university of groningen / faculty of mathematics and natural sciences / artificial intelligence

Pre-trained Deep Convolutional Neural Networks for Face Recognition

Siebert Looije

S2209276

January 2018

MSc. Thesis

Artificial Intelligence

University of Groningen, The Netherlands

Supervisors

Dr. M.A. (Marco) Wiering

K. (Klaas) Dijkstra, MSc.

ALICE Institute

University of Groningen

Nijenborgh 9, 9747 AG, Groningen, The Netherlands

“What we learn with pleasure
we never forget. ”’

- Alfred Mercier

“I think people need to understand that deep learning is making a lot of things, behind-the-scenes, much better. Deep learning is already working in Google search and in image search; it allows you to image search a term like ’hug’.”’

- Geoffrey Hinton

University of Groningen

Abstract

Faculty of Mathematics and Natural Sciences

Master of Science

Pre-trained Deep Convolutional Neural Networks for Face Recognition

by Siebert Looije
S2209276

Pre-training of models is important because of the unavailability of datasets and models that are becoming more complex. We investigate two aspects of pre-training using face recognition tasks. The first is the use of models that are pre-trained on face datasets and non-face datasets. We will evaluate five pre-trained models based on their results with freezing multiple layers and on robustness. The second aspect is to investigate universal features in pre-trained deep models. This is done by evaluating the performance using only the first few layers for pre-training. This is also investigated by swapping the first layers of the models.

We show that models pre-trained on face datasets achieve better results and are more robust in three face recognition tasks than models pre-trained on non-face datasets. The results with pre-training and swapping only the first layers show a significant difference between models that are pre-trained on face datasets and non-face datasets. From this, we conclude that it is important which dataset is used for pre-training the models and used for testing in face recognition. We also conclude that the first few layers of pre-trained models affect performance on face recognition.

Acknowledgements

I want to thank everyone who has helped me to finish this thesis. First, I want to thank my supervisors, Dr. M.A. Wiering and K. Dijkstra, for supporting me throughout the project. The guidance with making decisions during the research was very helpful. At points, where I got off the right path or I was overdoing things, they helped me to get on the right path again. Furthermore, they helped me to tackle some technical challenges with their expertise on machine learning.

Secondly, I want to thank my two great friends, Jos van de Wolfshaar and Matthia Sabatelli for trying to understand all the ideas I had, for drinking coffee and eating at the library. This support really was a very big help for me and it helped me to get motivated every time. I will never forget these times.

Lastly, I want to thank Timo, my girlfriend and father for support with correcting my grammar, which was often quite hard because of the subjects they never faced before. Hopefully, they learned something on this topic.

Siebert Looije
January 2018

List of Figures

List of Tables

Contents

Chapter 1

Introduction

1.1 Transfer learning

Transfer learning can be seen as using knowledge from one domain to get better in another domain. Transfer learning will reduce the need to recollect the training data for a specific domain [53]. As Pan et al. state, transfer learning allows the distributions, tasks, and domains to be different for the training and testing data [53]. Transfer learning is motivated by the fact that people can intelligently apply knowledge from previously learned solutions to solve new problems or find better solutions [53]. For example, it is easier to learn French if you already know Latin. Another example is: if you already learned to ride a scooter then it will be easier to learn to ride a motorcycle.

In the field of *machine learning*, transfer learning is often the same but it is used for sharing the weights of neural networks. The usual approach is to train all the layers of one network and then copying the first n layers of this network to a second network. This step is called *pre-training* of the network. When the layers are copied to the second network then the remaining layers of the second network can be randomly initialized and trained on the target task with a different dataset. There are two ways how the pre-trained weights can be used. The first way is to train these weights together with the weights of the last layers. The second approach is not to change the weights of the pre-trained layers. The weights are left frozen during the training on the target task. This second approach is used in this thesis.

Yosinski et al., noticed that the first layer features tend to be either *Gabor filters* or color blobs [82]. They saw that this is common on different training objectives and training datasets. These first layer features are called *general* or *universal* because they are standard to occur regardless of the cost function or the image dataset [82]. It can be important to know when a layer is general because it is likely to be used for transfer learning. The last layer features of a network are called *specific* because these features are only used for the target task and dataset.

1.2 Face recognition

Face recognition is increasing in the fields of information security, entertainment, smart cards, law enforcement and surveillance [84]. The upcoming trends, virtual reality and human-robot interaction, have given a boost in the interest for face recognition in entertainment. The strong need for systems that are user-friendly and protect our privacy, has a great impact on the biometric field and therefore in the field of face recognition. The increase of interest in face recognition specifically also has to do with the feasibility of available technologies after 30 years of research [84]. According to Abate et al., face recognition is a good compromise regarding reliability, social acceptance, security and privacy compared to other biometric technologies [2]. Fingerprint and iris scanners, for example, require interaction with a device. This may be considered intrusive and is often more expensive than the technology behind face recognition [84].

Zhao et al. give a clear definition of face recognition using machines: *given still or video images of a scene, identify or verify one or more persons in the scene using a stored database of faces* [84]. Using

this definition, face recognition can be divided into two categories: *identification* and *verification*. Face identification is a *one-to-many* matching problem because it needs to compare a face template with all the face templates existing in the database. Face verification is a *one-to-one* matching problem because it compares a face template with another face template. Face identification is seen as a more challenging problem because it needs to match the face template with more different face templates, whereas in verification it only needs to be matched with one face template [10]. Figure 1.1 shows an example of face verification and an example of face identification.

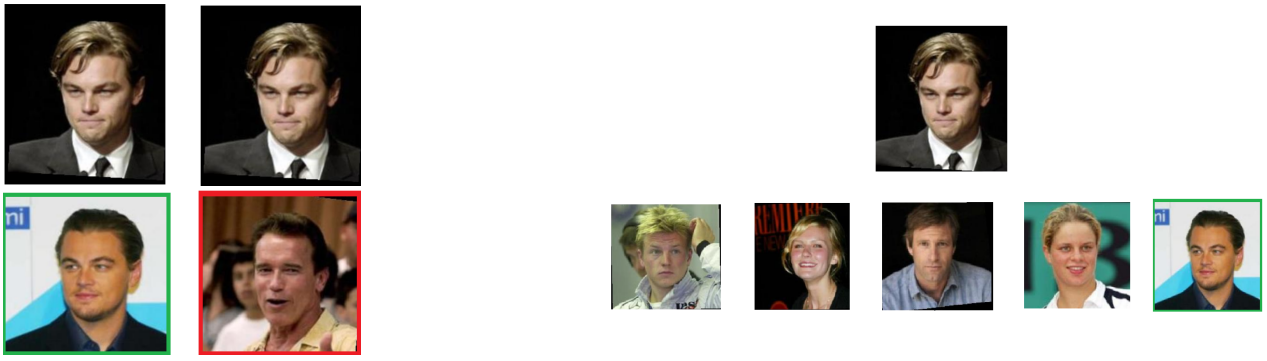

Figure 1.1: Face recognition can be divided into two categories: verification and identification. **Left:** an example of face verification. Two images are compared and the model predicts if it is the same person. This is a one-to-one matching problem. **Right:** an example of face identification. An image is compared and the model predicts to which person it belongs. This is seen as a one-to-many matching problem. The images are taken from the labeled faces in the wild benchmark [29].

1.3 Research questions

We are going to combine transfer learning with face recognition and investigate several aspects of transfer learning with face recognition. The two models already proved that they can achieve good results on image recognition tasks. The first model is called GoogleNet that was introduced by Szegedy et al. in 2015 [72]. The second model is known as Inception Residual Network V2, which was introduced by He et al. and won the ImageNet 2015 challenge [27].

