TUM Data Innovation Lab

Munich Data Science Institute (MDSI)

Technical University of Munich

&

TUM Chair for Geometry and Visualization cooperating with Deutsche Nationalbibliothek & TU Delft

Final report of project:

Detection and Classification of Historic Watermarks

Authors	Evghenii Beriozchin, Sebastian Pfaff, Paulina Weyh
Mentors	Prof. Dr. Martin Skrodzki (TU Delft), Lena Polke, M.Sc. (TU Munich), Dr. Ramon Voges (DNB)
Project Lead	Dr. Ricardo Acevedo Cabra (MDSI)
Supervisor	Prof. Dr. Massimo Fornasier (MDSI)

July 2023

Acknowledgements

First and foremost, we want to thank our mentors, Prof. Dr. Martin Skrodzki and Lena Polke, M.Sc., for their guidance, encouragement, and support. They have provided us with helpful feedback that allowed us to keep this project organized and on track. Additionally, we thank Dr. Ramon Voges and both the entire German National Libary, and the German Museum of Books and Writing, for granting us access to their remarkable collection and dataset and for sharing their expertise on the history and properties of paper and watermarks. This endeavor would not have been possible without their contribution. We also thank TU Delft for providing us access to their IT infrastructure, making it possible to build and train our models. Finally, we thank Dr. Ricardo Acevedo Cabra, Prof. Dr. Massimo Fornasier, and the Munich Data Science Institute (MDSI) team for organizing the TUM Data Innovation Lab and making the project possible in the first place.

Abstract

The study of watermarks holds significant importance for historical humanities. Watermarks offer valuable insights into the origins and production of paper, aiding in identifying papermakers, mills, and periods when a specific piece of paper was made. This information plays a crucial role in dating and verifying historical documents. By identifying watermarks in various documents, one can establish connections between manuscripts, trace paper sources, and track trade routes and distribution networks. However, the identification of such watermarks and comparison is currently a time-consuming manual process.

This report presents a novel approach to automatically finding similar watermarks within the database of the German Museum of Books and Writing and the German National Library (DNB) based on their digitized collection of historical papers, watermarks, and traced watermarks. As the DNB continuously adds new watermarks to the database, we aimed for an approach independent of the current digitization status and adaptive to new watermarks. Furthermore, as the watermarks are loosely labeled, the network needed to be independent of input-output pairs. In our approach, we use the image dataset provided by the DNB to create an unpaired dataset of watermarks and watermark sketches. The latter are generated by preprocessing watermark tracings. Using this dataset, we first preprocess the watermarks using different image processing techniques and then we train a CycleGAN neural network that can generate a sketch of a watermark present in a scan of a historical paper. With this model we generate sketches for all watermarks from the dataset and combine them with the sketches produced by preprocessing tracings. As a next step, we use a pre-trained ResNet18 neural network to extract a feature vector for each sketch. Finally, we use the Spotify Annoy algorithm, which enables an efficient approximate nearest neighbor search to compare a generated sketch for a new watermark with our entire database.

As a result, our full pipeline allows the user to input a watermark or a tracing thereof and get similar watermarks from the existing database. Within 25 nearest neighbors, we achieve an accuracy of more then 50%, and within 50 nearest neighbors, we achieve an accuracy of over 68% when comparing a new watermark to a database of over 6600 digitized watermarks and traced watermarks. The database can then be easily connected to the metadata provided by the DNB to get more information about each watermark.

Contents

1 Introduction

The history of paper in Europe dates back to the 12th century when paper making techniques were introduced to the region from China through the Islamic world. Before that, the primary writing materials used in Europe were parchment and vellum made from animal hides. The first European paper mills were established in Spain and Italy, followed by other European countries such as France and Germany in the 13th century. This early paper was made by hand from rags and linen. The process of paper production stayed more or less the same until the 19th century, when the invention of mechanical pulping techniques (by Friedrich Gottlob Keller in 1843) allowed the production of paper from wood fibers. [1] [2] [3] [4]

The process of making paper from rags involved several steps. Images that describe selected steps can be seen in Figure A.1. It typically began with the collection of rags, usually linen or cotton, which were first sorted and cleaned, see Figure A.1a. The sorted rags were cut into smaller pieces and beaten to break them down into tiny fibers, see Figure A.1b. These fibers were subsequently soaked in water to create a pulp, Figure A.1c. The pulp was transferred to a giant vat of water, stirred, and agitated to form an even suspension. A mold and deckle, frames with a wire mesh bottom, were used to scoop up the pulp from the vat. The mold was shaken to allow the water to drain through the mesh, leaving a layer of wet fibers on top. This layer was transferred onto a felt or cloth, and another layer of felt was placed on top. Pressure was applied to the stacked layers to remove excess water and compress the fibers. The whole skimming and pressing process of the paper can be seen in Figure A.1d. Afterwards, the sheets were dried by hanging or by pressing them between weighted boards, see Figure A.1e. Once dry, the paper was sized with animal glue or burnished to achieve a smoother surface. This part of the process was called glazing and can bee seen in Figure A.1f. Please note that the paper manufacturing process described here is simplified and illustrative, and it should not be considered the only method. Over the course of the more than 500 years between 1300 and 1800 A.D., the process of papermaking evolved significantly due to technical innovations, such as the introduction of the Hollander beater or the glazing hammer. [1] [5] [6]

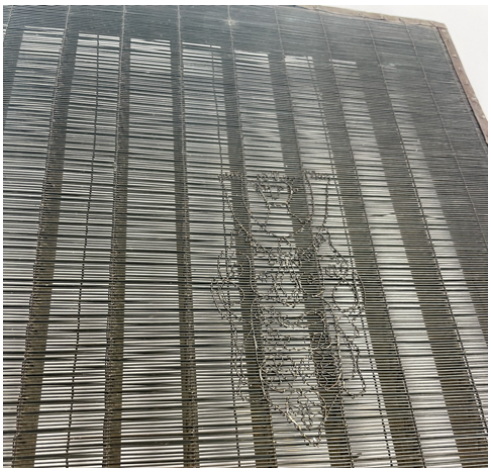
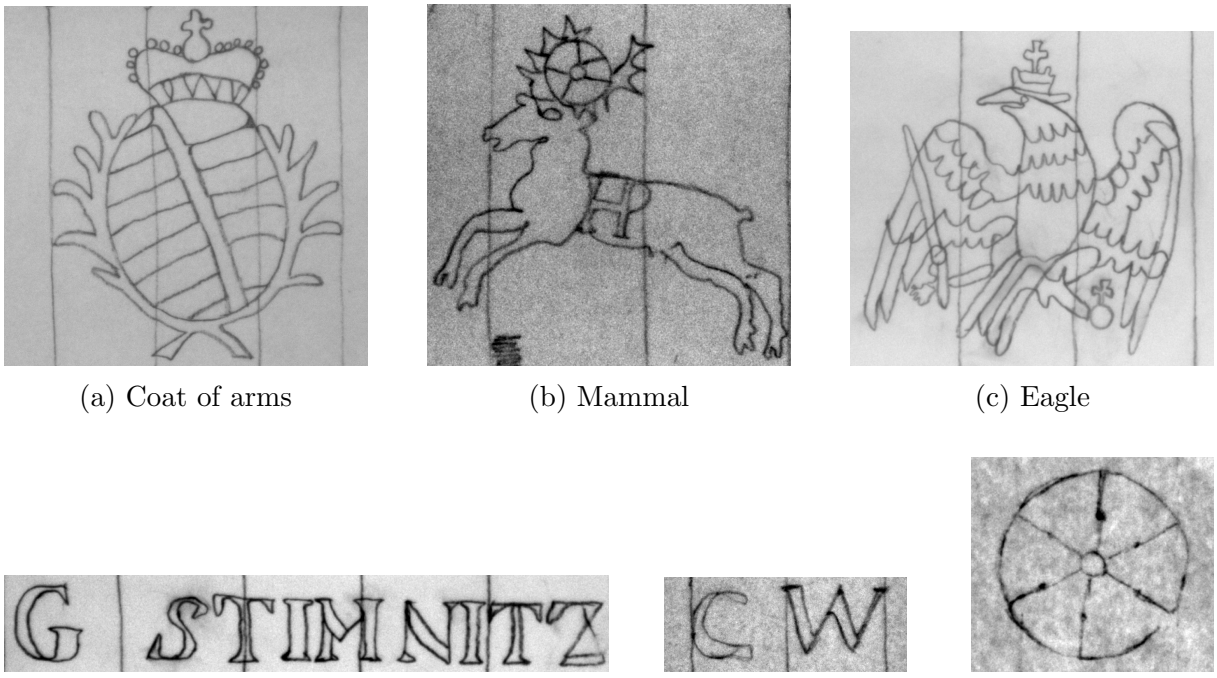


Figure 1: Wooden frame from the German Museum of Books and Writing used to create paper

The frames used to drain the water from the pulp were often embroidered with a metal wire that formed some design, pattern or motif, often images like coats of arms, eagles, or names. Figure 1 shows such a wooden frame belonging to the collection of the German Museum of Books and Writing in Leipzig.

The wire would leave a mark with slightly different thickness than the surrounding area, creating a thinner paper section. This difference in thickness resulted in a translucent area, which served as the watermark on the finished sheet of paper. When held up to the

light, the watermark would become visible, revealing the motif. Most paper producers used watermarks as identification. Paper manufacturers wanted potential customers to recognizes their paper, with high-quality paper being a source of prestige and incredibly expensive. Thus, colloquially speaking, watermarks served as early brand logos. Figure 2 shows examples of the different designs frequently used for watermarks. Watermarks were also used for other reasons, for example to describe properties of the paper such as size. [5] [6] [7] [8] [9] More information on the history of paper can be found in [1] by Hunter, a classical work in the field.



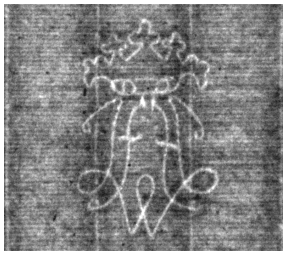
(a) Coat of arms (b) Mammal (c) Eagle

(d) Name (e) Initials (f) Small design

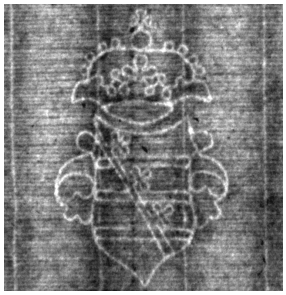
Figure 2: Tracings of common motifs for watermarks

In the previous paragraphs, we summarized the history of paper and explained how and why the paper was watermarked. Now we explain why the study of historical watermarks is of great interest to modern-day historians. Historians mainly use watermarks to date documents and identify their origin. Watermark designs were linked to individual paper mills, and the designs evolved over time. Additionally, the frames used to create the watermarks were built by hand. Therefore, no two frames are identical, even if they have the same motif. Thus, identical watermarks on two different pieces of paper indicate the use of the same wooden frame in the making of both papers. Since each handmade frame was only in use for a limited time, we know that papers containing identical watermarks were produced around the same time. Even papers containing very similar watermarks were likely produced around the same time. By examining the watermark design and its location on the paper and comparing it with known examples, historians can identify the papermaker. Furthermore, historians can date when the paper was produced if they can cross-reference the watermark on the paper of interest with other documents containing the same watermark that can be accurately dated, e.g., stamped government documents,

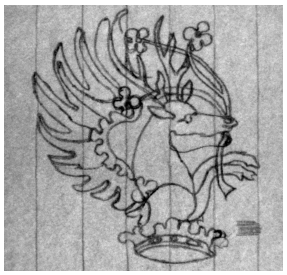
letters, or manuscripts. As paper was valuable, and usually used soon after being produced, we can conclude the time when document was written from the time the paper was produced. This analysis has been used by historians to establish connections between manuscripts, to track trade routes, and to understand paper distribution networks. For example, watermark analysis has been used in art history to authenticate works, to establish a timeline of the artists works or to determine the sources of materials. [10] [8] [7]
For the reasons mentioned above, historians and museums have collected watermarks for over a century. The watermarks were usually collected by holding the watermarks against a light source and tracing the outline of the watermark on a white piece of paper. Books containing entire collections of these tracings have been published for more than 100 years. [11] [12] Examples of watermarks and tracings can be seen in Figure 3.



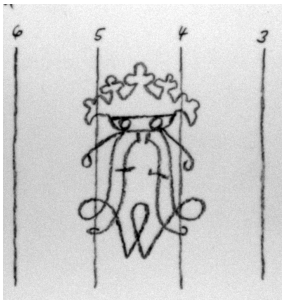
(a) Example watermark 1



(b) Example watermark 2



(c) Example watermark tracing 1



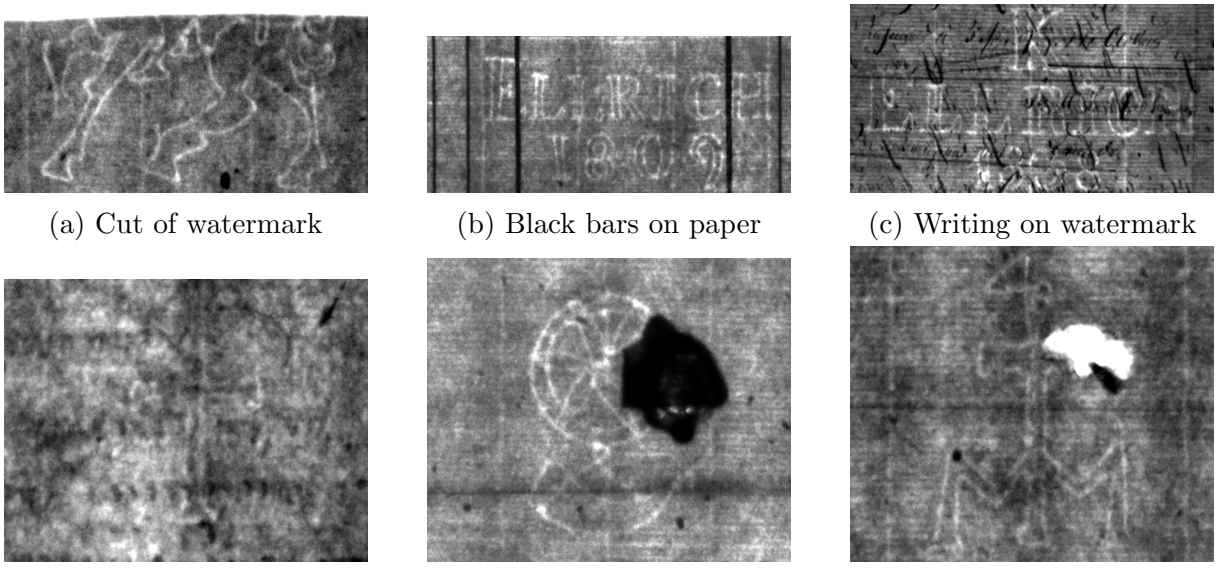
(d) Example watermark tracing 2

Figure 3: Images of watermarks and tracings from DNB dataset

With the advent of the internet and high-quality digital photography, it has become possible to efficiently provide these collections online. Creating large online collections of watermarks helps researchers as it allows them to easily and affordably access a vast amount of watermarks from anywhere in the world.
In fact, many collections of watermarks, such as the Briquet [13] and Piccard [14] collections, have been digitized and made available online. Collections have also been pooled to give researchers a central point of reference rather than searching through dozens of different online archives. For example, the Bernstein project, run by the Austria Academy of Science, contains images over 260,000 (February 2021) of watermarks and tracings from 49 collections and 20 states. [15] [16]
It is not easy to work with these databases. For example, the classification used in the Bernstein projects database contains 13 groups and more than 50 subgroups. The classification of watermarks is based on a semantical system based on symbolism. Understanding this symbolism requires a great deal of training and experience. Using these data bases effectively is difficult for non-experts. [17]
Even though hundreds of thousands of images of watermarks are available online, it remains challenging for non-watermark experts even among historians to find a match. As a result, the process of locating and identifying watermarks still requires a lot time and resources, which is hindering efficient research in this area. For example, the partners of this project, the German Museum of Books and Writing and the German National Library (DNB), pose a large collection of historical paper and watermarks. At the moment the museum and DNB get many requests from researchers to identify specific watermarks

by e-mail. This work is currently done manually by experts from the German National Library in Leipzig, who look up specific watermarks in large magazines. This work is very time consuming and expensive as it requires a great level of expertise. Automating the watermark identification process can greatly boost researchers' efficiency, improve online database accessibility, and empower historians to leverage historical watermarks in their research. Additionally, it opens up opportunities for hobbyists to engage in watermark analysis and search for matches. [18].

Efforts to automate the classification and reverse image search of historic watermarks have emerged as a response to the growing need for efficient and scalable methods in watermark analysis. Researchers and organizations have explored the application of machine learning and computer vision techniques to automate the identification and classification of watermarks. We will discuss the existing approaches and literature in subsection 2.1. However, the problem of automatic watermark detection has not been fully solved yet. This is because image recognition for watermarks is a particularly hard problem as watermarks by nature have properties that make them bad targets for automated image recognition. In particular, watermarks strongly vary in size and generally display very different motifs. We already saw typical watermark designs in Figure 2. At the same time, there exist many watermarks with very similar motifs, which makes finding an exact matching watermark difficult, as many images might be almost identical to a query image. An example of 5 watermarks with basically the same motif can be seen in Figure 5. Figure A.2 shows 20 similar watermarks. Furthermore, watermarks are often poorly preserved. Figure 4 shows different types of decay commonly found on watermarks.

