios. After manually inspecting the outputs of both models, we found that LayoutParser's Mask R-CNN model provided more robust inferences for our dataset, even though it is based on an older architecture. Despite the comparable mAP scores (around 90% for both models), the Mask R-CNN model demonstrated superior and more robust performance when it came to detecting the more complex documents within the SweMPer dataset. Thus, although YOLO-NAS is faster and more modern, we decided to proceed with LayoutParser's Mask R-CNN for its superior performance in our specific task.

Segmentation masks: Bounding boxes are a simple but effective way of representing object locations within an image, as they enclose each object with a rectangle. For layout detection, bounding boxes were used to annotate various elements on the page. However, since we chose the Mask R-CNN model, which requires segmentation masks instead of bounding boxes, we had to convert the bounding box annotations into segmentation masks. Segmentation masks provide a pixel-level representation of objects, making them more precise than bounding boxes, which are limited to rectangular shapes. To handle this, we used a script to convert the bounding boxes into pixel masks, ensuring the annotations were suitable for the Mask R-CNN model.

Transfer Learning and Fine-Tuning: Instead of training the Mask R-CNN model (He et al. 2017) from scratch, we used transfer learning to fine-tune a pre-trained model on our annotated dataset. Transfer learning is particularly useful when dealing with limited datasets because it allows us to leverage a model that has already learned a wide range of features from a large, general dataset and fine-tune it to perform well on a specific task. Instead of training a model from scratch, transfer learning allows us to build on an already powerful feature extractor.

In our case, the LayoutParser Mask R-CNN model was pre-trained on the PubLayNet dataset, which contains over 360,000 document pages. This pre-training gave the model a strong foundation for understanding document layouts, making it well-suited for our task of layout detection in historical Swedish medical journals. The classes in the PubLayNet dataset are: text, title, list, table, figure. The variant of the model used is the Mask R-CNN X-101-32x8d-FPN-3x Network.

The Mask R-CNN X-101-32x8d-FPN-3x network is a highly versatile model for both object detection and instance segmentation, composed of several key components:

- **Backbone** (**ResNeXt-101, Xie et al. 2017**): This extracts the main features from images.
- **FPN** (**Feature Pyramid Network**): Helps process images at different scales.
- **RPN** (**Region Proposal Network**): Suggests regions in the image that might contain objects.
- **RoI Align** (**Region of Interest Align**): Warps these regions of interest into a uniform size for further processing.
- **Bounding Box and Mask Heads**: Identifies the class of each object and draws bounding boxes around them, while also generating detailed masks.

When refining a pre-trained model, we freeze the early layers so they are not retrained. These layers focus on general features, like edges and textures, which are common across many types of images. For our model, we tested freezing 2, 3, and 4 sections and found that freezing the first 3 sections resulted in the best fine-tuning performance. This choice provided a balance between retaining generalizable features learned from PubLayNet and allowing the later layers to adapt to the more specific elements in our dataset.