1.3.1 Models pre-trained and trained on images of the same domain

Pre-training models with images of the same domain improves the training on images of the same domain [82]. This is concluded by a paper from Yosinski et al., they used and removed the images of this same domain in the pre-training phase. They noticed that the results in the training phase improved using the images in the pre-training phase and the results decreased when they removed the images in the pre-training phase. In similar research by Huh et al., they found results that contradict with the findings of Yosinski et al. [82]. They conclude that using images of the same domain in the pre-training phase, did not improve the results in the training phase [30].

In extension of these two papers, we investigate if using models that are pre-trained on images from the same domain have a positive influence on the target objective. Face recognition is the target objective we use. We investigate if models pre-trained on face datasets achieve better performance than models pre-trained on non-face datasets. We will try to answer the following research question: *Do models pre-trained on face datasets achieve better results in comparison with pre-training on non-face datasets in face recognition?*

This question is examined from different points of view. The first view is to compare the results of models pre-trained on face datasets with models pre-trained on non-face datasets. We also study if there is a difference in results using more pre-trained layers. The models pre-trained on face datasets should have features that are more common with the dataset that is used for training. This is the second topic, which is investigated in this thesis. This is done by looking at the difference in robustness between models pre-trained on face datasets and on non-face datasets. This results in the following sub-questions:

1. *Do models pre-trained on face datasets perform better in comparison with non-face datasets per n frozen layers?*
2. *Are models pre-trained on face datasets more robust than models pre-trained on non-face datasets?*

1.3.2 Universal features

Yosinski et al. stated that the first-layer features are often Gabor filters or color blobs for different training objectives [82]. Therefore they occur regardless of the dataset. This finding of Yosinski et al. should indicate these universal first layers having no influence on the performance. We investigate this by the research question: *Do the first few layers have an influence on the results in face recognition?*

We answer this question using two distinct approaches. The first approach is to pretrain only the first layers and training the rest of the model. This is the first approach to investigate this research question. In the second approach, we are going to swap the first layers of the pre-trained models and then train the model. These approaches are separated into two sub-questions:

1. *Is there a difference in performance when only using the first layers of the pre-trained models?*
2. *Is there a difference in performance when swapping the first layers of the pre-trained models?*

1.4 Contributions

Yosinski et al. and Huh et al. already investigated the influence of using pre-training and training on the images of the same domain but this was on the ImageNet dataset [30, 82]. This is different from our research. We compare models that are pre-trained on different datasets and then trained on another dataset. As far as we know this is not investigated before.

Furthermore, there is already plenty of research on the first-layer features. But the comparison of results between models that are only pre-trained on the first few layers is not fully explored. In this thesis, we investigate the influence of pre-training the first few layers with different datasets on the results. Lastly, the swapping of layers between different pre-trained models has not been done as far as we know. The removal and random sorting of the neurons within a network has already been done by Viet et al. [77], but swapping pre-trained layers between models has not been investigated.

1.5 Thesis outline

Chapter 2 describes the background information that is needed to understand the rest of the thesis. The related work on transfer learning and face recognition are also described in more details in this chapter. In Chapter 3, we describe which models are used in the experiments. In this chapter, we also describe which additional components are added to the models to obtain the results. Chapter 4 describes the experiments which are performed to answer the research questions. In Chapter 5 the results of these experiments are shown. The results of the experiments are discussed and the conclusions are given in Chapter 6.

Chapter 2

Background theory and related work

2.1 Neural networks

Neural networks are models of computation that are inspired by the biology of the brain [41]. Neural networks consist of a set of nodes that represent artificial neurons. The nodes are connected by directed edges. These edges represent the synapses of a biological neural network. Every neuron has an activation function l_j. The notation for the node where the edge starts is i and the node where the edge ends is j. Every edge is associated with a weight w_i^j, that corresponds to the directed edge between two nodes. The value of the neuron is calculated by the weighted sum of the values of the input nodes. This calculation is in literature [41], described as the *incoming activation*, notated as a_j and calculated as in equation 2.1. Secondly, the activation function on this weighted sum is calculated as in equation 2.2. An example of a simple artificial neuron, also known as *perceptron*, is shown in figure 2.1.

$$a_j = \sum_i w_i^j x_i \tag{2.1}$$

$$\hat{y}_j = l_j(a_j) \tag{2.2}$$

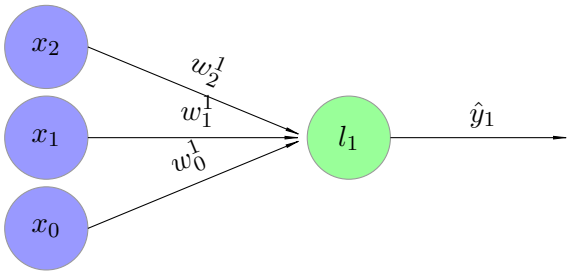

Figure 2.1: Example of an artificial neuron (perceptron), where x_0, x_1 and x_2 are the inputs, w_0^1, w_1^1 and w_2^1 make up the weight vector of the neuron and l_1 is the activation function. The output is defined as $\hat{y}_1$.

There are many different activation functions that can be used. The three most common activation functions are the *sigmoid*, *tanh* and *rectified linear unit (ReLU)* function. The activation in a sigmoid function $\sigma(a)$ is calculated by:

$$\sigma(a) = \frac{1}{(1 + e^{-a})}, \tag{2.3}$$

where a is the incoming activation. The activation of the tanh function $\phi(a)$ is calculated by:

$$\phi(a) = \frac{(e^a - e^{-a})}{(e^a + e^{-a})}, \tag{2.4}$$

where a is the incoming activation. The ReLU function can be seen as a piecewise linear function that prunes the negative side. It is calculated by the following equation:

$$f(a) = max(0, a). \tag{2.5}$$

When there are multiple classes that are needed to be predicted, the *softmax* function is often used. For K classes the softmax is calculated as follows:

$$f(a_k) = \frac{e^{a_k}}{\sum_{k'=1}^{K} e^{a_{k'}}}. \tag{2.6}$$

2.1.1 Feedforward neural network

Feedforward neural networks are a type of neural network. An example of a feedforward neural network is shown in figure 2.2. The neurons in a feedforward neural network are ordered into layers [70]. All the neurons of a layer in a feedforward neural network are connected with all the neurons in the next layer. There are no connections between the neurons in the same layer. The input, often denoted as x, is fed to the first layer of the network and each following layer computes the activation until the final layer is reached [41]. Each of the weights between the neurons is iteratively updated to minimize a *loss function* $\mathcal{L}(y, \hat{y})$, where $\hat{y}$ is the predicted output and y is the target output. In this thesis, we use a loss function that calculates the difference between the output of the last layer and the target that represents the class label.

An example of a simple feedforward network is a *multi-layer perceptron* (MLP). Figure 2.2 shows an example of an MLP. This MLP has three input nodes, two hidden layers with four and five hidden nodes and an output layer with two output nodes. The information flow goes from left to right. In this example the sigmoid activation function (σ) is used.