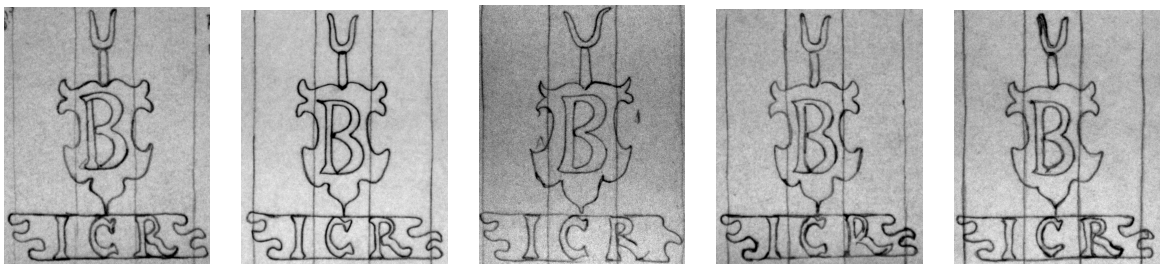


(a) Cut of watermark (b) Black bars on paper (c) Writing on watermark

(d) Faded watermark (e) Black dots on watermark (f) Wholes in paper

Figure 4: Examples for typical degenrations of watermarks

This project aims to add to the existing research field of automated watermark detection described in subsection 2.1 by designing and building a working system for automated



(a) (b) (c) (d) (e)

Figure 5: Five different watermarks with almost identical motifs

reverse image search of historical watermarks and tracings.
Our analysis is based on a dataset provided by DNB. Currently, DNB is digitizing its collection, having already created over 10,000 photos of watermarks and tracings. We describe our dataset and data annotation process in subsection 3.1.
Our approach differs from most existing literature, as we do not directly compare watermarks to each other or to tracings. Instead, our approach has an intermediate step, in which we produce an artificial sketch for each watermark. The sketches are made to roughly resemble the tracings, which make up a significant portion of our dataset. Tracings are easier to analyze due to their higher contrast and white background. Moreover, tracings exhibit less erosion, are less noisy, and encounter fewer artificial obstructions, such as writing. However, considering that tracings are also made on different pieces of paper and photographed under different conditions and may contain some annotations, we still had to preprocess the tracings to create a desired sketch format - a binary black and white image with a minimum amount of other annotations and noise. We describe our entire approach in detail in section 3 and section 4.
In subsection 3.1, we describe our dataset, the data acquisition, and our manual data annotation. We use different preprocessing pipelines depending on metadata we collect manually in subsection 3.1. Our image preprocessing is described in sections subsection 3.2 and subsection 3.3. We use various image-processing tools to denoise our images and reduce the impact of degradations. Finally, in subsection 4.3 we do a nearest neighbor search to find the closest match among all sketches in our dataset.
Before we go into the details of our approach in section 3 and section 4, we first examine the relevant existing research in section 2. In subsection 2.1, we compare our pipeline and approach to others in the watermark recognition field. In subsection 2.2, we describe Generative Adversarial Networks, their subtypes and applications in image translation.

To sum up, the ultimate goal of this project is to design and to build an automated reverse image search system, where a user inputs a photo of a watermark and gets the closest match identified among watermarks in a database. Such a system will make identifying watermarks faster and cheaper. Furthermore, it will make working with historical watermarks more accessible for non-experts. Over the course of this project[1] we build such a system for the German Museum of Books and Writing and the German National Library based on their immense collection of historical papers and watermarks.

[1]Code: https://github.com/EvgheniiBeriozchin/watermark-detection.git

2 Related Work

2.1 Watermark Recognition

Watermarks on historical papers have been collected and studied for a long time. Books containing entire collections have been published as far back as 100 years ago [11] [12] [19] [20]. For almost as long historians have used watermarks to date works of important artists such as Shakespeare or Rembrandt [21] [22] [23].

In the 1990s and 2000s, computer based approaches for the analysis of historical documents and watermarks have become used and studied. These approaches were built using classical image processing and computer vision. However, full reverse image search - the goal of this project - was not usually attempted. Only [24] contains such a feature. However, the approaches from this time were successfully used to extract watermarks from images and increase the visibility of watermarks. [25] [26] [27] [28] [29] [30] The classical approach that comes closest to ours is [24]. Rauber et. al implement a reverse image search based on computing a circular histogram of the binarized tracing around its centre of gravity. However, the paper only deals with high quality tracings. Therefore, the problem they solve is not comparable to ours, since we implement a reverse image search for watermarks and watermarks and tracings have very different properties.
In his doctoral thesis Hiary introduced two different approaches to localize watermarks, to enhance the quality and to extract a binarized graphical representation of the watermark from the original scan. The first approach combines image processing operations and the second approach uses a version of the back-lighting effect to find the location of the watermark. [29] Our own watermark preprocessing is inspired by Hiary's first approach and will be discussed in subsection 3.2.
Other approaches for computer aided recognition of watermarks have also appeared in the last couple of years. For example, [31] and [32] use decision trees based on user feedback to identify watermarks in Rembrandt etchings. This approach is highly accurate but requires detailed manual user inputs for each query image and a labeled dataset. For reference, the preparations required for the dataset here are more complex and take considerably more time and effort than the annotations in our dataset described in subsection 3.1. The approach is thus not scalable. It is however relatively easy to use for non-experts. In [33], user input is used to calculate the distances and ratios between watermarks and chain line intervals to accurately determine watermarks. This approach also suffers from similar issues as the decision tree approaches in [31] and [32].

With the popularization of deep learning for image recognition in the last 10 years, a whole new tool box for image recognition has been opened to researchers. These new methods have been used to compare and find watermarks in an automated way.
In [34] D. Picard, T. Henn and G. Dietz propose a dictionary learning approach for the watermark reverse image search problem. They learn the dictionary from a set of images based on a reconstruction criterion, with additional constraints. Afterwards they use a bag of words type approach to aggregate the local features into a single vector which represents the entire watermark. Their dataset consists of 658 tracings and the first relevant watermark has an average rank of 30/657. The accuracy is already useful for practition-

ers, as they typically only need to investigate the 30 closest watermarks to their query watermark. Looking up 30 watermarks is manageable and can be done by hand.
Pondenkandath et al. introduced a classification and similarity matching task using the ResNet18 convolution neural network (CNN) in [35]. They performed their tasks on the WZIS watermark database containing 106,000 images and 96,000 tracings from the dataset [14] compiled by Gerhard Piccard. This dataset contains almost entirely high quality tracings, and therefore does not compare to ours. Pondenkandath et al. compared a randomly initialized model to a model initialized with weights from the ImageNet dataset [36] and with the weights from the classification task. We also used the ResNet18 model for image comparison. However, we used a pre-trained model for feature extraction, not classification as that would require retraining the model whenever new watermarks are digitized and added to the dataset.
Shen et al. [37] also use a CNN based approach. They use cell phone photographs to retrieve an exact match from the Briquet catalog [13] of 16,753 tracings. They distinguish between global and local features. First, they calculate a global similarity score and then improve the matching by employing spatial information of the watermark and locally fine-tune features. This way, they manage to outperform classical CNN approaches and manage to recover the exact matching tracing with a 55 % accuracy where each tracing corresponds to an individual class of their CNN. Finding the exact image from a simple cell phone photo of a watermark, not a tracing, with more than 50 % accuracy is a very promising result given the bad properties of watermarks discussed in section 1.
Bounou et al. [38] build on the approach of Shen in [37]. In particular, they build a web application for automatic watermark recognition based on an input image of the watermark by a user. In order to speed up the similarity matching based lookup (one lookup using Shen's full model takes 27 minutes for the full database), they use a fast yet less accurate approach. As of July 2023, the Web application can no longer be found online.

Our approach contributes to the existing literature by bridging the divide between watermarks and tracings. Our model allows the search of tracings and watermarks from a dataset containing both tracings and watermarks. This expands the scope of previous research and provides a comprehensive new approach to the image recognition problem for historic watermarks. By generating sketches from watermarks, we establish a novel connection between tracings and watermarks, allowing us to leverage extensive existing datasets that contain many tracings. Furthermore, our approach eliminates the need for retraining the model when new data is digitized and added to the DNB database. This ensures efficient and continuous integration of new information without disrupting the system's performance.

2.2 Generative Adversarial Networks (GANs)

Generative Adversarial Networks were first introduced by Goodfellow et al. [39] in order to enable generative modeling of various objects. Usually, such models consist of a generator network and discriminator network. Given a dataset of objects, the generator network learns to generate realistic objects, whereas the discriminator learns to detect fake objects. As in our task objects are images, we will focus on image GANs.
To extend the usability of such models, Mirza et al. [40] introduced conditional GANs

(cGANs), which enable inputs - or conditions - to the generator. Thus, a cGAN does not simply generate realistic images which are similar to the training dataset, but takes some conditions as input and generates realistic images in accordance to the original dataset, as well as to the the input conditions. The inputs/conditions may vary, but for our use case image inputs are relevant.
Having images as inputs means that a cGAN can be perceived as an image translation model. The input image is transformed into an output image by the generator and the 'correctness' of the transformation is controlled by the discriminator. This idea was first introduced in the pix2pix model by Isola et al [41].
cGANs as image translation models were successfully used for various tasks like image colorization, sketch-to-image transformation [41], historical map stylization [42] or face (de)aging [43]. We tested a pre-trained image-to-sketch model on our watermarks [44]. However, it did not produce usable results, so we discarded this idea.
cGAN-based image translation models can be divided into 2 important subcategories based on the input dataset. The given inputs can be either paired or unpaired.
A paired dataset means that for each image of class A (input), there exists a corresponding image of class B (output) in the training dataset. The previously mentioned pix2pix (Isola et al. [41]) is a model for such datasets. A further such model is pix2pixHD by Wang et al [45], which is adapted for translation of high resolution images.
In constrast, an unpaired dataset does not contain a 1-to-1 correspondence between the images of the two classes, rather it only contains a set of images of each class. This makes the task of translation considerably more difficult, but with a large dataset which is representative of the original classes such a model is a viable option. Such a model was first introduced as CycleGAN in Zhu et al. [46]. Using cycle-consistent adversarial networks, they learned to transform horses into zebras, real pictures into Monet paintings etc. Improvements to this model architecture were later introduced in various works. For example, Contrastive Unpaired Translation (CUT) by Park et al. [47] made training faster and more memory-efficient. Augmented CycleGAN [48] enabled learning many-to-many mappings between classes. Shen et al. [49] improved the overall performance by adjusting the loss function and data augmentation. As these models were not directly applicable to our use case, and CUT did not significantly affect runtime on our 6000 image dataset (see next chapters), we focused on the direct application of CycleGAN.
Unpaired cGANs were successfully used for various tasks such as CBCT (tomography) image correction [50], face generation [51], molecular optimization [52] or seismic data denoising [53].
Note that many of these models, in particular pix2pix and CycleGAN simultaneously learn to translate images of class A into class B and from class B into class A. Thus, works in this area usually evaluate both directions of translation. As the translation of tracings into watermarks is not a focus of this work, we will not evaluate this side of the model, although the model inherently learns to perform this task as well.
In our case, given the loose structure of the original data - there is no explicit mapping between classes of watermarks and tracings - it is problematic to obtain a paired dataset. Thus, cGANs for unpaired datasets are the primary method used in this work. We further used a paired cGAN - pix2pix - in order to preprocess one class of our images, the tracings. This procedure as well as the way we acquired and preprocessed data will be explained in Chapter 3.

3 Data Exploration and Preprocessing

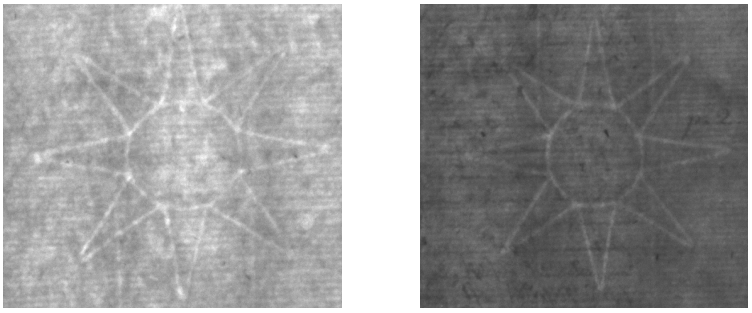
3.1 Data Acquisition and Cleaning

The data for this project is provided by DNB. It is based largely on the physical collection of watermarks gathered by Karl Theodor Weiss in the first half of the 20th century. The collection consits mainly of watermarks and tracings but also some early copies. These tracings were the most common way to document and archive watermarks for a long time. Books consisting of these have been published for decades. The physical collection is stored in an archive on sliding shelves (magazines) at the DNB headquarters in Leipzig. The entire DNB watermark and historical document collection contains more than 400,000 pieces. [18]
DNB began to digitize its archive in 2013. They started with watermarks from the German states of Saxony and Thuringia. To date, more than 10,000 images have already been digitized. The images were digitized using a special setup, see Figure A.3. DNB takes the images using a DMK 72AUC02, a monochrome USB 2.0 Camera with 5 mega pixels. They furthermore, use backlighting and an infrared filter to enhance the visibility of the watermarks and reduce the visibility of writing on the watermark. Metadata like an image class or whether the image contains a watermark or tracing is not accessibly provided by DNB.
The direct use of raw images for image comparison poses several challenges. Examples why automatic watermark detection is difficult can be seen in Figure A.6 and Figure 5. The watermarks, being only a small part of the actual image, introduce difficulties such as variations in size and location within the image. Some images may lack a watermark entirely. Additionally, certain images contain additional annotations, and the background color, particularly for watermarks, can vary significantly. In some cases, multiple watermarks may be present in an image. Examples of watermarks with varying motifs, sizes and background are shown in Figure 2. Furthermore, about half of our dataset consists of tracings, as tracing the watermark by hand was historically the most common way to collect watermarks. However, the watermarks and tracings have very different properties, recall Figure 3. Thus, comparing images from the two classes directly is very hard. Therefore, to enable such comparisons, we separately generate artificial sketches from both watermarks and tracings by first preprocessing the data and then feeding it into a neural network. In order to do that we need to know which images contain watermarks and which contain tracings. As mentioned above, this metadata was not available to us. We collected this metadata by hand using the open source labeling tool LabelStudio [54]. An example of our labeling in LabelStudio is shown in Figure A.4.
We use data annotation and preprocessing to overcome the aforementioned challenges in our data. To address some of these issues, we manually amend the dataset by incorporating bounding boxes around the watermarks and tracings. This approach effectively resolves problems associated with multiple watermarks, varying locations, empty papers, and the small size of watermarks within the image. It also mitigates the impact of background variations. Furthermore, we gathered metadata such as whether an image represents a watermark, tracing, contains writing, or is empty. In total, we annotated 6100 images containing 3941 watermarks and 2661 tracings.
However, since watermarks and tracings have such distinct properties, using the same

image preprocessing to extract the contour of the motif is not feasible. Thus, we use different preprocessing for watermarks and tracings. Both preprocessing approaches are described in detail in subsection 3.2 and subsection 3.3.

3.2 Preprocessing of Watermarks

In this subsection we will introduce processing steps for images of watermarks. The watermarks provided by DNB vary in shape and quality. Some watermarks have written text on top, some watermarks have a lighter or darker background. An example may be found in Figure 6. As the pipeline should be applicable to all watermarks, our requirement is to create a robust and flexible image processing architecture. The overall goal of preprocessing for watermarks is to enhance their visibility and, therefore, improve the quality of the CycleGAN for sketch generation.



(a) Light watermark. (b) Dark watermark with text.

Figure 6: Example on different types of available watermarks in the data base.

As described in section 2, Hiary also analysed methods to clean photographs of watermarks and make the watermark more visible [29]. Some of the same morphological transformations Hiary used in his work are also present in our procedure. Hiary first stretches the image and then uses a combination of erosion and dilation to reduce the writing on the watermarks. In a next step he applies top-hat transformation to remove the non-uniform background. However, the morphological transformations we are using are applied in a different order and using different parameters. Moreover, we use more steps for our preprocessing.
We use different methods to stretch the contrast of the images, remove noisy foreground and background, and combine different processing steps to get an optimal output. Figure 7 gives an overview of the applied steps in preprocessing. Figure 8 shows an example of the different preprocessing steps applied to a watermark. All steps inspired by Hiary are marked. In the following, all steps are described in detail.
Stretch Contrast To enhance the visibility of the watermark in the background of the image, we stretch the contrast using

$$image_{streched} = 255 \cdot \frac{img - \min(img)}{\max(img) - \min(img)} \tag{1}$$

where *img* is a grey-scaled image. An example is shown in Figure 8b.

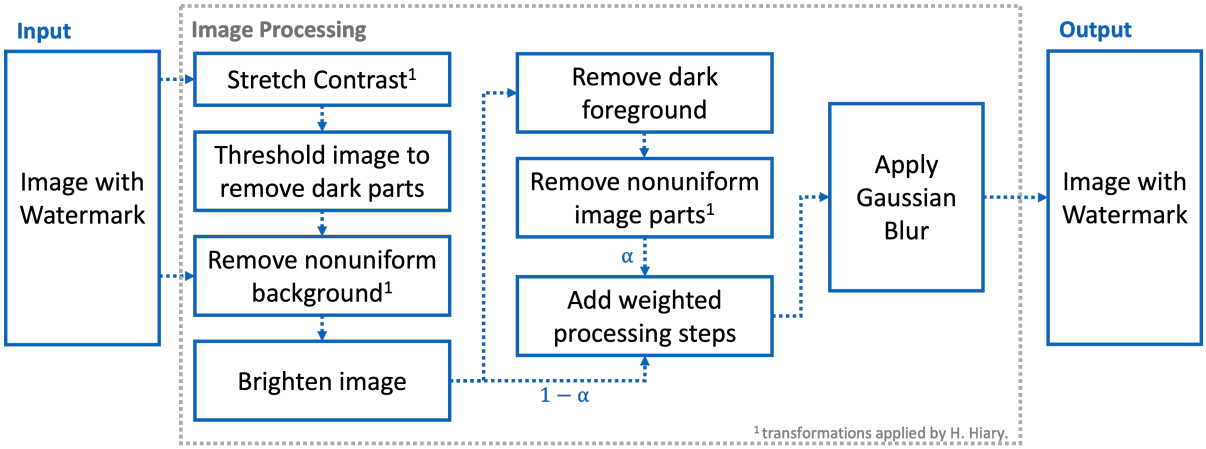


Figure 7: Flow Chart of the preprocessing steps for watermarks. The single steps are described in detail in subsection 3.2

Threshold image to remove dark parts The watermark is the lightest part of the image. To remove the darkest parts, especially the ink on a watermark, a binary mask of the watermark is produced. This mask is generated using inverse binary thresholding with a threshold value of $s = 70$ and a max binary value $\max_{bin} = 255$. The pixel intensity below the threshold is set to 255 (black) and the values above to 0 (white). The white parts of the mask then indicate which parts to in-paint in a next step using an algorithm based on Navier-Stokes equations [55]. An example is shown in Figure 8c.