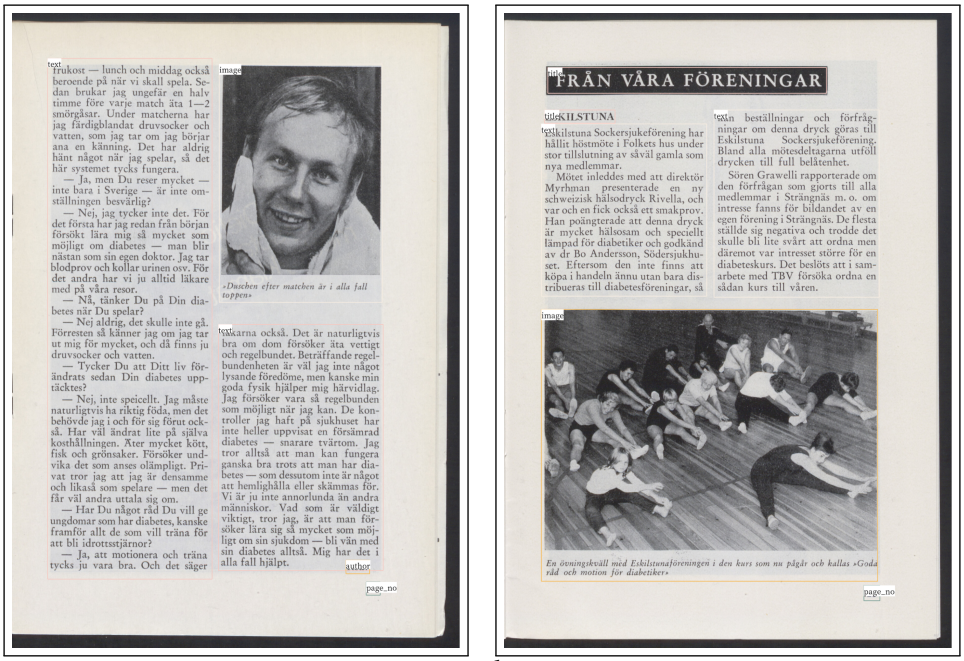
Through this approach, we maximize the benefit of transfer learning by maintaining the general document layout features learned from PubLayNet, while adapting the model to the specific challenges of historical Swedish medical journals. This ensures that the model performs well on layout detection tasks, as seen in Figure 9, and produces high-quality results in later steps, such as metadata extraction.

4.3.3 *Layout analysis*

Once the layout detection model has been trained, the next step in the SweMPer pipeline is layout analysis using the transfer learned model from above. The goal of layout analysis is to identify and classify different layout elements in historical journals. The images in Figure 10 shows the inference results using our trained layout detection model.

The table from Figure 10 contains information about the bounding boxes detected by the layout detection model:

- **x_1, y_1**: The top-left corner of the bounding box.
- **x_2, y_2**: The bottom-right corner of the bounding box.
- **block_type**: Describes the geometric shape of the layout element.



a. b.

Figure 9: Examples of layout detection results generated by our transfer learned LayoutParser model. The model accurately identifies and labels various layout elements such as “text”, “image”, “author”, “title”, and “page number”, preparing them for further steps like text extraction using OCR.

	x_1	y_1	x_2	y_2	block_type	type	score
0	144.569962	184.140015	811.390625	2298.991943	rectangle	text	0.999926
1	837.126099	1264.826416	1505.461792	2263.205811	rectangle	text	0.999638
2	840.967163	207.490494	1501.823608	1174.445557	rectangle	image	0.998965
3	1354.159546	2221.210449	1450.243896	2276.750732	rectangle	author	0.993870
4	1436.580078	2310.358398	1493.492188	2363.374756	rectangle	page_no	0.987107

Figure 10: An example output from the layout analysis model in a tabular format.

- **type**: The predicted class of the detected layout element.
- **score**: The confidence score of the predicted class, indicating how certain the model is in its classification.

This structured output provides an understanding of the layout elements present on the page. For instance, the first row in the dataframe indicates that a text block is detected within the coordinates defined by x_1, y_1, x_2, and

y_2 with a confidence score of 0.999926, meaning the model is almost certain of its prediction. By capturing these layout elements as bounding boxes, we can efficiently map the page structure and prepare it for subsequent steps, such as OCR and metadata extraction.

4.3.4 *OCR*

After the layout analysis step, the next task in the pipeline is to extract the textual content from the identified layout elements. For this, we use Optical Character Recognition (OCR) with tesseract (Smith 2007) to transform the visual information into machine-readable text. Our OCR process is efficiently integrated with the results from the layout detection model, ensuring that we only perform OCR on relevant portions of the page.

The key steps in this process are as follows:

1. **Filtering the Output by Element Type**: Once we obtain the bounding boxes and class predictions from the LayoutParser model, we filter the output DataFrame 10 to retain only the layout elements that contain textual content. Specifically, we focus on the following classes:

   - `text`
   - `title`
   - `page_no`
   - `header_or_footer`
   - `author`

2. **Cropping the Image Sections**: For each identified layout element in the filtered DataFrame, we crop the corresponding region of the original image. The bounding box coordinates (`x_1, y_1, x_2, y_2`) provided by the layout model are used to isolate the relevant portion of the page that contains text.