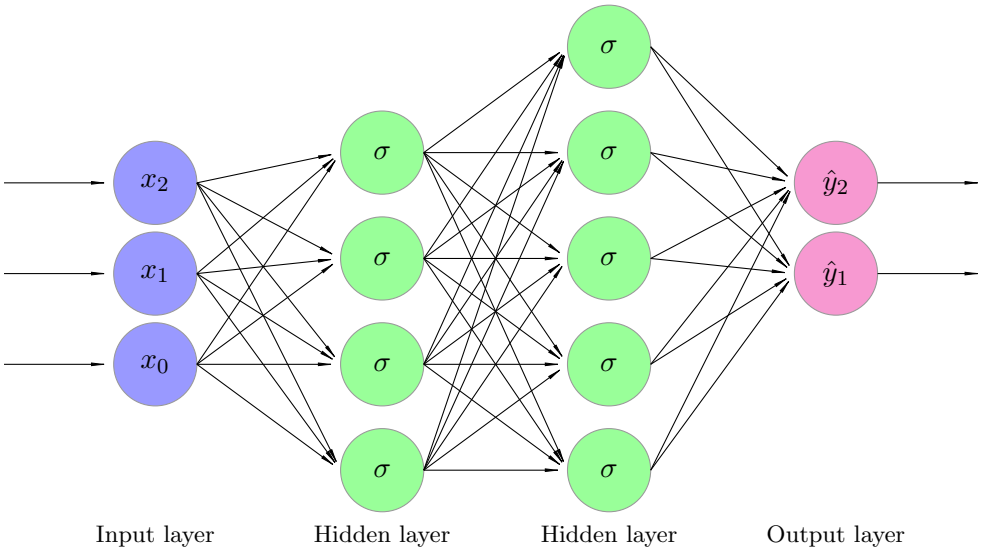

Figure 2.2: An example of a multi-layer perceptron (MLP). It has three input nodes (x_0, x_1 and x_2), two hidden layers with respectively four and five hidden neurons and an output layer with two neurons ($\hat{y}_1$ and $\hat{y}_2$). In this example, the neurons in the hidden layers have sigmoid (σ) activation functions.

2.1.2 Convolutional neural network

Convolutional neural networks are a type of neural network that are specialized to be used on data that has a *grid-like topology* [8]. Examples of grid-like data are time series data in 1D and image data in 2D. This is because an image can be seen as a 2D grid of pixels. The mathematical operation *convolution* is the basis of the convolutional neural network. Convolutional neural networks are feedforward neural networks that use the convolution operation instead of general matrix multiplication in the layers [8].

As an example, consider a noisy input signal $x(i)$ with measurements for time step i. To get a less noisy input signal it is helpful to get the average of several measurements. In practice, the more recent measurements are more relevant to the task, so a weighted average with more weight on the recent measurement gives a better input signal. This can be done by the weighting function $w(n)$, where n is the age of a measurement. The operation that is applied here is called the *convolution* operator:

$$\hat{y}(i) = \sum_n x(i-n)w(n). \tag{2.7}$$

Where we specify $\hat{y}$ as the output of the convolution operator to keep it consistent with the actual output of a neuron. Often the weighting average function is called the *kernel* and the output is called the *feature map* [8].

In deep learning applications, there is often more than one dimension in the data. Therefore, the input and the kernel have multidimensional arrays. The convolution operation can also be applied on more than one axis at a time. The convolution formula will be changed as follows:

$$\hat{y}(i,j) = \sum_m \sum_n x(i-m, j-n)w(m,n), \tag{2.8}$$

where i and j are the indexes of the two-dimensional input array and m and n are the indexes of the two-dimensional kernel array. As an example, we assume that there is a 2D matrix input x and 3x3 kernel w:

$$x = \begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 8 & 9 & 10 & 11 & 12 & 13 & 14 \\ 15 & 16 & 17 & 18 & 19 & 20 & 21 \\ 22 & 23 & 24 & 25 & 26 & 27 & 28 \\ 29 & 30 & 31 & 32 & \mathbf{33} & 34 & 35 \\ 36 & 37 & 38 & 39 & 40 & 41 & 42 \\ 43 & 44 & 45 & 46 & 47 & 48 & 49 \end{bmatrix} \quad w = \begin{bmatrix} 3 & 3 & 3 \\ 3 & 3 & 3 \\ 3 & 3 & 3 \end{bmatrix}.$$

Then to calculate, for example, the output value for 33. The sum of the pairwise multiplication of the following matrixes is calculated:

$$\begin{bmatrix} 25 & 26 & 27 \\ 32 & \mathbf{33} & 34 \\ 39 & 40 & 41 \end{bmatrix} * \begin{bmatrix} 3 & 3 & 3 \\ 3 & 3 & 3 \\ 3 & 3 & 3 \end{bmatrix}.$$

Which results in the output value for 33: 25∗3+26∗3+27∗3+32∗3+33∗3+34∗3+39∗3+40∗3+41∗3 = 891. If this is done for all the values of the matrix x then:

$$\hat{y} = \begin{bmatrix} 243 & 270 & 297 & 324 & 351 \\ 432 & 459 & 486 & 513 & 540 \\ 621 & 648 & 675 & 702 & 729 \\ 810 & 837 & 864 & 891 & 918 \\ 999 & 1026 & 1053 & 1080 & 1107 \end{bmatrix}$$

The size of output $\hat{y}$ is smaller than the size of the input x. The size will be smaller using more convolution operations. If the output $\hat{y}$ is smaller than a 1x1 matrix, which can happen if there are a lot of convolutional operations after each other, then it can not be used by the convolutional neural network anymore.

To solve this, the *padding* function is introduced. The idea of padding is simple, fill the rows and columns around the input x. The shape of $\hat{y}$ will be the same after the convolution as input x. To get the same shape again, the width and the height of the kernel w are used. In input x is inserted $floor(\frac{w_{height}-1}{2})$ above the first row and $floor(\frac{w_{height}}{2})$ below the last row. On the sides of the input is on the left side inserted $floor(\frac{w_{width}-1}{2})$ and on the right side is inserted $floor(\frac{w_{width}}{2})$. In the previous example the shape of the kernel is 3x3, which means that $w_{width} = 3$ and $w_{height} = 3$. If these values are filled in the formulas then: $floor(\frac{3-1}{2}) = 1$ and $floor(\frac{3}{2}) = 1$. Which means that one row below and above the matrix are filled with zeros and that on the left and right column of the input a new column is filled with zeros. After the convolution on the padded input, the shape of output $\hat{y}$, has the same shape as the original input x.

In the previous examples, images without color channels are used. In machine learning applications RGB images are often used. These images are represented as a 3-dimensional tensor with shape: (width, height, 3). If the image is an RGB image and the kernel dimension is 2-dimensional. Then the convolution operation of all three color channels are separately calculated and the outcome of the three convolutions are summed.

Pooling is another function that is often used in convolutional neural networks. Nowadays a lot of architectures are using pooling functions, an example of an architecture is the VGG-16 architecture proposed by Simonyan et al. [62]. The pooling function is used to reduce the size of the feature maps. The feature maps will still have the important information from the image. There are two pooling functions that are mainly used: max pooling [47] and average pooling [28]. Assume that the kernel size of the max pooling is 2x2 and stride 1. The input matrix x of the previous example is used. Then the result of max pooling is:

$$\hat{y} = \begin{bmatrix} 459 & 486 & 513 & 540 \\ 648 & 675 & 702 & 729 \\ 837 & 864 & 891 & 918 \\ 1026 & 1026 & 1080 & 1107 \end{bmatrix}$$

Often the convolution operation and the max pooling are set after each other. To complete this example, we show the output of the start matrix x with the convolution operation and max pooling.