Remove nonuniform background In order to remove noisy background and make the image more uniform, the black top-hat transformed image is subtracted from the white top-hat transformed image. The white top-hat transformation of a gray-scale image A and its structuring element B is described by

$$TopHat_{white} = A - (A \circ B), \tag{2}$$

where $\circ$ is morphological opening $A \circ B = (A \ominus B) \oplus B$ [56]. The white top-hat transformation returns the parts of an image, that are brighter than their surrounding and smaller than the structural element. Here, we use a filter size of 6. The black top-hat transformation of the image is described by

$$TopHat_{black} = A - (A \bullet B), \tag{3}$$

where $\bullet$ is morphological closing $A \bullet B = (A \oplus B) \ominus B$ [56]. The black top-hat transformation returns the parts of the image, that are darker than their surrounding and smaller than the structural element. Here, we use a filter size of 20. Hiary also uses this transformation in high work [29]. An example is shown in Figure 8d.

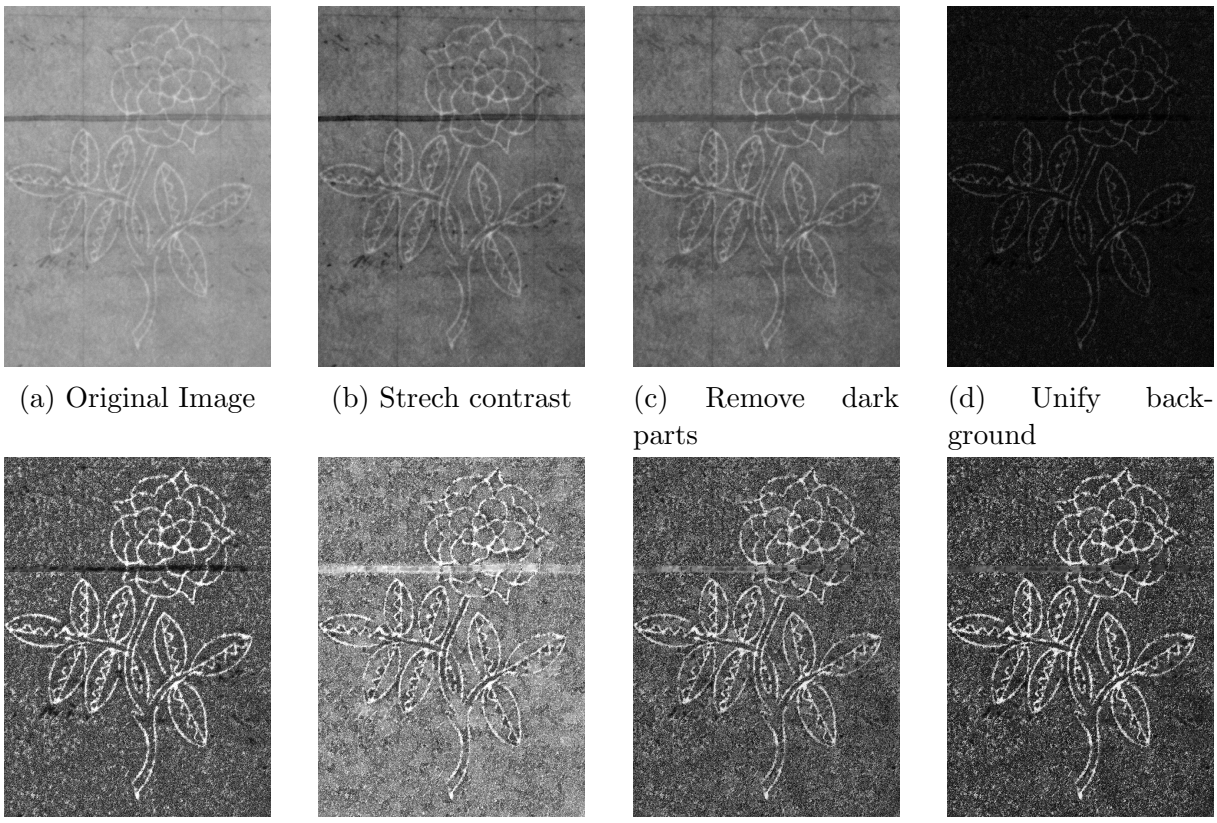
Brighten image As the image now contains a lot of dark gray and the difference between the background and the watermark is weak, the next step aims to increase the contrast of the image again and brighten it. Therefore, the image is scaled by the value $\alpha = 5$ and the value $\beta = 10$ is added to the scaled image. An example is shown in Figure 8e.

Remove dark foreground The darkest parts of the image can be partly removed by using the dilated image and getting the eroded image. Then the absolute difference between

the eroded image and the image from the brighten step is calculated. This produces a negative of the watermark. Then, the result is inverted again using bitwise inversion. As parameters, a structuring element size of three and six iterations are chosen. In dilation, the brighter areas get bigger, while the dark pars get smaller. Erosion operates on the opposite. The non-uniform background can then be improved by using morphological opening. Hiary again uses erosion and dilation in his work [29]. An example is shown in Figure 8f.

Remove nonuniform image parts The next step aims to remove non-uniform image parts. Therefore, morphological closing with a structuring element size of 20 is used. Closing of an image is used to "close" small lighter holes and irritations and unify the dark background. Then the image from the previous step is subtracted to increase the contrast of the watermark. An example is shown in Figure 8g.

Weighted image As different preprocessing steps have different advantages for different images, the image with the high contrast from the brightening step is added to the final image from the last processing step to get the final output for the watermark preprocessing. This avoids that some images get too bright or too dark. The brightened image is multiplied by 0.7 and added to 0.3 times the image from the last step. An example is shown in Figure 8h.



(a) Original Image (b) Strech contrast (c) Remove dark parts (d) Unify background

(e) Brighten (f) Remove dark foreground (g) Remove non-uniform background (h) Weighted image

Figure 8: Different preprocessing steps for watermarks.

As a final step, Gaussian blur is used to smooth unevenness's in the background. For 5% of the images, the preprocessing fails, resulting in very dark to black image. Primarily, if the original image is too dark, the preprocessing sometimes results in a bad output. More examples of watermark processing including examples of failures of the procedure are shown in Figure A.5.

3.3 Preprocessing of Tracings

3.3.1 Classical Image Processing Approach

In this subsection, we continue with the preprocessing of tracings. The tracings are already quite close to the state we eventually want them to be in for the comparison step. They are high contrast, less noisy and usually have no major degradations, such as written text on them. The background paper of the tracings is also usually in decent shape. However, most tracings still have aspects that need to be fixed to arrive at the watermark's contour. They often contain vertical and horizontal scaling lines as well as rulers for scale. Other tracings are either cut off or incomplete. Figure A.6 shows examples of these degradations. We also observe different background colours for individual images, which also makes preprocessing more difficult. This ca be due to the color of paper as well as lighting conditions.

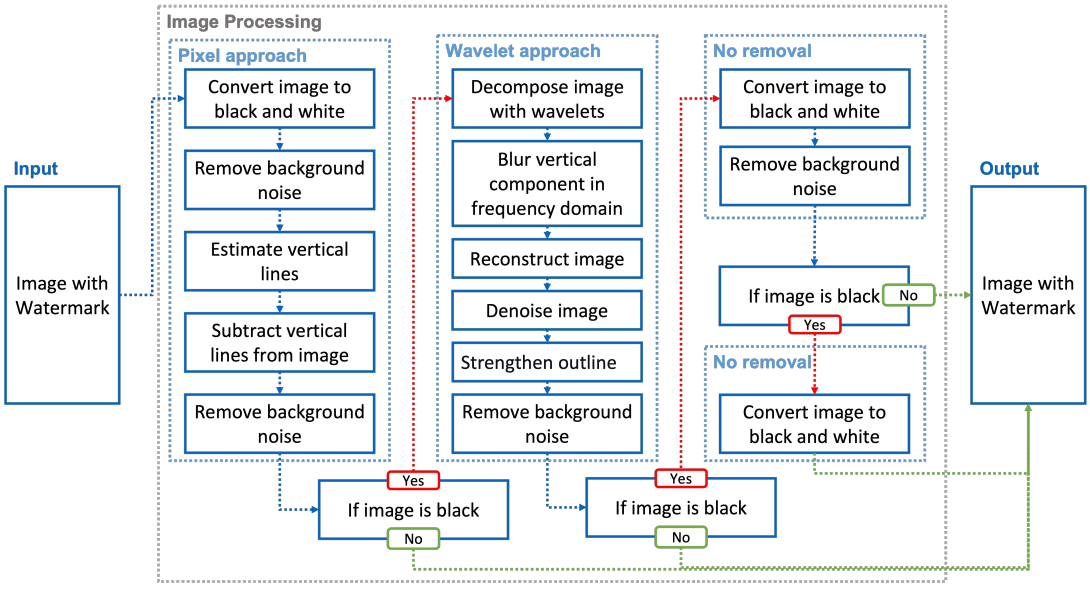


Figure 9: Preprocessing pipeline for tracings

The vertical lines particularly impact the overall outline of the watermark significantly. Furthermore, most tracings have these vertical lines. Thus the main objective of the preprocessing stage is to eliminate the presence of vertical lines in the tracings.
Detecting and removing vertical lines can be achieved with some success due to their distinct structure. We have attempted two approaches for this purpose. The first approach is pixel-based, which scans the image at a pixel level to identify vertical structuring elements. The second approach involves wavelet analysis [57], where we perform a base change, decompose the image into different components, and then blur the horizontal

components in the frequency domain. Our wavelet approached is based on [58]. Both approaches return the outline of the tracings as white curves on a black background.
Our two different approaches tend to work well for different types of images. They also sometimes erase too much of the image such that the outline is no longer recognizable. Therefore, we combine our two approaches to one overall pipeline. If one approach returns an unrecognizable outline, we use the other approach. We can determine with good accuracy if an image is almost entirely black and has no recognizable motif left by examining the percentages of entirely-black rows and columns. We first apply the pixel-based approach, and if it returns an almost entirely black image, we switch to the wavelet approach with relatively weak parameters. Using weak parameters in this sense means doing the wavelet transform on higher levels and using a weaker blurring in the frequency domain. Weak parameters lead to less consistent deletion of lines but, in turn, also produce an overall less blurred output image. If the wavelet approach also produces an almost black image, we perform noise removal as a preprocessing-step without attempting to delete the lines. Finally, if this process still results in an almost black image, we return a black-and-white version of the original image. For further details, please see the pipeline chart from Figure 9. We will now provide a more detailed explanation of the individual steps outlined in the pipeline chart in Figure 9. We begin with the function `If image is black`, followed by the functions in the pixel approach. Then, we explain the functions in the wavelet approach. The approach without deleting vertical lines is equivalent to the first two steps of the pixel approach. Observe, the function `Remove background noise` appears in all three approaches, but for the sake of space, we will explain it only once.

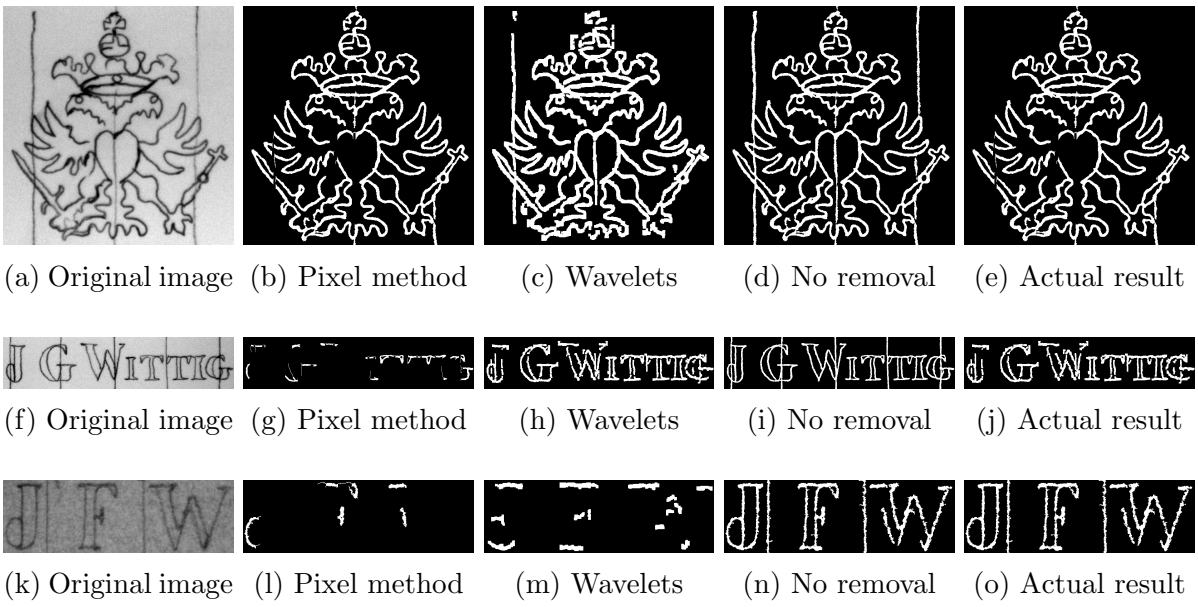
If image is black We calculate the percentage of rows and columns of a binary image that are entirely black. We found that images with more than 20% black rows or columns are too degraded to recognize the outline of the tracing.

Convert image to black and white We convert the input image into a black-and-white binary image. We then invert the resulting binary image.
Remove background noise The binary images often have noisy backgrounds full of little white dots. We remove these small white objects by removing all white objects with an area smaller than a specified value and an outline shorter than a specified length.
Find vertical lines This is the most essential part of the pixel method. Here we extract the vertical elements with at least one-pixel width and a specified length. We then use dilation and blurring to smooth these vertical lines and remove background noise.
Subtract vertical lines from the image We subtract the previously identified vertical elements from the image to create an output image free from vertical chain lines.

Decompose image with wavelets The image is decomposed into wavelets using Haar-wavelets [59], which are the most commonly used basis functions in wavelet transformations. We decompose the image into its horizontal, vertical, and diagonal elements.
Blur vertical element in the frequency domain We carry out a Fourier transformation on the vertical and blur the vertical elements in the frequency domain [60] using a Gaussian kernel.
Reconstruct image We reconstruct the image, first by applying the inverse Fourier transform [60] to the vertical elements and then by applying the inverse wavelet transfor-

mation for all elements.
Denoise image After blurring the image in the frequency domain to eliminate vertical lines, the resulting image tends to be quite noisy. Thus we use Otsu thresholding [61] and the Non-Local-Means algorithm [62] for denoising.
Strengthen outline The watermark is now usually weakly visible. We strengthen the outline by first dilating and then eroding the image. Finally, we invert the image such that the tracing is now a white line on a black background.

Figure 10 shows the results of our three preprocessing methods in columns two to four and the image our decisions logic ultimately chooses in the final column. In the first row the pixel approach gets chosen, in second row the wavelet approach and in third the chain lines are not removed at all, only the noise is removed from the image. Figure 11 illustrates the results of the individual steps in the pixel method. Similar Figures for the wavelet approach and preprocessing without vertical line removal can be found in the Appendix (Figure A.11 and Figure A.12, respectively).



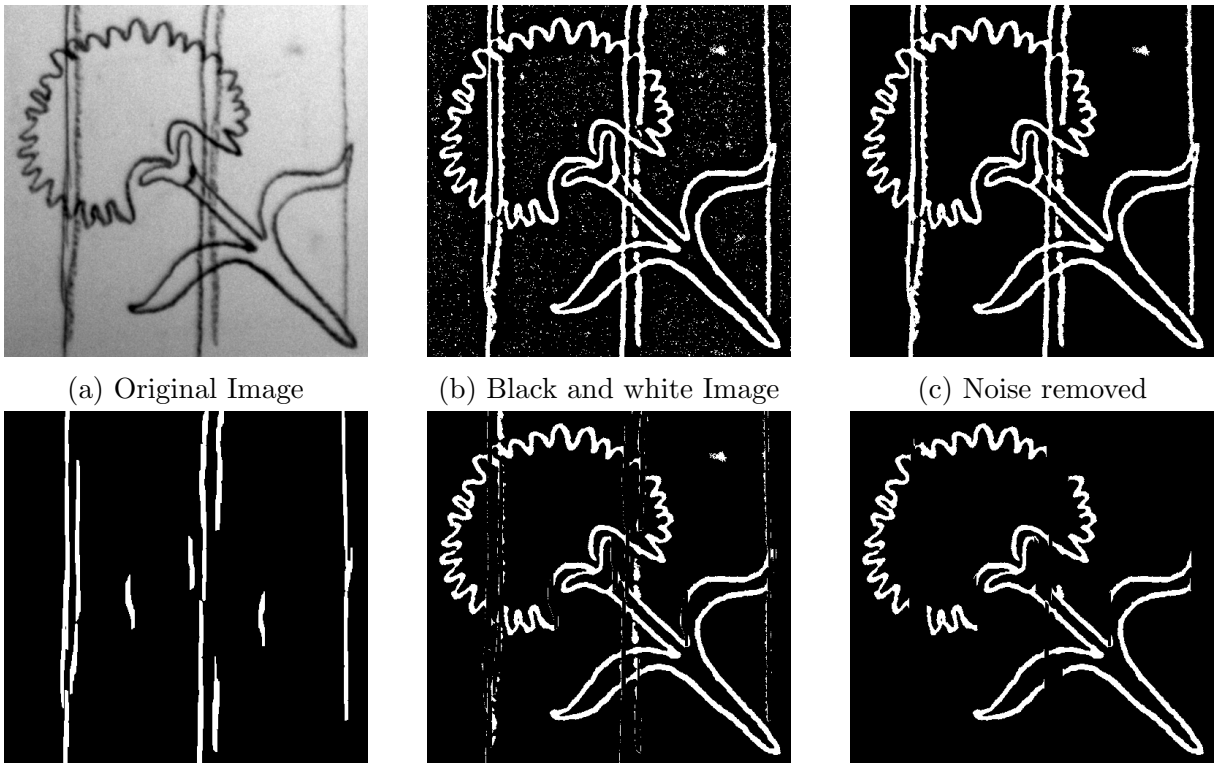
(a) Original image (b) Pixel method (c) Wavelets (d) No removal (e) Actual result

(f) Original image (g) Pixel method (h) Wavelets (i) No removal (j) Actual result

(k) Original image (l) Pixel method (m) Wavelets (n) No removal (o) Actual result

Figure 10: Selection logic for different methods in our preprocessing

While our pixel-based approach generally works well, it encounters difficulties when handling words since it sometimes unintentionally deletes parts of letters. On the other hand, the wavelet approach struggles with small and weakly drawn watermarks. The blurring of vertical components in the wavelet approach tends to mix the watermarks with other image components excessively, blurring the tracings motif and not merely removing the vertical lines. Both approaches appear to do better with larger watermarks and can struggle with small watermarks and writing. Examples of these issues can be seen in Figure A.7 and Figure A.8. Both methods have one more significant drawback: they do not have globally consistent parameters. Parameters that yield satisfactory results for one image might produce a bad result like an almost entirely black image for another tracing and vice versa. For examples of these consistency problems see Figure A.9 and Figure A.10.