3. **Running Tesseract OCR**: The cropped image sections are then passed through the Swediah version of Tesseract OCR engine. Tesseract is configured to work with the Swedish language (`--lang swe`). This configuration allows Tesseract to accurately recognize the text in these cropped regions.

4. **Text Extraction**: Tesseract processes each cropped section and returns the corresponding text for each layout element. The extracted text is then stored along with the bounding box coordinates and the type of the layout element in the output dataframe, facilitating further analysis and metadata generation.

4.3.5 *Metadata generation*

In the SweMPer project, generating metadata for each processed page is a critical step that consolidates information from multiple stages of the pipeline, including layout detection, OCR, and the inherent metadata embedded within the image filenames. The goal of metadata generation is to create a structured YAML file that captures essential details about each page, enabling the creation of a comprehensive, searchable database of journal content.

Once the text elements have been extracted through OCR and combined with the layout model's output, we proceed to generate metadata in a structured YAML format. This metadata contains key information such as:

1. **File Information**: Metadata about the file itself, including its name, size, creation date, and MD5 checksum. This helps maintain data integrity and track the origin of each file.

2. **Document-Specific Information**: Information from the file names is parsed and used to fill in fields like the journal name, year of publication, volume number, and issue number.

3. **Layout Elements**: Detected layout elements such as text blocks, tables, images, page numbers, titles, etc., are described with their bounding box coordinates, ocr content (for text elements), and confidence scores.

This structured metadata not only allows for easy indexing and retrieval of journal pages but also facilitates more advanced queries. For example, users could search for pages with a specific title, or for advertisements published in a particular year. Furthermore, by capturing the OCR text within the metadata, it becomes possible to perform full-text searches across the entire corpus, making the journal collection an invaluable resource for researchers and historians.

The metadata is saved as a YAML file (as seen in Figure 11), which is a human-readable format that also supports complex data structures. This makes it easy to parse programmatically for building databases or other digital collections. With such metadata in place, a well-structured, searchable digital library can be created, offering powerful tools for exploring and analyzing Swedish medical periodicals over time.

```
Swemper-page-descriptor:
  ContainerVolumeFile: '"Diabetes_1963_vol013_nr003_0012.yaml"'
  CreationDate: '"2024-08-01"'
  DocumentTopic: '"Diabetes Research"'
  FileSize: '"0.84MB"'
  FileType: '"JPEG"'
  IdentifiedLayoutElements:
    Advertisements:
    - BoundingBox:
      - 147.48545837402344
      - 1315.8365478515625
      - 1346.800552368164
      - 972.1468505859375
      Content: ''
      Score: '"0.9977514147758484"'
      Type: '"advertisement"'
    AuthorNames: []
    HeaderOrFooters: []
    Images: []
    Lists: []
    Pagination:
    - BoundingBox:
      - 1445.08544921875
      - 2315.408447265625
      - 54.815673828125
      - 52.37255859375
      Content: '"11"'
      ElementID: '"3"'
      Score: '"0.9865828156471252"'
      Type: '"page_no"'
    Tables: []
    TextBlocks:
    - BoundingBox:
      - 131.93858337402344
      - 809.3839111328125
      - 673.6935577392578
      - 444.4246826171875
      Content: '"Det saknas en rimlig generositet

        från stat, kommun och arbetsgiva-

        re när det gäller de sockersjuka

        ungdomarna, påpekar docent Ber-

        til Söderling i en artikel i Svenska

        läkartidningen. Man spekulerar i

        alltför hög grad i de tillstötande

        s. k. sensymtomen. Det tillkommer

        knappast ett handikappvänligt"'
      ElementID: '"1"'
      Score: '"0.9994248151779175"'
      Type: '"text"'
    - BoundingBox:
      - 830.3582763671875
      - 200.9127655029297
      - 672.113525390625
      - 1048.6465606689453
      Content: '"samhälle att härvidlag uppträda

        som ett försäkringsbolag, vilket

        inte önskar ta några risker för

        framtiden.

        Om och när senkomplikationer

        uppträder och om de medför in-

        skränkningar i arbetsförmågan, det

        är frågor som bör lösas vid den

        aktuella tidpunkten och inte vid

        anställningens eller utbildningens

        början, hävdar docent Söderling.

        Vi får inte glömma att vi också har

        en skara högt kvalificerade diabe-

        tiska män och kvinnor i olika yr-

        ken, personer som gjort och allt-

        jämt gör synnerligen betydelseful-

        la insatser. Det är ytterligt motbju-

        dande att en välståndskultur som

        vår inte fattat klarare ståndpunkt

        beträffande diabetisk och annan

        handikappad ungdoms = utbild-

        ene och anställningsförhållan-

        en."'
      ElementID: '"2"'
      Score: '"0.9997593760490417"'
      Type: '"text"'
    Titles:
    - BoundingBox:
      - 133.75193786621094
      - 198.21221923828125
      - 527.6947174072266
      - 406.5040283203125
      Content: '"Arbetsgivarna

        restriktiva

        mot diabetes-

        ungdomar"'
      ElementID: '"0"'
      Score: '"0.9932644367218018"'
      Type: '"title"'
  JournalName: '"Diabetes"'
  Language: '"Swedish"'
  NumberIndex: '"nr003"'
  PageFilename: '"Diabetes_1963_vol013_nr003_0012.jpg"'
  PageIndex: '"0012"'
  Source: '"diabetes(in-house scanned)"'
  Timestamp: '"2024-08-01T07:01:24.435795"'
  TotalPages: '"209"'
  VolumeIndex: '"vol013"'
  Year: '"1963"'
  md5_checksum: '"04e9d6f562eb98bde52513d821cc2b21"'
```

