JOHANNES KEPLER
UNIVERSITY LINZ

Author
Philipp Hofer

Submission
Institute of Computational Perception

Thesis Supervisor
a.Univ.-Prof. Dr. Josef Scharinger

April 2020

Gait recognition using neural networks

Master's Thesis
to confer the academic degree of
Diplom-Ingenieur
in the Master's Program
Computer Science

**JOHANNES KEPLER
UNIVERSITY LINZ**
Altenbergerstraße 69
4040 Linz, Österreich
www.jku.at
DVR 0093696

Abstract

Methods for recognizing people are both heavily researched presently and widely used in practice, for example by government and police. People can be recognized using various methods, such as face, finger and iris recognition, which differ in terms of requirements massively. *Gait recognition* allows identifying people despite large distances, hidden body parts and with any camera angle - which makes it a naturally attractive method of identifying people.

This approach uses the uniqueness of gait information in every person. Most of the current literature focuses on hand-crafting features, such as step and stride length, cadence, speed and hip angle. This thesis proposes a way of performing gait recognition using neural networks. Hence, features have not to be specified manually anymore, while also boosting current state-of-the-art accuracy of being able to recognize people.

First, in order to increase the robustness against cloth-changes, the silhouette from a person is extracted using Mask R-CNN. In order to capture spatial information about the subject, a convolutional neural network creates a gait-embedding based on each silhouette. To augment the quality, the next step is to take temporal information into account, using a long short-term memory network which uses the single-picture-based embedding of multiple images and computes its own, enhanced, embedding. Last but not least, the network should not be trained for every new person from scratch. Thus, a Siamese network is trained to be able to distinguish two people, which the network has (probably) never seen before.

Kurzfassung

Aktuell gibt es viele Forschungen und veröffentliche Paper im Bereich der Gangerkennung. Außerdem ist diese Methode bereits weit verbreitet, zum Beispiel bei der Regierung und Polizei. Es ist möglich, Personen aufgrund zahlreicher Eigenschaften zu identifizieren, unter anderem aufgrund des Gesichts, des Fingers und der Iris. Diese Möglichkeiten unterscheiden sich in ihren Anforderungen gewaltig. *Gangerkennung* ermöglicht die Identifikation einer Person trotz großer Entfernung, versteckten Körperteilen und unter allen möglichen Kamerawinkeln. Aus diesem Gründen ist Gangerkennung eine Methode, die ohne vielen Anforderungen auskommt.

Diese Herangehensweise nützt die Einzigartigkeit des Ganges von jeder Person. Ein Großteil der aktuellen Literatur fokussiert sich auf das manuelle Erstellen von Merkmalen, unter anderem Schrittlänge, Rhytmus, Geschwindigkeit und Winkel der Hüfte. Diese Arbeit schlägt eine Methode vor, um den Gang mittels neuronalen Netzwerken zu erkennen. Auf der einen Seite müssten Merkmale so nicht mehr manuell spezifiziert werden, auf der anderen Seite wird die Genauigkeit im Vergleich zu aktuellen Systemen verbessert.

Um gegen Kleidungswechsel robust zu sein, wird die Silhouette einer Person mittels Mask R-CNN Netzwerk extrahiert. Um die räumlichen Informationen eines Bildes zu extahieren, wird ein konvolutionelles neuronales Netzwerk verwendet, welches diese Informationen in einem hochdimensionalen Vektor speichert, welcher aus jeder einzelnen Silhouette erstellt wird. Die Qualität wird im nächsten Schritt erweitert, indem zusätzlich zeitliche Informationen hinzukommen. Es wird ein langes Kurzzeitgedächtnis verwendet, welches mehrere Bilder verwendet. Nachdem das Netzwerk nicht für jede Person neu trainiert werden soll, wird ein Zwillings-Netzwerk verwendet um zwei Personen, welche (wahrscheinlich) noch nie vom System gesehen wurden, zu unterscheiden.

Contents

Chapter 1

Introduction

Identifying people for specific purposes based on certain metrics is a fundamental component of life in today's advanced industrial societies. The applications are endless, the government, for example, wants to recognize known terrorists at airports or it is surely beneficial to detect someone with dementia on a street if they decided to go on a stroll all alone. Thus, being able to identify people for specific purposes should be of public interest.

One way of identifying people is using face recognition. Today, this method is so widespread, that it is not only used by official entities, such as the security screening in an airport, but also by regular people which can use face recognition to unlock their digital devices. Due to the impressive accuracy, face recognition has its right to exist. On the other hand, face recognition is not an all-in-one solution. One of the biggest downsides is that face recognition needs a relatively high quality picture, thus requiring either an expensive high-quality camera or a short distance to the subject. This aspect is not far-fetched but widespread - keeping to the example of airport security where the police wants to identify known terrorists, it might be too late when the camera got a decent picture. Another problem is that it is possible to obscure the face, for example by wearing clothing which covers parts of the face, which may render face recognition worthless.

A good method for identifying people allows for an unobtrusive way of doing it, despite large distances and hidden body parts. Furthermore, in order to be as practically useful as possible, the effectiveness should not depend on the camera angle. Thus, to address these problems, this thesis focuses on gait recognition, which has promisingly few requirements. Research in this field has focused on hand-crafting the features, for example creating GEI images [1]. Up to the authors knowledge, there has not been an approach using only neural networks for performing gait recognition, thus this thesis proposes a way of doing just that.

We propose a way of recognizing a person based on a few seconds worth of video using its gait information. The scope of this thesis is limited to the extent that the input video features only a single person. By way of contrast, there are not any other constraints on the video:

- The camera can be positioned anywhere (either the same height as the person, lower or higher than the person).

- The background can be cluttered.

- The step does not have to be continuous, and the video can start in any phase of the step.

- Other moving objects can be present.

- The algorithm should be robust against different surrounding conditions, such as different shoes, walking speed, carriage load, underground and clothing.

1.1 Motivation

As motivated in the previous section already, being able to recognize people is of broad interest. There exist a lot of methods for uniquely identifying a human, the following list is just a small excerpt of current, widely-used techniques. Table 1.1 summarizes these requirements. Section 2.1 deals with these aspects in more detail.

- Face Recognition
  In order for face recognition to work you either need to have a picture taken from a (very) close distance or you have to use a high-resolution camera. Furthermore, the subject has to be cooperative, if it does not look in the camera or hide its face, this technique will not work.

- Fingerprint Recognition
  Since a person has to physically place the finger on a specific device, this method requires most cooperation.

- Iris Recognition
  Similar to *Face Recognition* the input image needs to be of particularly high quality.

- Hand Recognition
  This technique may be less common, although studies have shown that also the

	Face	Fingerprint	Iris	Gait
Large distance / low resolution possible	×	×	×	✓
Body parts can be hidden	×	×	×	✓
Not dependent on specific view	×	×	∼	✓
Not dependent on cooperation of suspect	∼	×	∼	✓
Can be used for mass surveillance	✓	×	∼	✓

Table 1.1: Necessary requirements for common recognition methods

shape of the hand can be used to uniquely identify a person [2]. In 2016 Alpar et. al. [3] proposed a technique of using the unique shape of the back of the hand to recognize a person. For this type of recognition the algorithm needs to work with high-quality pictures and the hand needs to be in a certain position, i.e. it will not work if the person makes a fist.

To build a system which is robust against most of these drawbacks, this thesis proposes the use of gait information. This is a non-obtrusive way of recognizing people. Even without a state-of-the-art camera it is possible to detect people at distances of hundreds of meters. Although it is possible to obfuscate your identity to a gait recognition system, for example by putting a stone in your shoe, it is much harder to do than fooling any of the previously mentioned recognition methods. Furthermore, gait recognition does not depend on any specific view. In fact, this thesis uses a dataset which features 11 different camera perspectives, ranging from 0 degrees to 90 degrees and up to 180 degrees - thus including every possible view.

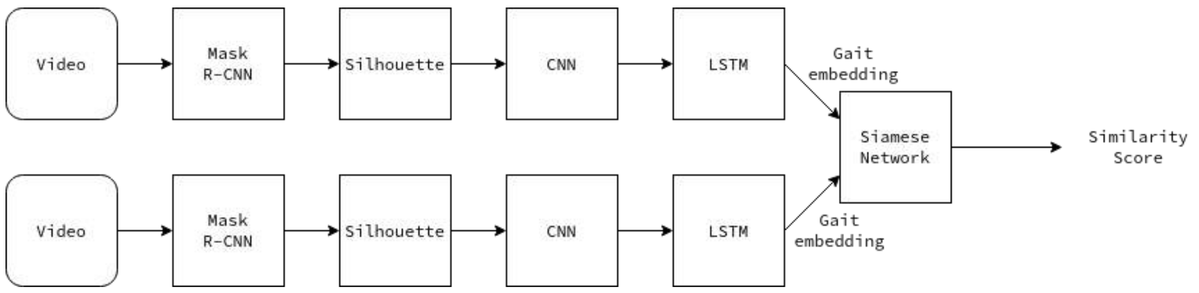


Figure 1.1: Architecture for the networks of this thesis

1.2 Research approach

This thesis proposes a method of recognizing people based on their gait information using neural networks. This is accomplished by first extracting the silhouette from each frame in the gait video using a Mask R-CNN neuronal network [4]. Next, both the spatial and temporal information of the silhouette is extracted using a convolution neural network and a long short-time memory [5], respectively. In order to be able to recognize people which the system has never seen before, a Siamese network [6] is trained to distinguish between two people. This process is visually shown in Figure 1.1.

1.3 Outline

The rest of this thesis is structured the following way:

- Chapter 2 outlines the basic concepts of all tools used in this thesis.
- The next chapter explains how to extract silhouettes from images using a neuronal network.

- How the spatial gait information is extracted using a convolutional neural network is described in Chapter 4.

- To also include the temporal information, a long short-term memory [5] is used. How this is done is explained in Chapter 5.

- Chapters 6 and 7 analyze how to combine the previous approaches to compute a similarity score.

- Last but not least, this thesis sketches related work and completes with a conclusion.

Chapter 2

Background

This chapter explains why and how biometrics can be used to recognize and identify people. This thesis uses different neural networks, thus in Section 2.2 we introduce convolutional neural networks, Mask R-CNN [4], recurrent neural networks and Siamese networks [6]. Furthermore, to drastically decrease training time, the concept of transfer learning is introduced as well. In the last section, we will talk about available gait datasets - which play an important role for training the neural networks, and thus impact the quality of the results immensely.

2.1 Biometrics

Every human being has many biometric features which can be used to uniquely identify a human [1]. This chapter will outline various benefits and disadvantages of different

[1]See list in Section 1.1.

biometric traits.

2.1.1 Face recognition

With face recognition it is possible to perform mass identification. It does not necessarily depend on the cooperation of the subject, Thakkar et. al. [7] argue, that in some instances the crowd is not even aware of the system. Furthermore, a study by Phillips et. al. shows, that algorithms are already superior to humans for matching frontal faces in images [8].

On the downside, face recognition requires a high-quality picture. A lot of factors may influence the result:

- **Illumination [7]:** Liu et. al. [9] showed, that illumination drastically changes the appearance of a face.
- **Expression [7]:** A smiling person may obfuscate a face recognition software. This is the reason why most countries require a neutral face expression in passport photos.
- **Pose [7]:** In 2017, [10] argues that *pose discrepancy between two face images is one of the key challenges in face recognition.*
- **Viewing angle:** [11] argues, that *Face recognition has been getting pretty good at full-frontal faces and 20 degrees off, but as soon as you go towards profile, there have been problems.*

2.1.2 Fingerprint recognition

Fingerprint recognition has high reliability and the systems are small and cheap. These are some reasons, why this technique is widely popular. Fingerprint sensors are used in many areas, such as in mobile phones [12], cars [13], doors [14], border control and airports.

Unfortunately, fingerprint recognition is not particularly useful for the scope of this thesis, because it is neither an unobtrusive way, nor can it be performed on many subjects at the same time, or operate from the distance.

2.1.3 Iris recognition

First things first, although Iris recognition systems, in contrast to fingerprint systems, do not need direct contact, a high-quality, close-up shot is required. Traditional approaches can handle distances of less than a meter only. A study in 2016 [15] helps to relax this constraint by using machine learning techniques, allowing distances of up to three meters. Vatsa et. al. [16] show, that it is possible to achieve 99.9% accuracy.

On the flip-side, iris recognition systems can be fooled with an image of an iris. Furthermore, the quality is highly affected by lighting and iris recognition system are quite expensive. Similar to fingerprint systems, they are not really unobtrusive, since it is not possible to get a high-quality image from a distance of more than 1 meter (yet[2]). Thus, this system is not usable for mass identification.

[2]See [15] for an approach which might make *iris at a distance* more feasible.

2.1.4 Gait recognition

Since gait recognition is not as popular as the previously mentioned methods, we will introduce the basic concept of how gait recognition works in this section.

Recognizing people based on gait information can be done with a lot of modalities, mainly through videos [17] [18], sensor mats [19], sensors in shoes [20], audio signals [21] [22] and smartphone sensors [23] [24]. All these approaches are featured in figure 2.1. As the cited paper argues, the characteristics of these features are unique to every person. For example, the skeleton and the way a person walks is different for everyone and the pressure of the foot onto the ground characterizes the gait.

Although gait is unique to everyone, there are many influencing factors at work. Over time, gait will change because of injuries [26], pregnancies [27], aging in general [28], weight change [29], and so on. Interestingly, gait is so expressive, that Perera et. al. argued in 2016 [30] that gait speed can make prediction about disability and mortality over a 3-year time frame.

With gait recognition it is possible to identify people unobtrusively, requiring only few assumptions [3], which makes it the perfect methodology for this thesis.

[3]See list in the introduction of Chapter 1.