(a) Original Image (b) Black and white Image (c) Noise removed

(d) Vertical lines (e) Deleted lines (f) Noise removed

Figure 11: Different preprocessing steps for tracings, removing vertical lines with pixel based method

We also unsuccessfully attempted to use Hough Lines [63] to identify the vertical lines. Utilizing Hough lines would offer the advantage of estimating mathematical linear objects, allowing us to investigate properties like spacing. However, we faced challenges finding suitable parameters, resulting in multiple Hough lines being estimated for each scaling line in the image. We could not effectively get around this problem. We also tried to use Canny edge detection [64] to eliminate the lines. However both our pixel based and wavelet approached had more promising results.

3.3.2 Conditional GAN Approach

As traditional image processing methods initially worked successfully only on a limited subset of images and failed on the others, we attempted another tracing preprocessing approach. Namely, we created an artificial dataset of sketches which resemble the tracings from out dataset and proceeded to train a conditional GAN with this paired dataset of sketches to learn how to generate the original (clean) images from the artificially noisy ones.
Specifically, we considered the dataset of 20,000 images from [65], which is a collection of grayscale rough sketches of a variety of objects. An example of a sketch can be seen in Figure 12a. The following effects were applied to the aforementioned sketches:

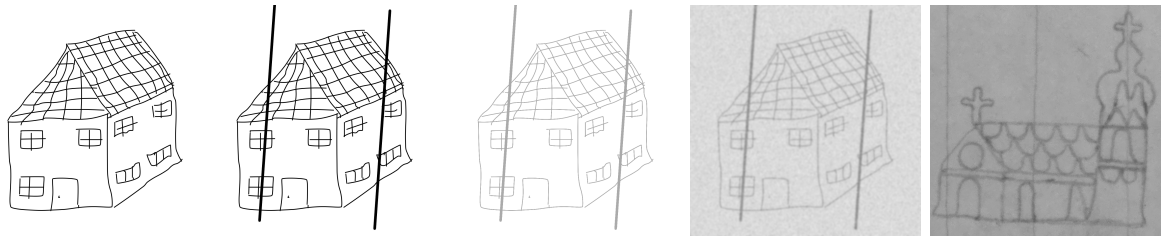
1. Roughly vertical lines were added on top of the image to emulate the traced delim-

iters of the paper sieve. These were randomly slightly inclined in either direction, see Figure 12b.

2. The sketch and the vertical lines were colored in gray to resemble a pencil, similarly to the tracings from our dataset, see Figure 12c.

3. Gray noise and blur were added across the entire image to emulate the overall look of the tracings, see Figure 12d.

Each step as well as the final result in comparison to the initial sketch and to a tracing from our DNB dataset can be seen in Figure 12.



(a) Raw sketch (b) Vertical Lines (c) Grey lines (d) Noisy sketch (e) DNB Tracing

Figure 12: Generation of the artifical sketch dataset for the tracing preprocessing cGAN

Using this preprocessing dataset, we trained a paired conditional GAN, namely pix2pix [41], to generate the initial sketches from the noisy ones.
Initially, we trained the GAN using the default hyperparameters of the model, the relevant ones being a learning rate of 0.0002 and a batch size of 1. The model was trained for 200 epochs, with a linear learning rate decay after 100 epochs.
The results of this model were visually indeterminate and during the training the generator model loss was not converging, rather jumping through high losses. This led us to decrease the learning rate to 0.00005 and 0.0000002 in 2 further trials. In the first, the loss was still jumping over local minima, whereas the second was steadily but slowly decreasing.
This led to a more structured approach to the hyperparameter search by:

- A shallow and wide search approach: randomly sampling 20 learning rate and batch size combinations within the intervals of [2e-8, 1e-4] for the learning rate and {2, 4, 8} for the batch size, and training these for 10 epochs with no learning rate decay.

- A deep and narrow approach: randomly sampling 12 combinations from the same interval for the learning rate and {2, 4} for the batch size, but training those for the full 200 epochs.

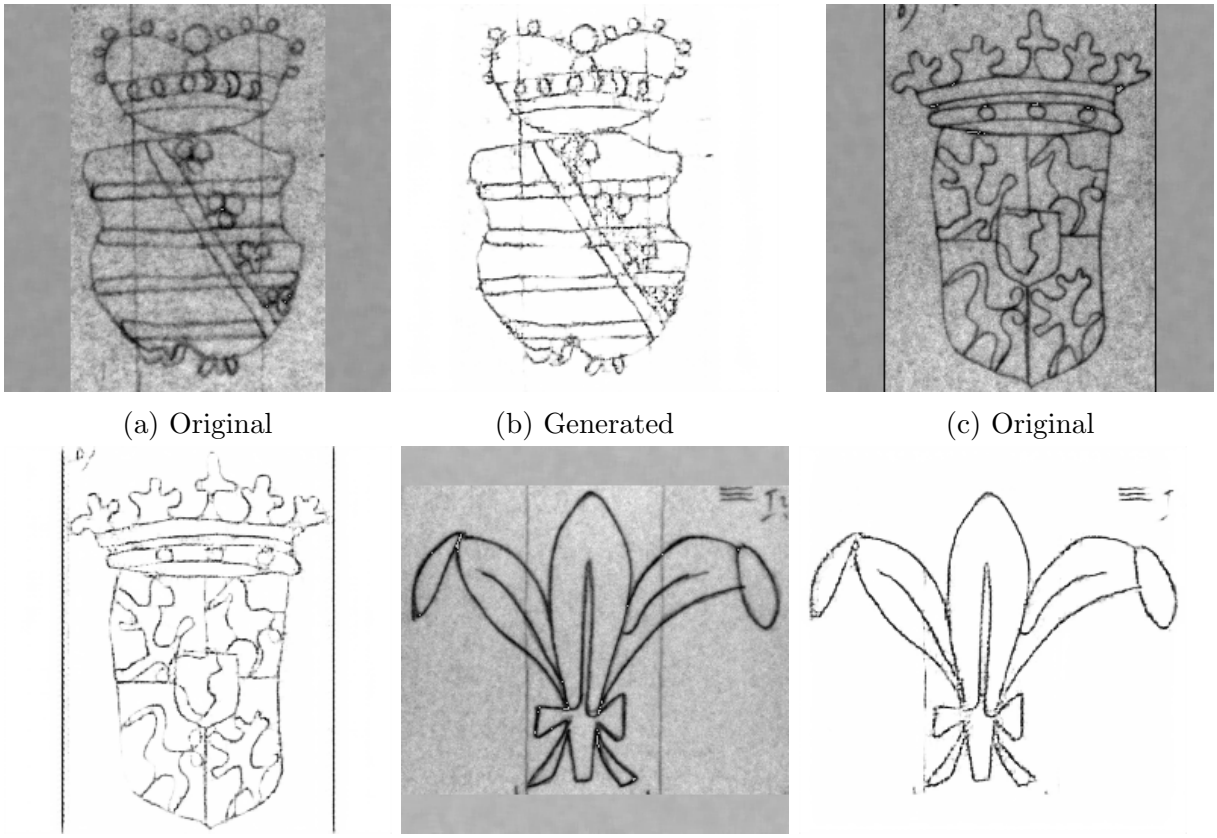
Larger batch sizes were not attempted due to the limitations of the GPU the models were trained on.
The results of the narrow approach are depicted in Table 2 and the results of the wide approach in Table 1. The calculated metrics were

1. the errors of the Generator Network: $G_{GAN} + G_{L1}$. G_{GAN} is the loss of the clean sketch generator, and G_{L1} is the so-called Cycle Consistency Loss, i.e. the difference between an original image and the result of this image being passed through the A-to-B generator and then the B-to-A generator. This loss helps to prevent the model from always returning the images from the training set.

2. the errors of the Discriminator Network: $D_{real} + D_{fake}$. D_{real} is the prediction loss for real images, i.e. the original sketches, and D_{fake} is the prediction loss for generated clean sketches.

As for our task we only need the Generator Network, because we are transforming a noisy sketches into clean ones, the relevant column is $G_{GAN} + G_{L1}$. The best models for each test were T12 with learning rate $1.029e-07$ and batch size 2 for the narrow test, and T18 with learning rate $1.06498e-06$ and batch size of 4 for the wide test.

The best result of the wide and shallow tests were then trained for the usual 200 epochs. Both models provided approximately the same results - they managed to create black and white images with, in most cases, minimal noise, but have not been able reliably remove the vertical lines from the images. Example transformations are depicted in Figure 13.



(a) Original (b) Generated (c) Original

(d) generated (e) Original (f) Generated

Figure 13: Original tracings and generated sketches using the sketch preprocessing cGAN

4 Pipeline architecture and model framework

4.1 Overall pipeline and data model

The pipeline and our models build upon the original dataset from DNB [18]. This data is cropped, then the watermarks are processed using classical image processing (subsection 3.2) and the tracings are processed using either classical image preprocessing or a cGAN (subsection 3.3). The CycleGAN for sketch generation is trained on the 2 classes of data - preprocessed watermarks and preprocessed tracings, in order to learn the transformation from one class to another. We only use the watermark to tracing generation, as preprocessed tracings are equivalent to a sketch. The generated sketches combined with the processed tracings are then stored in a new data bucket for the nearest neighbor search. Our full data model and its connection to the preprocessing for tracings and training of the CycleGAN is visualized in the first two rows of Figure 14.
The pipeline for a new image starts with an original scan or photograph of a watermark. This image is processed to enhance the visibility of the watermark, then the CycleGAN is used for sketch generation. We use a ResNet18 to extract a feature vector from the generated sketch. The most similar images from the "generated sketch and processed tracings" dataset is then found using nearest neighbor search. Finally, the n closest images are returned. Figure 14 shows a detailed overview of the processes.

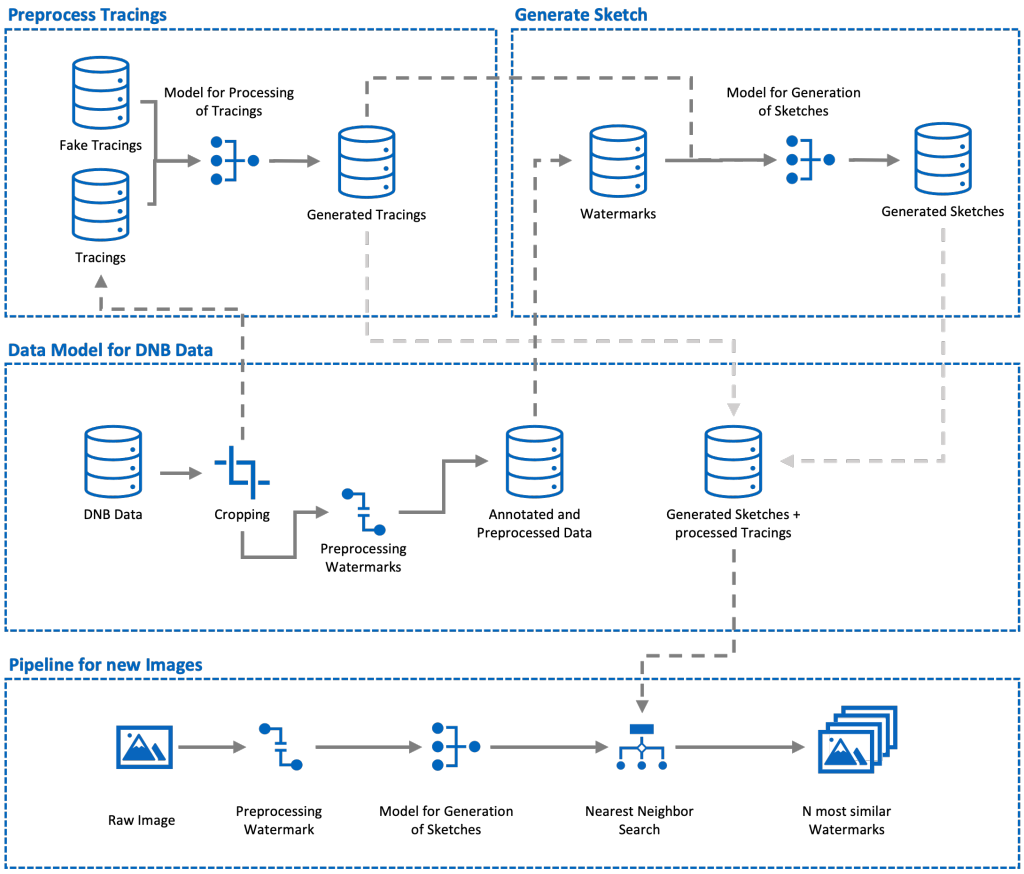


Figure 14: Connection between the different versions of the data and our full data model, preprocessing of tracings using a CycleGAN and the final pipeline for new watermarks.

4.2 Watermark translation

As mentioned, at the core of our approach is the idea of transforming pictures of watermarks into binary (black/white) sketches.
Considering the structure of our dataset, namely the lack of watermark - tracing pairs, and the lack of other large paired watermark datasets, we resorted to an unpaired image translation approach. We focused on the classical CycleGAN [46] for the following reasons:

1. Proven applicability to various image translation tasks.
2. Code availability and extensive documentation.
3. Improvements that followed the original paper mostly focus on runtime and memory efficiency, which was not an issue in our case, due to the size of our dataset.

In our case, the input to the CycleGAN is the watermarks and tracings, both preprocessed according to the procedures described earlier in this work (see 3.2 3.3). More specifically, we used the full preprocessing pipeline for watermarks and mostly focused on the cGAN approach for tracing due to the fact that is was more consistent across all images.
Apart from training multiple versions of the CycleGAN, we trained a CUT model [47] with an earlier version of our preprocessed data. This model did not land any palpable improvements in translation or runtime, presumably because the improvements in the original paper focus mostly on memory efficiency and runtime, which would only have a considerable impact on larger datasets. Thus, we later only trained the CycleGAN.
Similarly to the Sketch Denoising model, CycleGAN was initially trained by trial and error and finally a hyperparameter search was executed in order to find the optimal hyperparameter values from within a reasonable range.
Specifically, we first trained a proof-of-concept CycleGAN model on a different dataset [37] with only ∼150 images per class, in order to test the potential of the model.
Afterwards, a base version of a model was trained using the DNB data and the default hyperparameters of the model: learning rate of 0.0002 and batch size of 1. As this model exploded after ∼75 epochs and returned pitch black images, we retrained it with a learning rate of 0.00005 (4 times less) to receive first visible results.
After further preprocessing of both the watermarks and tracings and more data labeling, we performed a shallow and wide hyperparameter search - randomly sampling 20 learning rate and batch size combinations within the intervals of [2e-8, 1e-4] for the learning rate and {2, 4} for the batch size, and training these for 10 epochs with no learning rate decay. The results of the search are depicted in Table 3.
As the task at hand is transforming watermarks (class A) into sketches (class B), the column of importance is $G_A + G_{cycle_A} + D_B$, which contains the A to B generator loss and its cycle consistency loss [46], as well as the B class discriminator loss. From this test, we established an optimal learning rate of $5.6925e-05$ and a batch size of 4 ($T19$).
Finally, we trained the model with these parameters and tried the following 3 variations:

1. Standard 200 epoch training, with linear learning rate decay after 100 epochs.
2. 400 epoch training with linear learning rate decay after 200 epochs.
3. 400 epoch training with cosine learning rate decay after 200 epochs.

The results of the training will be discussed in the next chapter.