Figure 11: Example of a YAML file showcasing the metadata output from our layout detection model and OCR processing. Each file contains detailed information, including the bounding boxes of layout elements (e.g., text blocks, advertisements, page numbers), confidence scores for detected elements, extracted text content, and additional document metadata like file size, journal name, and timestamp.

4.4 *Future work*

4.4.1 *Active Learning*

In order to further improve the performance of the layout detection model, we will incorporated an Active Learning (Settles 2009) approach. Specifically, we will utilize uncertainty sampling (Settles 2009: 11–12). Active learning is a process where we focus on the most challenging cases for the model—those where the model is least confident in its prediction. By having humans review these difficult cases, the model learns from its mistakes, making it more accurate in future predictions.

One important consideration in this process is the potential for class imbalance and bias in model predictions. If certain layout elements are consistently misclassified or have lower confidence scores due to under-representation in the dataset, active learning allows us to correct this by prioritizing uncertain cases for annotation. This ensures that less frequent or more complex elements, such as tables and lists, are sufficiently represented in the training data. While OCR-related biases, such as errors arising from faded text or unusual typefaces, are different in nature from class imbalance in layout detection, both can contribute to errors in downstream tasks. Addressing these biases systematically through improved training and data selction strategies is an area for further study.

In our Active Learning loop, we will analyze the inferences made by our trained layout detection model and identify the elements whose class confidence scores were low and hovered around the threshold of 80%. Cases where the model exhibits uncertainty in classifying layout elements will be manually annotated and added to the training dataset. The goal of this strategy is to improve the model's performance by fine-tuning it on examples where its predictions were least confident, thereby ensuring that it learns from its own mistakes while also mitigating potential biases in class representation.

4.4.2 *Create a searchable database*

We aim to create a searchable platform to access the extracted metadata and images from the SweMPer project. This platform will be developed using GraphQL for efficient data querying and React for building the user interface. Users will be able to search, browse, and filter the dataset based on metadata, such as journal name, year, and layout elements. This database will serve as a valuable tool for researchers, allowing them to explore historical Swedish medical journals through an intuitive interface.