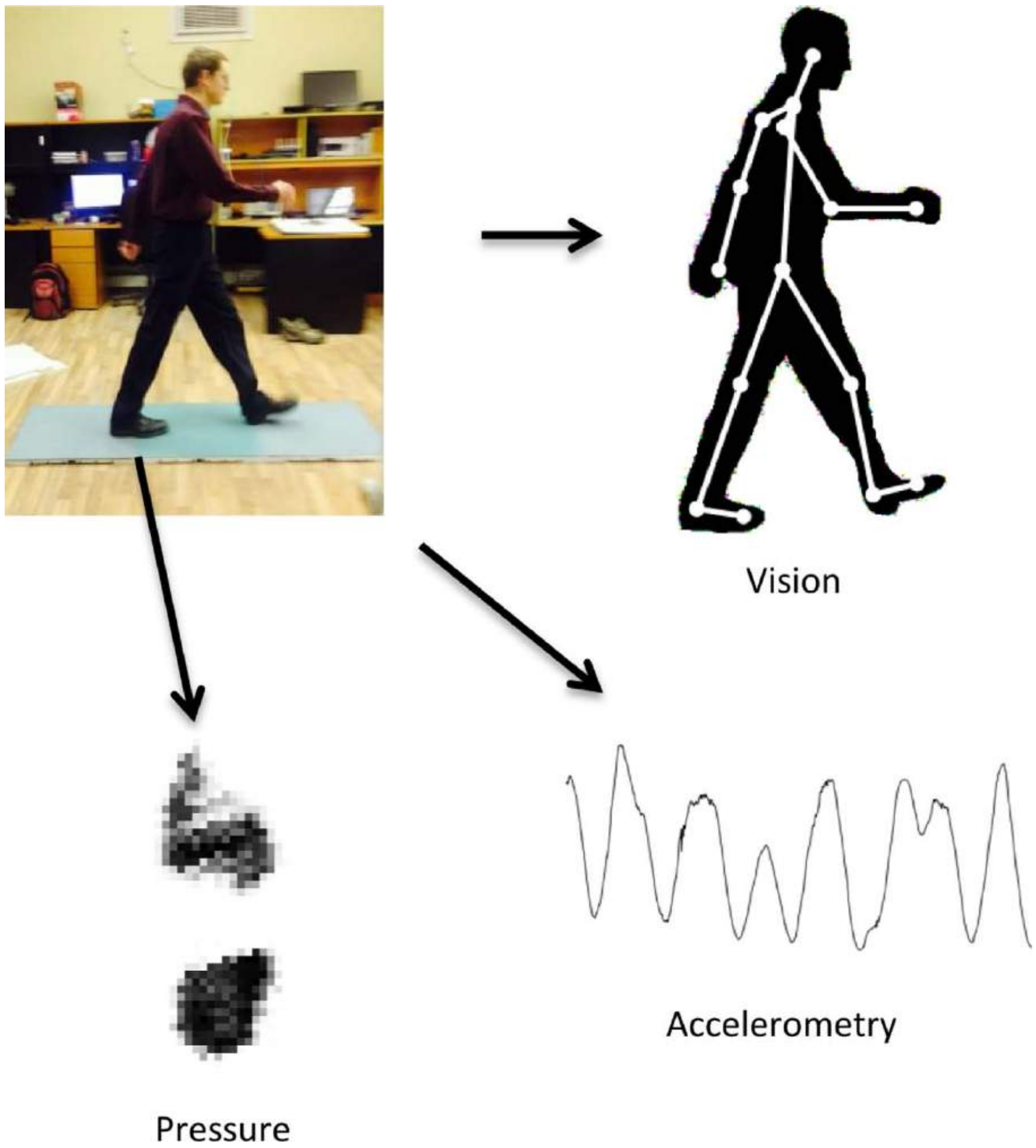


Figure 2.1: Different sensor modalities used for gait recognition. Figure taken from [25]

2.2 Neural networks

This chapter introduces the specific neural networks which are used in this thesis, Mask R-CNN [4], a convolutional neuronal network, LSTM [5], and Siamese network [6]. Furthermore, in order to dramatically reduce the training time, in Section 2.2.6 the concept of transfer learning and how it is used in this thesis is explained. To being able to train the network optimally, in Section 2.3 we go into detail about the two large datasets we used.

2.2.1 Convolutional neural networks

Convolutional neural networks (CNNs) are widely used in image related tasks. They consist of neurons with learnable weights and biases. Each neuron gets multiple inputs, computes a weighted sum over them and then applies an activation function. There are two main functions which can be applied: Convolution and Pooling. Convolution uses the assumption that neighboring pixel are highly correlated. A kernel is slid over the complete image and computes the dot product between the kernel and the part of the image covered by the kernel. It is possible (and very common) to have multiple kernel in every stage. Thus, the amount of layers increase in every step. This procedure is shown in Figure 2.2. In order to stop dimensionality explosion, pooling layers are used. They operate on each layer independently and combine neighboring pixels into one cell, as shown in Figure 2.3.

In this thesis, there are two CNNs at work:

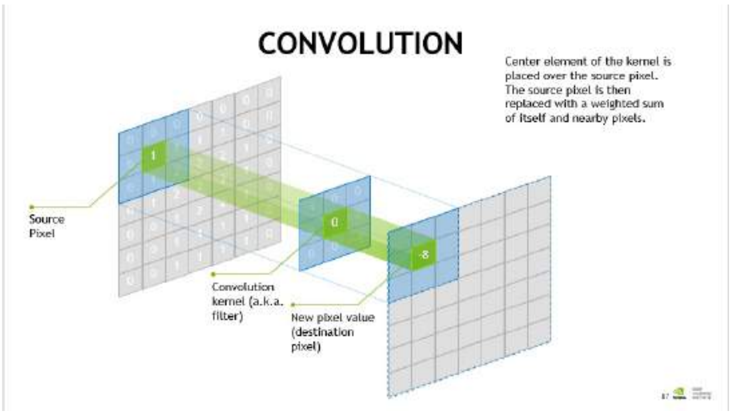

CONVOLUTION
Center element of the kernel is placed over the source pixel. The source pixel is then replaced with a weighted sum of itself and nearby pixels.
Source Pixel
Convolution kernel (a.k.a. filter)
New pixel value (destination pixel)

Figure 2.2: Concept of convolution. Figure taken from `https://blogs.nvidia.com/blog/2018/09/05/whats-the-difference-between-a-cnn-and-an-rnn/`

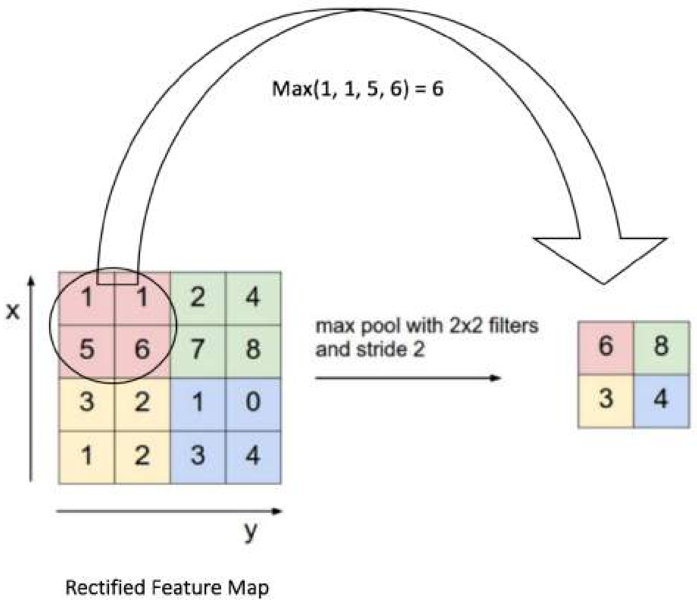


Figure 2.3: Concept of pooling. Figure taken from `https://ujjwalkarn.me/2016/08/11/intuitive-explanation-convnets/`

1. A CNN is used to extract the silhouette of a person from an image, c.f. Chapter 3.

2. Another CNN is used for calculating the gait embedding for a single image, c.f. Chapter 4.

2.2.2 Mask R-CNN

This thesis uses Mask R-CNN to extract the silhouette of a person. Thus, the approach of Mask R-CNN is explained in this section.

Instance segmentation Mask R-CNN[4] performs instance segmentation on images. This is significantly harder than both *classification* and *object detection*. Classification is able to infer that there is a person in the image. The result of object detection is that there is a person in a specific area. Finally, instance segmentation gives the concrete pixels of each person.

Algorithm On the high-level, Mask R-CNN[4] operates in two steps:

1. Process images and generate *proposals*, parts of the image where chances are high that they contain an object.

2. Classify proposals from step 1 and generate both bounding boxes and masks.

We can further split Mask R-CNN[4] into different parts:

Step 1

- Standard CNN to extract features, both low level (edges, corners, ...) and high level (person, building, glass, ...)

- Use sliding-window principle to find areas which contain objects. There are over 200.000 different windows for each image, with both different position and size to cover for all possibilities. The windows which have the highest probability of containing an object will be used further on.

Step 2

- Each of the extracted windows from step 1 are fed into another convolutional neural network. There are two outputs: 1. the class of the object 2. bounding box refinement to further refine both the size and the location of the bounding box.

- The last step is to generate a mask for each of the bounding boxes. This is done using yet another convolutional network. This output is the final one, a mask for every detected object.

2.2.3 Recurrent Neural Networks

In a recurrent neural network (RNN), the output does not only depend on the current input, but also on previous inputs (red arrow in Figure 2.4). This concept is shown in Figure 2.4. This is achieved by feeding the output from the previous step as input to the

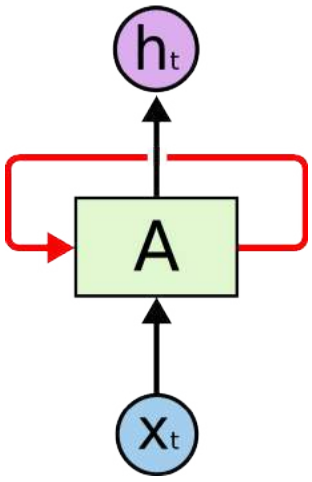


Figure 2.4: Structure of an RNN. Figure taken from `https://colah.github.io/posts/2015-08-Understanding-LSTMs/` and slightly edited

current step. RNNs are mainly used for three applications:

1. Sequence classification

2. Sequence labeling

3. Sequence generation

This thesis will use RNNs for *sequence classification*.

RNNs are used heavily in practice, for many problems: speech recognition, language modeling, translation, image captioning, embedding extraction and many more.

2.2.4 Long short-term memory

The long short-term memory (LSTM) is a special kind of RNN. The main problem with RNNs is, that the context is limited, because the gradient gets smaller and smaller with every layer [4]. The LSTM cell solves this problem by not storing every information, but rather choosing *some* information which is stored long-term. This is done by carefully removing or adding information to the current cell state, which is regulated by structures called *gates*.

2.2.5 Siamese network

Although the basics for Siamese networks have been introduced in the 1990s by Bromley et. al. [32] already, these neuronal networks are not as popular as, for example, CNNs. Thus, the basic idea behind Siamese networks are described in this section.

A regular neural network takes a lot of input data, feeds them through a series of layers and outputs its respective class probabilities. If the goal is to classify images of people into two different categories - lets say images which contain Konrad Zuse and images which contain Edsger Dijkstra, the classical way to go is to create a massive database with a lot of images of both people. Then, the network is trained to classify images correctly. After the training, the network is able to classify only between Mr. Zuse and Mr. Dijkstra. If the task changes, and you are interested to know whether there is a John Smith in the image, the network would have to be trained again, from scratch. An additional problem is that often this kind of training data do not exist. For example, if you want to classify a specific flower into classes, you might only have a few samples for every class.

[4]See literature for *Vanishing gradient problem*, e.g. [31].

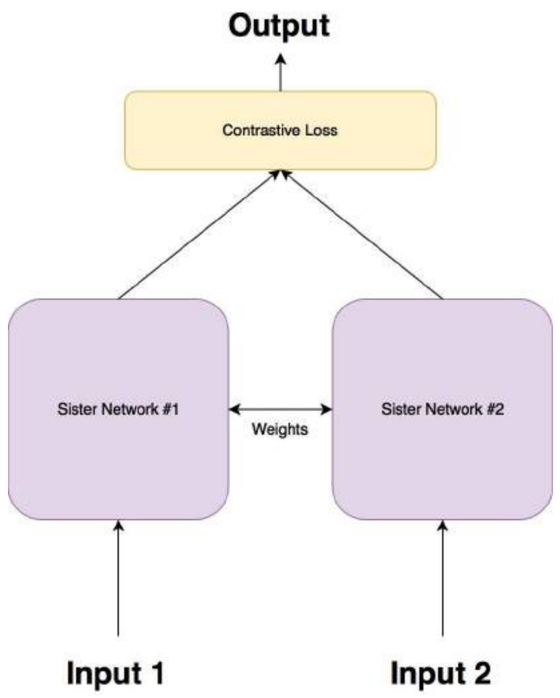


Figure 2.5: Architecture of a typical Siamese network. Figure taken from `https://hackernoon.com/one-shot-learning-with-siamese-networks-in-pytorch-8ddaab10340e`

A Siamese network offers a solution to these dilemmas. Instead of learning to classify specific objects, it will learn to discriminate two objects. If we take a look at the person-recognition domain, the Siamese network will take two images as input, and will compute a similarity score.

Figure 2.5 shows the typical architecture of a Siamese network. Such a network consists of two identical neural networks. In the case of this thesis, these identical neural networks compute a gait embedding for a given video. These embeddings are then fed to a loss function, which has the goal of calculating the similarity between the two images.

2.2.6 Transfer learning

Transfer learning is described in [33] as *the improvement of learning in a new task through the transfer of knowledge from a related task that has already been learned.*

In terms of convolutional neural networks, transfer learning is used by nearly all applications, mainly because it speeds up learning immensely. There exist pre-trained networks, which can classify a vast number of objects [5]. In order to be able to classify so many objects, the lower layers of the network learn low- and mid-level features, such as edges and corners. The main rationale why transfer learning works, is that many completely different high-level objects rely on the same (or similar) low- and mid-level features. Thus, if the initial network is set to pre-trained weights, there is no need to learn these low- and mid-level features from scratch, the network simply adapts them to its needs.

In this thesis we want to extract the silhouette of a person from an image. There are already quite advanced neural networks for detecting objects, such as cars, people and so on. Figure 2.6 shows a state-of-the-art neural network for instance segmentation. By using these neural networks as starting points for this thesis, a lot of computation time is saved because many low- and mid-level features can be reused from this network.

2.3 Datasets

Training data has the greatest influence on the quality of the resulting network. Thus, in pursuance of creating the most robust network, this section discusses the available options

[5] For example, such a network can be trained on the COCO dataset, c.f. Section 2.3.1.