$$\begin{bmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 \\ 8 & 9 & 10 & 11 & 12 & 13 & 14 \\ 15 & 16 & 17 & 18 & 19 & 20 & 21 \\ 22 & 23 & 24 & 25 & 26 & 27 & 28 \\ 29 & 30 & 31 & 32 & 33 & 34 & 35 \\ 36 & 37 & 38 & 39 & 40 & 41 & 42 \\ 43 & 44 & 45 & 46 & 47 & 48 & 49 \end{bmatrix} * \begin{bmatrix} 3 & 3 & 3 \\ 3 & 3 & 3 \\ 3 & 3 & 3 \end{bmatrix} = \begin{bmatrix} 243 & 270 & 297 & 324 & 351 \\ 432 & 459 & 486 & 513 & 540 \\ 621 & 648 & 675 & 702 & 729 \\ 810 & 837 & 864 & 891 & 918 \\ 999 & 1026 & 1053 & 1080 & 1107 \end{bmatrix}$$

$$\begin{bmatrix} 243 & 270 & 297 & 324 & 351 \\ 432 & 459 & 486 & 513 & 540 \\ 621 & 648 & 675 & 702 & 729 \\ 810 & 837 & 864 & 891 & 918 \\ 999 & 1026 & 1053 & 1080 & 1107 \end{bmatrix} * \begin{bmatrix} max & max \\ max & max \end{bmatrix} = \begin{bmatrix} 459 & 486 & 513 & 540 \\ 648 & 675 & 702 & 729 \\ 837 & 864 & 891 & 918 \\ 1026 & 1026 & 1080 & 1107 \end{bmatrix}$$

Convolutional neural networks have three important characteristics: *sparse interactions*, *parameter sharing* and *equivariance to translation* [8]. Neural network layers normally use matrix multiplication to describe the interaction between each input node and output node. Every output node has an interaction with each input node. But in CNNs there are typically sparse interactions. This is done by making the kernels smaller than the input. Therefore not all the input nodes have an interaction with the output node. As an example, the kernel has a width of 5 then only the next 5 output nodes are affected by that input node. Using an MLP for example would have affected all the output nodes instead of only 5 output nodes.

The definition of parameter sharing is that a parameter is used for more than one neuron in the model. An example of this are the kernels, which are used at every position of the input [8]. Parameter sharing means that for every convolution operator only one set of parameters is used. This is a contradiction with the recently used layer called the locally connected layer that is used by Taigman et al. [73]. In this layer, there is no parameter sharing and therefore it could better extract features from images with multiple features.

The last important characteristic is the equivariance to translation. This means that if the input translates, the output translates in the same way. This means for images that the convolution creates a 2D map which represents where certain features appear in the input. The representation of the input will move the same way as the output does by the convolution.

A convolutional neural layer consists in general of three stages [8]. In the first stage, the layer performs several parallel convolutions in order to produce a set of linear activations. This is called the convolution part of the layer. In the next stage, the incoming activation goes through a nonlinear activation function. In general, this is the ReLU activation function [48]. In the third stage, the pooling function provides the output of the layer such that it can be used in the next layer. The pooling function helps to make the representation become invariant to small translations in the input. This is one of the advantages that a CNN has over the multi-layer perceptron. The MLP is not invariant to small translations if this network receives 2-D images as input [37]. Because the layer has a pooling function, the values of most of the pooled outputs are not changed if the input is a 2D image.

2.1.3 Inception module

The new trend in deep learning is to make the networks wider and deeper. Wider means that it has more neurons in each layer and deeper means that there are more layers in a network. The first drawback of making the networks bigger is that the network consists of more parameters to train. The networks can suffer in performance because it is more prone to overfitting. The other drawback is that if there are more parameters to train then there are more computational resources required.

Adding sparse connections to the networks is one of the ways that can solve the drawbacks [72]. Szegedy et al. suggest moving from fully connected to sparsely connected architectures [72]. These sparse connections are sparse weights of a fully connected layer except that there are zero weights in place. This reduces the computational resources that are needed because adding and multiplying of these zeros are not needed. They also stated that the numerical calculation of non-uniform sparse data structures is not efficient and therefore they introduced an intermediate step. In this intermediate step is extra sparsity added to the architecture but the calculations are still done on dense matrices. The *inception module* started as a hypothetical experiment by Arora et al. [5] but it proved its success in object recognition [23] and localization [24]. The main focus of the inception module is to make the intermediate step that is discussed before. This means that an optimal local sparse structure is found and produced by available dense components [72].

Figure 2.3 shows an example of the first inception module proposed by Szedegy et al. [72]. They added the pooling layer in the inception module because pooling achieved good results using the convolutional neural networks. If these modules were stacked upon each other then it was leading to a computational blow-up. To avoid this, Szedegy et al. proposed another inception module that is shown in figure 2.4. The 1x1 convolutional layers are added to get more dimensionality reduction. The 1x1 convolutions are first calculated and therefore used as reductions. Then the expensive 3x3 and/or 5x5 convolutions are calculated. Extra non-linearity is added to the architecture because the 1x1 convolutions layers also have a ReLU activation function. The inception architectures proposed by Szedegdy et al. [71] and He et al. [27] have more complex inception modules that are deeper and wider. This resulted in better performances in different computer vision task.

In this thesis, we want to specify the kernel dimensions for each inception module, therefore we first divide the inception modules into inception *branches*. These branches are divided into inception *components*. The division is shown in figure 2.5. If we are speaking of branch 1 then we mean the most left path from the previous layer to the concatenation component. Branch 2 is the path that is right of branch 1 etc. It is possible that a branch has multiple inception components. In this case, the first kernel dimension in the table is the lowest component in the figure. An example of the kernel dimensions is in table 2.1. In the example, branch 1 got one value 64. This means that the 1x1 convolution layer of branch 1 shown in figure 2.5 has 64 kernels. The second branch got the values (96,128), this shows that the lowest layer with the 1x1 convolution layer has 96 kernels and the second layer with the 3x3 convolution layer has 128 kernels. In figure 2.5 and table 2.1, it must be noticed that branch 4 has two components but has one kernel dimension. This is because the max pooling does not have a kernel dimension, the 32 is the kernel dimension of the 1x1 convolution layer.

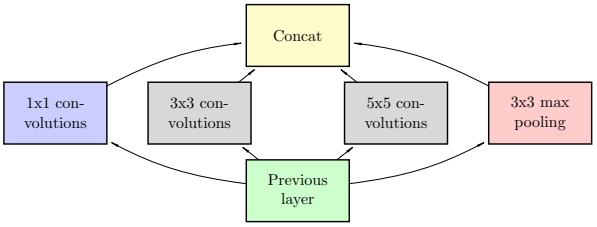

Figure 2.3: The naïve inception module proposed by Szedegy et al., which contains four branches [72]. From left to right: 1x1 convolution layer, 3x3 convolution layer, 5x5 convolution layer and 3x3 max pooling layer.

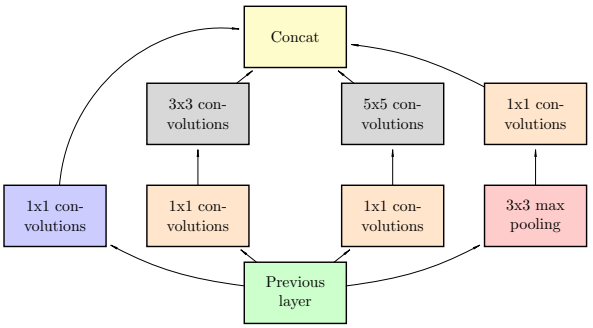

Figure 2.4: Revised inception module from GoogleNet, which is proposed by Szegedy et al. [72]. In comparison with the naïve version, Szegedy et al. added a 1x1 convolution layer before the 3x3 and 5x5 convolution layer [72]. After the 3x3 max pooling layer is also a 1x1 convolution layer added.