5 *Hands-on practical introduction*

5.1 *Converting Labelme rectangles to YOLO format*

Link to the code for converting Labelme Rectangles to YOLO Format: https://www.kaggle.com/code/daliaortizpablo/hands-on-labelme-to-yolo

5.2 *Code for selecting images in Section 4.3.2*

Link to the code used for selecting images: https://github.com/CDHUppsala/swemper_ML/blob/main/beast_code/utils/data_prep.py

5.3 *Code to create metadata for Section 4.3.5*

Link to the code: https://github.com/CDHUppsala/swemper_ML/blob/main/beast_code/create_metadata_v1.py

5.4 *Other code and instructions*

You can find more code related to the project and instructions at the GitHub repository here: https://github.com/CDHUppsala/swemper_ML/tree/main

References

Aangenendt, Gijs, Maria Skeppstedt & Ylva Söderfeldt. 2024. Curating a historical source corpus of 20th century patient organization periodicals. In *Huminfra conference*, 76–82.

AI, Deci. 2023. *Yolo-nas: a new state-of-the-art object detection model*. https://www.deci.ai/yolo-nas/. Accessed: September 23, 2024.

Arroyo-Ramirez, Elvia. 2016. Invisible defaults and perceived limitations: Processing the juan gelman files. *Medium*.

Bahrami, Mahdi & Amir Albadvi. 2024. Deep learning for identifying Iran's cultural heritage buildings in need of conservation using image classification and grad-cam. *ACM Journal on Computing and Cultural Heritage* 17(1). 1–20.

Bender, Edward A. 2000. *An introduction to mathematical modeling*. Courier Corporation.

Bolukbasi, Tolga, Kai-Wei Chang, James Zou, Venkatesh Saligrama & Adam Kalai. 2016. *Man is to computer programmer as woman is to homemaker? Debiasing word embeddings*. https://arxiv.org/abs/1607.06520.

Bunge, Mario & Mario Bunge. 1973. Mathematical modeling in social science. *Method, Model and Matter*. 131–142.

Colavizza, Giovanni, Tobias Blanke, Charles Jeurgens & Julia Noordegraaf. 2022. Archives and ai: An overview of current debates and future perspectives. *Journal on Computing and Cultural Heritage* 15(1). 1–15. DOI: 10.1145/3479010.

Ginzburg, Carlo. 1979. Clues: Roots of a scientific paradigm. *Theory and Society* 7(3). 273–288.

Goodfellow, Ian. 2016. *Deep learning*.

Google. N.d. *Machine learning glossary*. https://developers.google.com/machine-learning/glossary#t. Accessed: September 9, 2024. Updated: September 4, 2024.

Gruber, Ivan, Pavel Ircing, Petr Neduchal, Marek Hrúz, Miroslav Hlaváč, Zbyněk Zajíc, Jan Švec & Martin Bulín. 2020. An automated pipeline for robust image processing and optical character recognition of historical documents. In *International conference on speech and computer*, 166–175.

He, Kaiming, Georgia Gkioxari, Piotr Dollár & Ross Girshick. 2017. Mask r-cnn. In *Proceedings of the ieee international conference on computer vision*, 2961–2969.

He, Kaiming, Xiangyu Zhang, Shaoqing Ren & Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the ieee conference on computer vision and pattern recognition*, 770–778.

Hill, Mark J & Simon Hengchen. 2019. Quantifying the impact of dirty ocr on historical text analysis: Eighteenth century collections online as a case study. *Digital Scholarship in the Humanities* 34(4). 825–843.

Jaillant, Lise & Annalina Caputo. 2022. Unlocking digital archives: Cross-disciplinary perspectives on ai and born-digital data. *AI & society* 37(3). 823–835.

Jarlbrink, Johan & Pelle Snickars. 2017. Cultural heritage as digital noise: Nineteenth century newspapers in the digital archive. *Journal of Documentation* 73(6). 1228–1243.

Jockers, Matthew L. 2013. *Macroanalysis: Digital methods and literary history*. University of Illinois Press.

Kaur, Ravpreet & Sarbjeet Singh. 2023. A comprehensive review of object detection with deep learning. *Digital Signal Processing* 132. 103812.

Krišto, Mate, Marina Ivasic-Kos & Miran Pobar. 2020. Thermal object detection in difficult weather conditions using yolo. *IEEE access* 8. 125459–125476.

