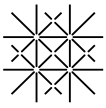

University of Basel

Hand Gesture Retrieval in Art Image Collections

Master project

Natural Science Faculty of the University of Basel
Department of Mathematics and Computer Science
Databases and Information Systems
https://dbis.dmi.unibas.ch

Examiner: Prof. Dr. Heiko Schuldt
Supervisor: Dr. Mahnaz Parian-Scherb

Manuel Rosenthaler
manuel.rosenthaler@unibas.ch
12-067-831

June 30, 2021

Abstract

The aim of this project was to build up a process chain with which gestures in artistic images – e.g. paintings – can be recognized. Gestures of a person are very much expressed by posture of the hands (see e.g.Wongphati et al. [20]), which suggests as a first step a segmentation of the hands and forearms. The forearms are important to have a reference for the position of the hands, because especially in artistic images (e.g. a ceiling painting) there is not necessarily a clear vertical order. For this step, the "Cross-Domain Complementary Learning Using Pose for Multi-Person Part Segmentation" (for Short CDCL) of Lin et al. [13], which was slightly adapted to segment only the hands and forearms, is used. Despite, being developed as a body parts segmentation program, the segmentation also works satisfactorily on artistic images.
After the body parts segmentation, the original image is then masked with the isolated hands and forearms. Since CDCL partially cuts off the fingers or other parts, the mask is dilated beforehand to get the whole hands/forearms.
The next step is to compute feature vectors using the Very Deep Convolutional Networks for Large-Scale Image Recognition (Vgg16) developed by Simonyan and Zisserman [18]. Although Vgg16 is designed to segment objects in complete images, applying it to masked images of hands/forearms still yields usable feature vectors.
In a final step, similar images are found for a reference image with respect to the gesture. This search is based on a similarity search. However, if it is not wished to have a similarity search, the database still groups images with similar hand gestures together by using k-means clustering.

Table of Contents

1 Introduction

The digital transformation is not only omnipresent in technology-related fields, but is also increasingly taking hold in the areas of culture and society. Cultural institutions in particular, which are grouped together under the term "GLAM", i.e. galleries, libraries, archives and museums, are digitizing their collections and making them available on the Internet. Since many of these institutions receive substantial funding, there is also a demand that these cultural assets are made freely available to the public under "open access". For this purpose, web platforms are often developed that allow access to these artifacts. Large collections can contain many thousands of digital representations. For example, the Denver Public Library's Western History [1] collection contains over 1.1 million photographs. Europeana [2], a European project, offers access to many millions digitized cultural objects from over 4000 European cultural institutions. These cultural artifacts are not only an important part of our cultural heritage and are therefore of great interest to society. They are also a subject of research in disciplines such as art history, general history, archaeology, literature, etc.

A major problem, however, is how to search these online collections in a targeted and efficient manner. Text-based sources can be converted into a searchable form by transcription and/or OCR. Powerful indexing systems such as "elastic search[3]" then allow complex, elaborate searches to find interesting passages of text. Statistically based methods, which can be summarized under the term "distant reading", allow the investigation of large text corpora.

For other types of sources, e.g. images (photographs, paintings, sketches, etc.), video and film, search strategies normally rely on metadata, which usually have to be painstakingly created by hand. Therefore, it has long been a desideratum to develop content-based search for such objects. The development of machine learning that is based on neural networks has made rapid progress in recent years. An important application area is the analysis, segmentation and processing of

[1] See Denverpubliclibrary [5]
[2] See Europeana [8]
[3] See Elastic [7].

natural images. Therefore, reliable and efficient content-based image search and image characterization is becoming more and more possible. From this context came the motivation for my project: can modern neural networks and machine learning help to orient oneself in a database of paintings? To narrow down the question, I asked whether hand gestures can be recognized and used as search features. Gestures basically encompass the whole body (posture), but besides facial expressions, hands are among the most important body parts. Wongphati et al. [20] have shown that hands play an important role in 88% of gestures. In addition, I wanted to focus on paintings, i.e. non-natural (non-photographic) images. A typical example that demonstrates the significance of the hands is Albrecht Dürer's drawing "Praying Hands" which shows only the hands and forearms and yet expresses praying with tremendous strength.

Figure 1.1: Albrecht Dürer's "Praying Hands". Source: Commons [4].

Thus, the goal of this project is to use neural networks within artworks to recognize and retrieve gestures of the depicted people. It was the first time for me to deal practically with machine learning and neural networks outside of lectures. Since my research did not uncover a general and widely recognized corpus of hand gestures (especially for art paintings), I decided that the image search should be based on the hand gestures of an example image. These considerations

resulted in the following design of a process chain:

1. Isolation of hands and forearms on the images from the image corpus. In other words, the hands and forearms in all images have to be recognized and segmented.
2. Generate feature vectors of these image parts.
3. Perform a similarity search of a user supplied reference image. This implies, that the isolation of body parts and the extraction of the feature vectors has to be applied to the reference image. The resulting feature vectors are then compared to the corpus using a distance measure.

Thus as a result, the process should return images from the corpus that contain similar hand gestures as the, by the user given, example image.
For the first two steps, it was obvious to use neural networks. However, in the short time of this project work it was not possible for me to construct my own neural network, to construct a learning set with images of the body parts that are important for gesture recognition and to carry out the learning process for the recognition of the body parts and the generation of meaningful feature vectors. Therefore, I searched for pre-trained neural networks, which I could adapt and combine into a meaningful process chain for my task.

The paper is arranged chronologically according to my work process. As a first step, I looked into the theory of regional convolutional neural networks (R-CNN), which are considered suitable for this kind of tasks.
In a next step, I will describe the body parts segmentation program "Cross-Domain Complementary Learning Using Pose for Multi-Person Part Segmentation" (CDCN for short) I used to detect and segment hands and forearms.
My approach to gesture retrieval is then discussed, as well as the advantages and disadvantages of my method. As a last chapter I touch on points where a subsequent researcher could follow up on.

2
Approach

There are many papers about the significance of gestures in human communications, e.g. Susan Goldin-Meadow and Martha Wagner Alibali (2013)[4]. At the beginning, however, I had to ask myself, what does the word gesture even mean? Gestures can include a variety of head movements, body movements, hand and arm movements. However, most scientific papers seem to focus heavily on the hands and arms when discussing gestures. Take as an example, the book "Beredte Hände – Die Bedeutung von Gesten in der Kunst des 16. Jahrhunderts bis zur Gegenwart" ("Eloquent hands - the meaning of gestures in art from the 16th century to the present day"), which was published to accompany the exhibition with same name that took place in the Residenzgalerie in Salzburg in 2004[5]. In this and similar publications, the importance of the *hands* is repeatedly emphasized. Other body parts such as the head, torso, etc. also might play a role in gestures, but the posture of the hands seems to be one of the strongest non-verbal physical means of expression (perhaps with the exception of facial expressions, which also play a decisive role in conveying feelings when talking directly to a counterpart). From this insight, I decided that the first subtask in my working pipeline was to isolate the hands from the rest of the images, to allow further post processing. Since the arms have a large influence on hand gestures, I determined to isolate them along with the hands for my gesture retrieval as well.