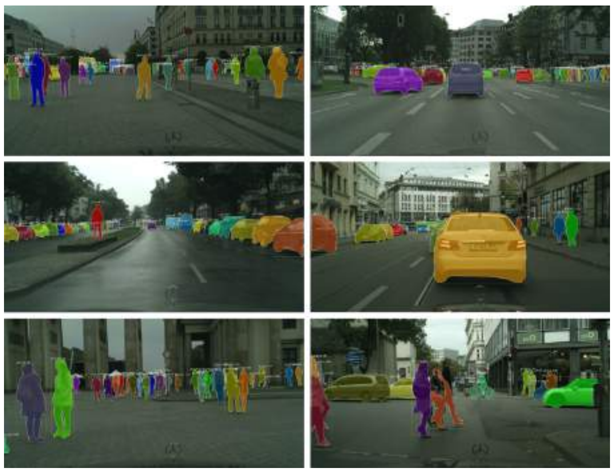


Figure 2.6: Example of state-of-the-art instance segmentation, Figure taken from [4]

	Number of images	Quality	Variety
COCO	200.000	~	✓
Supervisely Person	5.000	✓	✓
CASIA	~1.1 million	~	✓

Table 2.1: Overview of different datasets

for different training data. We will introduce three different datasets, which are all used in this thesis. They have been selected mainly because of their quality, size and variability [6]. The differences between them are highlighted and the advantages of each one is shown in this chapter. Table 2.1 gives an overview of the different datasets.

2.3.1 COCO

This dataset has (as of February 2020) over 200.000 annotated images. They feature more than 250.000 people, which is excellent for the purpose of extracting the silhouette. An example of this dataset is shown in Figure 2.7.

[6]For example, having different camera angles, different load carriage, different people, different background and so on.

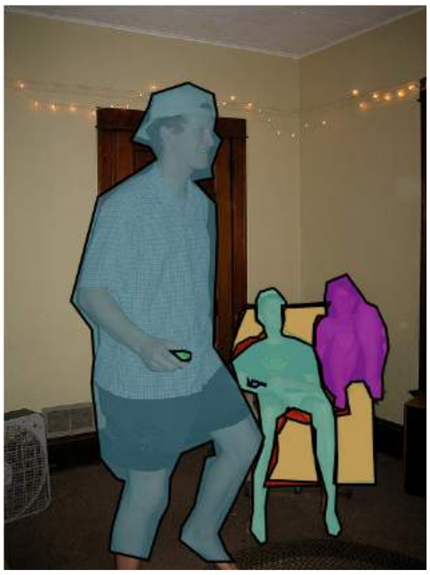


Figure 2.7: Example of annotated image in COCO dataset, Figure taken from (`http : //cocodataset.org/#explore?id=353270`)

While this dataset with 200.000 images and many object categories is quite unbeatable in terms of quantity, the quality could certainly be improved. Figure 2.7 shows clearly that there is still room for improvement. In this instance, particularly the legs are not annotated perfectly.

2.3.2 Supervisely Person

Since the COCO dataset does not meet our expectation in terms of quality, we want another, possibly much smaller dataset which has high quality annotations. The Supervisely Person dataset fulfills this property, although it contains just about 5.000 images of people. In exchange, the quality is really near-perfect, as shown in Figure 2.8.

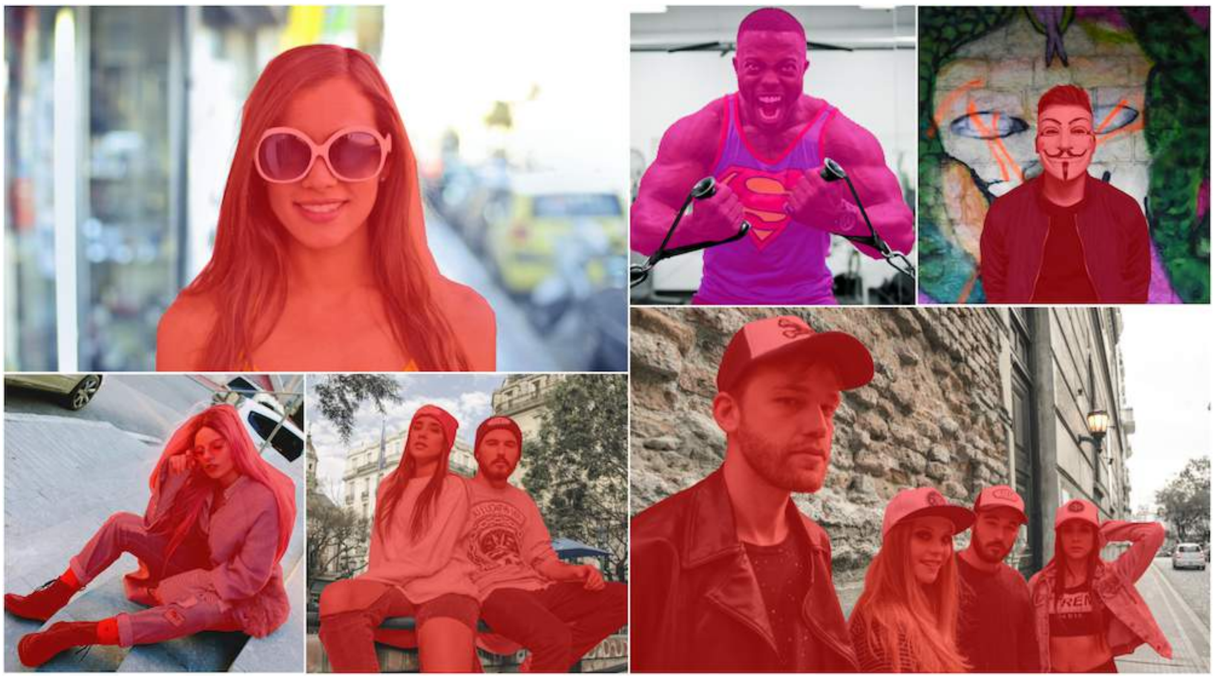


Figure 2.8: Example of annotated image in Supervisely Person dataset (`https : / / supervise.ly/`), Figure taken from hackernoon.com

2.3.3 CASIA gait database

This dataset is a large database[7] which comes with mainly 3 variations:

1. Viewing angle: 11 different views, from 0 degrees to 90 degrees up to 180 degrees
2. Clothing
3. Carrying condition: 3 different variations: normal, in a coat and with a bag.

Figure 2.9 shows an example from the CASIA database.

[7]124 subjects

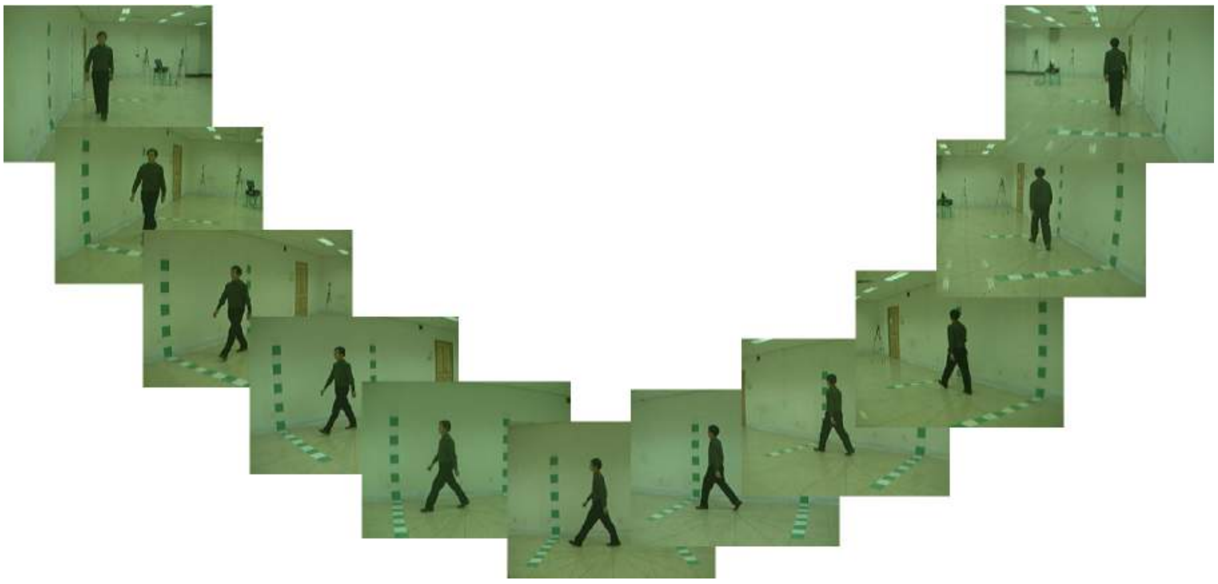


Figure 2.9: Example of CASIA gait database, Figure taken from `http://www.cbsr.ia.ac.cn/english/Gait%20Databases.asp`

Chapter 3

Silhouette extraction

The method this thesis proposes receives a gait video of a person as input[1]. In order to be more robust to clothing, the gait recognition method this thesis proposes will not work with the images itself, but rather with silhouettes. Thus, we will propose a current state-of-the-art neural network for extracting the silhouette.

Section 3.1 will discuss which data we use and how the training data is retrieved and split into training and validation set. Next, in order to feed the images to the neural network, the images are loaded together with information about where people are visible in the image. Finally, in Section 3.3 the Mask R-CNN [4] neuronal network is created and trained. At long last, Section 3.4 shows the performance of the network.

[1]c.f. Research approach in Section 1.2

3.1 Data

This section prepares the data to be used in a neural network. First, we will discuss the rationale behind selecting the datasets. Next, the images from different datasets are retrieved and afterwards split into training and validation set.

First things first, in order to create a network with the highest possible quality, we need both a lot of images and images with high quality. Unfortunately, there is no dataset available which fulfills our expectations. Thus, we combine two different datasets - COCO and Supervisely Person (c.f. Sections 2.3.1 and 2.3.2, respectively). Although the data of both datasets are similar, they fulfill different purposes: The COCO dataset has an immense volume, thus it is used as starting point for the silhouette extraction. Furthermore, the COCO dataset is quite popular, so fortunately there exists a pre-trained model which was trained on this dataset. This reduces the training time of the whole network substantially. In contrast to this, the second dataset is used to increase the accuracy. This is possible, because the Supervisely Person dataset is of superb quality, as described in 2.3.2.

Since two datasets are used, they have to be imported:

COCO dataset As hinted in Section 3.1 already, there is a pre-trained model available. Therefore, in order to exploit the immense volume of the COCO dataset, we download the pre-trained weights from [34]. Since these weights have been trained with the COCO dataset, we implicitly use this dataset.

Supervisely dataset The ∼5.000 images can be downloaded from `https://supervise.ly`.

In order to know when to stop training, the computation of a validation loss is needed. Hence, the Supervisely dataset is split into training and validation images. This thesis uses 80% of the images as training data and 20% as validation data.

Listing 3.1 shows the Python code for splitting the data to respective subfolders. Line 5 iterates over every image in the Supervisely Person dataset. For all images (line 6) a random number between 0 and 100 is generated (line 7). Since we want to have 20% validation data, we move the file to the validation directory if the random number is less than 20. Otherwise, the image will serve in the training data set.

```
1  import os
2  import random
3
4  img_path = "/path/to/dataset/folder"
5  for filename in os.listdir(img_path):
6      if filename.endswith(".jpeg") or filename.endswith(".png"):
7          if random.randint(0,100) < 20:
8              folder="val"
9          else:
10             folder="train"
11         os.rename(img_path+"/"+filename, img_path+"/"+folder+"/"+filename)
```

Listing 3.1: Code for splitting images in training and validation set

3.2 Loader

The images are stored locally in a training and a validation folder. In order to use images for training the network, they need to be loaded in Python. The code for doing this is shown in Listing 3.2. First, line 3 specifies the name of the type of object which should be classified, *silhouette* in this case. The *for*-loop in line 9 iterates over all files in either the training or the validation folder. One of the most important operation to do at this point is parsing the ground-truth silhouettes for each image. This is done by first loading the correct annotation file (lines 14 and 15), and then extracting the x and y coordinates of the silhouette for the image. A *polygons* list is created where the silhouette is stored. At the end of the function (line 30+) the image is added to the set of training or validation images. This process enables the usage of the images in Python, i.e. we are able to feed the images to our neural network, as discussed in the next section.

```
 1 def load_silhouette(self, dataset_dir, subset):
 2     self.add_class("silhouette", 1, "silhouette")
 3 
 4     assert subset in ["train", "val"]
 5     dataset_dir = os.path.join(dataset_dir, subset)
 6 
 7     directory = os.fsencode(dataset_dir)
 8 
 9     for file in os.listdir(directory):
10         filename = os.fsdecode(file)
11         if filename.endswith(".png") or filename.endswith(".jpeg"):
12             image_path = os.path.join(dataset_dir, filename)
13 
14             with open(image_path + '.json') as f:
15                 data = json.load(f)
16 
17             polygons = []
18             for object in data["objects"]:
19                 points = object["points"]["exterior"]
20                 x = [a[0] for a in points]
21                 y = [a[1] for a in points]
22                 obj = dict()
23                 obj["name"] = "polygon"
24                 obj["all_points_x"] = x
25                 obj["all_points_y"] = y
26                 polygons.append(obj)
27             image = skimage.io.imread(image_path)
28             height, width = image.shape[:2]
```

```python

            self.add_image(
                "silhouette",
                image_id=filename,  # use file name as a unique image id
                path=image_path,
                width=width, height=height,
                polygons=polygons)
```

Listing 3.2: Code for loading the images with their annotations in Python

3.3 Network architecture and training

After loading the images, the next step is to prepare the neural network and start training it. The input to the network are images of people. The network should learn to recognize the silhouette of a person. This problem is an example of *Image segmentation*[2]. Since this is a well-researched topic, there is no need to create a neural network from scratch. The current state-of-the-art image segmentation network is called Mask R-CNN. The architecture of this network is used, as described in Section 2.2.2.

This chapter will discuss the configuration of Mask R-CNN to be able to be used to recognize silhouettes. Next, Section 3.3.2 explains how to use transfer learning. Last but not least, the images are fed into the Mask R-CNN network and the network is trained.