Labellerr. 2023. *A detailed guide to best practices in image annotation*. https://www.labellerr.com/blog/best-practices-in-image-annotation/.

Written by Akshit Mehra. Accessed: August 29, 2024. Published: Nov 23, 2023.

Lee, Benjamin Charles Germain, Jaime Mears, Eileen Jakeway, Meghan Ferriter, Chris Adams, Nathan Yarasavage, Deborah Thomas, Kate Zwaard & Daniel S. Weld. 2020. The newspaper navigator dataset: Extracting and analyzing visual content from 16 million historic newspaper pages in chronicling america. *CoRR* abs/2005.01583. https://arxiv.org/abs/2005.01583.

Małecki, Piotr. 2006. *Road*. Wikimedis Commons. Licensed under the GNU Free Documentation License and Creative Commons Attribution-Share Alike 3.0 Unported License. Modified from the original. https://creativecommons.org/licenses/by-sa/3.0/ (25 June, 2024).

Marciano, Richard, Victoria Lemieux, Mark Hedges, Maria Esteva, William Underwood, Michael Kurtz & Mark Conrad. 2018. Archival records and training in the age of big data. In *Re-envisioning the mls: Perspectives on the future of library and information science education*, vol. 44, 179–199. Emerald Publishing Limited.

Matschak, Tizian, Florian Rampold, Malte Hellmeier, Christoph Prinz & Simon Trang. 2022. A digitization pipeline for mixed-typed documents using machine learning and optical character recognition. In *International conference on design science research in information systems and technology*, 195–207.

Medium. 2020. *Introducing label studio, a swiss army knife of data labeling*. https://towardsdatascience.com/introducing-label-studio-a-swiss-army-knife-of-data-labeling-140c1be92881. Written by Nikolai Liubimov. Accessed: September 6, 2024. Published: Januaty 8, 2020.

Medium. 2024. *A quick reference for bounding boxes in object detection*. https://medium.com/@rajdeepsingh/a-quick-reference-for-bounding-boxes-in-object-detection-f02119ddb76b/. Written by Rajdeep Singh. Accessed: June 25, 2024. Published: Jan 17, 2024.

Mikołajczyk, Agnieszka & Michał Grochowski. 2018. Data augmentation for improving deep learning in image classification problem. In *2018 international interdisciplinary phd workshop (iiphdw)*, 117–122.

Moretti, Franco. 2005. *Graphs, maps, trees: Abstract models for a literary history*. London/New York: Verso.

Naiman, Jill P, Peter KG Williams & Alyssa Goodman. 2023. The digitization of historical astrophysical literature with highly localized figures and figure captions. *International Journal on Digital Libraries*. 1–21.

Nanonets. N.d. *Annotation best practices for object detection*. https://nanonets.github.io/tutorials-page/docs/annotate. Accessed: August 29, 2024.

Neudecker, Clemens, Konstantin Baierer, Mike Gerber, Christian Clausner, Apostolos Antonacopoulos & Stefan Pletschacher. 2021. A survey of ocr evaluation tools and metrics. In *Proceedings of the 6th international workshop on historical document imaging and processing*, 13–18.

Nicoletti, Leonardo & Dina Bass. 2023. Humans are biased. Generative AI is even worse. *Bloomberg Technology + Equality*. https://www.bloomberg.com/graphics/2023-generative-ai-bias/.

Ntoutsi, E., P. Fafalios, U. Gadiraju, V. Iosifidis, W. Nejdl, M. Vidal, S. Ruggieri, F. Turini, S. Papadopoulos, E. Krasanakis, I. Kompatsiaris, K. Kinder-Kurlanda, C. Wagner, F. Karimi, M. Fernández, H. Alani, B. Berendt, T. Kruegel, C. Heinze, K. Broelemann, G. Kasneci, T. Tiropanis & S. Staab. 2020. Bias in data-driven artificial intelligence systems—an introductory survey. *WIREs Data Mining and Knowledge Discovery* 10. DOI: 10.1002/widm.1356.