Under the assumption that the first step of hand and arm isolation can be successfully executed, the question arose as to what the subsequent step should be. It was obvious that the isolated body parts, as hands and forearms, had to be characterized. In other words, feature vectors had to be extracted. Of course, it would be optimal to use a neuronal network trained specifically for the hand positions to create the feature vectors. But again, since training a neuronal network myself was unfeasible I had to find a different solution. For this reason, I decided to look for a generic feature extraction network for images, with the idea to only put

[4] https://www.ncbi.nlm.nih.gov/pmc/articles/PMC3642279/
[5] https://www.deutsche-digitale-bibliothek.de/item/4HPLSSIBW3AAE3LTWVOS3JP6SX3TYDOL

the segmented image sections of the hands and forearms as an input for the feature extraction. This way I hoped to still get usable feature vectors.

Assuming that the steps up to this point produce usable feature vectors, the question now arose as to how to proceed. A first consideration was to group the feature vectors using a k-mean clustering so that I have groups of similar hand gestures. However, to make this meaningful, an image corpus annotated with respect to gestures would have been necessary. Otherwise I end up with several groups of images with no real semantic meaning. Unfortunately, neither I nor my supervisor found such a corpus. As an alternative, I decided - similar to the content based image retrieval program vitrivr[6] - to search an image corpus for images that are have similar gestures as a reference image with a certain gesture that is provided by the user. Therefore, the final step would be to find similar images with a "similarity search", based on the extracted feature vectors.

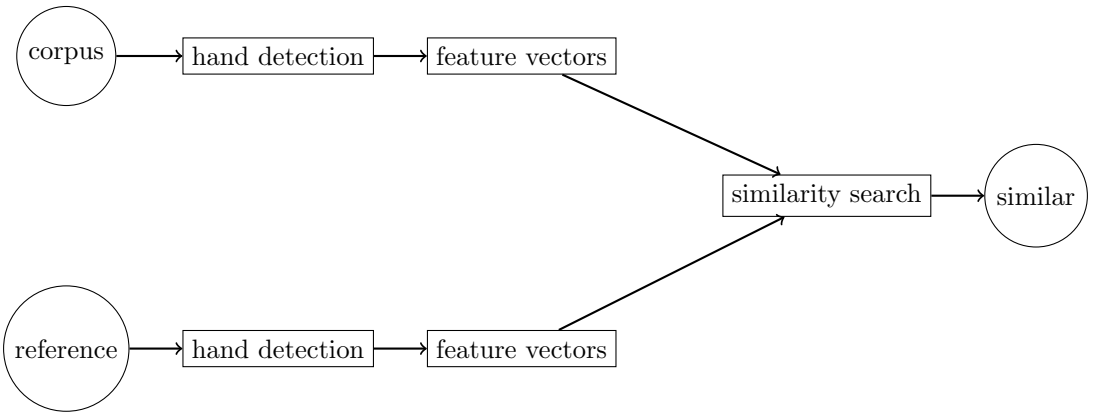

Figure 2.1: Workflow as implemented in this project.

2.1 Preparation

As a form of preparation, I was advised to read various papers on the topic of hand and gesture recognition. This should give me the theoretical foundation for the project. The papers I have read are: Ufer et al. [19], Gonthier et al. [11], Shen et al. [17] and Yin et al. [22].
In these papers I also found the first hints on software that would be applicable during the project. First and foremost, the faster R-CNN caught my attention. Since i spent a considerable amount of time of my project with analyzing the use of this software branch for my project, we will have a closer look into the theory of R-CNN - despite me not using it in the end. In the discussion part it will be explained, why i didn't use this software branch.

[6] See https://www.vitrivr.org

2.2 R-CNN

R-CNN was developed by Girshick et al. [10] to enable efficient automatic image segmentation. The special feature of the R-CNN family is that before the input image is run through a neural network, it is searched for regions of interest (RoI for short). These RoI are then sent into a Convolutional Neuronal Network (CNN) to extract feature vectors of these regions. The feature vectors are sent yet again into a support vector machine to correlate the extracted feature vectors with actual objects.

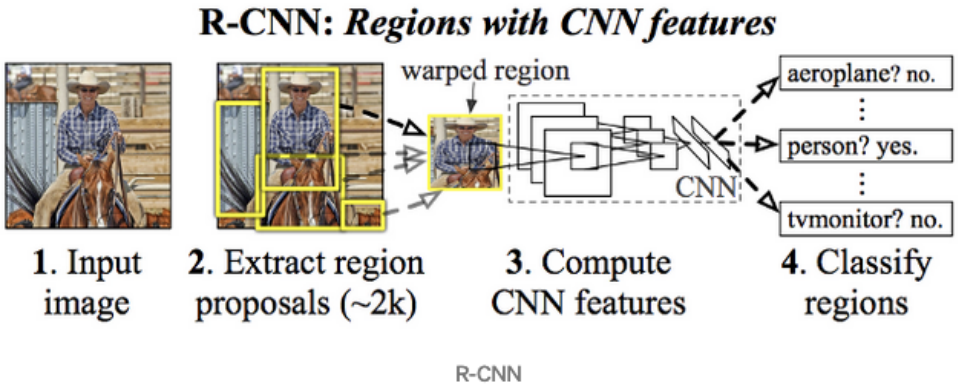

Figure 2.2: Work pipeline of a R-CNN. Source: Girshick et al. [10].

This three fold concept (2-4 in the figure) established itself as one of the modern approaches in image segmentation. However, as it is often the case with scientific method development, not soon after the development of the R-CNN, many advancements have been able to be applied to the already present R-CNN.

In the following subsections, we will briefly discuss each of the evolutions in a really superficial way and discuss the differences to the previous models.

2.2.1 Fast R-CNN

Instead of first searching for RoI and then passing those RoI through a neural network, the fast R-CNN from Girshick [9] first computes the whole image using the CNN, and only then the RoI. The RoI are extracted from the CNN in a special way. They are taken directly from the output layer of the CNN and are then reshaped. This new method is called RoIPooling. Underlying is a selective search to generate the RoI.

2.2.2 Faster R-CNN

The Faster R-CNN from Ren et al. [16] is a neural network according to modern standards. As a basis, faster R-CNN uses the regional CNN that was developed by R. Girshick et al. in 2014.

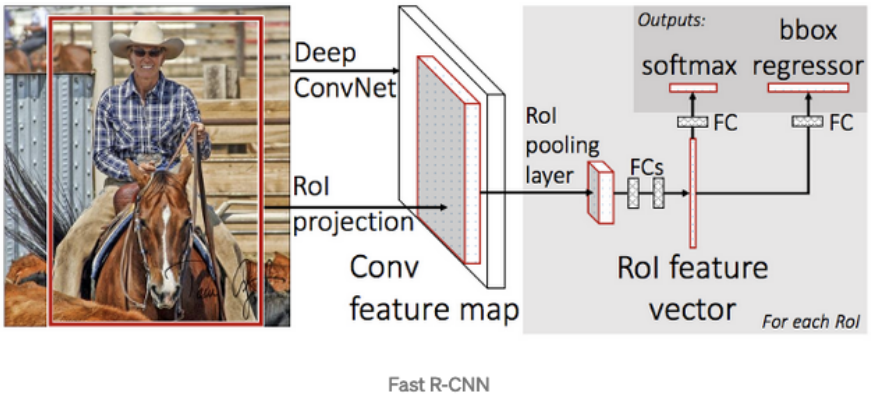

Figure 2.3: Work pipeline of a Fast R-CNN. Source: Girshick [9].