Fortunately, Matterport [34] has already implemented the algorithms proposed by [4]. Therefore, this code is adapted to our situation of recognizing silhouettes - no need to write everything from scratch. More specifically, the following changes need to be made:

[2]c.f. Section 2.2.2

- Create a SilhouetteConfig class.
- Load the COCO dataset at the very beginning to reap the benefits of transfer learning.
- Get the images which have been preprocessed as described in Section 3.1 and 3.2.

3.3.1 SilhouetteConfig class

Mask R-CNN requires certain configuration parameters. These are stored in a Config-class, which is called *SilhouetteConfig* in the case of this thesis. As shown in 3.3 it defines the name of the class (line 2), how many images are processed at the same time (line 3) and how many classes there are (line 4). Furthermore, through empirical testing the remaining two variables (*STEPS_PER_EPOCH* and *DETECTION_MIN_CONFIDENCE*) are set to 70 and 0.9, respectively.

```
1 class SilhouetteConfig(Config):
2     NAME = "silhouette"
3     IMAGES_PER_GPU = 1
4     NUM_CLASSES = 1 + 1  # background + silhouette
5     STEPS_PER_EPOCH = 70
6     DETECTION_MIN_CONFIDENCE = 0.9 # Skip detections with < 90% confidence
```

Listing 3.3: Code for SilhouetteConfig

3.3.2 Transfer learning

Performing image segmentation is a well-researched topic. A lot of work has been done by combining popular datasets with popular neural networks. Fortunately, Matterport [34] trained the Mask R-CNN network with the COCO dataset and publicly released the resulting weights. In order to exploit transfer learning as described in Section 2.2.6, our weights are set to the pre-trained COCO weights: *model.load_weights(weights_path)*.

3.3.3 Prepare images and start training

Next, the images have to be fed into the neural network. Therefore, as shown in Listing 3.4, both the training and validation images are loaded. Finally, by calling the *model.train(...)* function, the training is started.

As shown in Listing 3.4 on line 17, the *model.train(...)*-function takes a *layers* argument. This specifies which layers of the Mask R-CNN network are trained. The weights for all the rest of the layers are not changed while training the network. These weights are left at the values which have been loaded in Section 3.3.2. There are five possible values for choosing which layers to train:

1. heads
2. 5+
3. 4+

4. 3+

5. all

In ascending order, more and more layers of the network are trained. *Heads* only trains the final layer, while *all* trains the whole network. Doing the latter will take considerably more time. Through empirical testing the value *5+* was chosen for this thesis. This seems like a reasonable trade-off between performance benefits from transfer learning and being more flexible by using the Supervisely dataset. The result of the training is described in the next Section.

```python
def train(model):
    """Train the model."""
    # Training dataset.
    dataset_train = SilhouetteDataset()
    dataset_train.load_silhouette(args.dataset, "train")
    dataset_train.prepare()

    # Validation dataset
    dataset_val = SilhouetteDataset()
    dataset_val.load_silhouette(args.dataset, "val")
    dataset_val.prepare()

    print("Training␣network␣heads")
    model.train(dataset_train, dataset_val,
                learning_rate=config.LEARNING_RATE,
                epochs=100,
                layers='5+')
```

Listing 3.4: Code for preparing images

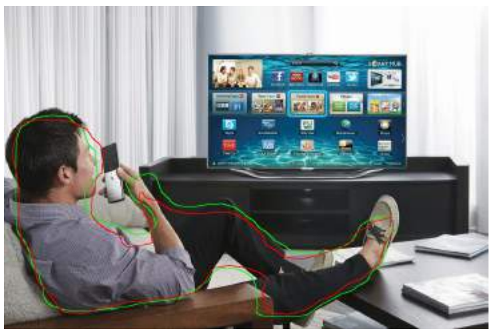
3.4 Result visualization

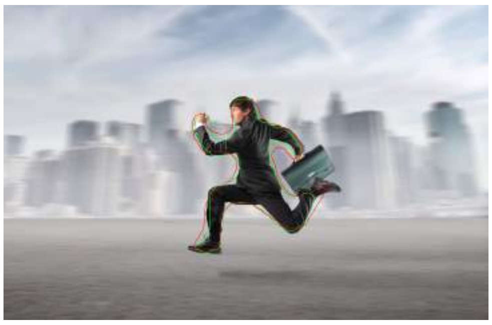
After training the network, in this chapter we want to verify how good the network performs on new, unseen data. In order to visually see the output of the network, a silhouette mask is generated. This can be done for both the pre-trained COCO dataset model (red line) and the newly trained network (green line). Hence, the performance improvement of our network can be seen versus the pre-trained model.

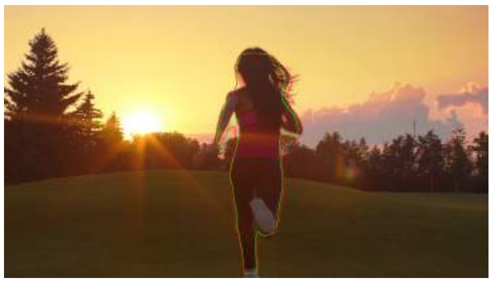
Figure 3.1 shows some real-world examples obtained through manual web-searches. The most interesting parts are described in the following paragraph. In (a) the performance improvement can be shown by looking at the left arm of the person. The newly trained model clearly makes a better job for predicting the arm. In (c) the result from the network which combined both datasets is way closer to the actual body than the pre-trained model. Hence, the silhouette is more accurate. A similar effect can be seen in (d), the area under the left armpit is way more accurate. In (e) and (f) yet another benefit is visible: The newly trained model is better at predicting the arms and legs. It is especially obvious in (e) and (f) because the area between the legs is (correctly) not assigned to the silhouette.

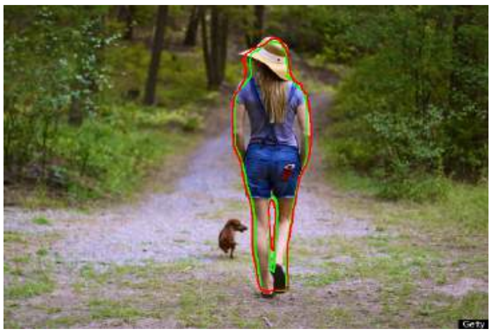
The scope of this thesis is to recognize a person based on gait videos featuring this person only. As discussed in Chapter 9, future work could focus on allowing multiple people in the gait video. Figure 3.2 shows that the network this thesis proposes only detects a single person in the whole picture.

(a)


(b)


(c)


(d)


(e)

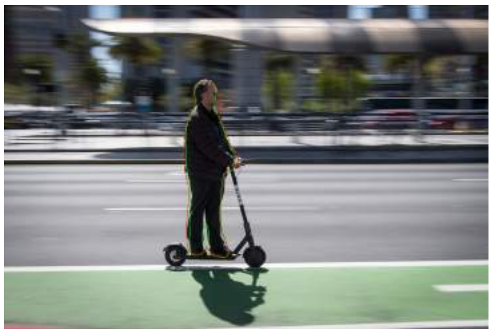

(f)

Figure 3.1: Output of trained network

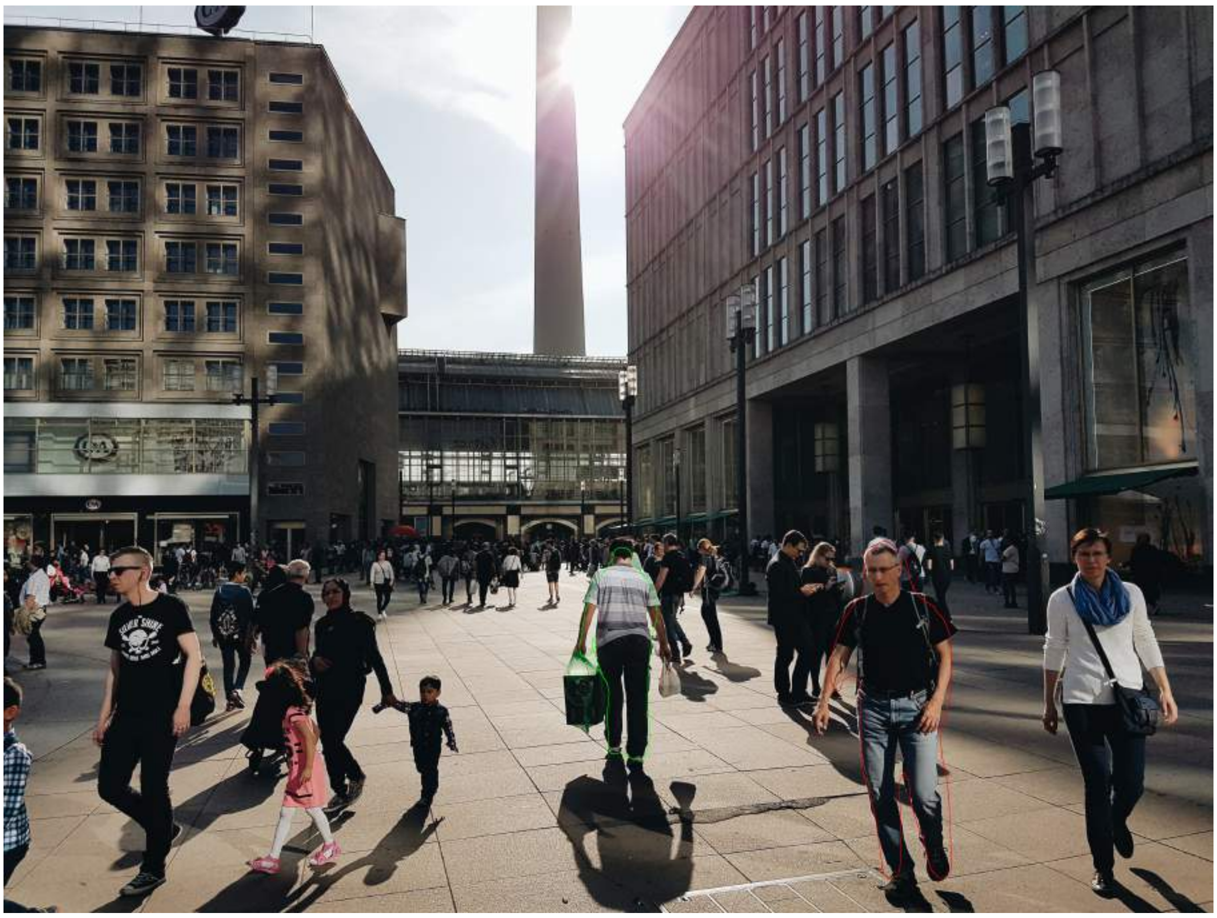


Figure 3.2: Example of bad silhouette extraction due to multiple people.

Chapter 4

Spatial information extraction using a CNN for Gait Embedding

Chapter 3 proposed a way of extracting silhouettes from images. These silhouettes will be used to extract a gait embedding, which in turn should be able to characterize a walking person. If there are multiple videos of the same person, the embeddings should be similar, even if the setting is completely different. For instance, the camera might be at a different angle, the person has different clothing, the underground might be different and the person might even be a few years older. In contrast to this, the gait embedding of a different person - even though he is in a very similar setting - should be as different as possible.

As hinted in Section 1.2, the gait embedding is extracted in two steps:

1. Extract an embedding from a single image, for the sake of simplicity, let us call this type of embedding *cnn-gait-embedding*.

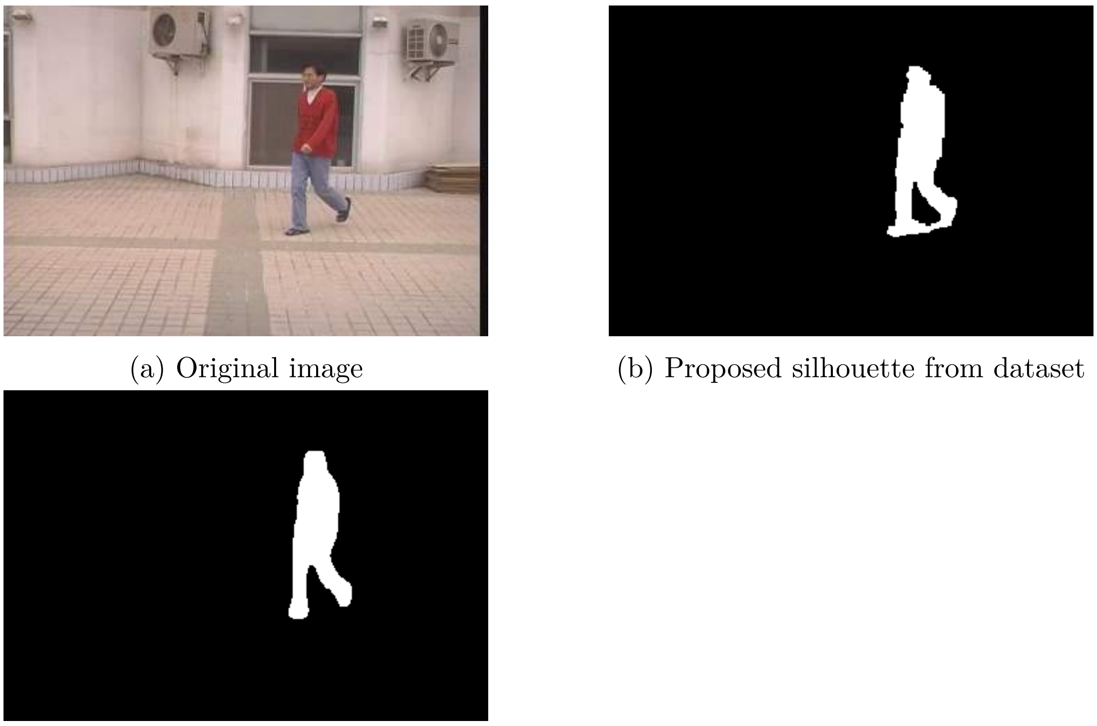
2. Use multiple cnn-gait-embeddings and calculate another embedding on top of it, thus including also the temporal information. We will call that embedding *lstm-gait-embedding*.