4.3 Sketch Comparison

In this section we describe the comparison of sketches in order to return nearest neighbors. Firstly, our comparisson method needs to be fast to ensure quick retrieval of results from our already large dataset. Additionally, scalability is crucial as DNB is continuously digitizing its collection, so the method should handle increasing datasets without retraining. We also expect a certain level of consistency, where if we search for a watermark showing a bird for example, we want to see predominantly bird-related images. Lastly, the method should be stable even when dealing with specific types of transformations, such as noise or short breaks in the outline of the images. We use a nearest neighbour search based on feature vectors of a pre-trained CNN to find similar images.
We investigated Hu-Moments [66] and SIFT [67] for nearest neighbor comparison, with our SIFT approach being based on [68]. As direct nearest neighbor comparisons of key point vectors are computationally unfeasible, we simplified the key points by using k-means [69] on the key point descriptors. Then, we employed bag-of-words (BoW) [70] and term frequency-inverse document frequency (TF-IDF) [71] approaches with the groups found in the k-means step to reduce the size of the fixed point vectors. Both Hu-Moments and SIFT methods had major drawbacks: Hu-Moments were slow and did not reurn images with similar motifs. SIFT struggled with eroded and non-smooth binary images due to its reliance on local gradients.
Using a pre-trained model offers several advantages. Firstly, we possess poorly labeled data, making it hard to properly train our own network. Secondly, leveraging pre-trained models saves considerable time and computational resources as we can directly extract feature vectors without training a new model from scratch. This is especially advantageous since DNB is still in the process of digitizing its collection, eliminating the need for regular retraining. Finally, pre-trained models are robust and transferable since they are trained on large and diverse datasets. This robustness is beneficial for our application, given the significant differences between images dataset.
We tested different pre-trained CNNs, including the commonly used VGG16 [72] and ResNet18 [73]. We eventually settled on ResNet18. We extract the network's final layer before classification to obtain a feature vector for the input image. However, these large feature vectors make direct nearest neighbor comparisons impractical. To mitigate this issue, we experimented with Principal Component Analysis (PCA) [74] for dimension reduction, but ultimately, we used the `Spotify Annoy` [75] library for approximate nearest neighbor search. The `Spotify Annoy` algorithm builds binary trees to divide a high-dimensional space and create efficient data structures, which are easier to search.
We compared the different CNNs and nearest neighbor approximation methods by applying transformations to our preprocessed tracings and then comparing the average position of the original image, its median position, and frequency, being among the top 10, top 20, and top 50 nearest neighbors. Figure A.13 shows an example of the transformations used. No method emerged as clearly superior. However, due to its strong performance with partially disintegrated watermarks and high overall accuracy, we opted for the ResNet18 network for feature extraction and the Spotify annoy algorithm for the nearest neighbour search. One drawback is that the Annoy Index needs to be restructured whenever new watermarks are digitized, but this process is simple and relatively fast. Our results will be discussed in section 5.

5 Results

5.1 Qualitative results of the sketch generation

All 3 models described in the previous chapter, provided very similar outputs, with the last two having marginally better results. Towards the end of the project, the classical preprocessing approach was further improved and we trained a CycleGAN model with the same parameters for 200 epochs. This model provided mixed results due to noisiness. Thus, our best model(s) used classical preprocessing for watermarks and cGAN preprocessing for tracings. They were trained for 400 epochs with a learning rate of $5.6925e-05$ and a batch size of 4, with cosine/linear learning rate decay after 200 epochs. All of the following results regard these 2 models.

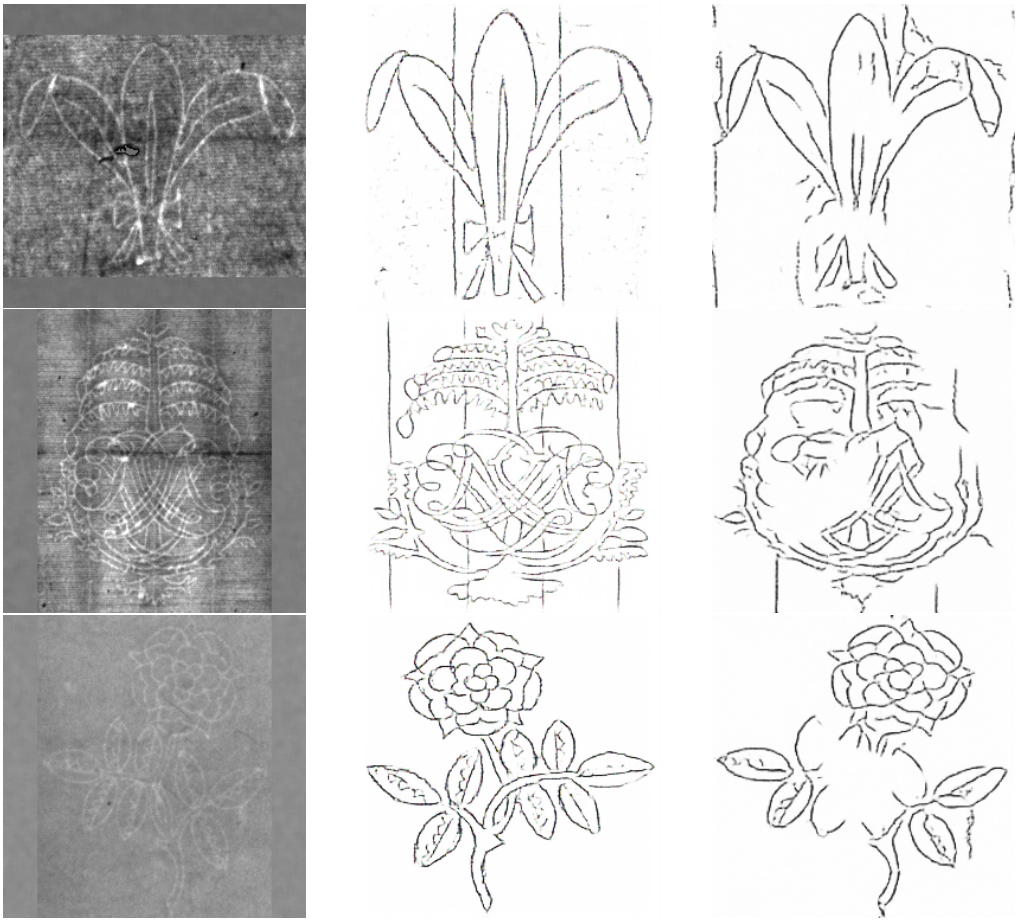


Figure 15: Example outputs of CycleGAN in triplets: Original watermark - Sketch of a similar tracing - Output of the CycleGAN

Example results for CycleGAN predictions for watermarks from the DNB dataset are depicted in Figure A.14 on the right. The image on the left is the original watermark and the image in the middle is a similar preprocessed tracing, which estimates the expectations for a good output. Further examples are linked in the Appendix.
It can be inferred that the model has the ability to generate visually similar sketches, fully preserving the position and rotation of the original watermark. It accurately redraws dif-

Figure 16: Unsuccessful example output of CycleGAN. Original watermark - Sketch of a similar tracing - Output of the CycleGAN

ficult curves and is often able to reproduce even low level details in the watermarks. In a majority of cases the model can also alleviate the difference in contrast within and across different watermarks.
However, the well-generated images can also contain various artifacts, like unnecessary lines or white speckles, where the expected outline is missing. It also often includes the vertical lines of the sieve in the resulting image. Although they are often visible in the original watermark image, they are more often introduced due to their presence in the preprocessed sketches which we feed to the network.
Although a minority, the model also happens to generate images from which it is difficult to comprehend what the original watermark represented. Such an example can be observed in fig:Cyclegan-bad-outputs. From our observations this happens more likely in the following types of images:

- Very low contrast images, where the watermark is hard to see even with the naked eye.
- Textual watermarks with improper (squashed) resizing - although this was subsequently improved with proper resizing.
- Partial watermarks where only part of the outline is visible.

Despite this issues, the CycleGAN was able to generate reasonable sketches for a majority of watermarks, and for clean and high-contrast watermarks images, created sketches almost indistinguishable from preprocessed tracings.
We believe the model can be further improved to alleviate the problems described above by better preprocessing and further model tuning.

5.2 Quantitative analysis of the full pipeline

In this subsection, we analyse the accuracy of our pipeline on a quantitative scale. For this purpose, we manually created a test dataset containing 22 classes of watermarks and corresponding tracings from the dataset provided by DNB. The count number of observations for watermarks and tracings may vary on a large scale within one class ranging from two observations to 146 per class. The data is cropped and annotated in an analogous manner, the unique image identifier and the corresponding class identifiers are

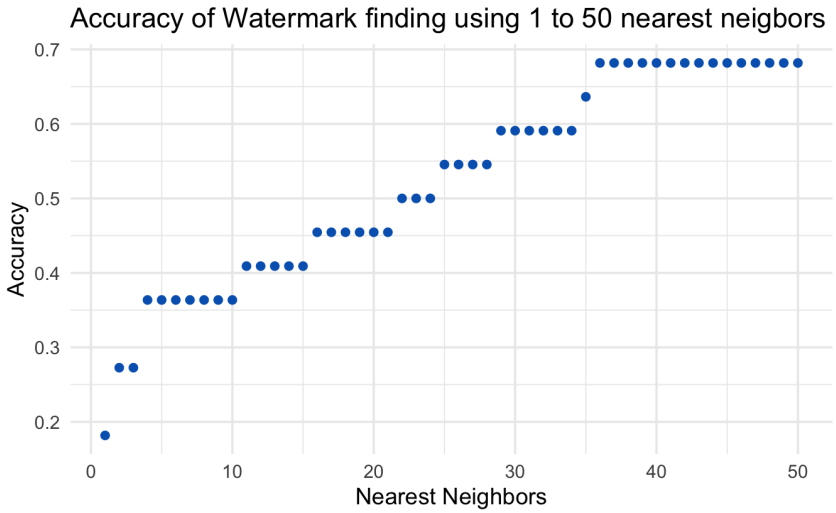


Figure 17: Accuracy of the latest model over the number of nearest neighbors taken into account.

stored in a table. This table contains 567 images in total. The test set was excluded from later trainings of our CycleGANs. Afterward, all watermarks and tracings from the test set are added to the data for sketch comparison.
For testing, we first randomly sample one watermark from each class in the test set. Next we use our pipeline. Here, we apply our image preprocessing (subsection 3.2)to all watermarks in the test set, and then generate a sketch using a CycleGAN (subsection 4.2). Finally, we get the n nearest neighbors using the sketch comparison from subsection 4.3. The accuracy of our pipeline depending on n neighbors is calculated by

$$accuracy_n = \frac{\sum^{22} \mathbb{1}_{\{\#\text{ NN from same class} \geq 1\}}}{22} \tag{4}$$

where we count for how many classes we can find at least one pair of watermark and corresponding sketch(es) within the n nearest neighbors (NN). The larger we choose n, the higher the probability to find the sketch(es) we are interested in.
The accuracy of the model is evaluated using different values for n. The actual calculation is partly supported by a script. However, as the test set is incomplete and some watermarks and tracings were missed in some classes during the collection of the test data, the calculation of the accuracy is supported by a manual component where we double checked the output of the pipeline and actively searched for the matching watermark(s). The position and the name of the first match is then added to a table.

We use our latest model to estimate the accuracy of our pipeline. Using 25 nearest neighbors we can achieve an accuracy of more than 50% while for 50 nearest neighbors we achieve an accuracy of more than 68%. Figure 17 shows the accuracy over the number of nearest neighbors taken into account. For 15 out of 22 test watermark classes, at least one matching watermark can be found within the 50 nearest neighbors in the full data base which contains over 6600 processed images.

6 Conclusion

Our primary goal in this project was to develop a framework which will find similar watermarks and tracings in a database, when we present a new photograph or scan of a historic watermark. We tackled this problem by training a CycleGAN to generate a sketch of the presented watermark. Using this model, we generated sketches for all watermarks from the dataset and combined them with the sketches produced by preprocessing tracings. We then used a pre-trained ResNet18 neural network to extract a feature vector from the sketch. Finally, we used the Spotify Annoy algorithm, which enables an efficient approximate nearest neighbor search to compare a specific sketch with our entire database. By this approach we could get an accuracy of over 68% when using 50 nearest neighbors.
The novelty of our approach, compared to previous methods presented in the literature, lies in the use of a generative adversarial network. This enables us to add new watermarks and tracings to the comparison database without the need to retrain the CycleGAN each time. Additionally, our approach is not dependent on predefined class labels for all watermarks, making it more time-efficient for the unlabeled dataset of DNB. As we keep record of the original name of the image, the database can be easily connected to the metadata provided by the DNB to get more information about each watermark. Furthermore, our approach has the potential to be extended to other data bases of watermarks and to include them in the comparison data base.
In a next step, our framework to find similar watermarks and tracings in a database can be integrated with a front-end application to provide users the opportunity to use our pipeline for their own research on historic watermarks. Moreover, the examples for which our pipeline fails can potentially be reduced by including an user interface for watermark preprocessing to allow for manual adjustments for some parameters of the image processing functions. Thereby, watermarks with an unusable output after preprocessing can be avoided. Beyond that, different GANs, as well as other preprocessing approaches, can be explored to improve the quality of the generated sketches. Finally, one could train a CNN (e.g., a ResNet18) from scratch to obtain the features for nearest neighbor search. This could potentially increase accuracy in image comparison. However, this approach would necessitate manual effort to create a labeled dataset.
In conclusion, this work contributes to the goal of facilitating streamlined and accelerated research on watermarks. The usage of deep learning techniques demonstrated in our project showcases the enormous support artificial intelligence can provide to libraries, such as the DNB, in their efforts to digitize and preserve historic works.

References

[1] D. Hunter, *Papermaking: The History and Technique of an Ancient Craft*. Dover Books Explaining Science, Dover Publications, 1978.

[2] "Die Geschichte von Papier." https://papierkannmehr.de/innovation/die-geschichte-von-papier. Accessed: June 26, 2023.

[3] P. Schule, "Geschichte der Papierproduktion." https://www.papiermachtschule.at/papierproduktion/geschichte/. Accessed: June 26, 2023.

[4] J. Damberger, "Geschichte der Papierherstellung," *LWF - aktuell*, vol. 54, pp. 43–45, Jul 2006. ISSN 1435 - 4098.

[5] T. Barrett and et al., "European papermaking techniques 1300-1800." Paper through Time: Nondestructive Analysis of 14th- through 19th-Century Papers, 2022. Last modified September 02, 2022.

[6] N. B. Fuller, "A brief history of paper.." Website, July 2002. Accessed: June 26, 2023.

[7] L. Mueller, "Understanding paper structures, watermarks, and a conservator's passion." https://harvardartmuseums.org/article/understanding-paper-structures-watermarks-and-a-conservator-s-passion, May 2021. Accessed: June 26, 2023.

[8] N. Harris, *Paper and Watermarks as Bibliographical Evidence*. Institute d'histoire du livre, Lyon, 2017.

[9] "Watermarks & foolscaps: Exploring the history of paper production." Alembic Rare Books Blog, June 19 2015.

[10] J. Dabrowski, *Paper Manufacture in Central and Eastern Europe Before the Introduction of Paper-making Machines*. July 2008.

[11] C.-M. Briquet, *Les filigranes: dictionnaire historique des marques du papier des leur apparition vers 1282 jusqu' en 1600*. Geneve: F. Burger, 1907.

[12] W. A. Churchill, *Watermarks in Paper in Holland, England, France, etc. in the XVII and XVIII Centuries and Their Interconnection*. Amsterdam: M. Hertzberger, 1935.

[13] C.-M. Briquet, "Briquet online." https://briquet-online.at/BR.php?IDtypes=100&lang=fr, 2021. Accessed: July 11, 2024.

[14] G. Piccard, "Piccard online." https://www.piccard-online.de/start.php. Accessed: July 11, 2024.

[15] "The bernstein consortium - the memory of paper." https://www.memoryofpaper.eu/BernsteinPortal/appl_start.disp. Accessed: July 11, 2024.

[16] "The bernstein consortium - the memory of paper image based paper expertise and history." https://bernstein.oeaw.ac.at. Accessed: 2023-06-25.

[17] International Association of Paper Historians, "Standards - paper history." http://www.paperhistory.org/Standards/. Accessed: June 26, 2023.

[18] "Historical paper collections." https://www.dnb.de/EN/Sammlungen/DBSM/PapierhistorischeSammlungen/papierhistorischeSammlungen.html. Accessed: 2023-06-25.

[19] E. Heawood, *Watermarks Mainly of the 17th and 18th Centuries*. Hilversum: Paper Publications Society, 1950.

[20] G. Piccard, *Die Wasserzeichenkartei Piccard im Hauptstaatsarchiv Stuttgart: Findbuch*. Stuttgart: Kohlhammer, 1961–1996.

[21] A. H. Stevenson, "Shakespearian dated watermarks," *Studies in Bibliography*, vol. 4, pp. 159–164, 1951.

[22] N. E. Ash, "Watermark research: Rembrandt prints and the development of a watermark archive," *The Paper Conservator*, vol. 10, no. 1, pp. 64–69, 1986.

[23] K. Nicholson, "Making watermarks meaningful: Significant details in recording and identifying watermarks," in *The Book and Paper Group Annual*, vol. 1, American Institute for Conservation of Historic and Artistic Works, 1982.

[24] C. Rauber, P. Tschudin, and T. Pun, "Retrieval of images from a library of watermarks for ancient paper identification," in *Proceedings of EVA'97 - Elektronische Bildverarbeitung und Kunst, Kultur, Historie*, vol. vol. 14, (Berlin, Germany), November 12-14 1997.

[25] D. Stewart, R. A. Scharf, and J. S. Arney, "Techniques for digital image capture of watermarks," *Journal of Imaging Science and Technology*, vol. 30, no. N, pp. 261–267, 1995.

[26] P. Zamperoni, "Wasserzeichenextraktion aus digitalisierten bildern mit methoden der digitalen bildsignalverarbeitung," *Das Papier*, vol. 43, no. Jahrgang, Heft 4, pp. 133–143, 1989.

[27] D. L. Gants, "Pictures for the page: Techniques in watermark reproduction, enhancement, and analysis," *Unpublished*, April 1994.

[28] D. L. Gants, "The application of digital image processing to the analysis of watermarked paper and printer's ornament usage in early printed books," in *New Ways of Looking at Old Texts II* (W. S. Hill, ed.), (Tempe, AZ), pp. 133–148, Renaissance English Text Society, 1998.

[29] H. Hiary, *Paper-based watermark extraction with image processing*. PhD thesis, 10 2008.

[30] V. Belov, V. Esipova, V. Kalaida, and V. Klimkin, "Physical and mathematical methods for the visualization and identification of watermarks," *Solanus*, vol. 13, pp. 80–92, 1999.

[31] C. R. Johnson, "Decision trees for watermark identification in rembrandt's etchings," *Journal of Historians of Netherlandish Art*, vol. 12, no. 2, 2020.

[32] A. C. Weislogel, C. Richard Johnson, A. House, K. Martucci, S. Siegler, S. J. Lim, K. Ferreira, and M. Canfield, "The wire project at cornell: An interactive decision tree approach for the rapid identification of watermarks in rembrandt's etchings," in *2018 52nd Annual Conference on Information Sciences and Systems (CISS)*, pp. 1–6, 2018.