Obeso, Abraham Montoya, Jenny Benois-Pineau, Alejandro Álvaro Ramirez Acosta & Mireya Saraí García Vázquez. 2017. Architectural style classification of mexican historical buildings using deep convolutional neural networks and sparse features. *Journal of Electronic Imaging* 26(1). 011016–011016.

OpenCV. 2024. *Data annotation – a beginner's guide*. https://opencv.org/blog/data-annotation/. Written by Alvi Farooq. Accessed: June 4, 2024. Published: Februry 21, 2024.

Pastaltzidis, Ioannis, Nikolaos Dimitriou, Katherine Quezada-Tavarez, Stergios Aidinlis, Thomas Marquenie, Agata Gurzawska & Dimitrios Tzovaras. 2022. Data augmentation for fairness-aware machine learning: Preventing algorithmic bias in law enforcement systems. In *Proceedings of the 2022 acm conference on fairness, accountability, and transparency*, 2302–2314.

Pavlopoulos, Georgios A, Maria Secrier, Charalampos N Moschopoulos, Theodoros G Soldatos, Sophia Kossida, Jan Aerts, Reinhard Schneider & Pantelis G Bagos. 2011. Using graph theory to analyze biological networks. *BioData mining* 4. 1–27.

Rafael Padilla, Amy Roberts & the Hugging Face Team. 2023. *Open object detection leaderboard*. https://huggingface.co/spaces/hf-vision/object_detection_leaderboard.

von Ranke, Leopold. [1824] 1885. *Geschichten der romanischen und germanischen völker von 1494 bis 1514*. Verlag von Dunder und Humblot.

Redmon, Joseph, Santosh Divvala, Ross Girshick & Ali Farhadi. 2016. You only look once: Unified, real-time object detection. In *Proceedings of the ieee conference on computer vision and pattern recognition*, 779–788.

Reitermanova, Zuzana. 2010. Data splitting. In *Wds*, vol. 10, 31–36.

Roboflow. 2024a. *Best ocr models for text recognition in images*. https://blog.roboflow.com/best-ocr-models-text-recognition/. Written by Leo Ueno. Accessed: September 23, 2024. Published: March 16, 2024.

Roboflow. 2024b. *How to label image data for computer vision models*. https://blog.roboflow.com/tips-for-how-to-label-images/. Written by Joseph Nelson. Accessed: August 29, 2024. Published: January 5, 2024.

Russell, Bryan C, Antonio Torralba, Kevin P Murphy & William T Freeman. 2008. Labelme: A database and web-based tool for image annotation. *International journal of computer vision* 77. 157–173.

Settles, Burr. 2009. Active learning literature survey.

Shen, Zejiang, Ruochen Zhang, Melissa Dell, Benjamin Charles Germain Lee, Jacob Carlson & Weining Li. 2021. Layoutparser: A unified toolkit for deep learning based document image analysis. In *Document analysis and recognition–icdar 2021: 16th international conference, lausanne, switzerland, september 5–10, 2021, proceedings, part i 16*, 131–146.

Shorten, Connor & Taghi M Khoshgoftaar. 2019. A survey on image data augmentation for deep learning. *Journal of big data* 6(1). 1–48.

Smith, Astrid J. & Bridget Whearty. 2023. All the work you do not see: Labor, digitizers, and the foundations of digital humanities. In Lauren F. Klein & Matthew K. Gold (eds.), *Debates in the digital humanities 2023*, 27–46. University of Minnesota Press.

Smith, Ray. 2007. An overview of the tesseract ocr engine. In *Icdar '07: Proceedings of the ninth international conference on document analysis and recognition*, 629–633. Washington, DC, USA: IEEE Computer Society. https://storage.googleapis.com/pub-tools-public-publication-data/pdf/33418.pdf.

Thomas, Julia & Irene Testini. 2024. Capturing captions: Using AI to identify and analyse image captions in a large dataset of historical book illustrations. *Digital Humanities Quarterly* 18(2).