In contrast to it's predecessor fast R-CNN, faster R-CNN integrates the determination of the RoI inside of the CNN itself instead of calculating them separately and outside of the CNN via a selective search.

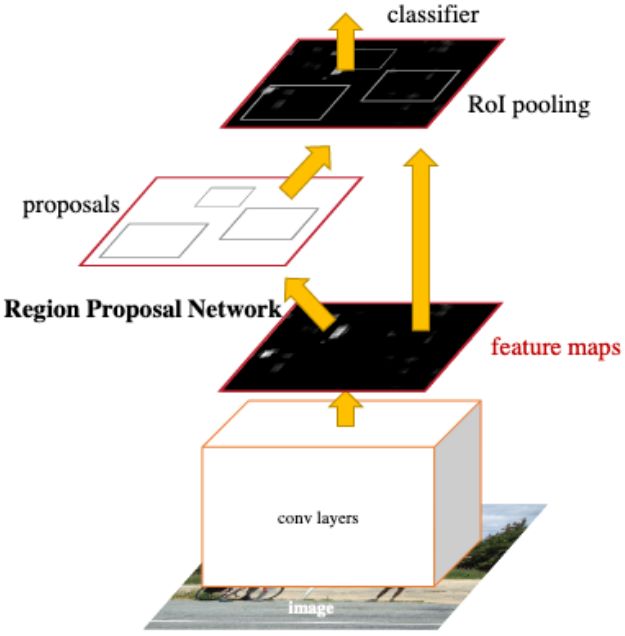

Figure 2.4: "Faster R-CNN is a single, unified networkfor object detection. The RPN module serves as the'attention' of this unified network" - Ren et al. [16]. Imagesource: Ren et al. [16].

2.2.3 Mask R-CNN

Mask R-CNN from He et al. [12] is based on Faster R-CNN. However, while Faster R-CNN focuses on object segmentation, Mask R-CNN tries to introduce instance segmentation. So unlike Faster R-CNN, Mask R-CNN does not just try to put bounding-boxes around RoI, but it tries to put an exact mask over the RoI. For this purpose, the mask formation does not run sequentially after the calculation of the RoI by the CNN, but parallel to it. To calculate the masks 1 x 1 convolution is used. Mask R-CNN also uses a fully connected neural network, which allows the spatial dimensionality of the image not to be lost when generating the feature vectors of the RoI. That spatial dimensionality can then be used to accurately determine the masks. To further improve the accuracy, the RoI-pooling layer was replaced by an RoI-align layer. Normal RoI-pooling uses approximations for the calculations, which produced large inaccuracies in the masks. The RoI-align, on the other hand, does not use rounding and therefore determines the masks more accurately.

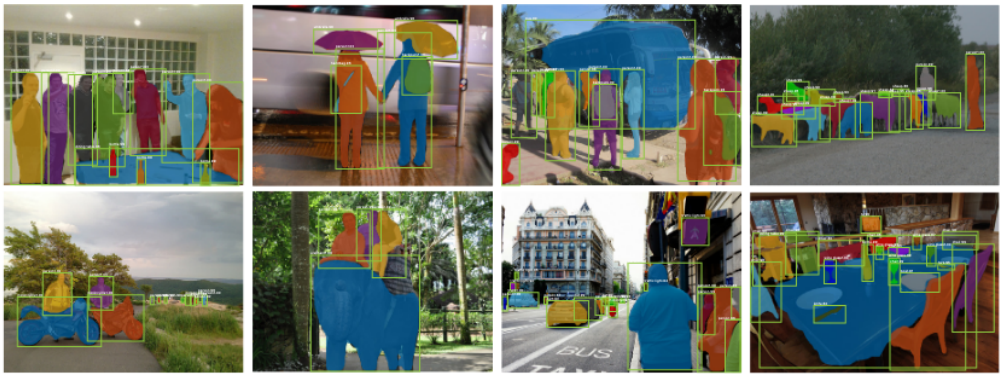

Figure 2.5: Example of Mask R-CNN in action. Source: He et al. [12].

2.3 Detectron 2

Detectron 2 is a conglomeration of many different neural networks and systems for object detection developed by Wu et al. [21] at Facebook. On the website[7] it is described as "Facebook's AI Research's next generation library that provides state-of-the-art detection and segmentation algorithms". It includes Faster R-CNN and Mask R-CNN as well as other state of the art object detection frameworks.

[7] https://github.com/facebookresearch/detectron2

2.3.1 Experience and conclusion

The reason I decided to take a closer look at Detectron's suitability for gesture retrieval was that Detectron 2 is currently about the most modern when it comes to image segmentation. It is a conglomeration of the most important image segmentation approaches, works with recent software dependencies and offers a large selection of pre-trained models on the Internet, which gives a wide range of recognition possibilities.

Although Detectron 2 is a modern and very powerful system that achieves impressive results, it turned out that I could not use it in the context of my work. The reason for this is that I did not find any pre-trained models that would allow Detectron 2 to perform body part segmentation.
One consideration to circumvent this problem was to train a model for Detectron 2 myself so that it would be able to recognize hands and extremities. However, in discussions with my assistant, Dr. Mahnaz Parian-Scherb, we concluded that such an endeavor would involve great effort and an inherent risk. The risk is that the final result would not be satisfactory and the hand detection would be insufficiently accurate. In order to train a neural network, one needs a very large dataset of training data, which would have to be divided into a training set and a validation set yet again. It requires a lot of work and time to create such a dataset. Moreover, there is still a risk that the final result will not be satisfactory and i would lose a lot of time with no real outcome.

A further consideration was to use the keypoint detection built into Detectron 2 to generate hand and extremity segmentation.
Detectron 2 implements the COCO Person Keypoint Detection Baselines with Keypoint R-CNN. This includes several pre-trained models that allow Detectron 2 to calculate the skeletal structure of persons. Using this skeletal structure, it would be possible to localize and subsequently

Figure 2.6: Example of the keypoints detection. It's one of the first test pictures I used to test the keypoints detection. As it can be seen: The hands are not part of the keypoints detection.

isolate the hands and extremities. However, there was a problem with this method. The skeleton segmentation only included head, torso, arms and legs. The hands, however, were not included in the skeleton model. To locate the hands, I would have had to find the end of the forearms and then estimate the region of the hands. Of course, one could assume that the hands would simply be in a straight line leading from the forearm. However, the world of gestures is large and diverse, where the position of the Hand relative to the forearm plays a decisive role. I.e. there are many gestures, where the hand is angled at a 90° angle from the forearm. In those cases in particular, as well as in all cases where the hand would not be exactly in line of the forearm, this method would lead to great inaccuracies. Therefore, this method appeared to me as insufficient for a reliable gesture detection and I therefore discarded this possibility.

Based on the above considerations, I decided to drop the approach of using the Detectron 2 and to look for an alternative to it.