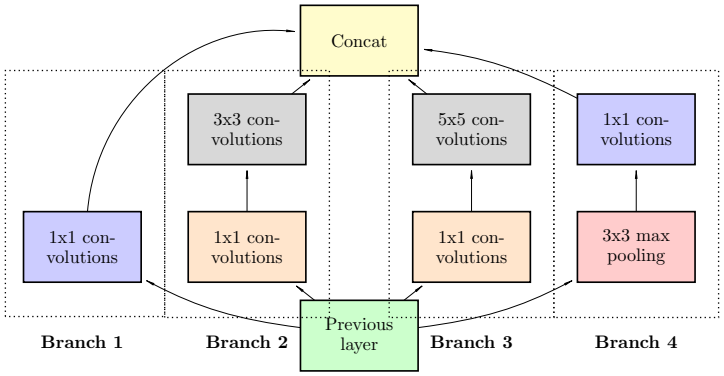

Figure 2.5: Inception module from figure 2.4, which is divided into four branches. A branch is defined as a path from the previous layer until the concatenation part and visualized as a dashed block. Together with table 2.1, the kernel for each part of this module is specific.

Branch 1	Branch 2	Branch 3	Branch 4
64	(96,128)	(16,32)	32

Table 2.1: Specification of the kernel dimensions of each branch of the inception module that is shown in figure 2.5. If there is more than one value for the kernel dimensions then it is specific from bottom to top. An example with figure 2.5: branch 2 has a kernel dimension of 96 for the 1x1 convolution layer and a kernel dimension of 128 for the 3x3 convolution layer. Note: max pooling layer never has a kernel dimension, because of this there is only one value in branch 4.

2.2 Gradient descent methods

The *gradient descent techniques* and *optimizer*, which are relevant for this thesis, are discussed in this section. First, we look at *gradient descent*, then secondly on an extension of gradient descent called *stochastic gradient descent*. Lastly, we will give a brief description of *RMSProp*, the optimizer that is used in our experiments.

Gradient descent was proposed in 1847 by Cauchy as a method to generic function minimization [12]. The function to minimize is often referred to as the *loss function* or *cost function* ($\mathcal{L}$). Input values and target output values are used to minimize a loss function for the model ($f(x)$). The input values and target output values are often in literature referred as examples [9]. These examples are referred to as $z = (x, y)$, where x is the input value and y the target output value [9]. In a supervised learning setting the function in equation 2.9 is often calculated, where $\hat{y}$ is the predicted output.

$$\hat{y} = f(x) \tag{2.9}$$

The loss function is defined as $\mathcal{L}(y, \hat{y})$. The loss function is to minimize the difference between the target output y and the predicted output $\hat{y}$ for the model ($f(x)$). The equation 2.9 and the loss function can be combined to get $\mathcal{L}(y, f(x))$. This equation shows that both of the arguments of example z are used. The model has weights and the loss function is minimized with respect to the weights w. Therefore the model is often written as $f_w(x)$. To minimize the loss function, the gradients of the weights are used

to find the direction of the steepest descent in the parameter space. The weight vector w is iteratively updated as in equation 2.10, where the gradient at time t, g_t, is defined as in equation 2.11. $\eta \in (0, 1)$ is the learning rate in which the step rate of the updates is defined and ∇_w is defined as the gradient with respect to w.

$$w_t = w_{t-1} - \eta * g_t \tag{2.10}$$

$$g_t = \nabla_w \mathcal{L}(f_w(x), y) \tag{2.11}$$

The loss function $\mathcal{L}(f_w(x), y)$ can be used in multiple different forms. For simplicity, we discuss regression. The loss function for regression is usually defined as the sum of squared errors:

$$\mathcal{L} = \frac{1}{2}\sum_{i}^{N}(\hat{y}_i - y_i)^2 = \frac{1}{2}\sum_{i}^{N}(f_w(x_i) - y_i)^2, \tag{2.12}$$

where N is the number of examples.

The purpose of training neural networks is to generalize the network for unseen test data. One of the problems that can occur with the trained neural networks is overfitting. *Overfitting* means that the networks also train the random noise and not only the underlying relationships in the data. The performance on the unseen test data can decrease because the network is using the noise. When the gradient of the function is followed accurately, the weights can possibly steer in *zero-gradient regions* like local minima and plateaus.

Stochastic gradient descent is an extension on gradient descent which approximates the gradient by considering a random ordering of the data [8]. The equation for the stochastic gradient descent is shown in 2.13. This gradient descent is called *stochastic* because it approximates the gradient and because it is using a random ordering of the data. An advantage of stochastic gradient descent over gradient descent is the efficiency because it is only used on a subset of the dataset.

$$\mathcal{L} = \frac{1}{2}\sum_{i}^{M}(f_w(x_i) - y_i)^2, \tag{2.13}$$

where $1 \leq M < N$.

Stochastic gradient descent has problems with parameter spaces that have surface curves that are more steep in one direction than in another direction [69]. In these cases, stochastic gradient descent makes great progress in the steep curves but in the other direction the progress is low. Momentum proposed by Quan et al. is a method that helps with speeding up in the other direction and the method also helps with damping the steeper direction [56]. Momentum works by taking a fraction of the previous update step into the current update step. The (stochastic) gradient descent update rule changes from equation 2.10 into:

$$v_t = \gamma * v_{t-1} + \eta * g_t, \tag{2.14}$$

$$w_t = w_{t-1} - v_t. \tag{2.15}$$

In this thesis, the RMSprop optimizer is used to minimize the loss function. This optimizer was first introduced in a lecture by Tieleman and Hinton in 2012 [75]. RMSprop adapts the gradient update step according to a root of the running average of the squared gradients. The equation of 2.10 is changed into the following equation:

$$w_t = w_{t-1} - \frac{\eta}{\sqrt{MA(g^2)_t + \epsilon}} * g_t, \tag{2.16}$$

$$MA(g^2)_t = \beta * MA(g^2)_{t-1} + (1 - \beta) * g_t^2, \tag{2.17}$$

where $MA(g^2)$ is the moving average of the squared gradients, β is the decay parameter and ϵ is the fuzzy factor. The decay parameter β corresponds to the ratio that the moving average of the squared gradient of the previous time step is taken in this update step.

2.3 Backpropagation

Neural networks can be trained using *backpropagation*. This algorithm was reinvented by Rumelhart et al. in 1985 [58]. Backpropagation is used for computing the gradients of the loss function $\mathcal{L}$ by recursively applying the *chain rule*. The chain rule is explained in extra 2.3.1.

Extra 2.3.1: Chain rule

Leibniz used the chain rule for the first time in 1676 [52]. In calculus, it is used to calculate the derivative of a composition of two or more functions. If $f(x)$ and $g(x)$ are the two functions then the composition is written as

$$F(x) = f(g(x)).$$

The derivate of this composition can be written as

$$F'(x) = f'(g(x))g'(x).$$

Leibniz used another notation for the chain rule:

$$\frac{dy}{dx} = \frac{dy}{du} \cdot \frac{du}{dx},$$

which is related to the previous notation. If $y = f(u)$ and $u = g(x)$, then

$$\frac{dy}{dx} = \frac{dy}{du} \cdot \frac{du}{dx} = F'(x) = f'(g(x))g'(x).$$

The gradient is a vector of partial derivatives. In equation 2.18 is the partial derivate of the loss function $\mathcal{L}$ with respect to weight w. Using this equation, we can calculate how the output of $\mathcal{L}$ changes in relation with weight w for every training sample. This partial derivate denotes how fast the cost changes when the weights are changing. The goal of the backpropagation is to calculate the partial derivates of the function with respect to all the weights and the biases.

$$\frac{\partial \mathcal{L}}{\partial w} \tag{2.18}$$

2.4 Regularization

Lately, regularization is becoming important in deep neural networks [8]. The deep learning book of Goodfellow et al. nicely defines regularization as: *any modification we make to a learning algorithm that is intended to reduce its generalization error but not its training error* [8]. In this subsection, a brief overview of the regularizers that are important in this thesis is given.