This chapter will focus on the first step, thus extracting the spatial information from the silhouette. There are multiple classical ways of doing this, the most common one is to create a GEI [1]. However, this thesis tries to outperform current state-of-the-art methods, thus a convolutional neural network will be used to extract spatial information. The first section will reason about the kind of data that is used for training the network and how the data is prepared. Next, Section 4.2 will introduce the network architecture. Afterwards, the network is trained and Section 4.4 discusses its performance. Finally, the cnn-gait-embedding is extracted which will be used in the next chapters.

4.1 Data Preparation

In order to train the CNN, it needs a lot of silhouettes of different people. One of the largest collections is the CASIA dataset (`http://www.cbsr.ia.ac.cn/english/Gait%20Databases.asp`) which features 124 people, from 11 different camera angles, with different settings - carrying a backpack, different backgrounds and so on. In total there are over a million images available. The dataset also provides the silhouettes of the people, unfortunately the quality leaves something to be desired. Figure 4.1 shows in a) the original image, in b) the proposed silhouette from the CASIA dataset itself and c) our proposed silhouette using the approach from Chapter 3.

Since our silhouette extraction seems to work better than the silhouette which is proposed


(a) Original image

(b) Proposed silhouette from dataset

(c) Proposed silhouette of our method, as described in Chapter 3

Figure 4.1: Different silhouette suggestions from different networks.

by the dataset itself, the latter will not be used. Instead, we feed our network from Chapter 3 every single image from the dataset, and save the silhouette. Thus, we can use this silhouette as training data for our CNN, which should learn the cnn-gait-embedding based on the spatial information present in the silhouette.

4.2 CNN Network Architecture

CNNs are the most common method of performing image classification, thus a lot of different methods have been proposed. One of them, a heavily engineered one, is called Inception network [35]. There are different versions of the Inception network, in this thesis

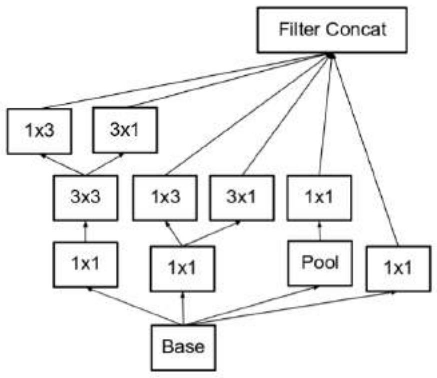


Figure 4.2: Architecture of the inception network v2, Figure taken from `https://towardsdatascience.com/a-simple-guide-to-the-versions-of-the-inception-network-7fc52b863202`

we use version 2. Before Szegedy et. al. [35] proposed the inception network in 2017, a common way of achieving a higher accuracy was to create a deeper network. This approach obviously meant much longer training time. The main idea of the inception network [35] is to combine different convolution layers cleverly, as displayed in Figure 4.2.

Since the inception network [35] is quite popular in the machine learning domain, [36] created and published the model. Thus, it is not necessary to re-create the model from scratch. Instead, the model is downloaded from `https://github.com/tensorflow/hub/tree/master/tensorflow_hub/tools/make_image_classifier`. The convolutional neural network receives a silhouette as input, which was extracted in Chapter 3. The network then to identify a person based on this silhouette. Since the dataset consists of 124 people, the output of the network is a 124 dimensional vector, where each element specifies the likelihood that the received silhouette belongs to the respective person.

4.3 Training the network

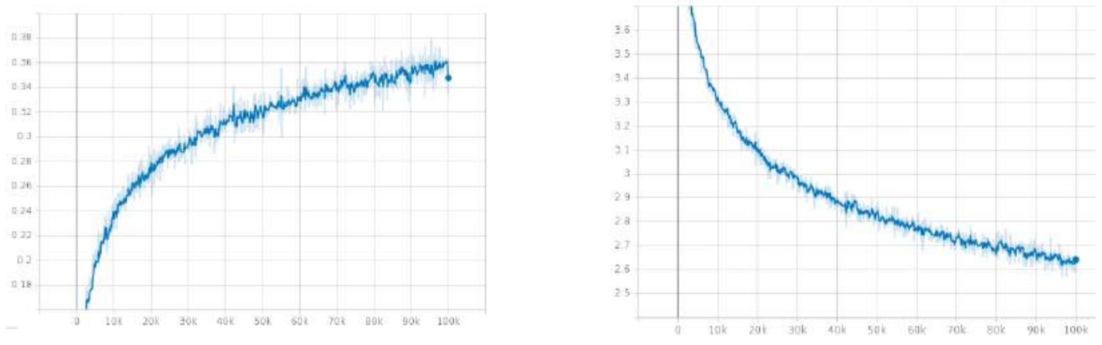
The training data - in our case silhouettes of different people - has been prepared in Section 4.1, the network architecture has been created in Section 4.2. Thus, everything is in place to start training the network. This is done with issuing the following command:

```
/path/to/dowloaded/model/image_retraining/retrain \
    --bottleneck_dir=/tf_files/bottlenecks \
    --model_dir=/tf_files/inception \
    --output_graph=/tf_files/retrained_graph.pb \
    --output_labels=/tf_files/retrained_labels.txt \
    --image_dir /tf_files/silhouettes
```

The model has been trained on a single computer with an NVIDIA GeForce MX130 graphics card. After tuning the hyper-parameters, the final training took about 6 hours. The results are explained in the next section.

4.4 Performance

The accuracy and the loss on the validation set is plotted in Figure 4.3. In the end, the CNN achieves an astonishing 35% accuracy on the validation set. This is remarkable, because the CNN looks at just a single image and classifies the image to one of the 124 people in the dataset.



(a) Accuracy (b) Cross-entropy loss

Figure 4.3: Accuracy and loss of the validation set on the CNN

4.5 Gait embedding extraction using a CNN

The CNN, which is described in this chapter, is able to extract spatial information from a silhouette. At the end, the neural network classifies the silhouette into the 124 people. The next step is to increase the 35%-accuracy observed in Section 4.4 by using an additional neural network on top of this CNN approach, which will be explained in Chapter 5. Since the spatial information has already been extracted by the convolutional neural network in this chapter, we do not want the additional network to learn the spatial features again, but rather supply the additional network with the spatial information. Thus, the final classification of the CNN, which is described in this chapter, is removed, Hence, the output of the CNN is a 2048 dimensional vector, which captures the spatial information of a silhouette. This vector is called *cnn-gait-embedding*.

In order to be able to use this 2048 dimensional vector, the cnn-gait-embedding needs to be extracted for all input images. Even though the extraction for a single image takes just a few seconds, since there are over 1 million images, the extraction will need a considerable amount of time.

In order to decrease the time necessary, Listing 4.1 shows how to make use of all CPU cores. If a machine has 8 cores, this reduces the time to $\frac{1}{8}$ of the original duration. The code basically uses the multi-threading capability of Python. In line 2 a Pool is created, the number of processes is, by default, set to the number of available CPU processes. Next, line 3 iterates over our dataset and extracts the cnn-gait-embedding for every image, by calling the `extractor.extract (...)` function in line 15 for every image. This function is shown in Listing 4.2.

Normally, the CNN which is described in this chapter would return the class to which the image belongs. Since we want to train a LSTM [5] in the next chapter on top of this, we are interested in the gait of a person. Thus, we remove the very last layer of our CNN (where the classification happens) by setting our output layer to the layer before the last, which is done by calling module_apply_default/hub_output/feature_vector/SpatialSqueeze in our case. The resulting embedding is then saved and will be used in the next chapter.

```
 1 def extract_features():
 2     pool = Pool()
 3     for person_id in os.listdir(input_path):
 4         if not os.path.exists(output_path + "/" + person_id):
 5             os.makedirs(output_path + "/" + person_id)
 6         imgs = os.listdir(input_path + "/" + person_id)
 7         pool.starmap(handle_image, zip(imgs, repeat(person_id)))
 8 
 9 def handle_image(img, person_id):
10     path = output_path + "/" + person_id + "/" + img[:-4] + ".npy"
11     if os.path.isfile(path):
12         print("Skipping␣"+img)
13     else:
14         print("Processing␣"+img)
15         embedding = extractor.extract(input_path+"/"+person_id+"/"+img)
16         save(path, embedding)
```

Listing 4.1: Multithreaded extraction of gait embedding

As is shown in Listing 4.2, the constructor mainly deals with setting the correct configuration for the network, such as specifying the model and label file in lines 4 and 5, respectively. Furthermore, the input and output layer are set to the values as described in the last paragraph. The extract (...) function loads the input image, passes it to the CNN and returns the output as a numpy-array.

```
 1  class Extractor:
 2    def __init__(self) -> None:
 3        super().__init__()
 4        model_file = "/path/to/model/file.pb"
 5        self.label_file = "/path/to/label/file.txt"
 6        input_layer = "Placeholder"
 7        output_layer = 'module_apply_default/hub_output/feature_vector/SpatialSqueeze'
 8
 9        self.graph = self.load_graph(model_file)
10
11        input_name = "import/" + input_layer
12        output_name = "import/" + output_layer
13
14        self.input_operation = self.graph.get_operation_by_name(input_name)
15        self.output_operation = self.graph.get_operation_by_name(output_name)
16
17    def extract(self, file_name):
18        t = self.read_tensor_from_image_file(
19            file_name,
20            input_height=self.input_height,
21            input_width=self.input_width,
22            input_mean=self.input_mean,
23            input_std=self.input_std)
24
25        with tf.compat.v1.Session(graph=self.graph) as sess:
26            results = sess.run(self.output_operation.outputs[0], {
27                self.input_operation.outputs[0]: t
28            })
29        return np.squeeze(results)
```

Listing 4.2: Extractor function for CNN

In summary, the first three steps as shown in Figure 1.1 are completed. The silhouette was extracted from an image in Chapter 3. In the current chapter the cnn-gait-embedding has been calculated. Since this embedding is based on a single image only, the next step is to add temporal information by using a recurrent neural network. This process is shown in the next chapter.

Chapter 5

Temporal information extraction using LSTM for Gait Embedding

In Chapter 4 the spatial information, such as hip angle, ratio between body and feet and so on, was extracted. These traits have not been specified manually, but rather learned implicitly by a CNN. The main idea of gait recognition is to use even more features, such as step and stride length, cadence and speed. It is not possible to extract this kind of features from a single image, thus it cannot be done with a CNN. Fortunately, we not only have one image but a whole video. This enables us to learn temporal features with a different neural network type.

The focus of this thesis is to increase the accuracy of gait recognition by using neural networks. A sequence of images has to be processed, thus a recurrent neural network[1] is used. More specifically, since we want to remember long-term dependencies, we use an LSTM [5], as explained in Section 2.2.4.

[1]c.f. Section 2.2.3

This chapter will show what data is used for training, how the architecture of the LSTM [5] cell looks like, the training process and its performance. The output is an enhanced gait embedding, which takes temporal features into account. This embedding will be called lstm-gait-embedding.

5.1 Data Preparation

In Section 4.5 the CNN extracted a gait embedding from a single image and saved the result in a file for every image. The LSTM [5] uses this compact representation[2] of an image. This has the benefit of shorter training time, because since the spatial information is already encoded in the cnn-gait-embedding, there is no need to learn it again. The LSTM [5] can focus on learning the temporal information.

In order to get a video as input, the cnn-gait-embeddings of n multiple images are stitched together. Through empirical testing we fixed the value for n as 15. In a 20-frames-per-second video this will roughly be a single step. The code for doing this is shown in Listing 5.1. Lines 2 and 9 iterate over all people and setting for each person, respectively. Next, for every setting and person, we look at every cnn-gait-embedding of this specific setting (line 15). Then, 15 successive embeddings are stacked together. In our dataset, the first image always represents the start of a step. Since we do not want to be dependent on any phase of the step (i.e. it should be possible to identify a person if in the reference video he or she is in the middle of a step), additional sequences are created. The code uses every third image (line 21) as starting image of a new sequence, which is 15 images long again. This process is shown in Figure 5.1.

[2]a 2048-d vector

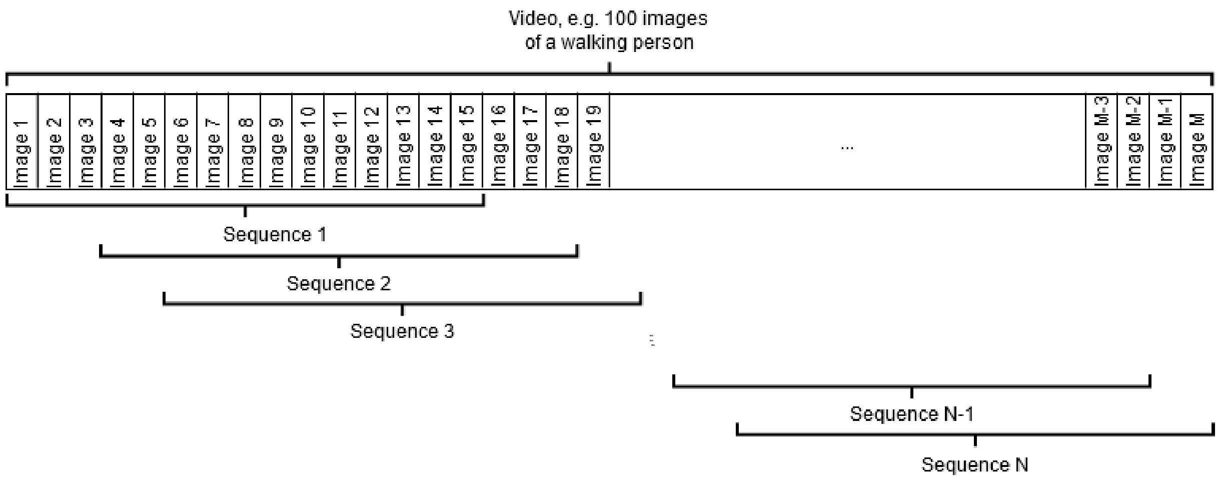


Figure 5.1: Sequence extraction from cnn-gait-embedding

```python
def save_sequences_from_imgs():
    for person_id in os.listdir(output_path):
        if not os.path.exists(sequence_dir + "/" + person_id):
            os.makedirs(sequence_dir + "/" + person_id)
        __create_sequence_for_person(person_id)

def __create_sequence_for_person(person_id):
    settings = __split_files_to_settings(output_path + "/" + person_id)
    for setting in settings.values():
        __create_sequence_for_setting(person_id, setting)

def __create_sequence_for_setting(person_id, files):
    sequences = []
    count = 0
    for file in files:
        sequences = __remove_seq_if_large_enough(sequences, person_id)
        x = load((output_path + "/" + person_id + "/" + file))
        for index, sequence in enumerate(sequences):
            sequences[index] = vstack((sequence, x))
        count = count + 1
        if count % 3 == 0:
            sequences.append([x])

def __remove_seq_if_large_enough(sequences, person_id):
    amount_stitch_together = 15
    remove = []
    for index, sequence in enumerate(sequences):
        if len(sequence) >= amount_stitch_together:
            remove.append(index)

    for r in remove:
        save(sequence_dir + "/" + person_id + "/" + str(__getNextNumber(sequence_dir + "/" + person_id)) + ".npy", sequences[r])
        del sequences[r]

    return sequences
```

Listing 5.1: Creating the input for the LSTM by stitching together 15 frames.