2.4 Cross-Domain Complementary Learning (CDCL) Using Pose for Multi-Person Part Segmentation

2.4.1 Structure of the CDCL

In the work of Lin et al. [14] they wanted to take advantage of the fact that there are already very good methods for image segmentation (namely ResNet101). Once ResNet101 has found a human, it calculates where the human skeleton should be located. Since the human skeleton is strongly correlated with the localization of the human extremities, K. Lin et al. first compute the skeletal structure, then use a neural network to compute the extremities' localities. More precisely, they use a cross-domain complementary learning framework to achieve their part segmentation. This means that K. Lin et al. use two modules, one with synthetic inputs and one with real inputs, to compute keypoint maps in parallel and thus achieve more accurate results for body part segmentation than previous methods.

2.4.2 Experience and conclusion

The reason I decided to use CDCL to perform the hand and arm segmentation was both because Dr. Mahnaz Parian-Scherb recommended this library to me and because it is one of the most advanced and accurate tools to calculate body segmentation.

The fact that CDCL provides a Dockerfile was also very convenient for me. After I was able to remove a few bugs from the Dockerfile and adapt it to my needs, the installation of the necessary Docker container on the cluster was very easy.

Out of the box, the CDCL was able to recognize different body parts such as hands, fore-

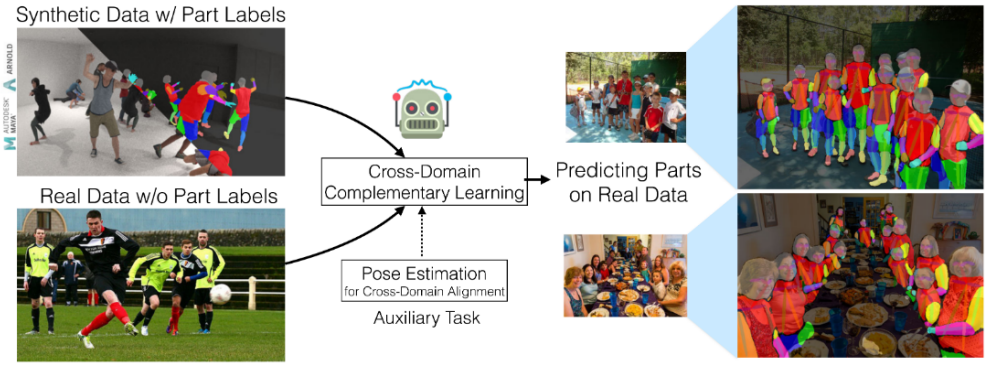

Figure 2.7: Pipeline of CDCL. Lin et al. [14] are using a complementary learning technique with a synthetic and a real dataset. Source: Lin et al. [14].

arms, upper arms, head, etc. and mask them in different colors. I wanted to use this location

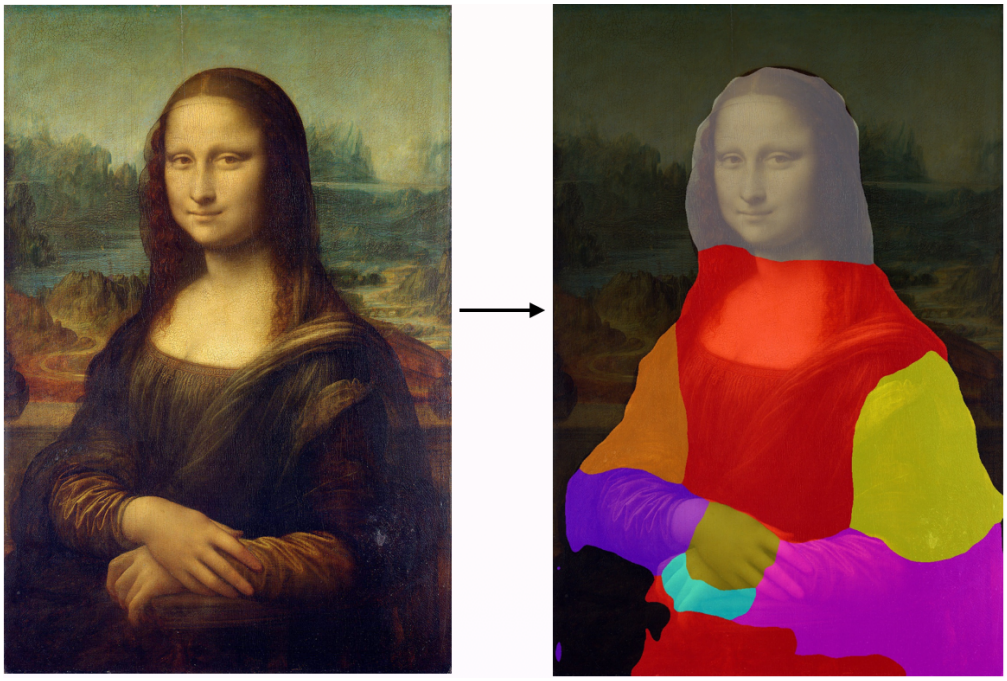

Figure 2.8: One of the first Tests with CDCL. As it can be seen, the body parts of Mona Lisa get segmented in a really precise manner - even tho CDCL is applied on a painting. Note however, that the fingers of the Hand are not precisely masked.

information to delete everything on the original image except the hands, forearms and upper arms. To achieve this, I edited the function "human_seg_combine_argmax" within CDCL to store only the mask pixels of the hands, under arms and upper arms. Then the color of each mask given by CDCL is replaced so that the background is completely white and the masks are

completely black and finally all sub-masks are added together to form a single large mask. Then the color values of the original image minus the color values of the mask can be calculated, which has the effect of erasing everything on the original image except the masks of the hands and arms.

However, I encountered the problem that the fingers were often not within the calculated masks. To correct this, I implemented an additional dilatation that builds a small padding around the calculated masks. With this implementation I prevented on the one hand that the shape of the arms later became too important when calculating the feature vectors and on the other hand I achieved that from that point on the fingers were more often inside the mask. So at this point

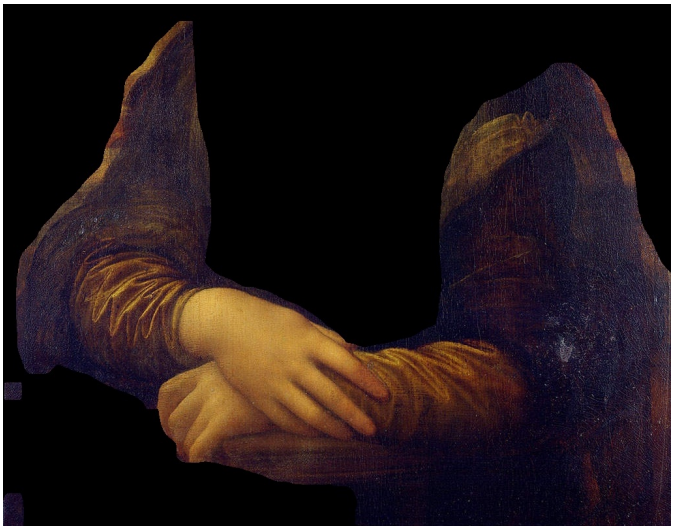

Figure 2.9: The Mona Lisa picture after its segmentation process. As you can see, the background as well as all other body parts besides the hands and arms are black. You can also notice the dilation that's around the arms and hands to ensure that the fingers are part of the segmentation.