Dropout proposed by Hinton et al. is an example of such regularizer [28]. The idea of dropout is to randomly drop nodes of the neural network with all their incoming and outgoing connections during the training phase. The dropout is always specified as a ratio between 0 and 1, which determines the probability neurons of that layer are randomly dropped during that epoch. Dropout is described in more detail in the paper by Srivastava et al. [65]

Batch normalization is another regularizer that is introduced in 2015 by Ioffe and Szegedy [31]. They wanted to reduce the *internal covariance shift* of the deep learning methods. The distribution of the activations of hidden nodes by a network can change, this is known as the internal covariance shift. One of the advantages due to batch normalization is that the learning rate can be set higher [31]. This is because it is less dependent on the gradient flow of the initial parameters. A second benefit is: there is less need for dropout because the batch normalization also regularizes the model.

The normalization is set before the activation function and after the calculation of the weights. Equation

2.1 is changed into:

$$a_j = BN(\sum_i w_i^j x_i). \tag{2.19}$$

The batch normalization is defined as BN. The normalization is done for each activation separately, by making it mean zero and unit variance [31]. The layer with input $x = (x_i \dots x_n)$ is independently normalized with the following equation:

$$\hat{x_i} = \frac{x_i - E[x]}{\sqrt{(Var[x] + \varepsilon)}}, \tag{2.20}$$

where E is the expectation over the dataset or subset. Var is the variance over the dataset or subset and ε is the fuzzy factor. The expectation and the variance are also updated with a moving average on the values. Ioffe and Szegedy introduced an extra variable for the updating of the moving averages, the decay Γ. The ratio of the decay specifies how much the moving average is updated by the current moving average and by the moving averages in the past. The equation is:

$$MA(x_i) = \Gamma * MA(x_{i-1}) + (1 - \Gamma) * MA(x_i). \tag{2.21}$$

Ioffe and Szegedy note that with this normalization the representation of the features can be lost [31]. They solve this by introducing for each input x^n, a pair Ψ, Υ. The Ψ is called the scale factor and Υ is called the shift factor, which they use to represent the features again. They use this to modify $\hat{x_i}$ in equation 2.20 to the following equation:

$$y_i = \Psi x_i + \Upsilon. \tag{2.22}$$

Ψ and Υ are trained with gradient descent. Ψ and Υ are initialized with one and zero.

2.5 Transfer learning

Deep learning methods seem to be well suited for representing high-level abstractions [7]. In each layer of deep methods, new feature abstractions are added and the low-level abstractions are reused and composed. Some of the features that are useful in one domain can also be useful in another domain, which make deep methods well suited for transfer learning. One of the fields where this is used is in handwriting recognition. A research by Ciresan et al. is one of the examples that used transfer learning in handwriting recognition [17]. They found out that a deep model that is pre-trained on Chinese characters is better at recognizing uppercase Latin letters. It takes the low-level features that are already pre-trained on the Chinese characters and uses these features to train the high-level features that are needed for the recognition of uppercase Latin letters. In 2016, Tang et al. used a convolutional neural network that was pre-trained on modern Chinese characters and fine-tuned on historical Chinese characters [74]. This also resulted in better results. Training of a pre-trained model saves time and it requires less computer power because the initial values of the weights are better than the random initialization.

A lot of researchers are using models that are pre-trained on the *ImageNet ILSVRC 2012* dataset and are achieving good results, a research by Oquab et al. is an example [51]. They use the model proposed by Krizhevksy et al. and pre-train it on the ImageNet dataset and then train it on the *Pascal VOC 2012* dataset [36, 51]. Van de Wolfshaar et al. used the same model and dataset to pre-train and train it on the gender classification dataset *Adience*. They also achieved good results by first pre-training the model and later training it on another data set [76]. On other image classification datasets impressive results were also obtained [22, 60]. The ImageNet pre-trained models achieve good results not only on the image classification datasets but also on action recognition [61], human pose estimation [11], optical flow [78] and image captioning [21, 32].

Huh et al. were wondering why the ImageNet dataset is good for pre-training of deep neural networks [30]. They tested the models that were pre-trained with the ImageNet dataset on three different tasks: object detection, action classification and scene classification. They defined a couple of criteria that used to test which aspect of the pre-training with the ImageNet dataset is important. The three criteria were: the number of examples per class, the number of classes that are used in pre-training and the availability of target classes in pre-training. The consensus by Huh et al. is that the key to learn

generalizable features, is a large number of training examples and classes. Furthermore, they mention in their paper that it is not necessary to have the target class in pre-training [30]. This is in contradiction with the findings that Yosinki et al. showed [82]. Yosinki et al. showed in their results that using the same classes in pre-training improves the performance.

2.6 Face recognition

The error rates in face recognition have decreased the last twenty-five years by three orders of magnitude [55]. One of the reasons for the decrease is the increase of development in machine learning methods. In this section, we describe two popular machine learning directions in face recognition. The first direction are the machine learning methods where handcrafted features or dimension reduction is used, called *shallow learning* methods. The overview of these shallow learning methods in face recognition is described in subsection 2.6.1. The second direction are machine learning methods which are using raw input into a neural network with multiple hidden layers and is called *deep learning*. An overview of the deep learning methods in face recognition is given in subsection 2.6.2. Another reason for the decrease of the error rate is the increase of the size of datasets for face recognition. An overview of these datasets with celebrities can be viewed in subsection 2.6.3.

2.6.1 Shallow learning

Most face recognition methods are using handcrafted features [73]. These methods are called shallow learning methods. These handcrafted features are used to extract a feature representation of the face template in the image. Examples of handcrafted features that can be used for face recognition are *SIFT* [16, 44, 63], *Gabor* [42], *LBP* [3] and *HOG* [19, 64]. When the features are extracted from the images, an overall face description is made from the features by using a pooling function such as the *Fisher vector* [54]. But Taigman et al. stated that such methods are often sensitive for intra-person variations like lighting, expression, occlusion and aging [73, 85].

Another way how shallow learning methods are used is with dimensionality reduction. Face recognition is performed in a high-dimensional space, which means a lot of computation is needed to find the match especially when the size of the images increases. Dimensionality reduction techniques can be used to solve this problem. Kirby and Sirovich were one of the first researchers who tried this. They used the *Eigenfaces* algorithm [35], which uses *principal component analysis* (PCA) to reduce the dimensionality. PCA works well for face recognition because the template of faces in images have similar structure and therefore the features can be represented in a lower-dimensional space [35].

In 2002, *independent component analysis* (ICA) was introduced by Barlett et al. as a more powerful tool for face recognition [6]. ICA can be seen as a generalization of PCA but it has advantages in comparison with PCA [6]. It extracts a better characterization of the data. The vectors found by ICA reduce the reconstruction error and extract discriminant features that also consider high-order statistics [2].

Another good alternative for PCA is linear discriminant analysis (LDA) for Face Recognition [45, 46]. The aim of LDA is to find a base of vectors providing the best discrimination amongst the classes. LDA performs better than PCA when the training set is bigger [46]. Furthermore, LDA overcomes the limitations of PCA by using the Fisher's linear discriminant criterion [13]. This criterion gives a better separation between different classes than the criterion used by PCA. A further development of LDA is *discriminant common vectors*, which is introduced by Cevikalp et al. in 2005 [13]. Discriminant common vectors eliminates the differences of samples in the classes and therefore extract the common properties in each class. This common vector is obtained by removing all the features that are not representable for the class [13]. They obtained better results than the eigenfaces algorithm because the eigenfaces could not take into account the light conditions.

One of the issues with the aforementioned algorithms is that they are commonly used for linear problems whereas face recognition is a non-linear problem. Neural networks provide the means to deal with non-linear problems. One of the first studies that addressed this solution was Cottrell & Fleming in 1990

[18]. However, their experiments showed neural networks did not provide better results than eigenfaces. In 1997, Lin et al. used a *probabilistic decision based neural network* for the recognition of faces [40].