These video are retrieved by using a generator which returns a random video sequence. The code is shown in Listing 5.2. Lines 3 and 4 retrieve the relevant data and choose either training or validation set. Then, a random sample is chosen in line 11, and its cnn-gait-embedding is extracted (line 12). This procedure is repeated until there are *batch_size*-elements produced. The function get_class_one_hot(...) encodes a given value in the one-hot encoding. For the sake of simplicity, let us assume that there are only 5 possible classifications. The one-hot encoding of value 4, and thus the result of get_class_one_hot(4), would look like this:

$$\begin{pmatrix} 0 \\ 0 \\ 0 \\ 1 \\ 0 \end{pmatrix}$$

```python
1 @threadsafe_generator
2 def frame_generator(self, batch_size, train_test):
3     train, test = self.split_train_test()
```

```
    data = train if train_test == 'train' else test

    print("Creating %s generator with %d samples." % (train_test, len(data)))

    while 1:
        X, y = [], []
        for _ in range(batch_size):
            sample = random.choice(data)
            sequence = self.get_extracted_sequence(sample)
            if sequence is None:
                raise ValueError("Can't find sequence. Did you generate them?")
            X.append(sequence)
            y.append(self.get_class_one_hot(sample[1]))

        yield np.array(X), np.array(y)
```

Listing 5.2: Generator for retrieving cnn-gait-embeddings

5.2 LSTM Network architecture

Since we have already extracted a lot of (spatial) information in Chapter 4, only temporal information has to be learned in this chapter. Thus, through empirical testing we decided to go with a simple structure, which is depicted in Figure 5.2.

As regularization technique, this network uses two 50% dropouts. The first one is in the LSTM cell, the second one is located between the two Dense layers. The first Dense layer uses a sigmoid activation function and should represent the new lstm-gait-embedding. The second Dense layer uses this gait embedding to classify the input video into one out of 124 identities.

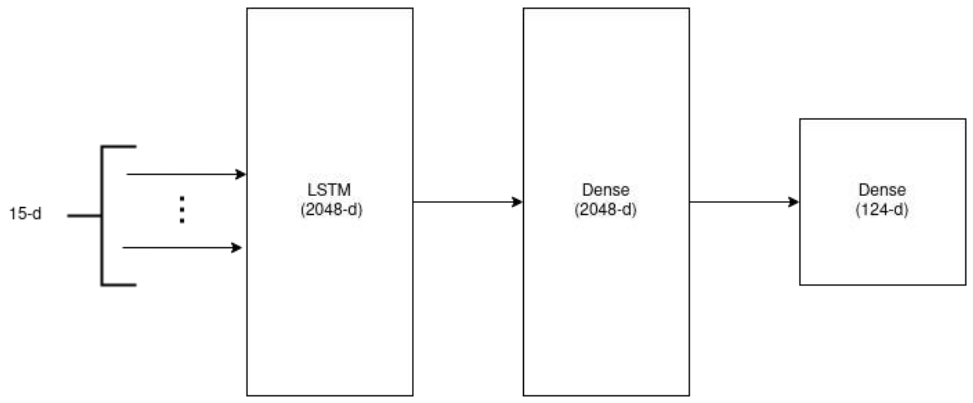


Figure 5.2: Architecture of the LSTM

5.3 Training and Experimental Results

After determining the training data (Section 5.1) and defining the network architecture (Section 5.2), the network can be trained. The code for doing so is shown in Listing 5.3. The training took about 2 days and 9 hours on the same hardware as described in Section 4.3, and the network achieves an accuracy of 86% on the test dataset.

```
rm.model.fit_generator(
    generator=generator,
    steps_per_epoch=steps_per_epoch,
    epochs=nb_epoch,
    verbose=1,
    callbacks=[tb, csv_logger, checkpointer, early_stopper], # early_stopper
    validation_data=val_generator,
    validation_steps=40,
    workers=4)
```

Listing 5.3: Code for training the LSTM

After extracting the spatial information in Chapter 4, this chapter focused on extracting temporal information. In order to indirectly use the spatial information, the cnn-gait-embedding is used as input for an LSTM. This increases our accuracy by 245% - the CNN-only approach achieved 35%, while the LSTM achieved 86% accuracy. These findings are

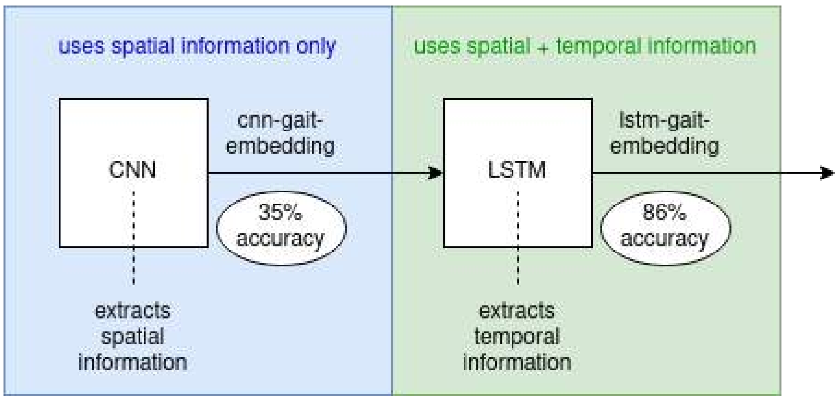

uses spatial information only
uses spatial + temporal information
CNN
cnn-gait-embedding
35% accuracy
extracts spatial information
LSTM
lstm-gait-embedding
86% accuracy
extracts temporal information

Figure 5.3: Results of CNN and LSTM networks, including their accuracy and scope.

highlighted in Figure 5.3.

Chapter 6

Gait-Recognition using Siamese networks

After Chapter 5 the network is able to distinguish 124 people from the training dataset, which is shown in Figure 6.1. The main disadvantage of this procedure is that every person which the network can identify will have to be known beforehand. This implies, that a lot of training images for every recognizable person is needed. This has two main disadvantages:

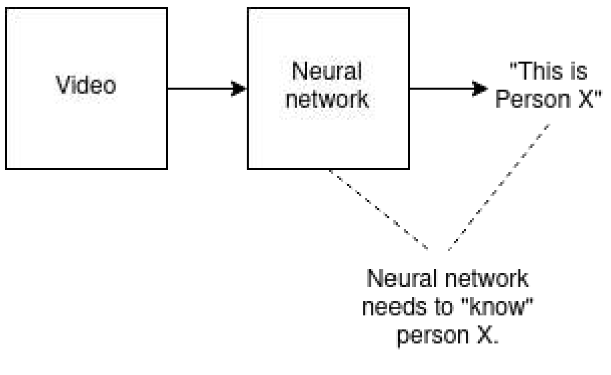


Figure 6.1: Operating principle of the network (so far)

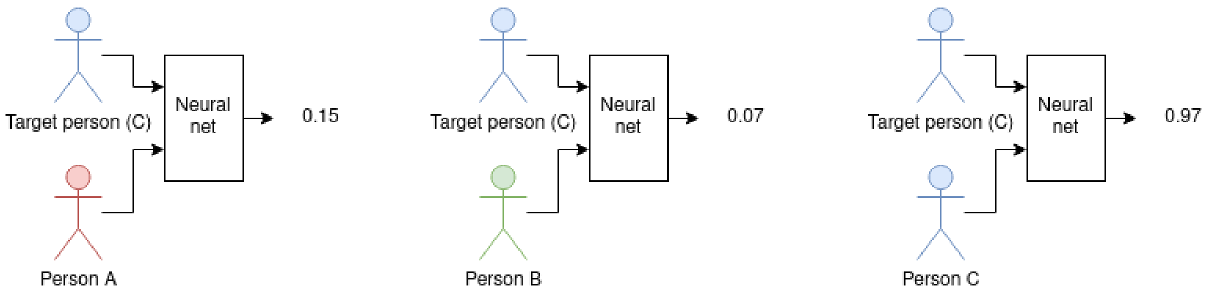


Figure 6.2: Process of using a Siamese network

1. It is very unlikely that the real-world provides the needed quantity of such training samples. For example, if someone robs the bank, there is just a few minutes worth of video, not millions of pictures.

2. Even if enough data is available, the network would have to be re-trained. Because this takes both a lot of processing power and time it is very impractical.

As discussed in Section 2.2.5 already, practically it would be far more useful to train the network just once. This creates a new challenge: It is not possible to train a network which can identify people which the network has never seen before. For example, the network can not output *This is person X*, if the neural network has never seen X before. Thus, a different objective is needed.

Instead of identifying people, the network learns to discriminate between people. It will learn the difference between humans. If two people are fed to the Siamese network, it will calculate the similarity between them.

Let us suppose such a network exists. As Figure 6.2 demonstrates, the query image (i.e. target person) could be fed together with all other people to the neural network and a similarity score is calculated. Then, one can simply select the person with the highest similarity as the detected identity. Furthermore, an application might also decide on a threshold value (e.g. 90%). If the observed similarity is below this threshold, the network will not consider the respective person as possible candidate. This is useful if the target person is not in the database. The main benefit of this approach is that you only need a single shot[1] of a person, without retraining the network.

6.1 Siamese Network Architecture

The architecture is shown in Figure 6.3. The neural network takes the same input as the LSTM network - gait embeddings from the CNN. The first part of the Siamese network consists of a slightly modified version of the LSTM which was introduced in Chapter 5. The only difference is that the last Dense layer is chopped off, thus the LSTM used in this chapter will not do any classification but rather learn an embedding. This new LSTM takes a series of cnn-gait-embeddings and calculates a lstm-gait-embedding. This new embedding is calculated for two different gait videos and fed into the Siamese network. The Siamese network then tries to discriminate between people based on this *lstm-gait-embedding* and create a similarity score.

In Listing 6.1 you can see the model. First, the two inputs are defined. Next, the RNN with a single LSTM cell is configured. Afterwards, the rnn-gait-embeddings for the two input videos are calculated and stored in encoded_l and encoded_r, respectively. Then the loss function is described according to Section 2.2.5.

```
def get_siamese_model(input_shape):
    left_input = Input(input_shape)
    right_input = Input(input_shape)

```

[1]c.f. *One-shot learning in Section 2.2.5*

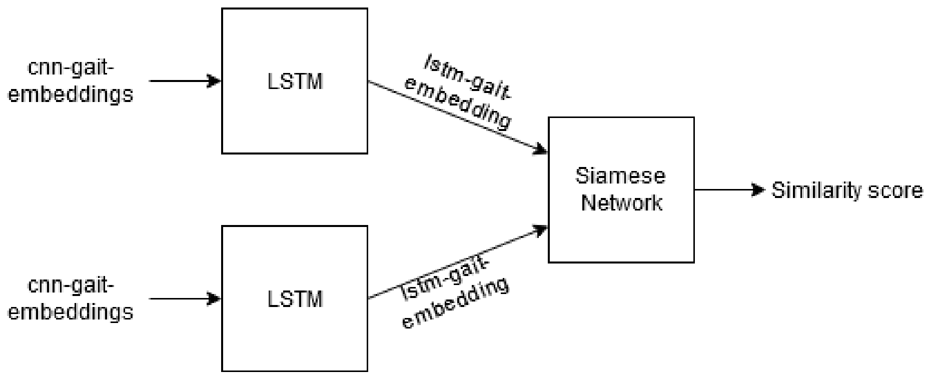


Figure 6.3: Architecture of the Siamese network

```
    # Recurrent Neural Network
    rnn = Sequential()
    rnn.add(LSTM(256, return_sequences=False,
                 input_shape=(15, 2048),
                 dropout=0.35, recurrent_dropout=0.2))
    rnn.add(Dropout(0.2))

    # Generate the embeddings for the two input gait videos
    encoded_l = rnn(left_input)
    encoded_r = rnn(right_input)

    L1_layer = Lambda(lambda tensors:K.abs(tensors[0] - tensors[1]))
    L1_distance = L1_layer([encoded_l, encoded_r])

    prediction = Dense(1, activation='sigmoid')(L1_distance)

    siamese_net = Model(inputs=[left_input, right_input], outputs=prediction)

    return siamese_net
```

Listing 6.1: Code for Siamese network creation

6.2 Training and Experimental Evaluation

Even though we changed the network architecture and its purpose, we can use the very same code for training the network, because we still use a generator for providing the input, as shown in Listing 5.3.

On our machine the training took approximately 1 day and 7 hours, and achieved an accuracy of 98.3%. In order to show the performance of the Siamese network, we will look at an example. We have three different videos:

1. **Video A:** Person A walking, the camera is positioned at 90 degrees. The images are shown in Figure 6.4.

2. **Video A2:** Person A walking, the camera is positioned at 180 degrees. The images are shown in Figure 6.5.

3. **Video B:** Person B walking, the camera is positioned at 90 degrees. The images are shown in Figure 6.6.

Figure 6.4 and Figure 6.6 look quite similar, because the camera was positioned at the same angle. Similar to the concept shown in Figure 6.2 we calculate the similarity output of each pair. The results are shown in Table 6.1. At first view, it seems reasonable, that the similarity score of the same person (although different camera angle) is the highest one. As a sanity check, this is already a good sign. On the other hand, the network is 48% confident, that the person from Figure 6.4 and Figure 6.6 are the same person. This should be as close to zero as possible. One explanation of the quite high value is

Input A	Input B	Similarity
A	A2	0.79
A	B	0.48
A2	B	0.07

Table 6.1: Similarity score for all test-image pairs.

that many conditions, mainly the camera angles, are exactly the same. Further work is proposed in Chapter 9.