I had first the original image with one or more people on it, and secondly the segmented image with just the hands and arms of the people on it. How could I use this information to make a gesture retrieval? The simplest solution was to calculate feature vectors of those segmented images and then make comparisons on a purely mathematical level with the feature vectors, which in turn would allow conclusions to be drawn about the original images.

2.5 Feature vectors

Feature vectors are mathematical representations of traits, of characteristics of a particular observation. In the context of an image, the feature vector contains features of the image that have been carried to the surface during its computation.

Feature vectors are very useful for dealing with image content on a purely mathematical level. If a feature vector of an image is available, the computer can completely concentrate on mathematical transformations and work detached from the semantic context of the image.

The content of the detected features depends on the methodology of how the feature vector was created. How I created the feature vectors in my program, as well as what could be determined with them afterwards, we will look at in the following couple of chapters.

2.5.1 The creation of feature vectors

There are a number of completely different ways to compute feature vectors. Probably the most modern method, however, is to use a CNN that learns to recognize different features. This was also the approach I chose.

Feature vectors are inherently used when a neural network learns object segmentation. In object segmentation, the goal is to distinguish and possibly highlight all objects on an image that are different from each other. In order for an object segmentation program to be able to distinguish the different objects, the neural network must learn the different features of the objects that it wants to distinguish.

The last layer of a neural network is the output layer. Each node of the output layer usually represents one of the objects to be distinguished. For example, the uppermost node might represent the object "cat". If an image with a cat is passed through the neural network, only the top node for "cat" in the output layer should be equal to 1 and all other nodes of the output layer should be 0. Because the output layer indicates either 1 for cat or 0 for cat based on the

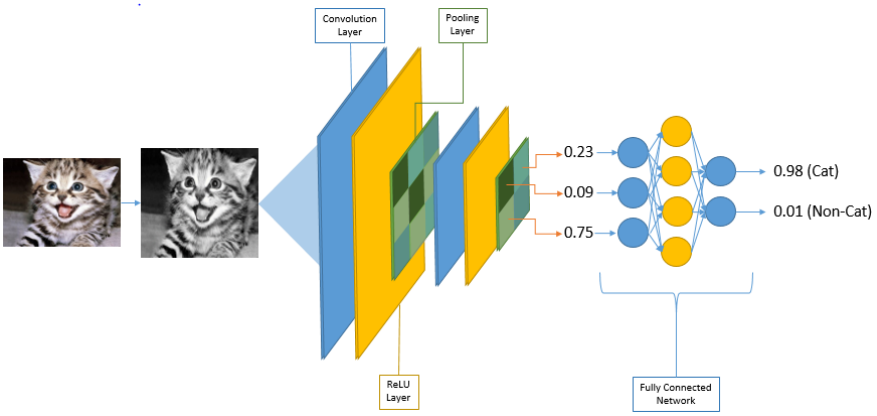

Figure 2.10: Visual representation of a CNN. The output layer decides if it sees a cat or not. The second last layer is the feature vector of the cat. Source: codeproject.com [3].

features of the objects to be distinguished, this must mean that the immediately preceding layer must contain the features that the neural network finds on the image. Otherwise, it would not be possible to make a decision in the output layer about which object it is in question. The fact that the feature vector is located in the second last layer of a neural network can be exploited.
In my project, I used a neural network called Vgg16 by Simonyan and Zisserman [18] from Keras (see Chollet et al. [2]), which is trained to recognize objects in an image. However, one could also use any other neural network that has been trained to recognize objects in an image.
If one intervenes in the second last layer and cuts away the last layer, the output layer, the Vgg16 network returns the desired feature vector of the image, which it has calculated based on its pre-trained weights.
I run Vgg16 over the already pre-segmented images, where only the hand and arms are visible. So the idea would be that Vgg16 calculates the feature vectors, which then contains only features of the hands and arms - the general gesture I'm looking for. Now that I have the feature vectors of the hands and arms on the images, what can I do with them?

2.5.2 k-means clustering

First I had the idea that I would do a k-means clustering using the generated feature vectors. In a k-means clustering, a statistical optimization algorithm is used to assign points within an n-dimensional vector space to a certain set of groups. This method allows to join clusters of points that are related to each other.

A feature vector is nothing else than an n-dimensional vector, pointing to a point in the also n-dimensional space. Feature vectors pointing to approximately the same location in space are therefore similar to each other. Since the feature vector is a representation of the different features of an image, the images whose feature vectors are similar to each other are also similar. With the help of k-means clustering, it is therefore possible to group similar images together.

Originally, this was the goal I wanted to achieve. However, I did not find a dataset where the gestures were annotated with their semantic meaning. Without such a dataset, the built clusters would not have a clear semantic meaning. Therefore, a k-means clustering seemed to me to be somewhat Unsatisfactory. However, there is another way to retrieve gestures - the similarity search.

2.5.3 Similarity search

In a similarity search, you provide the computer with an instance that you would like to search for. Then the computer tries to gather all instances that are similar to the input instance and provide them to the user as output. To describe this with a concrete example: Suppose you have a database with pictures of thousands of different animals. If you were to give the computer a

picture of a guinea pig, the computer would try to find all the guinea pigs in the database and filter them out.
This mechanism can be used in my case to search not for guinea pigs or cats but for certain gestures. If you had a picture with a certain gesture on it, the computer would filter out all pictures that have the same - or at least a similar - gesture.

In fact, a Similarity search is not far from a k-means clustering. You also need the feature vector of the image to have a pure mathematical representation of the image. Then you can compare the feature vector of the input image with the feature vectors of the database and return similar images as output. But how do you recognize that an image is similar to the input image?
Basically there are many methods that can calculate the similarity of features vectors. Almost always the distance between the points in the n-dimensional space plays a central role.

As we have already seen earlier, the closeness that the feature vectors have to each other represents their similarity. The closer the points represented by the vectors are, the more similar the feature vectors are and the more similar are the instances of the things represented by the feature vectors. If two feature vectors were identical in all their n entries - they would be the same vectors - then both vectors point to the same point in the n-dimensional space, so they are maximally similar.

There are different methods how to calculate the distance of n-dimensional vectors. The simplest calculation is the well-known euclidean distance, another one is the cosine distance, and there are many more. Which distance metric is best suited to determine the distance of concrete feature vectors is not easy to determine. Often, one does not find a definitive answer as to which distance metric yields the optimal result - unless one uses a specially trained neural network that learns to estimate the optimal distance calculation from scratch. However, I did not have the time to do this in my project. Therefore I stayed with using the cosine distance as similarity metric, because this proved to be slightly more reliable in my experiments as the euclidean distance (however, no exact statistics have been done).

2.5.4 Results

I tested the process chain with a sample of about 150 images. Since I could not find a suitable arts image collection, where only people are painted, I had to manually selected different images in a time-consuming process from publicly available image databases. The requirement for images was, of course, that they contained hands. Another requirement was that the images appeared reasonably "natural". This limited the epochs essentially to the range from the Renaissance (from about 1420) to Realism (until about 1850). In the epochs before the Renaissance, the pictures are often very flat and the properties are often not correct. In Impressionism and later

periods, the quasi "photographic" image is increasingly replaced by a freer interpretation of form and color, until in abstract painting the "natural" forms disappear completely.