[33] S. F. Gorske, C. R. Johnson, Jr., W. A. Sethares, M. H. Ellis, and P. Messier, "Moldmate identification in pre-19th-century european paper using quantitative analysis of watermarks, chain line intervals, and laid line density," 2020.

[34] D. Picard, T. Henn, and G. Dietz, "Non-negative dictionary learning for paper watermark similarity," pp. 130–133, 2016.

[35] V. Pondenkandath, M. Alberti, N. Eichenberger, R. Ingold, and M. Liwicki, "Identifying cross-depicted historical motifs," 2018.

[36] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255, Ieee, 2009.

[37] X. Shen, I. Pastrolin, O. Bounou, S. Gidaris, M. Smith, O. Poncet, and M. Aubry, "Large-scale historical watermark recognition: dataset and a new consistency-based approach," 8 2019.

[38] O. Bounou, T. Monnier, I. Pastrolin, X. Shen, C. Bénévent, M.-F. Limon-Bonnet, F. Bougard, M. Aubry, M. Smith, O. Poncet, and P.-G. Raverdy, "A web application for watermark recognition," *J. Data Min. Digit. Humanit.*, vol. 2020, 2020.

[39] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial networks," 2014.

[40] M. Mirza and S. Osindero, "Conditional generative adversarial nets," *arXiv preprint arXiv:1411.1784*, 2014.

[41] P. Isola, J.-Y. Zhu, T. Zhou, and A. A. Efros, "Image-to-image translation with conditional adversarial networks," in *Computer Vision and Pattern Recognition (CVPR), 2017 IEEE Conference on*, 2017.

[42] H. J. Andrade and B. J. Fernandes, "Synthesis of satellite-like urban images from historical maps using conditional gan," *IEEE Geoscience and Remote Sensing Letters*, vol. 19, pp. 1–4, 2020.

[43] J. Song, J. Zhang, L. Gao, X. Liu, and H. T. Shen, "Dual conditional gans for face aging and rejuvenation.," in *IJCAI*, pp. 899–905, 2018.

[44] M. Li, Z. Lin, R. Mech, , E. Yumer, and D. Ramanan, "Photo-sketching: Inferring contour drawings from images," *WACV*, 2019.

[45] T.-C. Wang, M.-Y. Liu, J.-Y. Zhu, A. Tao, J. Kautz, and B. Catanzaro, "High-resolution image synthesis and semantic manipulation with conditional gans," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 8798–8807, 2018.

[46] J.-Y. Zhu, T. Park, P. Isola, and A. A. Efros, "Unpaired image-to-image translation using cycle-consistent adversarial networks," in *Computer Vision (ICCV), 2017 IEEE International Conference on*, 2017.

[47] T. Park, A. A. Efros, R. Zhang, and J.-Y. Zhu, "Contrastive learning for unpaired image-to-image translation," in *European Conference on Computer Vision*, 2020.

[48] A. Almahairi, S. Rajeshwar, A. Sordoni, P. Bachman, and A. Courville, "Augmented cyclegan: Learning many-to-many mappings from unpaired data," in *International conference on machine learning*, pp. 195–204, PMLR, 2018.

[49] Z. Shen, S. K. Zhou, Y. Chen, B. Georgescu, X. Liu, and T. Huang, "One-to-one mapping for unpaired image-to-image translation," in *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pp. 1170–1179, 2020.

[50] J. Harms, Y. Lei, T. Wang, R. Zhang, J. Zhou, X. Tang, W. J. Curran, T. Liu, and X. Yang, "Paired cycle-gan-based image correction for quantitative cone-beam computed tomography," *Medical physics*, vol. 46, no. 9, pp. 3998–4009, 2019.

[51] Y. Lu, Y.-W. Tai, and C.-K. Tang, "Attribute-guided face generation using conditional cyclegan," in *Proceedings of the European Conference on Computer Vision (ECCV)*, September 2018.

[52] Ł. Maziarka, A. Pocha, J. Kaczmarczyk, K. Rataj, T. Danel, and M. Warchoł, "Mol-cyclegan: a generative model for molecular optimization," *Journal of Cheminformatics*, vol. 12, no. 1, pp. 1–18, 2020.

[53] W. Li and J. Wang, "Residual learning of cycle-gan for seismic data denoising," *Ieee Access*, vol. 9, pp. 11585–11597, 2021.

[54] M. Tkachenko, M. Malyuk, A. Holmanyuk, and N. Liubimov, "Label Studio: Data labeling software," 2020-2022. Open source software available from https://github.com/heartexlabs/label-studio.

[55] M. Bertalmio, A. Bertozzi, and G. Sapiro, "Navier-stokes, fluid dynamics, and image and video inpainting," in *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, vol. 1, pp. I–I, 2001.

[56] P. Maragos, "Chapter 13 - morphological filtering," in *The Essential Guide to Image Processing* (A. Bovik, ed.), pp. 293–321, Boston: Academic Press, 2009.

[57] I. Daubechies, "The wavelet transform, time-frequency localization and signal analysis," *IEEE Transactions on Information Theory*, vol. 36, no. 5, pp. 961–1005, 1990.

[58] B. Münch, P. Trtik, F. Marone, and M. Stampanoni, "Stripe and ring artifact removal with combined wavelet — fourier filtering," *Opt. Express*, vol. 17, pp. 8567–8591, May 2009.

[59] A. Haar, "Zur theorie der orthogonalen funktionensysteme," *Mathematische Annalen*, vol. 69, no. 3, pp. 331–371, 1910.

[60] R. N. Bracewell and R. N. Bracewell, *The Fourier transform and its applications*, vol. 31999. McGraw-Hill New York, 1986.

[61] N. Otsu, "A threshold selection method from gray-level histograms," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 9, no. 1, pp. 62–66, 1979.

[62] A. Buades, B. Coll, and J.-M. Morel, "A non-local algorithm for image denoising," in *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, vol. 2, pp. 60–65 vol. 2, 2005.

[63] R. O. Duda and P. E. Hart, "Use of the hough transformation to detect lines and curves in pictures," *Communications of the ACM*, vol. 15, no. 1, pp. 11–15, 1972.

[64] J. Canny, "A computational approach to edge detection," *IEEE Transactions on pattern analysis and machine intelligence*, no. 6, pp. 679–698, 1986.

[65] M. Eitz, J. Hays, and M. Alexa, "How do humans sketch objects?," *ACM Trans. Graph. (Proc. SIGGRAPH)*, vol. 31, no. 4, pp. 44:1–44:10, 2012.

[66] M. Hu, "Visual pattern recognition by moment invariants," *IRE Transactions on Information Theory*, vol. 8, pp. 179–187, 1962.

[67] D. G. Lowe, "Object recognition from local scale-invariant features," in *Proceedings of the seventh IEEE international conference on computer vision*, vol. 2, pp. 1150–1157, Ieee, 1999.

[68] V. Mishra, "Bag of visual words (bag of features)," November 2020. Published in Analytics Vidhya.

[69] S. P. Lloyd, "Least squares quantization in pcm," *IEEE Transactions on Information Theory*, vol. 28, no. 2, pp. 129–137, 1982.

[70] J. Sivic and A. Zisserman, "Efficient visual search of videos cast as text retrieval," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 31, no. 4, pp. 591–606, 2009.

[71] H. P. Luhn, "A statistical approach to mechanized encoding and searching of literary information," *IBM Journal of Research and Development*, vol. 1, no. 4, pp. 309–317, 1957.

[72] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2015.

[73] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015.

[74] K. P. F.R.S., "Liii. on lines and planes of closest fit to systems of points in space," *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, vol. 2, no. 11, pp. 559–572, 1901.

[75] E. Bernhardsson, "ANNOY library." `https://github.com/spotify/annoy`, 2018. Accessed: 2023-07-13.

A Appendix

A.1 Additional material for chapter 1

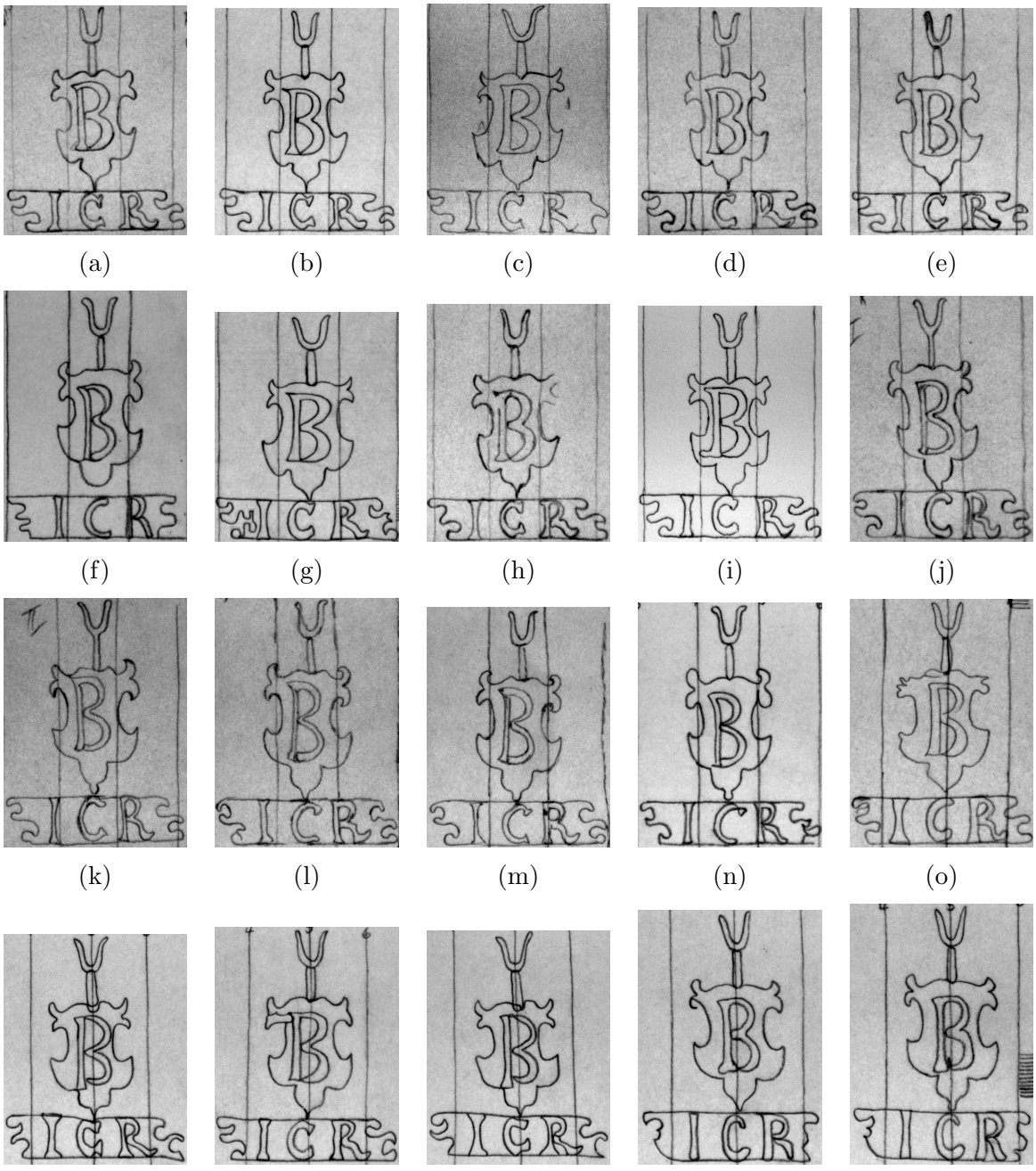
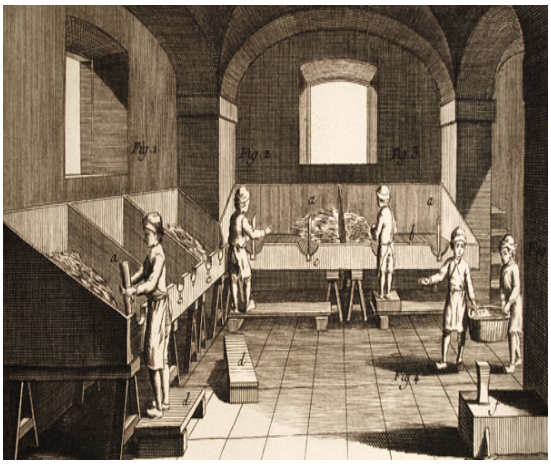
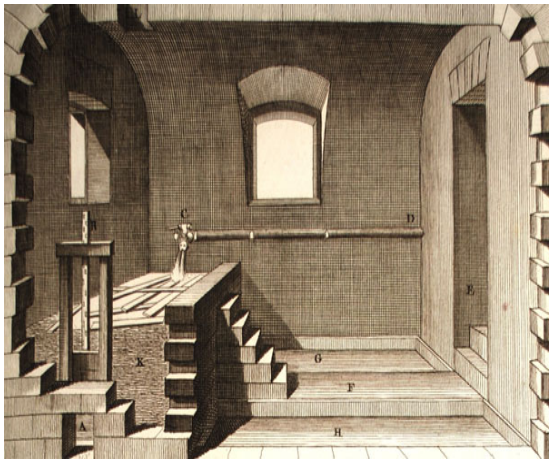


Figure A.2: 20 different watermarks with an almost identical motifs

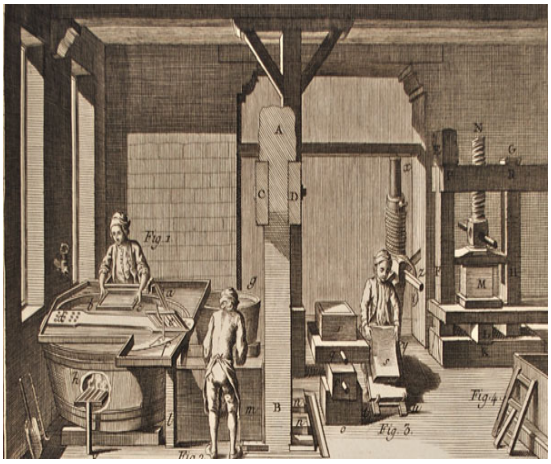
(a) Sorting rags by size, colour, and fabric



(b) Cutting the sorted rags into small pieces

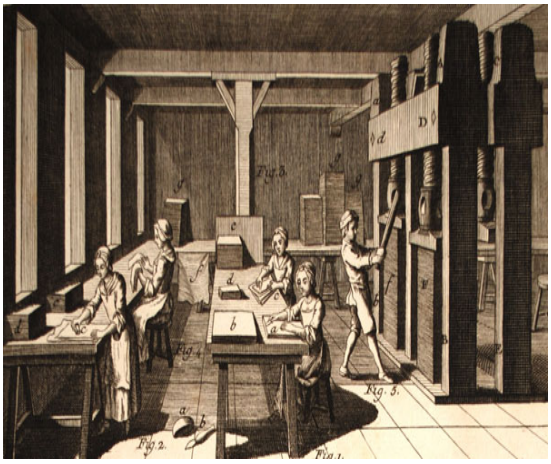


(c) Fermenting of rags in water



(d) Scooping and pressing the sheets

(e) Drying sheets of newly pressed paper



(f) Glazing to smooth the surface

Figure A.1: Selected steps of the manual paper making process in Europe between 1300 and 1800 A.D.