Ultralytics. 2023. *Object detection datasets overview*. https://docs.ultralytics.com/datasets/detect/. Written by Glenn Jocher, Rizwan Munawar, Ivor Zhu, Laughing-Q. Accessed: June 26, 2024. Published: Nov 12, 2023. Updated: Jun 2, 2024.

van Strien, Daniel, Kaspar Beelen, Mariona Ardanuy, Kasra Hosseini, Barbara McGillivray & Giovanni Colavizza. 2020. Assessing the impact of OCR quality on downstream NLP tasks. In *Proceedings of the 12th international conference on agents and artificial intelligence*, 484–496. Valletta, Malta: SCITEPRESS. DOI: 10.5220/0009169004840496.

Villaespesa, Elena & Seth Crider. 2021. A critical comparison analysis between human and machine-generated tags for the metropolitan museum of art's collection. *Journal of Documentation* 77(4). 946–964.

Wang, Rui-Sheng, Assieh Saadatpour & Reka Albert. 2012. Boolean modeling in systems biology: An overview of methodology and applications. *Physical biology* 9(5). 055001.

Wevers, Melvin & Thomas Smits. 2020. The visual digital turn: Using neural networks to study historical images. *Digital Scholarship in the Humanities* 35(1). 194–207.

Xailient. 2024. *Exploring data labeling and the 6 different types of image annotation*. https://xailient.com/blog/exploring-data-labeling-and-the-6-different-types-of-image-annotation/. Written by Sabina Pokhrel. Accessed: June 25, 2024. Published: March 11, 2024.

Xie, Saining, Ross Girshick, Piotr Dollár, Zhuowen Tu & Kaiming He. 2017. Aggregated residual transformations for deep neural networks. In *Proceedings of the ieee conference on computer vision and pattern recognition*, 1492–1500.

Xie, Xiaoyuan, Joshua WK Ho, Christian Murphy, Gail Kaiser, Baowen Xu & Tsong Yueh Chen. 2011. Testing and validating machine learning classifiers by metamorphic testing. *Journal of Systems and Software* 84(4). 544–558.

Xu, Yun & Royston Goodacre. 2018. On splitting training and validation set: A comparative study of cross-validation, bootstrap and systematic sampling for estimating the generalization performance of supervised learning. *Journal of analysis and testing* 2(3). 249–262.

Yosinski, Jason, Jeff Clune, Yoshua Bengio & Hod Lipson. 2014. How transferable are features in deep neural networks? *Advances in neural information processing systems* 27.

Zhang, Yuanyuan, Yixuan Zhang, Bence Mark Halpern, Tanvina Patel & Odette Scharenborg. 2022. Mitigating bias against non-native accents. In *Proceedings of the annual conference of the international speech communication association, interspeech*, vol. 2022, 3168–3172.

Zhong, Xu, Jianbin Tang & Antonio Jimeno Yepes. 2019. Publaynet: Largest dataset ever for document layout analysis. In *2019 international conference on document analysis and recognition (icdar)*, 1015–1022.

Zhu, Xizhou, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang & Jifeng Dai. 2020. Deformable detr: Deformable transformers for end-to-end object detection. *arXiv preprint arXiv:2010.04159*.

Zou, Zhengxia, Keyan Chen, Zhenwei Shi, Yuhong Guo & Jieping Ye. 2023. Object detection in 20 years: A survey. *Proceedings of the IEEE* 111(3). 257–276.

List of abbreviations

AI	Artificial Intelligence
CDHU	Centre for Digital Humanities and Social Sciences (Uppsala University)
CNN	Convolutional Neural Network
mAP	mean Average Precision
ML	Machine Learning
NLP	Natural Language Processing
OCR	Optical Character Recognition
RCNN	Region Based Convolutional Neural Networks
SweMPer	Communicating Medicine: Digitalisation of Swedish Medical Periodicals, 1781–2011
ROI	Region of Interest

Corresponding authors

Dalia Ortiz Pablo
Department of ALM
Uppsala University
dalia.ortiz_pablo@abm.uu.se

Sushruth Badri
Department of ALM
Uppsala University
sushruth.badri@abm.uu.se