The results of the similarity search surprised me very positively, as the examples below show. As a reference image I used - among others - Leonardo Da Vinci's "Mona Lisa". Figure 2.11 shows three of the found images that are similar in terms of gestures. While the top right image clearly shows a great similarity of the whole posture, the similarity of the others is not directly obvious. However, in Figure 2.12 you can see that the posture of both of the left hand is quite similar to that of Mona Lisa. In the final analysis, it can be stated that the basic process chain works, but that there is certainly still considerable potential for optimization.

Figure 2.11: The result of my project. Mona Lisa is given as a input picture. The three pictures on the right are pictures, that have similar gestures as Mona Lisa.

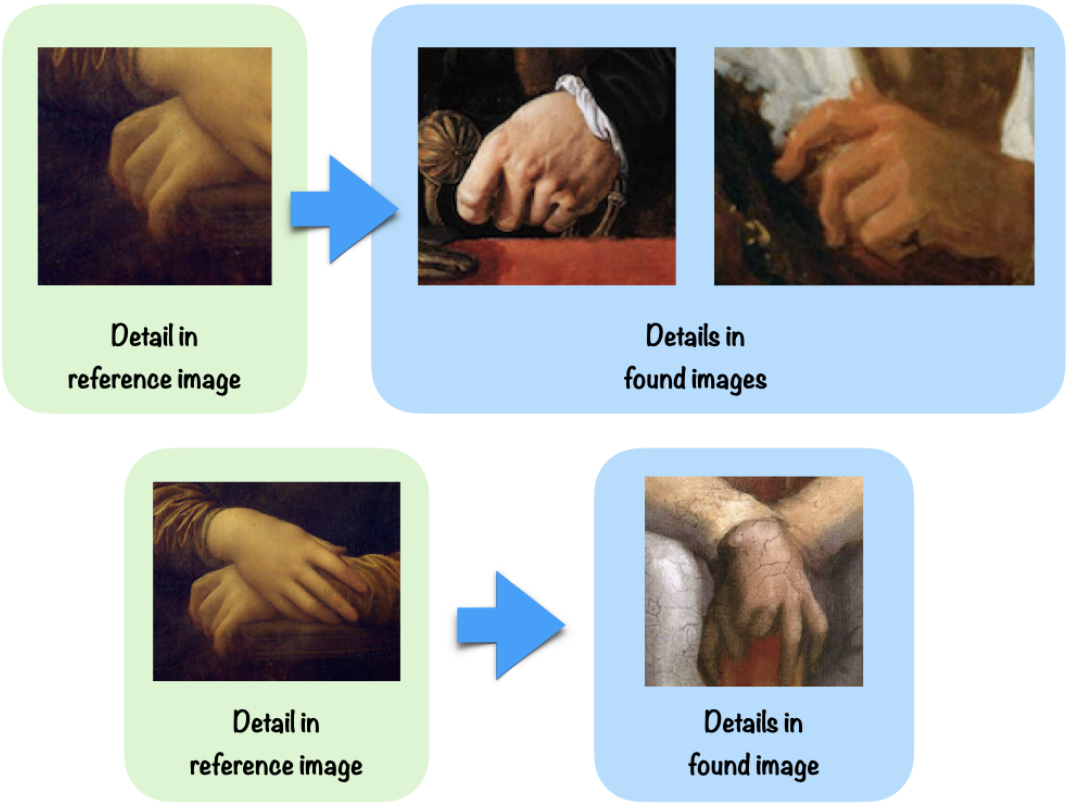

Figure 2.12: Detail view of the search result. In the upper part, the left hand of the Mona Lisa corresponds very exactly to the hand position in the found images. In the lower part, the similarity of the posture of the two hands in the two images is very great.

2.6 Implementation challenges

It was the first time in my studies of computer science where I had to implement a somewhat more complex workfow for machine learning. As expected, some difficulties arose, whose solutions were quite time-consuming due to the lack of experience and led to considerable delays, especially in the first part of the project. In the following subchapters I will go into the individual problems, explain what did not work as expected and where improvements could be made.

2.6.1 High Performance Cluster of the Department of Computer Science

The first attempts to install the programs on my personal computer (laptop and deskside), intel-based Apple with OS X, failed because the graphics cards were not of the "Nvidia GTX" type. In particular, "tensorflow" uses the Nvidia-specific library "CUDA". So unfortunately I could not use my personal computers for the development. After consultation with my supervisor I got an account on the HP-Cluser of the department, which is equipped with the necessary graphic cards. For understandable reasons I did not get privileged ("root") access. A first step was to

build a development environment for myself. I chose the IDE PyCharm from Intellij, which is relatively easy to configure for remote access.

2.6.2 Version dependencies ("the dependency hell")

Since I relied on existing libraries, which implement complete processes (processing pipelines), which in turn were based on a variety of other libraries, I came into contact with what is commonly known as "the dependency hell". Both the Detectron 2 and then also CDCL needed very specific - and even at the present time - already outdated versions of libraries (e.g. a special CUDA version). Over the course of the whole project I needed to install all the appropriate versions for the libraries I used on the cluster on my account. However, since Detectron 2 and CDCL had overlapping libraries, downloading the same libraries of different versions multiple times led to complications that resulted in my supervisor having to restart my part of the cluster several times. With such experiences, it had become clear to me that working on the cluster, without sudo rights as well as having the potential to crash my part of the cluster with my supervisor needing to reset the system, does prevent me from making efficient progress. I had to find a solution where I could afford to make mistakes without having to deal with a waiting period afterwards.

2.6.3 Linux PC with Nvidia

During a conversation, a good friend offered to provide me for a time with his own powerful PC equipped with a Nvidia graphics card. However, in order for this computer to be useful to me I had to install a compatible operating system (Ubuntu Linux) as dual-boot next to MS Windows itself.

Based on the experiences with the cluster of the department, I was looking for a way to bring the system into a previously saved state in case of an error or incompatibility caused by my working endeavours. Snapshotting, i.e. creating a momentary record of the system, as well as the possibility of easily restoring that state if necessary, seemed to be the best solution. My research revealed that the best variant for snapshotting on a ubuntu system is the combination of the "BTRFS" file system combined with the program "Timeshift" for the actual snapshotting itself. BTRFS is a "copy-on-write" file system that allows for easy snapshotting and fast recovery to a previous state. This is because copy-on-write basically only records files that changed, but not all the other files. So to return to a previous system state the program only has to undo all file changes that have been registered since the time of the snapshot. The combination of BTRFS and Timeshift helped me often and much during my experiments since I was able to reset the system to a stable working state within seconds.