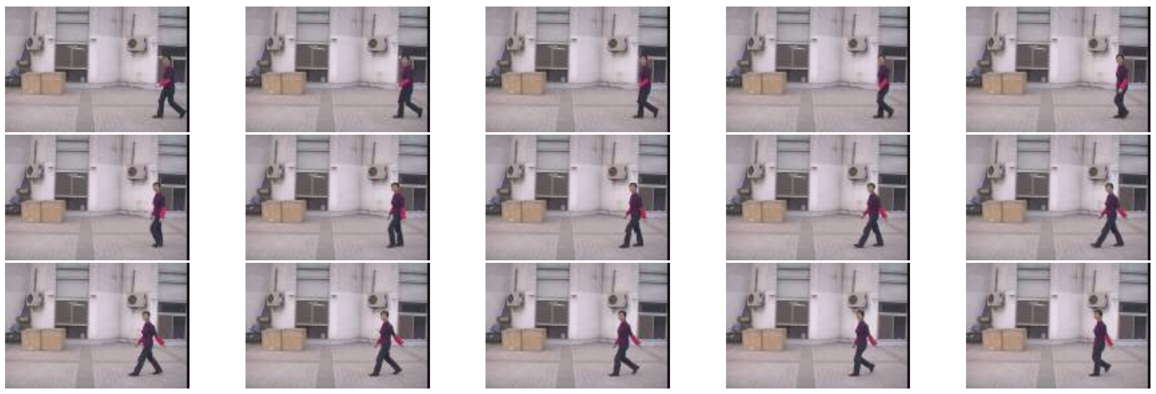


Figure 6.4: Images show person A walking with a camera angle of 90 degrees.

The Siamese network, which was introduced in this chapter, is able to distinguish two people and calculate a similarity score. In the next chapter, we will introduce a method of combining all chapters which performed gait-embedding extraction into a single algorithm.

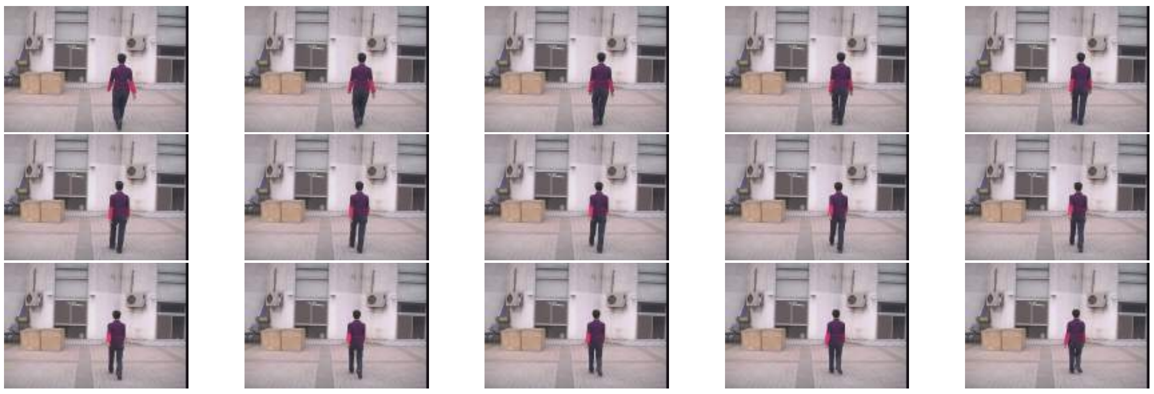


Figure 6.5: Images show person A walking with a camera angle of 180 degrees.

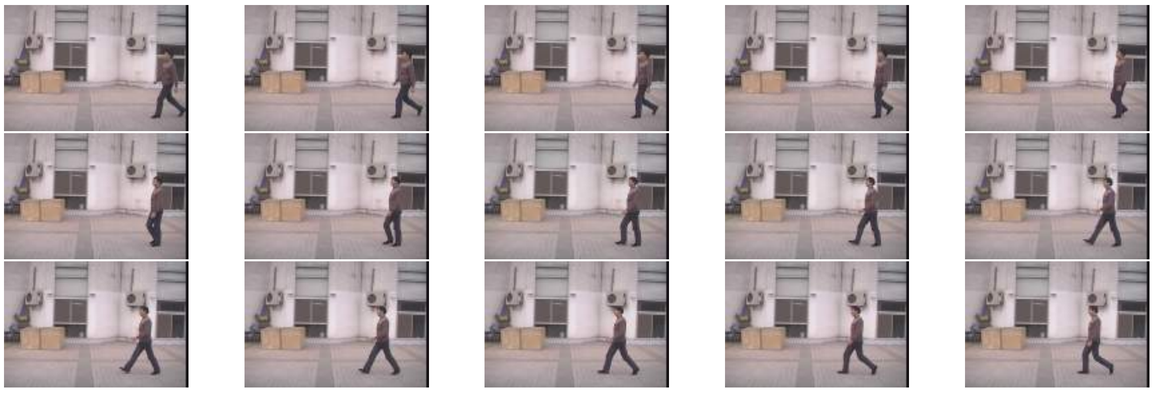


Figure 6.6: Images show person B walking with a camera angle of 90 degrees.

Chapter 7

Combination

So far, three networks have been introduced.

1. In Chapter 3 a Mask R-CNN network extracts the silhouette of a person, from each video frame.

2. Spatial information is extracted from each silhouette in Chapter 4 using a CNN.

3. A LSTM [5] network calculates a lstm-gait-embedding in Chapter 5, which dramatically increases the accuracy by adding important temporal information.

So far, these networks are executed in isolation. In order to make the networks more practically usable, this chapter will combine all three of them into a single application. Thus, it is possible to give the algorithm two gait videos, and receive a similarity score as output. Internally, the algorithm first extracts the silhouettes, calculates both spatial

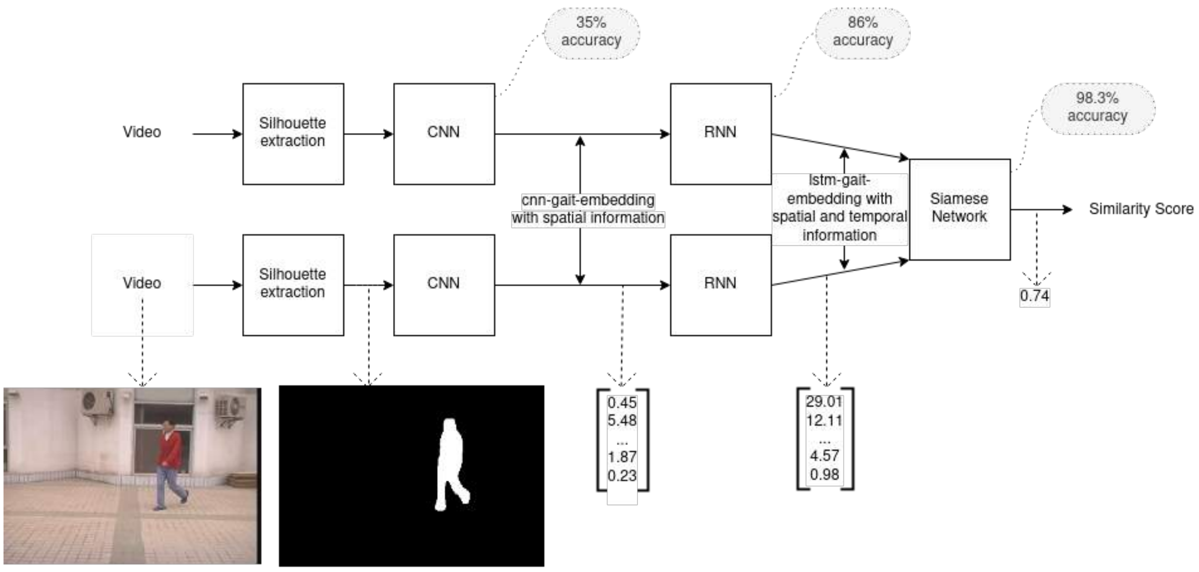


Figure 7.1: Architecture for the networks of this thesis

and temporal information and last but not least, will feed the resulting embeddings into the Siamese network to retrieve the final similarity score.

Listing 7.1 shows how to combine Chapters 4, 5 and 6 to extract the similarity between two people. The only input the proposed algorithm receives, are two datasets. The algorithm assumes that they are both in distinct folders, which are specified in lines 1 and 2. In Section 4.5 the Extractor() was already introduced, which will extract the silhouette and calculate the cnn-gait-embedding. The for-loop in line 8 iterates over all images and calculates its cnn-gait-embedding, which is stored in an array. Next, line 16 loads the LSTM/Siamese network. In the next line, the two arrays with the cnn-gait-embeddings are fed to the LSTM/Siamese network, which returns the final similarity score.

Figure 7.1 shows the high-level view of the algorithm. The dashed lines at the bottom visualize some example output after every step. Furthermore, the accuracy improvement in every step is shown.

```
1 a = "/path/to/images/1"
```

```
b = "/path/to/images/2"

extractor = Extractor()

def get_cnn_embedding(path):
    ret = []
    for person_id in os.listdir(path+"/sil"):
        ret.append(extractor.extract(path + "/" + person_id))
    return np.asarray(ret)


cnn_embedding_a = get_cnn_embedding(a)
cnn_embedding_b = get_cnn_embedding(b)

model = load_model("/path/to/siamese/network.hdf5")
prediction = model.predict([np.expand_dims(cnn_embedding_a, axis=0),np.expand_dims(
    cnn_embedding_b, axis=0)])

print(str(prediction[0][0]*100)+"␣%")
```

Listing 7.1: Code for calculating similarity between two people

The final network needs quite some time computing the similarity score. Future work could focus on reducing the computation time, as discussed in Chapter 9. To calculate a similarity score for two 15-frames videos takes about 1.5 minutes. Most time (64%) is spent on retrieving the cnn-gait-embeddings. 35% of the time is spent on loading the LSTM/Siamese network. Since this is a constant factor, it will not play a significant role if the gait recognition system is run continuously. The distribution of the computation time is shown in Figure 7.2.

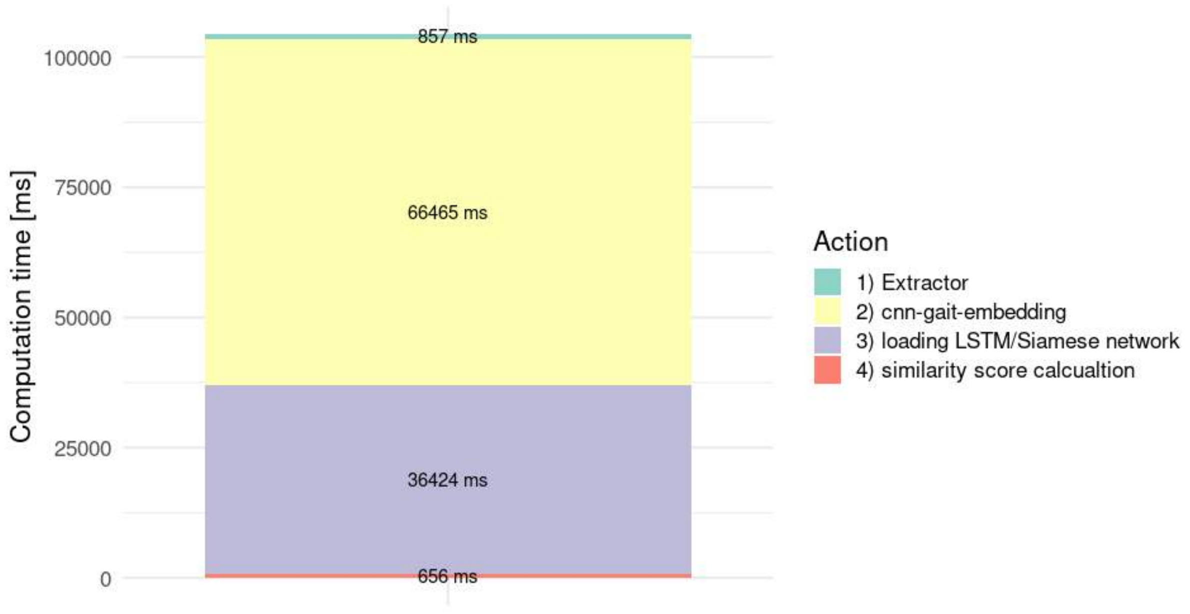


Figure 7.2: Computation times needed for the networks of this thesis

Chapter 8

Related work

In 2016 Zhang et. al. [37] proposed a similar method for gait recognition. In this paper a gait-energy image (GEI) is created from input videos. Next, just like in this thesis, a Siamese network is used to differentiate two people. The paper used a different dataset ([38]) which features more people but in fewer variations. There are only four different camera angles (vs. 11 with the dataset used in this thesis) and people are not carrying backpacks. The proposed network in this paper achieves an accuracy of 96.02%. The main difference to this thesis is that the input to the neural network are GEI images in [37], while the method proposed by this thesis uses unprocessed videos as input. Even though there is more variation regarding the camera angle, the method this thesis proposes achieves a slightly better accuracy ($\sim$ 2 percent points). Since a GEI image is a more compact form of the original video, some information is lost. This might explain the slightly lower accuracy, if compared to the results in Chapter 6.

Another paper [39] focused its work on relaxing the constraint of image quality. This is accomplished by learning high-level descriptors from low-level motion features.

Setting	Accuracy
Normal	87.2%
With bag	76.2%
In a coat	58.2%

Table 8.1: Accuracy for different settings by method proposed by [41]

This method proposed by Castro et. al. [39] achieves state-of-the-art results while using just 80x60 pixels-images. The authors of the paper [39] used the TUM-GAID dataset [40], which contains gait-videos of 305 people. Castro et. al. [39] achieved an accuracy of 98.0%.

The authors of [41] proposed a Spatial-Temporal Graph Attention Network to recognize people based on their gait information. Since this paper uses the same dataset (CASIA), a comparison to the method proposed by this paper is especially interesting. The paper published the accuracy of their model for each category of the CASIA dataset, which is shown in Table 8.1. In Chapter 5 this thesis computed a single accuracy, which includes all settings. In order to compare the accuracy with our result, we need to average over the different settings. Since the data is roughly distributed uniformly between the different settings, the different settings are weighted equally. This gives an overall accuracy of 73.9% for the method proposed by Wu et. al. [41]. As shown in Section 5.3, our proposed method achieves an accuracy of 86%.