A.2 Additional material for chapter 3

Figure A.3: Set up for digitization of the watermark dataset at DNB in Leipzig

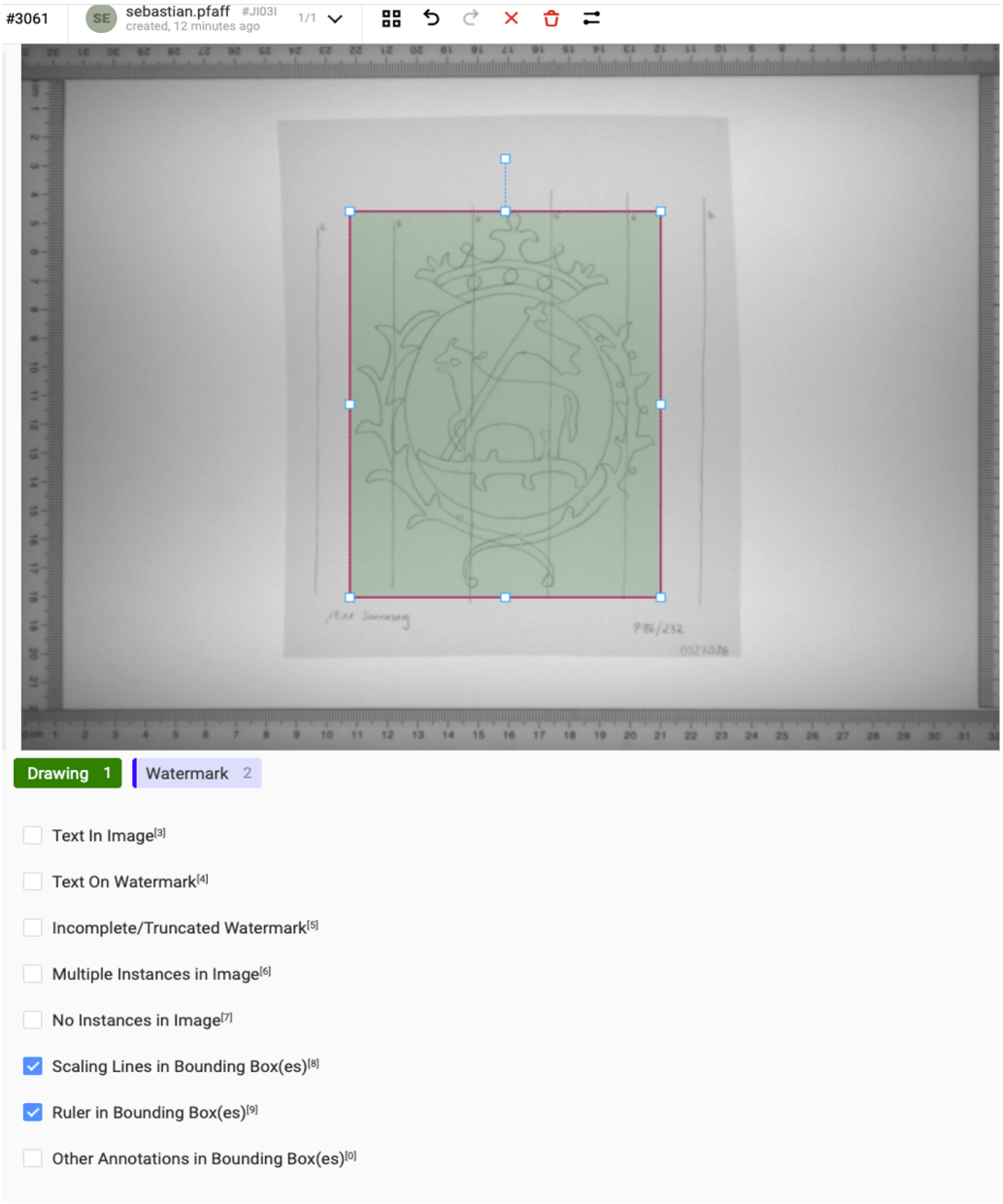
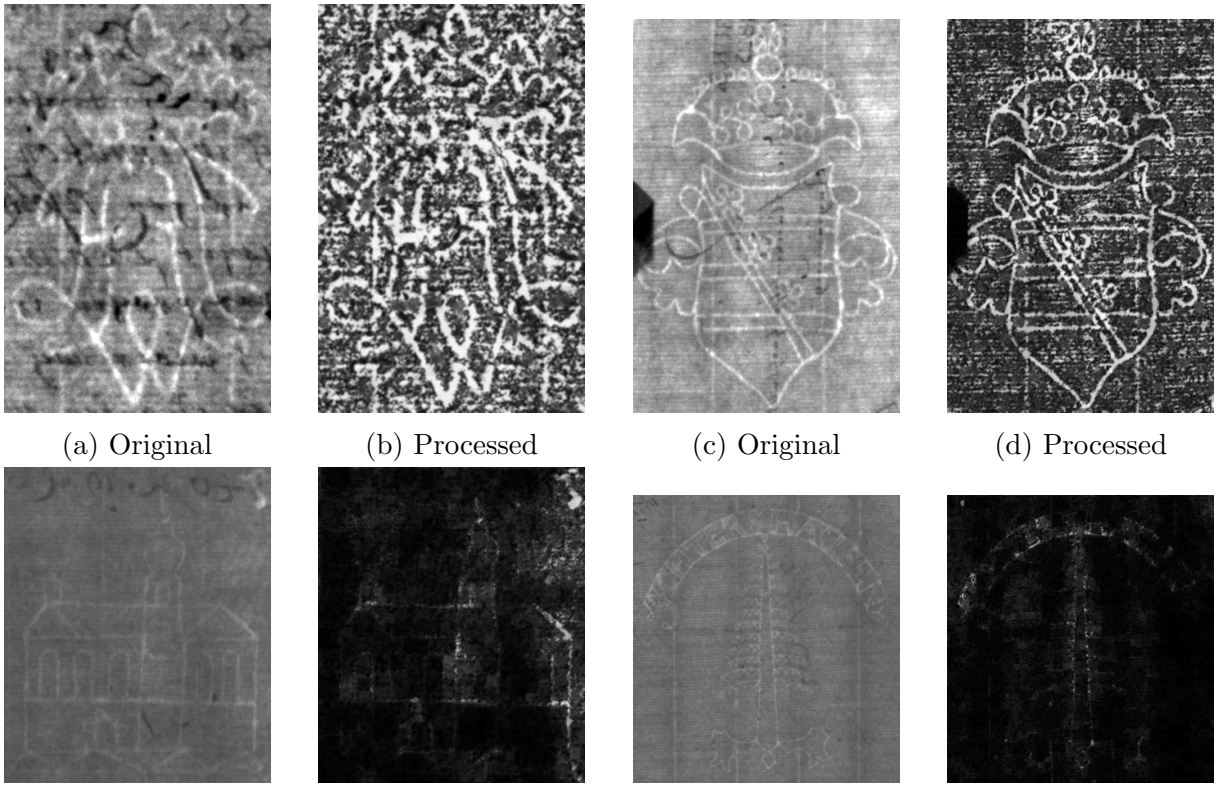


Figure A.4: Set up in LabelStudio for annotating the images from the data base. A bounding box is drawn around the relevant image part and either labeld as watermark or drawing (=tracing).



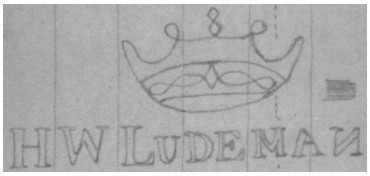
(a) Original (b) Processed (c) Original (d) Processed

(e) Original (f) Processed (g) Original (h) Processed

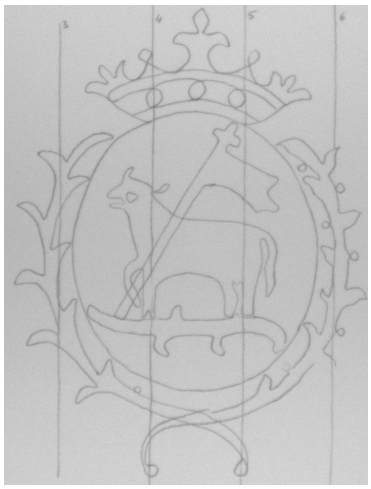
Figure A.5: Additional examples for preprocessing of watermarks. The first row shows successful results, the second row shows failures of the process.

(a) Clean tracing

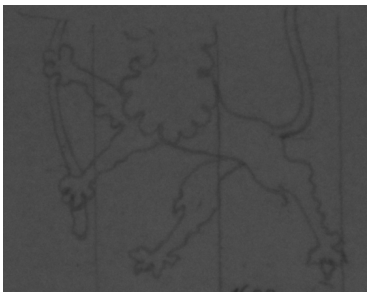
(b) Vertical lines



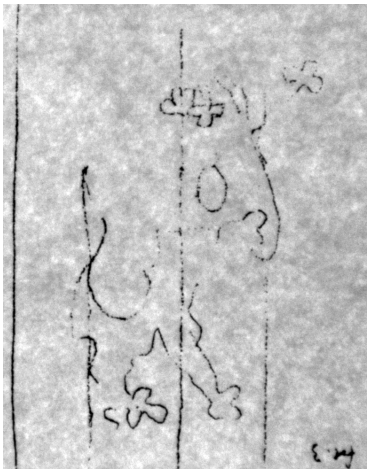
(c) Black bars



(d) Visible ruler

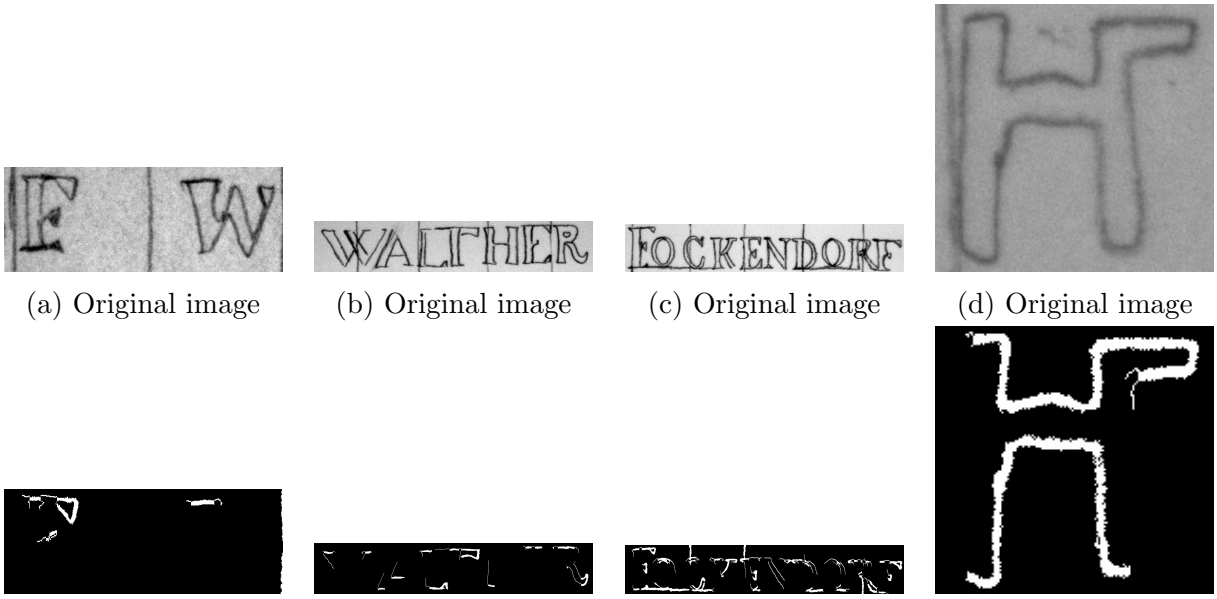


(e) Cut off tracing



(f) Incomplete tracing

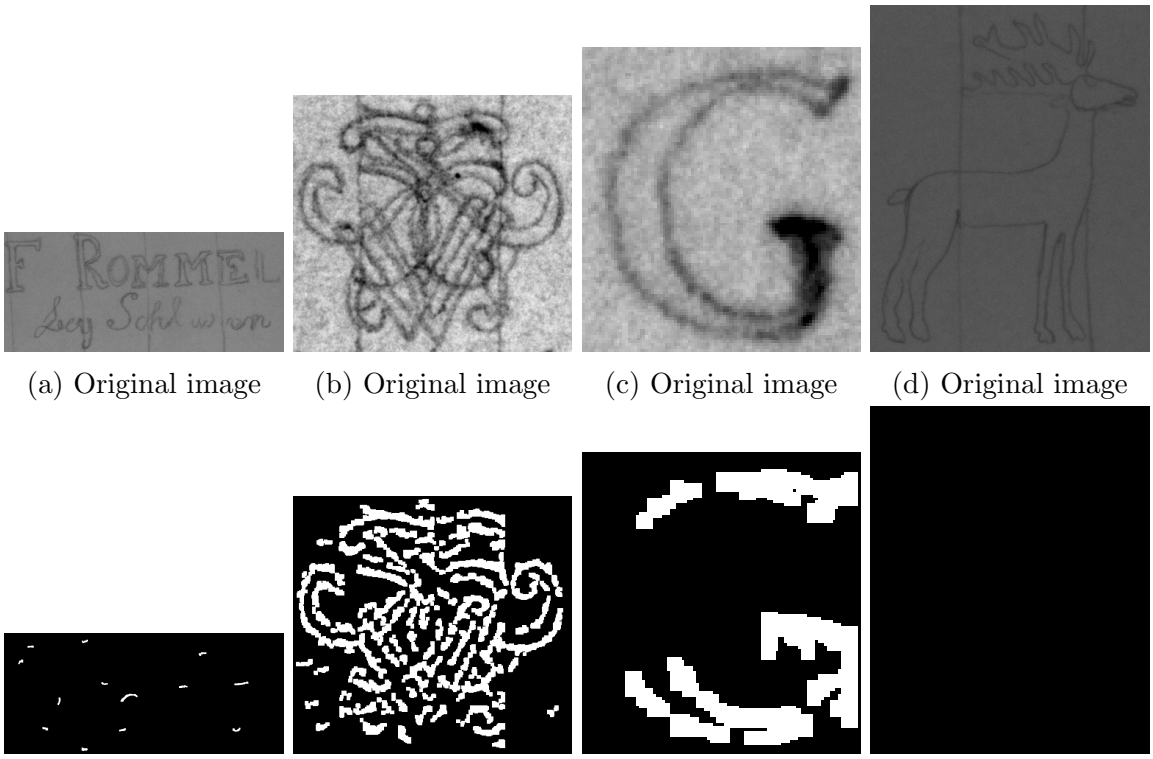
Figure A.6: Common degradation types on tracings



(a) Original image (b) Original image (c) Original image (d) Original image

(e) Preprocessed image (f) Preprocessed image (g) Preprocessed image (h) Preprocessed image

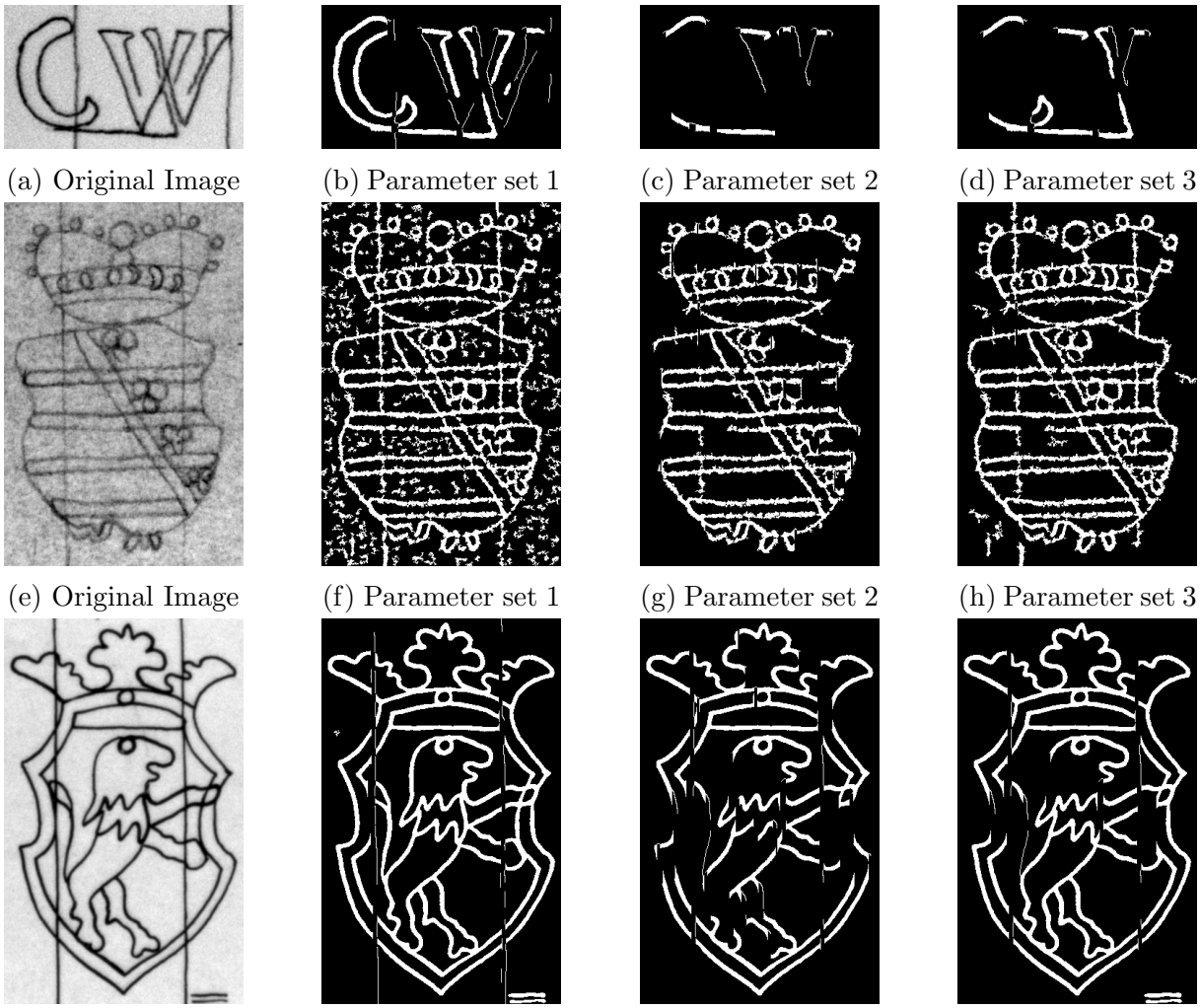
Figure A.7: Examples where pixel method deletes part of tracing



(a) Original image (b) Original image (c) Original image (d) Original image

(e) Preprocessed image (f) Preprocessed image (g) Preprocessed image (h) Preprocessed image

Figure A.8: Example where wavelet method struggles

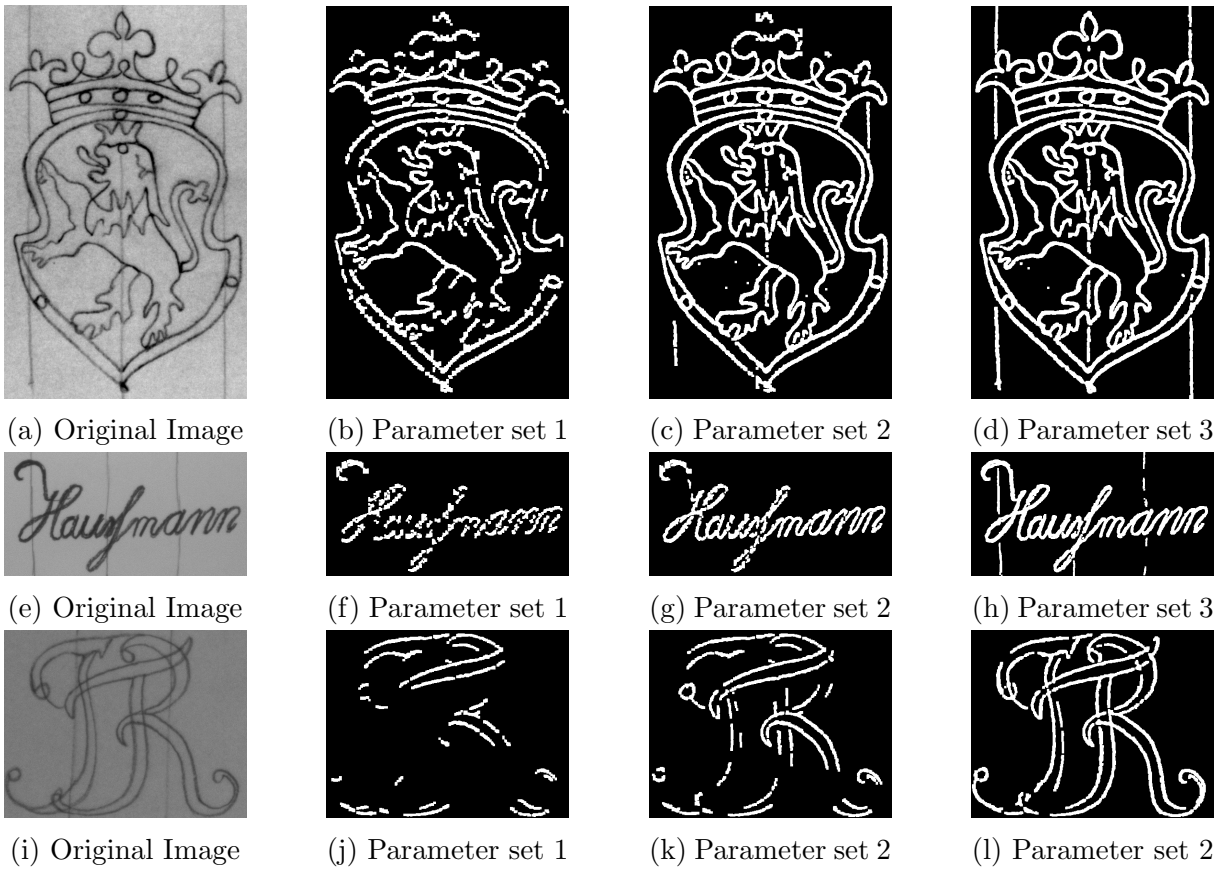


(a) Original Image (b) Parameter set 1 (c) Parameter set 2 (d) Parameter set 3

(e) Original Image (f) Parameter set 1 (g) Parameter set 2 (h) Parameter set 3

(i) Original Image (j) Parameter set 1 (k) Parameter set 2 (l) Parameter set 3

Figure A.9: Pixel method with different parameters for different images. The parameter set 1 corresponds to the vector [10,1000,15,35], parameter set 2 to the vector [300,50,100,100], and parameter set 3 to the vector [150,400,140,100].