Unfortunately, however, it became apparent during the course of the experiments with the newly

provided computer of my friend that the new Nvidia graphics card was not compatible with the required versions of the necessary libraries. Although the programs ran without errors, the results were quite obviously faulty and not usable. With that, I had to somehow get the pro-

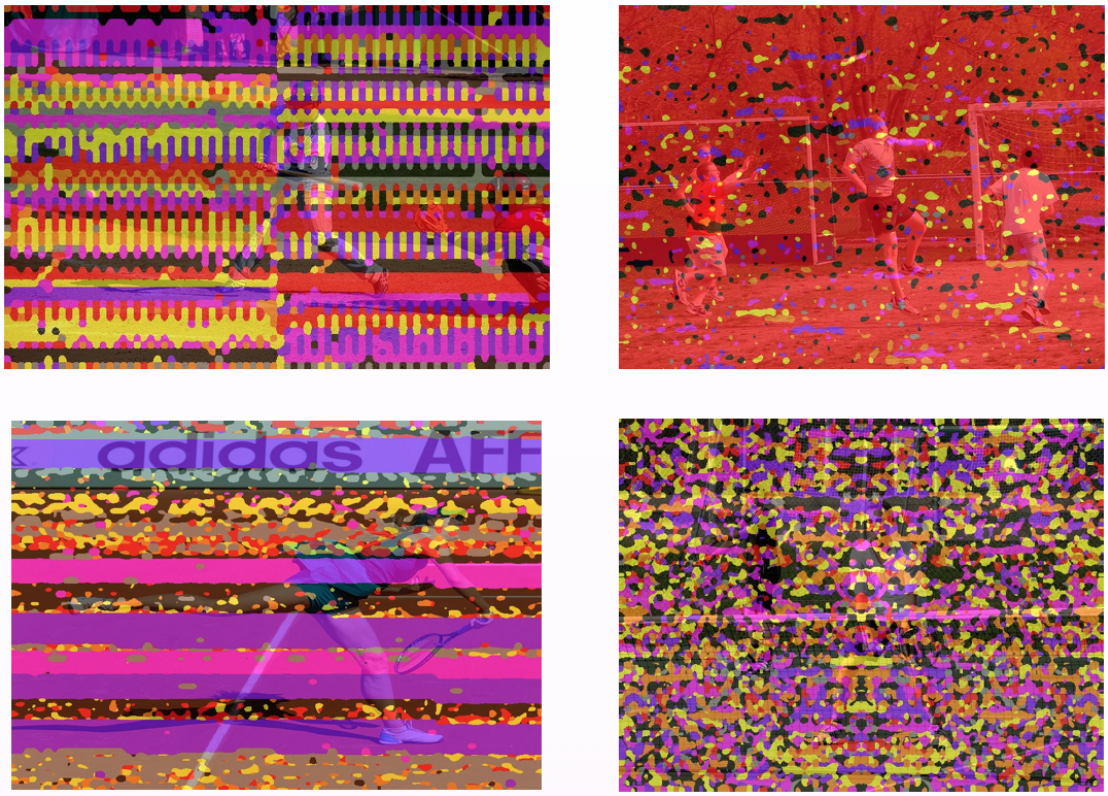

Figure 2.13: Some of the "modern art" my computer produced. It's clear, that, despite the program running without an error, something goes horribly wrong.

grams to run error-free on compatible hardware. To further improve the encapsulation as well as maybe get a working system running one the cluster, Docker seemed to be a possible solution - especially since Detectron2 as well as CDCL provided its users with a already existing Docker file.

2.6.4 Docker as a Solution

Docker is one way of "weak" virtualization. A Docker container runs in the operating system of the host system and uses its kernel functions, but all other components and libraries are isolated from the host system. This means that any Linux environment can be virtualized in a Docker container, which runs highly efficiently on the host and can also use its hardware - specifically also, for example, graphics card. This gave me a tool to generate exactly the environment my workflow required and run it on any system, that had the necessary hardware. I could develop

the Docker configuration on my local PC without risk, even though the results computed with it were unusable because of the non-compatible graphics card and afterwards install the Docker via the configuration file on the cluster and run the containers there. To get the Docker container run on my provided computer, i needed however to change and adjust the existing Docker file as well as contribute my own code to it. However, in order to run Docker containers on the cluster, I needed to be granted privileged access. This was eventually granted to me.

Fortunately, initial experiments quickly showed that I could then run my workflow on the cluster as a Docker image very efficiently and without errors or obviously false results. This finally gave me the opportunity to work on the content of my task.

It was an exhausting and time consuming experience to see how much effort it took to get research programs that were several years old up and running. On the one hand, the limited possibilities to install old library versions on the cluster, on the other hand, non-transparent incompatibilities within the Nvidia graphics family, where suddenly a single card of the same basic type leads to unusable results, surprised me and the development of a solution cost me a lot of valuable project time. Furthermore, I have gained invaluable experience and I have learned a lot. In particular, the principle of Docker has impressed me positively.

3
Discussion

The task of my project work was to adapt existing pre-trained neural networks (if necessary) and assemble them into a workflow that enables automatic gesture recognition and retrieval on artistic images. To characterize the gestures, I focused on the hands because they have a prominent role in gestures. The initial design of the process chain (hand detection → isolation of hands → generic feature vectors → similarity search) has proven to be a rather successful strategy.

The CDCL network could be adapted to detect and isolate only the hands and forearms relatively reliably. However, the algorithm often cut off the fingers. Since the whole hand including the fingers is very important especially for gestures, this posed a problem. This seems to be a problem not only with artistic images (paintings), but also with photographs. However, with the help of a dilation, the region around the hands could be extended, so that the calculation of the feature vector should take the whole hand into account.
But although CDCL is one of the most advanced body segmentation programs, it also has flaws. Sometimes it doesn't recognize the hands correctly or produces mask artifacts that shouldn't be there. However, I didn't dare hope to improve the accuracy of the body segmentation, since it was developed by a research group with many more people who have more time, resources and experience. A more accurate hand recognition would however surely improve the gesture retrieval.

When calculating the similarity search, there are two major potential sources of error that reduce the accuracy of the similarity search.
First, the feature vector is calculated from the whole mask-image, i.e. from the mask created during image segmentation and from the black background. It is not obvious whether the feature vectors calculated by Vgg16 really take the image content of the segmented body parts and not also the shape of the masks into account.
Second, I use Vgg16, which is trained for image segmentation i.e. recognizing arbitrary objects. It was not trained to recognize hands and arms. This affects the accuracy of the created feature

vectors.
Vgg16 is used instead of the CDCL because the Vgg16 net is already provided as default model by Keras. At the end of my project I was not able to extract the second last layer of the neural network from CDCL due to lack of time, so I used the default version of Vgg16.

The k-means clustering I initially wanted to implement would require a corpus of annotated gesture images. Such a corpus of images does not seem to exist and would certainly be a desideratum. A small literature search revealed that while there are some works on gesture in painting, they often refer to modern abstract painting (e.g. Frozen Gesture – Gesten in der MalereI Bitterli et al. [1]). A dissertation of the University of Tübingen with the title "Die Gesten im malerischen und zeichnerischen Werk Raffaels" by Martin Franz Mäntele [15] unfortunately only contains a textual analysis of the gestures. The book "Strukturen und Schauplätze der Gestik - Gebärden und ihre Handlungsorte in der MalereI des ausgehenden Mittelalters" Deuchler [6] contains a small picture atlas, but it is only in poor quality and not digitized. The image database of DaSCH of the archive of the Kunsthalle Basel[8] contains a lot of images of modern painting, which is not suitable for the task. Finally I found an online database, the "Web Gallery of Art"[9], which is searchable by keywords. There I was able to compile by hand a small corpus of about 150 images, which contain artistic representations relevant to gestures. This - actually too small - corpus allowed me to successfully test the process chain. However, an image corpus of only 150 images is by far not sufficient to test the system in a thorough way. A database with at least 1000 images would be desirable.
Due to the too small image corpus it seemed also pointless to me to perform statistical tests with my gesture retrieval. The statistical statements that could be made on the basis of such a small corpus would have no significance at all.