2.6.2 Deep learning

The introduction of deep learning methods in face recognition gave better results than the methods before them. These methods were better able to represent the features of face templates. This was due to the size of the networks and the introduction of convolutional neural networks (CNN). Deep learning methods have more than one hidden layer in opposed to most of the neural networks that were used before. For example, Cottrell & Fleming used in 1990 only a one hidden layer neural network and this one was not suitable to represent all the features of the face template [18]. The multiple layered neural networks are better suitable to extract these features. A benefit that the deep learning methods have over the shallow methods is that they are using raw input images instead of the handcrafted features.

The introduction of convolutional neural networks had an impact on image recognition because the convolutional layers are better suitable to represent spatial features [38] and therefore also better suitable for face recognition. One of first researches that was done on face verification is by Chopra et al. in 2005 [15]. They were using *energy-based models* (EBM) to learn a similarity metric. Because they did not have that many training examples at that time, they were using distance-based methods, like the Euclidean distance metric. With these methods, the similarity between the input image and the test image is computed. Chopra et al. were using a *siamese architecture* with two CNNs that resulted in an energy-based model [15].

One of the best results lately was by Taigman et al. in 2014 [73]. They proposed a deep learning architecture called DeepFace. There are two essential parts of their architecture. The first part is that they are using three locally connected layers without weight sharing instead of the default convolutional layers [73]. The locally connected layer does not share weights and the kernels are localized, therefore the features in the different regions of a face are better extracted. The second part is that they also improved the alignment of the faces with a 3D model of the faces. They decreased the error rate with more than 27% and received the state of the art at that moment on the *labeled faces in the wild validation set* (LFW) [29].

In the same year, Zhu et al. proposed an architecture to use multiple views of a face template and make a frontal view of it [85]. This frontal view is made out of images that have different poses, expressions and lighting. Therefore their architecture is more robust for classifying faces from images. They were using a three-layered CNN with a fully connected layer [85]. Sun et al. proposed in 2014 a CNN with *DeepID* features and in the same year the *DeepID2* features for the recognition of faces [66, 67]. The idea of the DeepID feature layer is that it is a fully connected layer that receives input from the last convolutional layer and the layer before the last layer. These two inputs are flattened to one input. Lately Sun et al. proposed the *DeepID2+* architecture [68], which is the improved one of DeepID2, where Sun et al. increased the dimensions of hidden representations and added supervision to the first convolutional layers. In DeepID2+ the feature maps are larger than in DeepID2. DeepID2+ has got a verification accuracy of 99.47 % on LFW, which outperforms the human verification with 0.27%.

Lately, the trend is to make the neural networks even deeper than they were before. One of the networks that started with this is the *GoogleNet* or also called *Inception V1* proposed by Szegedy et al. [72]. They obtained very good results with their 22-layered neural network on for example the ImageNet 2014 challenge. They also introduced the inception module, which is nowadays often used. For the 2015 ImageNet challenge, He et al. used an even deeper network, called Inception Residual Network V2, with 152-layers and so-called *residual connections* [27]. A residual connection is not connected directly to the next layer but skips one or more layers. This resulted in good results. Following the trend of making the networks deeper, Schroff et al. proposed the architecture called *FaceNet* [59]. In this architecture, extensions on the inception module are made. FaceNet achieved an even higher accuracy than the architectures that were using the DeepID methods.

2.6.3 Celebrities datasets

The improvement on face recognition tasks also has to do with the increase of available training datasets. In this thesis we are going to use a couple of these training datasets. In this subsection an overview of these celebrity face datasets is given.

Table 2.2: Collection of face recognition datasets, which vary in: number of images, classes, availability and year.

Dataset	# Number of images	Classes	Availability	Year
CelebA [43]	202,599	10,177	public	2015
Cross-Age Celebrity Dataset [14]	163,446	2,000	public	2015
IMDB + wikipedia Face dataset [57]	523,031	20,284	public	2015
LFW [29]	13,233	5,749	public	2016
CelebFaces [67]	202,599	10,177	public	2014
Facescrub [50]	106,863	530	public	2014
CASIA WebFace [81]	494,414	10,575	public	2014
MegaFace [49]	4.7M	672,000	public	2017
FaceBook [73]	4.4M	4,030	private	2014
Ms-celeb-1M [26]	8.5M	99,892	public	2016
Google [59]	200M	8M	private	2015

The size of the datasets has been increased by the introduction of web applications like Facebook, Google, Instagram and Youtube but sadly the datasets are not always publicly available. An overview is listed in table 2.2. Big companies like Facebook and Google are using the datasets mostly for training their own algorithms. An example is FaceNet that Schroff et al. proposed in 2015 [59]. This algorithm uses the Google dataset for training their algorithm and they achieved good results on the LFW validation data set. A second example is DeepFace, that was proposed by Taigman et al. that uses the dataset from Facebook [73].

Fortunately, there are researchers [14, 29, 43, 57, 67] that made their datasets available for training and testing different methods. Also, the introduction of various face verification and identification challenges has a big impact on the increased results on this problem. An example is the *MS-Celeb-1M: Challenge of recognizing one million celebrities in the real world* [26] developed by Microsoft. They gave the researcher the chance to use a dataset of 8.5 million images to train and test the models. Another example of a challenge is called the *MegaFace benchmark: 1 million faces for recognition at scale* [33]. This challenge gave teams from different companies and universities the chance to compete with each other on two different challenges with two different datasets. One challenge was that participants could use their own training dataset and verify it on the datasets that are given by MegaFace [33]. The second challenge was to show which method is the best when the training set is equal [49]. So the organizers of MegaFace gave the training set and the validation/test set and the participants could also compete with each other.

Chapter 3

Methods

3.1 Models

In this thesis two deep neural network architectures are used. The first architecture was proposed in 2015 by Szegedy et al. and is called GoogleNet or Inception V1 [72]. This architecture is explained in subsection 3.1.1. The second architecture is the winner of the ImageNet Large-Scale Visual Recognition Competition in 2015 (ILSVRC'15) and is called Inception Residual Network V2 or, shorter, Resnet V2. This architecture was proposed by He et al. [27]. In subsection 3.1.2 the Resnet V2 is explained.

3.1.1 Inception V1

The paper 'going deeper with convolutions' by Szegedy et al. [72] introduced GoogleNet. The GoogleNet architecture was the winning architecture of the ILSVRC'14 challenge. This was one of the first architectures that used more than 15 layers and achieved good results. The architecture consists of multiple convolutional layers, pooling layers, inception layers and fully connected layers. Table 3.1 shows the architecture with the kernels, strides, depth and parameters for each layer. The model is not an exact copy of the GoogleNet proposed by Szegedy et al. [72]. The default input image that they are using is 224 by 224 in comparison with the 160 by 160 that is used in this thesis. The reason for down sizing the image is to consume less computing power and it costs less time to train models. The purpose of this thesis is not to improve the accuracy of the GoogleNet but to compare different pre-trained models with the same settings, thus it will not have an influence on the final comparison.

Figure 3.1 shows the schematic overview of the inception module that is used in the GoogleNet. The explanation of the inception module has already been given in subsection 2.1.3. The architecture has a total number of 22 trainable layers. Between the last inception module and the dropout layer is a *global average pooling* layer. Instead of using this global average pooling, there often is used a fully connected layer, like is done in the VGG networks [62]. But it is pointed out by Lin et al. [39], that it is better to use a global average pooling layer. One of the reasons is that there are no parameters in this pooling layer and therefore no chance to get overfitting in this layer. There is a linear layer, often called the *bottleneck*, which is initialized before the softmax layer. This layer is added to easily adapt the model to another validation or testing dataset. The architecture is divided into five different blocks and is shown in table 3.1. A block consists of n layers. For example block 1 of the Inception V1 architecture has three convolutional layers and two max pooling layers. The division in the blocks is done for later use in Chapter 4. In the experiments, we will investigate the difference between training and freezing the different number of blocks.