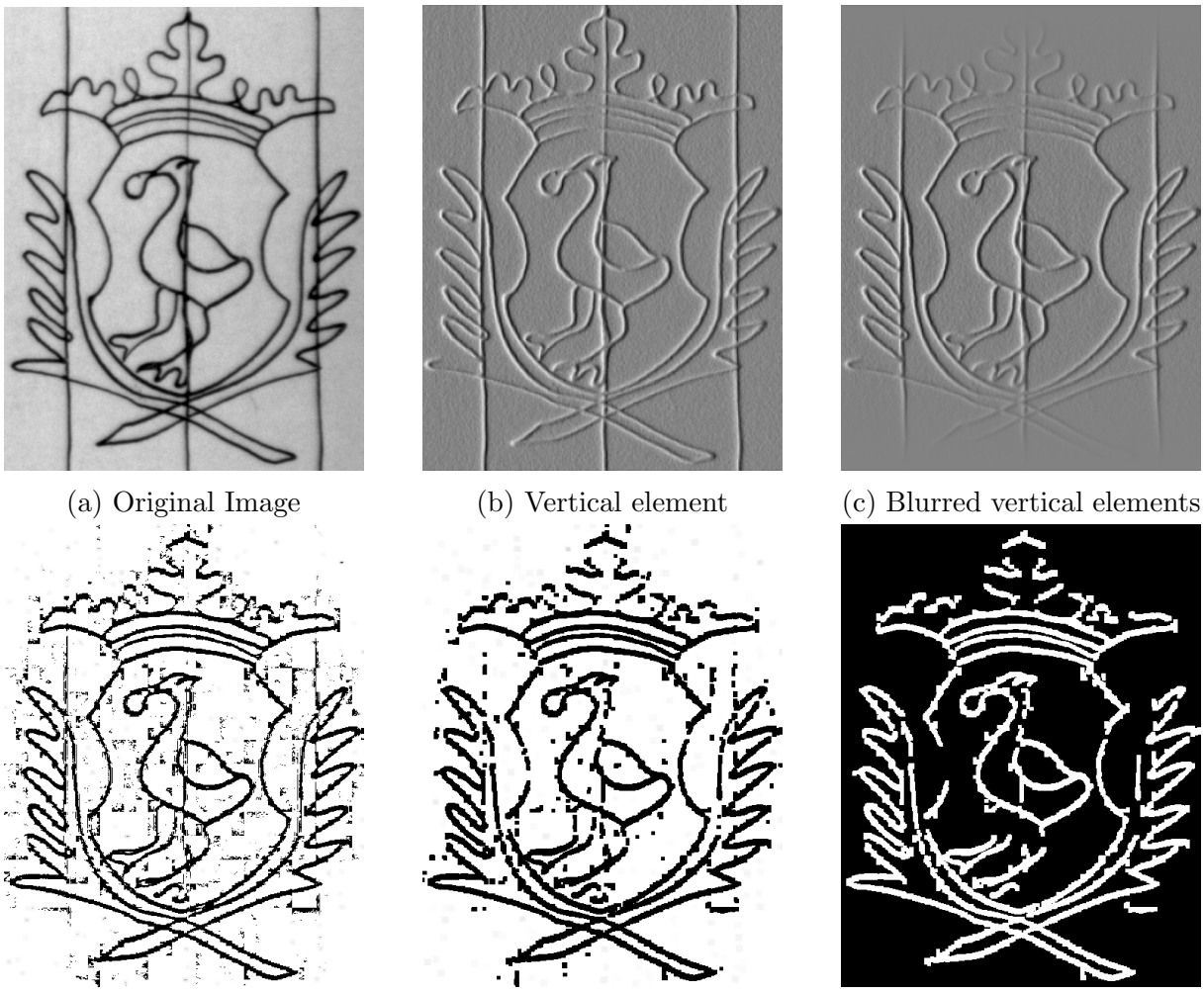


(a) Original Image (b) Parameter set 1 (c) Parameter set 2 (d) Parameter set 3

(e) Original Image (f) Parameter set 1 (g) Parameter set 2 (h) Parameter set 3

(i) Original Image (j) Parameter set 1 (k) Parameter set 2 (l) Parameter set 2

Figure A.10: Wavelet method with different parameters for different images. The parameter set 1 corresponds to the variables [basis = 'haar', level = 10,sigma = 50], parameter set 2 to the vector [basis = 'haar', level = 5,sigma = 10], and parameter set 3 to the vector [basis = 'haar', level = 2,sigma = 3].



(a) Original Image (b) Vertical element (c) Blurred vertical elements

(d) Reconstructed image (e) Strengthen outline (f) Noise removed

Figure A.11: Different pre-processing steps for tracings, removing vertical lines with wavelet method

(a) Original Image (b) Black and white image (c) Removed noise

Figure A.12: Different pre-processing steps for tracings without removing vertical lines

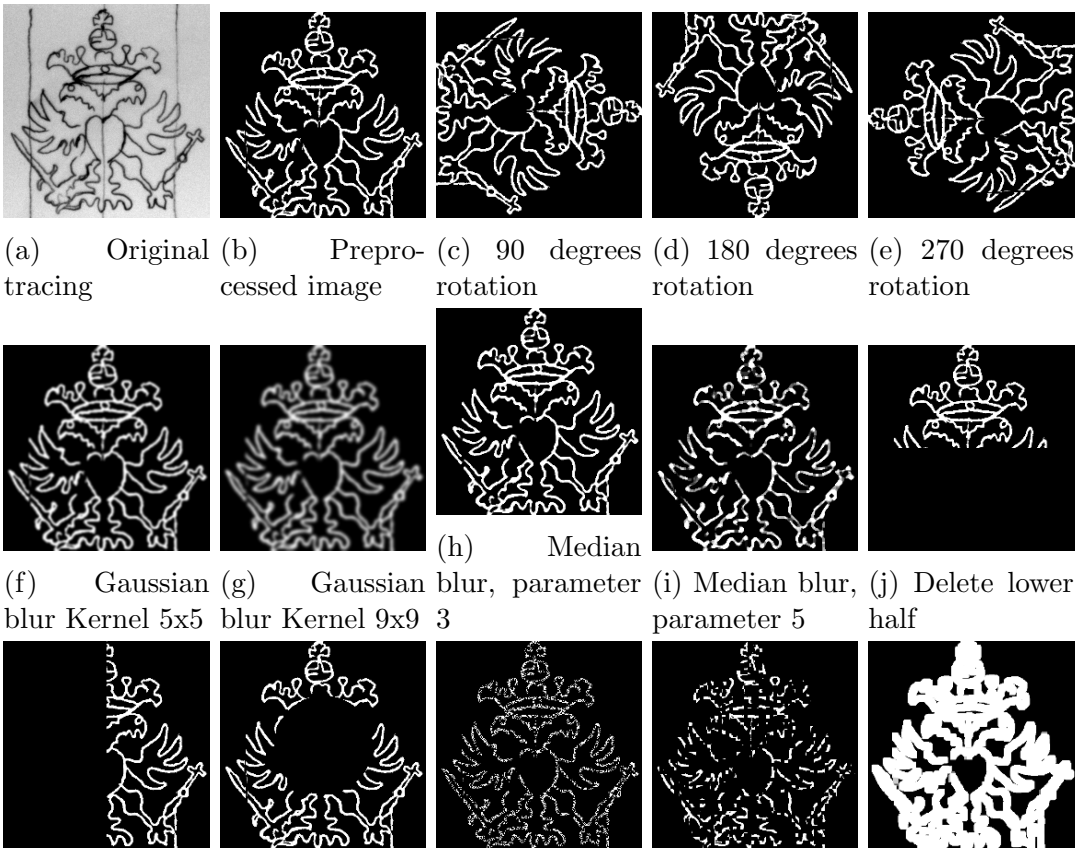
Trial	Batch Size	Learning Rate	$G_{GAN} + G_{L1}$	$D_{real} + D_{fake}$
T1	2	2.00051e-06	5.25926	1.00961
T2	4	1.38991e-06	3.16192	1.09442
T3	4	3.81642e-07	2.57181	1.33789
T4	4	1.94083e-05	3.58563	0.0924803
T5	2	5.80044e-05	7.03524	0.0367336
T6	4	8.93365e-06	4.3206	0.0807234
T7	2	4.90901e-08	23.5636	0.156984
T8	4	7.52065e-08	13.0767	0.0544864
T9	4	2.93331e-06	3.07164	0.617392
T10	4	2.3932e-08	59.367	0.0556893
T11	2	8.41496e-08	9.85447	0.164904
T12	4	2.47146e-08	52.232	0.0327645
T13	4	5.11003e-07	3.42068	0.889978
T14	2	6.63845e-07	1.61644	1.25707
T15	4	1.15535e-06	9.32557	0.483405
T16	4	1.70442e-06	2.47208	0.818634
T17	4	1.45459e-06	5.28097	0.353632
T18	4	1.06498e-06	1.29618	1.11996
T19	2	2.12062e-08	55.4039	0.0641136
T20	4	6.97207e-08	13.6767	0.149564

Table 1: Shallow and wide hyperparameter search for the Sketch Preprocessing cGAN - 20 samples, learning rates in range [2e-8, 1e-4] and batch sizes in {2, 4} trained for 10 epochs with no linear learning rate decay

Trial	Batch Size	Learning Rate	$G_{GAN} + G_{L1}$	$D_{real} + D_{fake}$
T1	2	3.24259e-07	4.1465	0.214329
T2	8	2.00194e-07	2.7701	1.08082
T3	8	7.6798e-06	5.33517	0.458044
T4	4	4.70091e-06	6.90337	0.0165826
T5	4	6.12553e-06	3.64774	0.0731736
T6	4	1.2455e-05	5.20814	1.6953
T7	8	9.63624e-08	2.14888	1.02393
T8	2	9.67572e-05	9.656	0.0027981
T9	8	3.69893e-09	54.0128	0.0426551
T10	8	2.23672e-05	3.84058	0.0782349
T11	4	5.01381e-08	3.21848	1.0627
T12	2	1.02913e-07	2.13291	1.14488

Table 2: Deep and narrow hyperparameter search for the Sketch Preprocessing cGAN - 12 samples, learning rates in range [2e-8, 1e-4] and batch sizes in {2, 4, 8} trained for 200 epochs with a linear learning rate decay after 100 epochs

A.3 Additional material for chapter 4



(a) Original tracing (b) Preprocessed image (c) 90 degrees rotation (d) 180 degrees rotation (e) 270 degrees rotation

(f) Gaussian blur Kernel 5x5 (g) Gaussian blur Kernel 9x9 (h) Median blur, parameter 3 (i) Median blur, parameter 5 (j) Delete lower half

(k) Delete left half of image (l) Delete circle in center (m) Delete random pixel (n) Delete 4x4 in-square (o) Erode (inverse) image

Figure A.13: Transformations used to evaluate different image comparison methods

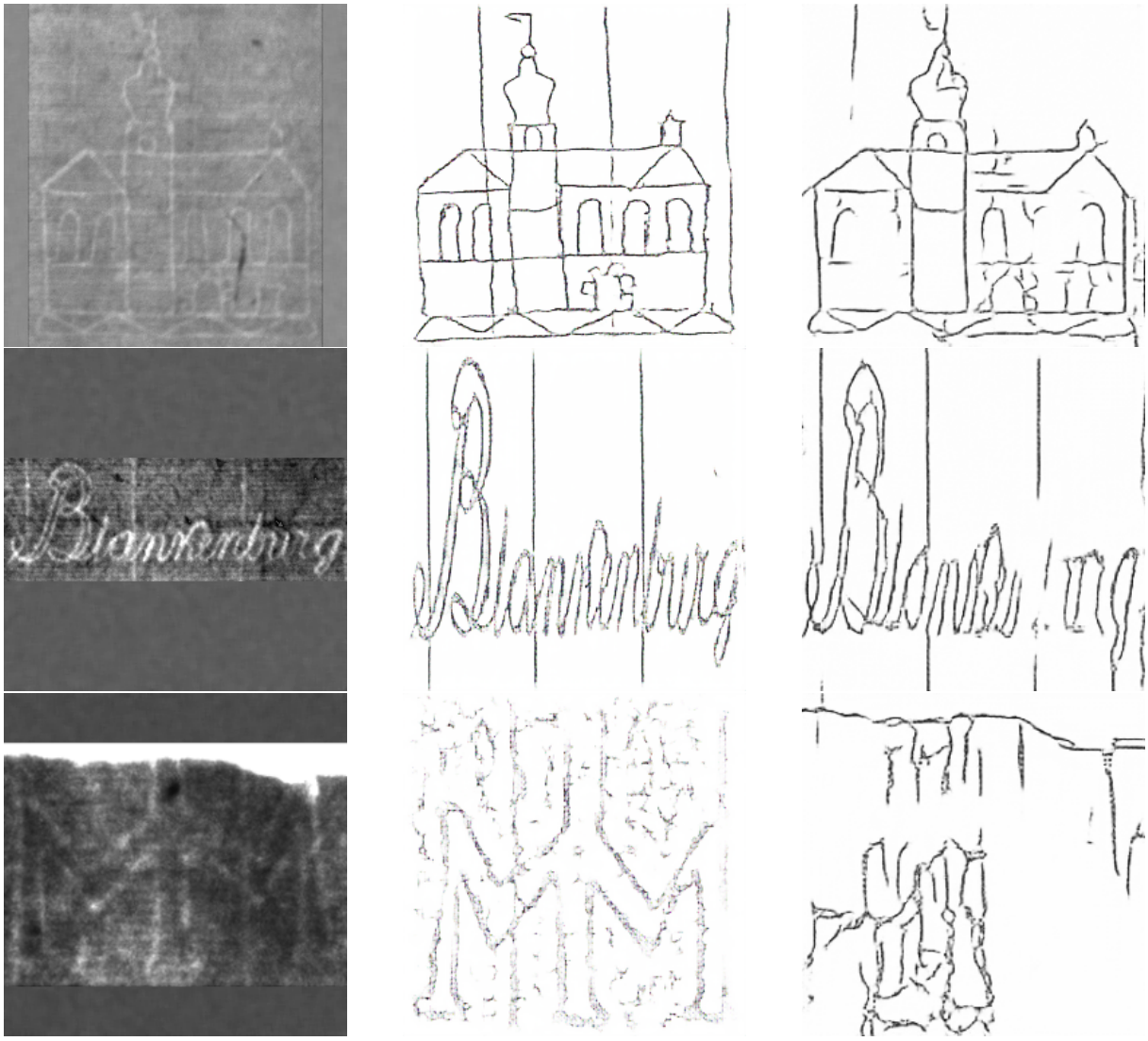


Figure A.14: Further good and bad outputs of the trained CycleGAN model in Triplets: Original Watermark - Sketch of a similar tracing - Output of the CycleGAN

A.4 Additional material for chapter 5

Trial	Batch Size	Learning Rate	$G_A + G_{cycle_A} + D_B$	$G_B + G_{cycle_B} + D_A$
T1	4	1.43514e-05	0.948875	1.1426
T2	2	1.03988e-06	1.24144	0.934863
T3	4	3.47172e-07	1.72982	1.25698
T4	4	1.60101e-07	1.51567	1.59304
T5	4	2.11067e-05	0.743989	1.01791
T6	2	1.11122e-07	1.63381	2.03171
T7	4	7.6539e-08	2.80261	2.01568
T8	2	1.96426e-07	1.32769	1.70412
T9	2	2.29912e-08	6.78732	7.79293
T10	2	4.19336e-05	0.961617	0.815696
T11	2	3.17404e-07	1.19164	1.24681
T12	2	3.04474e-05	5.27114	1.06706
T13	4	1.42223e-07	1.36262	1.67656
T14	2	3.10465e-07	1.48368	1.50432
T15	4	2.04385e-06	1.09923	1.04807
T16	4	1.60899e-06	1.03511	1.02395
T17	4	2.72688e-08	11.4191	5.63914
T18	2	6.26597e-05	1.07834	1.15261
T19	4	5.69252e-05	0.711055	1.09885
T20	2	2.38607e-05	0.884326	1.14593

Table 3: Shallow and wide hyperparameter search for the CycleGAN - 20 samples, learning rates in range [2e-8, 1e-4] and batch sizes in {2, 4} trained for 10 epochs with no linear learning rate decay