In 2017, [42] created a deep convolutional neural network for gait recognition. Compared to the method this thesis proposes, the network preprocesses the gait video by calculating GEI images. Next, as the title already implies, a deep CNN with 8 layers is trained. Even though Alotaibi et. al. [42] use the same dataset as this thesis, they do not use every image. More specifically, only images with a camera angle of 90 degrees are used. Since this thesis uses 11 different camera angles, it is clear that the performance of [42] with an average accuracy of 90.43% of [42] is better.

An interesting approach is used by [43]. Instead of relying on neural networks only, Mohualdeen et.al. [43] pre-computed features. These features are then used as input for back propagation neural networks, which perform the classification. The benefit of this approach is, that the training time might be reduced dramatically. The authors of the paper used a different dataset [44] and achieved an accuracy between 88 and 98.8%, depending on the setting.

Chapter 9

Conclusion and outlook

This thesis aimed to show the potential of neural networks for gait recognition. The recognition should be robust to different settings, such as different background, shoes, underground, and so on. This is important because this allows for an unobtrusive way of recognizing people.

This is solved by using different neural networks to extract as much information from a silhouette as possible. In the end, the neural networks return a similarity score for two given videos and achieves an accuracy of 98.3% for identity verification on the CASIA gait database.

We propose three possibilities for future work:

1. **Use a model-based approach**
   By training a CNN, LSTM and Siamese network without restrictions, this thesis implicitly focuses on model-free approaches of gait recognition. Another approach

would be to follow a model-based approach. Recent work in machine learning, such as Densepose [45] make it fairly straight-forward to not only segment the silhouette of a person, but also its body parts. This additional knowledge may allow for an even higher accuracy.

2. **Real-time gait recognition**

   As discussed in Chapter 7, the framework this thesis proposes can not perform gait recognition in real-time. Further work could focus on reducing the run-time complexity.

3. **Multiple people**

   This thesis proposes a method of identifying a person based on a gait video featuring this person only. Work in the future could focus on relaxing this constraint, thus allowing the presence of multiple people in the gait video.

List of Figures

List of Tables

Listings

Bibliography

[1] Ju Han and Bir Bhanu. Individual recognition using gait energy image. *IEEE transactions on pattern analysis and machine intelligence*, 28(2):316–322, 2005.

[2] Erdem Yoruk, Ender Konukoglu, Bülent Sankur, and Jérôme Darbon. Shape-based hand recognition. *IEEE transactions on image processing*, 15(7):1803–1815, 2006.

[3] Orcan Alpar and Ondrej Krejcar. Dorsal hand recognition through adaptive ycbcr imaging technique. In *International Conference on Computational Collective Intelligence*, pages 262–270. Springer, 2016.

[4] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 2961–2969, 2017.

[5] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[6] Gregory Koch, Richard Zemel, and Ruslan Salakhutdinov. Siamese neural networks

for one-shot image recognition. In *ICML deep learning workshop*, volume 2. Lille, 2015.

[7] Danny Thakkar. Top five biometrics: Face, fingerprint, iris, palm and voice. *Bayometric [online].[cit. 2019-03-04]. Available from: https://www. bayometric. com/biometricsface-finger-iris-palm-voice*, 2017.

[8] P Jonathon Phillips and Alice J O'toole. Comparison of human and computer performance across face recognition experiments. *Image and Vision Computing*, 32(1):74–85, 2014.

[9] Dang-Hui Liu, Kin-Man Lam, and Lan-Sun Shen. Illumination invariant face recognition. *Pattern Recognition*, 38(10):1705–1716, 2005.

[10] Luan Tran, Xi Yin, and Xiaoming Liu. Disentangled representation learning gan for pose-invariant face recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1415–1424, 2017.

[11] Mark Williams. Better face-recognition software. *Technology Review*, 30, 2007.

[12] Ming Gao, Xihong Hu, Bo Cao, and Dianxin Li. Fingerprint sensors in mobile devices. In *2014 9th IEEE conference on industrial electronics and applications*, pages 1437–1440. IEEE, 2014.

[13] Yao Qing-jun Shan Jia-jia and Yang Yi-lin Jiang Fa-chao. Fingerprint identification car door lock hardware and software design. *Microcomputer Information*, 11, 2010.

[14] A Aditya Shankar, PRK Sastry, AL Vishnu Ram, and A Vamsidhar. Finger print based door locking system. *International Journal Of Engineering And Computer Science*, 4(3):10810–10814, 2015.

[15] Nianfeng Liu, Haiqing Li, Man Zhang, Jing Liu, Zhenan Sun, and Tieniu Tan. Accurate iris segmentation in non-cooperative environments using fully convolutional networks. In *2016 International Conference on Biometrics (ICB)*, pages 1–8. IEEE, 2016.

[16] Mayank Vatsa, Richa Singh, and P Gupta. Comparison of iris recognition algorithms. In *International Conference on Intelligent Sensing and Information Processing, 2004. Proceedings of*, pages 354–358. IEEE, 2004.

[17] Khalid Bashir, Tao Xiang, and Shaogang Gong. Feature selection on gait energy image for human identification. In *2008 IEEE international conference on acoustics, speech and signal processing*, pages 985–988. IEEE, 2008.

[18] Tanmay T Verlekar, Paulo L Correia, and Luís D Soares. View-invariant gait recognition system using a gait energy image decomposition method. *IET Biometrics*, 6(4):299–306, 2017.

[19] Gang Qian, Jiqing Zhang, and Assegid Kidane. People identification using floor pressure sensing and analysis. *IEEE Sensors Journal*, 10(9):1447–1460, 2010.

[20] Takeshi Yamakawa, Kazuhiko Taniguchi, Kazunari Asari, Syoji Kobashi, and Yutaka Hata. Biometric personal identification based on gait pattern using both feet pressure change. In *2010 World Automation Congress*, pages 1–6. IEEE, 2010.

[21] Jürgen T Geiger, Martin Hofmann, Björn Schuller, and Gerhard Rigoll. Gait-based person identification by spectral, cepstral and energy-related audio features. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 458–462. IEEE, 2013.

[22] Jürgen T Geiger, Maximilian Kneißl, Björn W Schuller, and Gerhard Rigoll. Acoustic gait-based person identification using hidden markov models. In *Proceedings of the 2014 Workshop on Mapping Personality Traits Challenge and Workshop*, pages 25–30, 2014.

[23] Hoang Minh Thang, Vo Quang Viet, Nguyen Dinh Thuc, and Deokjai Choi. Gait identification using accelerometer on mobile phone. In *2012 International Conference on Control, Automation and Information Sciences (ICCAIS)*, pages 344–348. IEEE, 2012.

[24] Felix Juefei-Xu, Chandrasekhar Bhagavatula, Aaron Jaech, Unni Prasad, and Marios Savvides. Gait-id on the move: Pace independent human identification using cell phone accelerometer dynamics. In *2012 IEEE Fifth International Conference on Biometrics: Theory, Applications and Systems (BTAS)*, pages 8–15. IEEE, 2012.

[25] Patrick Connor and Arun Ross. Biometric recognition by gait: A survey of modalities and features. *Computer Vision and Image Understanding*, 167:1–27, 2018.

[26] Herbert P Von Schroeder, Richard D Coutts, Patrick D Lyden, E Billings, and Vernon L Nickel. Gait parameters following stroke: a practical assessment. *Journal of rehabilitation research and development*, 32:25–25, 1995.

[27] Theresa Foti, Jon R Davids, and Anita Bagley. A biomechanical analysis of gait during pregnancy. *JBJS*, 82(5):625, 2000.

[28] Jeffrey M Hausdorff, Susan L Mitchell, Renee Firtion, Chung-Kang Peng, Merit E Cudkowicz, Jeanne Y Wei, and Ary L Goldberger. Altered fractal dynamics of gait: reduced stride-interval correlations with aging and huntington's disease. *Journal of applied physiology*, 82(1):262–269, 1997.

[29] Lois Finch, Hugues Barbeau, and Bertrand Arsenault. Influence of body weight support on normal human gait: development of a gait retraining strategy. *Physical Therapy*, 71(11):842–855, 1991.

[30] Subashan Perera, Kushang V Patel, Caterina Rosano, Susan M Rubin, Suzanne Satterfield, Tamara Harris, Kristine Ensrud, Eric Orwoll, Christine G Lee, Julie M Chandler, et al. Gait speed predicts incident disability: a pooled analysis. *Journals of Gerontology Series A: Biomedical Sciences and Medical Sciences*, 71(1):63–71, 2016.

[31] Sepp Hochreiter. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, 6(02):107–116, 1998.

[32] Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. Signature verification using a" siamese" time delay neural network. In *Advances in neural information processing systems*, pages 737–744, 1994.

[33] Lisa Torrey and Jude Shavlik. Transfer learning. In *Handbook of research on machine*

learning applications and trends: algorithms, methods, and techniques, pages 242–264. IGI Global, 2010.

[34] Waleed Abdulla. Mask r-cnn for object detection and instance segmentation on keras and tensorflow. `https://github.com/matterport/Mask_RCNN`, 2017.

[35] Christian Szegedy, Sergey Ioffe, Vincent Vanhoucke, and Alexander A Alemi. Inception-v4, inception-resnet and the impact of residual connections on learning. In *Thirty-first AAAI conference on artificial intelligence*, 2017.

[36] Martín Abadi, Paul Barham, Jianmin Chen, Zhifeng Chen, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Geoffrey Irving, Michael Isard, et al. Tensorflow: A system for large-scale machine learning. In *12th USENIX Symposium on Operating Systems Design and Implementation (OSDI 16)*, pages 265–283, 2016.

[37] Cheng Zhang, Wu Liu, Huadong Ma, and Huiyuan Fu. Siamese neural network based gait recognition for human identification. In *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 2832–2836. IEEE, 2016.

[38] Md Zasim Uddin, Thanh Trung Ngo, Yasushi Makihara, Noriko Takemura, Xiang Li, Daigo Muramatsu, and Yasushi Yagi. The ou-isir large population gait database with real-life carried object and its performance evaluation. *IPSJ Transactions on Computer Vision and Applications*, 10(1):5, 2018.

[39] Francisco Manuel Castro, Manuel J Marín-Jiménez, Nicolás Guil, and Nicolás Pérez De La Blanca. Automatic learning of gait signatures for people identification. In *In-*

ternational Work-Conference on Artificial Neural Networks, pages 257–270. Springer, 2017.

[40] Martin Hofmann, Jürgen Geiger, Sebastian Bachmann, Björn Schuller, and Gerhard Rigoll. The tum gait from audio, image and depth (gaid) database: Multimodal recognition of subjects and traits. *Journal of Visual Communication and Image Representation*, 25(1):195–206, 2014.

[41] Xinhui Wu, Weizhi An, Shiqi Yu, Weiyu Guo, and Edel B García. Spatial-temporal graph attention network for video-based gait recognition. In *Asian Conference on Pattern Recognition*, pages 274–286. Springer, 2019.

[42] Munif Alotaibi and Ausif Mahmood. Improved gait recognition based on specialized deep convolutional neural network. *Computer Vision and Image Understanding*, 164:103–110, 2017.

[43] Molhema Mohualdeen and Magdi Baker. Gait recognition based on silhouettes sequences and neural networks for human identification. *Indonesian Journal of Electrical Engineering and Informatics (IJEEI)*, 6(1):110–117, 2018.

[44] Martin Hofmann, Shamik Sural, and Gerhard Rigoll. Gait recognition in the presence of occlusion: A new dataset and baseline algorithms. 2011.

[45] Rıza Alp Güler, Natalia Neverova, and Iasonas Kokkinos. Densepose: Dense human pose estimation in the wild. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 7297–7306, 2018.

Philipp Hofer

Lebenslauf

Personal data

Name	*Philipp Hofer*
Birthday	*29.01.1997*
Birthplace	*Linz*
Nationality	*Österreich*

Education

current	**Johannes Kepler University Linz**, *Computer Science*, 8. Semester.
Feb. - Aug. 2019	**Semester abroad ETH Zurich, Switzerland**.
Aug. - Dec. 2017	**Semester abroad Oxford Brookes, United Kingdom**.
5 years	**HTL Perg**, *Computer science and business economics*.
4 Jahre	**High school**, *BRG Fadingerstraße*, Linz.

Professional Experience

June - July 2019	**Self-employed**, *Luftenberg*, Software for Fifth and Missing (Barrie, ON, Kanada) created (it-results.at/portfolio/work-fifth.html).
Juli - Aug. 2018	**Self-employed**, *Luftenberg*, Software for Fresenius Kabi AG created (it-results.at/portfolio/work-fres.html).
February 2018	**Self-employed**, *Luftenberg*, Software for Kulturhaus Bruckmühle (Pregarten, Österreich) created (www.bruckmuehle.at).
May 2017	**Foundation Sole Proprietorship**, *IT-Results*, www.it-results.at.
July - Aug. 2017	**Catalysts GmbH**, *Linz*, Software Development.
July - Sept. 2016	**Catalysts GmbH**, *Linz*, Software Development.
August 2015	**JKU**, *Linz*, Diploma thesis for HTL.
August 2014	**Fabasoft AG**, *Linz*, Programming and testing department.
Juli 2013	**Chamber of commerce Upper Austria**, *Linz*, Informatics department.
August 2012	**Municipality Luftenberg**, Building department.

Sonstiges

2017 - derzeit	**Layout designer of municipal paper of Luftenberg, Austria**.
2012 - 2016	**Class representative**, *HTL Perg*.
2014 - 2015	**School representative**, *HTL Perg*.

Hobbys

Scout, *Leader (since 2015)*.

Running und biking.

Rubinweg 8 – 4225 Luftenberg
✉ *philipp.hofer@protonmail.com*

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Masterarbeit selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt bzw. die wörtlich oder sinngemäß entnommenen Stellen als solche kenntlich gemacht habe.

Die vorliegende Masterarbeit ist mit dem elektronisch übermittelten Textdokument identisch.

Linz, am 29.04.2020

Philipp Hofer