Despite all justified criticism, this work shows that by cleverly adapting and assembling existing neural networks, a task such as recognizing and retrieving gestures in artistic representations, especially paintings, is in principle possible. It is obvious that the results can be significantly improved by deeper adaptations of the individual networks and the whole process chain. This would also require a much larger corpus of suitable images, if possible even unannotated with respect to gestures.

Another point this project managed to highlight is that if possible, experiments should be done in a containerized environment like Docker from the beginning. The ability to configure each individual container for the necessary environment allows process chains to be built where the individual parts have different and mutually incompatible demands on the environment.

[8] See https://www.salsah.org/kuhaba/.
[9] https://www.wga.hu/index1.html

4
Outlook

If I could continue my project, I would first of all revise the distance metric of my similarity search. One of the weakest links in my project is the missing connection between feature vector and distance metric to calculate the similarity. Optimally, one would have a distance metric that is exactly tailored to my hand detection problem. In the optimal case, a distance metric would be trained together with the neural network, which is responsible for the creation of the feature vectors.

Using Vgg16 to produce the feature vectors is another starting point to improve my work. I use an - albeit modern - generic image segmentation network that recognizes cats and guinea pigs just as well as it recognizes gestures (presumably it recognizes dogs and cats better than gestures). So I use a neural network that is not designed to distinguish different gestures from each other. If the work could be taken further, surely a reasonable approach would be to have the feature vectors generated by a neural network programmed to recognize gestures, fingers, or at least hands and their position. Since this is not the case in my work, the created feature vectors are inaccurate, which in turn allows only a poor accuracy of the similarity measurement.

In addition, a larger image database that has suitable images is absolutely necessary to further test the system and make statistical analyses of the accuracy of the program. Creating this image database would be a necessary step in the further development of gesture retrieval.

As a final point, it should be noted that CDCL is a neural network that has been trained to compute body segmentation on photographs. It is true that CDCL does a pretty good job on most realistic paintings and sketches, but a neural network trained specifically to recognize human extremities in different art and style environments would be certainly more accurate.

5
Acknowledgements

I would like to thank Prof. Schuldt as examiner who made this project possible.

Furthermore, I would like to thank Dr. Mahnaz Parian-Scherb for the time she invested and the great support she gave me.

Bibliography

[1] Konrad Bitterli, Lynn Kost, and Andrea Lutz, editors. *Frozen Gesture – Gesten in der Malerei*. HIRMER, 2019.

[2] François Chollet et al. Keras. https://github.com/fchollet/keras, 2015.

[3] codeproject.com. Visualisation of a cnn, 2021. URL https://www.codeproject.com/Articles/5160467/Image-Classification-Using-Neural-Networks-in-NET.

[4] Wikimedia Commons. Albrecht dürer, praying hands, 2021. URL https://en.wikipedia.org/wiki/Prayer.

[5] Denverpubliclibrary. Webpage of the denver public library, 2021. URL https://history.denverlibrary.org/western-history-collection.

[6] Florens Deuchler. *Strukturen und Schauplätze der Gestik – Gebärden und ihre Handlungsorte in der Malerei des ausgehenden Mittelalters*. De Gruyter, 2014.

[7] Elastic. Webpage of elastic, 2021. URL https://www.elastic.co/de/elasticsearch/.

[8] Europeana. Webpage of the europeana, 2021. URL https://pro.europeana.eu/about-us/mission.

[9] Ross Girshick. Fast r-cnn, 2015.

[10] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation, 2014.

[11] Nicolas Gonthier, Yann Gousseau, Said Ladjal, and Olivier Bonfait. Weakly supervised object detection in artworks. *Computer Vision – ECCV 2018 Workshops*, pages 692–709, 2019. ISSN 1611-3349. doi: 10.1007/978-3-030-11012-3_53. URL http://dx.doi.org/10.1007/978-3-030-11012-3_53.

[12] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn, 2018.

[13] Kevin Lin, Lijuan Wang, Kun Luo, Yinpeng Chen, Zicheng Liu, and Ming-Ting Sun. Cross-domain complementary learning using pose for multi-person part segmentation. *IEEE Transactions on Circuits and Systems for Video Technology*, 2020.

[14] Kevin Lin, Lijuan Wang, Kun Luo, Yinpeng Chen, Zicheng Liu, and Ming-Ting Sun. Cross-domain complementary learning using pose for multi-person part segmentation. *IEEE Transactions on Circuits and Systems for Video Technology*, 31(3):1066–1078, 2021. doi: 10.1109/TCSVT.2020.2995122.

[15] Martin Franz Mäntele. *Die Gesten im malerischen und zeichnerischen Werk Raffaels*. PhD thesis, Eberhard-Karls-Universität Tübingen, 2010.

[16] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks, 2016.

[17] Xi Shen, Alexei A. Efros, and Mathieu Aubry. Discovering visual patterns in art collections with spatially-consistent feature learning, 2019.

[18] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition, 2015.

[19] Nikolai Ufer, Sabine Lang, and Björn Ommer. Object retrieval and localization in large art collections using deep multi-style feature fusion and iterative voting. In Adrien Bartoli and Andrea Fusiello, editors, *Computer Vision – ECCV 2020 Workshops*, pages 159–176, Cham, 2020. Springer International Publishing. ISBN 978-3-030-66096-3.

[20] Mahisorn Wongphati, Yusuke Kanai, Hirotaka Osawa, and Michita Imai. Give me a hand — how users ask a robotic arm for help with gestures. In *2012 IEEE International Conference on Cyber Technology in Automation, Control, and Intelligent Systems (CYBER)*, pages 64–68, 2012. doi: 10.1109/CYBER.2012.6392528.

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

[22] Rujie Yin, Eric Monson, Elizabeth Honig, Ingrid Daubechies, and Mauro Maggioni. Object recognition in art drawings: Transfer of a neural network. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2299–2303, 2016. doi: 10.1109/ICASSP.2016.7472087.

Declaration on Scientific Integrity
Erklärung zur wissenschaftlichen Redlichkeit

includes Declaration on Plagiarism and Fraud
beinhaltet Erklärung zu Plagiat und Betrug

Author — Autor
Manuel Rosenthaler

Matriculation number — Matrikelnummer
12-067-831

Title of work — Titel der Arbeit
Hand Gesture Retrieval
in
Art Image Collections

Type of work — Typ der Arbeit
Master project

Declaration — Erklärung
I hereby declare that this submission is my own work and that I have fully acknowledged the assistance received in completing this work and that it contains no material that has not been formally acknowledged. I have mentioned all source materials used and have cited these in accordance with recognised scientific rules.

Hiermit erkläre ich, dass mir bei der Abfassung dieser Arbeit nur die darin angegebene Hilfe zuteil wurde und dass ich sie nur mit den in der Arbeit angegebenen Hilfsmitteln verfasst habe. Ich habe sämtliche verwendeten Quellen erwähnt und gemäss anerkannten wissenschaftlichen Regeln zitiert.

Basel, June 30, 2021

Signature — Unterschrift