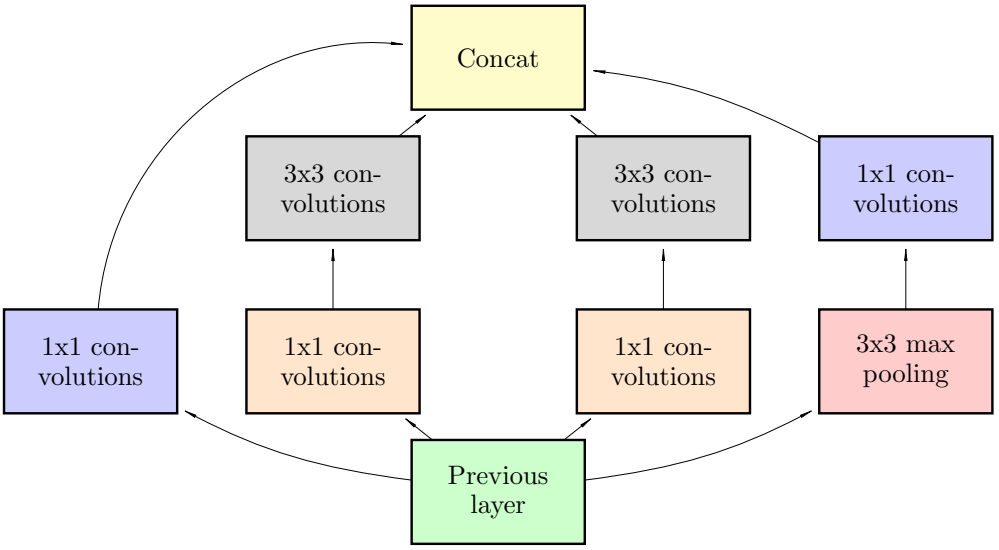

Figure 3.1: Modified version of the inception module (figure 2.4) proposed by Szegedy et al. [72]. They use an image size of 224x224, but in this research, it is set to 160x160. Because of this, the 5x5 convolutional layer is modified to a 3x3 convolutional layer.

Table 3.1: GoogleNet architecture with inception modules by Szedegy et al. [72]. The architecture is divided into five blocks. The kernel dimensions of the inception modules are specified in the branch 1, 2, 3 and 4. A branch is the path from the previous layer to the concatenation. Figure 3.1 shows the branches. An explanation how the kernel dimension can be extracted from the table is given in subsection 2.1.3. Note: the original GoogleNet architecture is modified because the image size is changed from the original 224x224 to 160x160.

Block	Type	Depth	Filter / stride	Output size	Branch 1	Branch 2	Branch 3	Branch 4	Params
1	Convolution	1	7 x 7 / 2	80 x 80 x 64					9K
1	Max pool	0	3 x 3 / 2	40 x 40 x 64					
1	Convolution	2	3 x 3 / 1	40 x 40 x 192					112K
1	Max pool	0	3 x 3 / 2	20 x 20 x 192					
2	Inception (3.1)	2		20 x 20 x 256	64	(96,128)	(16,32)	32	155K
2	Inception (3.1)	2		20 x 20 x 480	128	(128,192)	(32,96)	64	340K
2	Max pool	0	3 x 3 / 2	10 x 10 x 480					
3	Inception (3.1)	2		10 x 10 x 512	192	(96,208)	(16,48)	64	363K
3	Inception (3.1)	2		10 x 10 x 512	160	(112,224)	(24,64)	64	425K
3	Inception (3.1)	2		10 x 10 x 512	128	(128,256)	(24,64)	64	486K
3	Inception (3.1)	2		10 x 10 x 528	112	(144,288)	(32,64)	64	572K
3	Inception (3.1)	2		10 x 10 x 832	256	(160,320)	(32,128)	128	803K
3	Max pool	0	2 x 2 / 2	5 x 5 x 832					
4	Inception (3.1)	2		5 x 5 x 832	256	(160,320)	(32,128)	128	978K
4	Inception (3.1)	2		5 x 5 x 1024	384	(192,384)	(48,128)	128	1346K
5	Avg pool	0	5 x 5 / 1	1 x 1 x 1024					
5	Dropout (80%)	0		1 x 1 x 1024					
5	Linear	1		1 x 1 x 10575					11k
5	Softmax	0		1 x 1 x 10575					

3.1.2 Inception residual network V2

During the ILSRVC 2015 classification task, the Inception Residual Network V2 was introduced by He et al. [27]. Using this network, they achieved the best result on this competition with an error of 3.57%. The default image input size is 224x224 but our image input size is 160x160. This is because we modified the depth and the kernels of the architecture to be suitable for the 160x160 input size. So this is not an exact copy of the architecture proposed by He et al. [27]. The modified network consists of 134 layers, which was very deep at that time. This architecture has multiple convolutional layers, pooling layers, inception layers and fully connected layers. Table 3.2 presents the architecture with the kernels, strides,

depth, repeat and parameters for each layer. Here 'repeat' means how many times the inception module is repeated. As an example, inception A is repeated ten times after each other. The Resnet V2 has six different types of inception modules. They are shown in figures 3.2, 3.3, 3.4, 3.6 and 3.7, 3.8. The kernel dimensions of the branches for the inception modules are shown in table 3.2. This table presents the division of the architecture in eight different blocks. This is done for later use in the experiments in Chapter 4, where the training and freezing of different blocks is compared with each other.

Table 3.2: Inception Resnet V2 architecture with inception modules by He et al. [27]. The architecture is divided into seven blocks, which is used in Chapter 4. The kernel dimension of the inception modules are specific in the branch 1, 2, 3 and 4. An explanation, how the kernel dimension can be extracted from the table is given in subsection 2.1.3. The repeat column is added to specify how many times layers are repeated. Note: the original Inception Resnet v2 architecture is modified because the image size is changed from the original 224 by 224 to 160 by 160.

Block	Type	Repeat	Depth	Filter / stride	Output size	Branch 1	Branch 2	Branch 3	Branch 4	params
1	Convolution		1	3 x 3 / 2	79 x 79 x 32					896
1	Convolution		1	3 x 3 / 1	77 x 77 x 32					9K
1	Convolution		1	3 x 3 / 1	77 x 77 x 64					18K
1	Max pool		0	3 x 3 / 2	38 x 38 x 64					
1	Convolution		2	3 x 3	36 x 36 x 192					143K
1	Max pool		0	3 x 3 / 2	17 x 17 x 192					
2	Inception (5a, 3.4)	1	3		17 x 17 x 320	96	(48,64)	(64,96,96)	64	268K
3	Inception (A, 3.2)	10	3		17 x 17 x 320	32	(32,32)	(32,48,64)		1229K
4	Inception (6a, 3.6)	1	3		8 x 8 x 1088	384	(256,256,384)			2663K
5	Inception (B, 3.3)	20	3		8 x 8 x 1088	192	(128,160,192)			22522K
6	Inception (7a, 3.7)	1	3		3 x 3 x 2080	(256,384)	(256,288)	(256,288,320)		3878K
7	Inception (C, 3.8)	10	3		3 x 3 x 2080	192	(192,224,256)			18311K
8	Convolution		1	1 x 1	3 x 3 x 1536					31996K
8	Avg pool		0	3 x 3 / 1	1 x 1 x 1536					
8	Dropout (80%)		0		1 x 1 x 1536					
8	Linear		1		1x1 x 10575					11k
8	Softmax		0		1x1 x 10575					

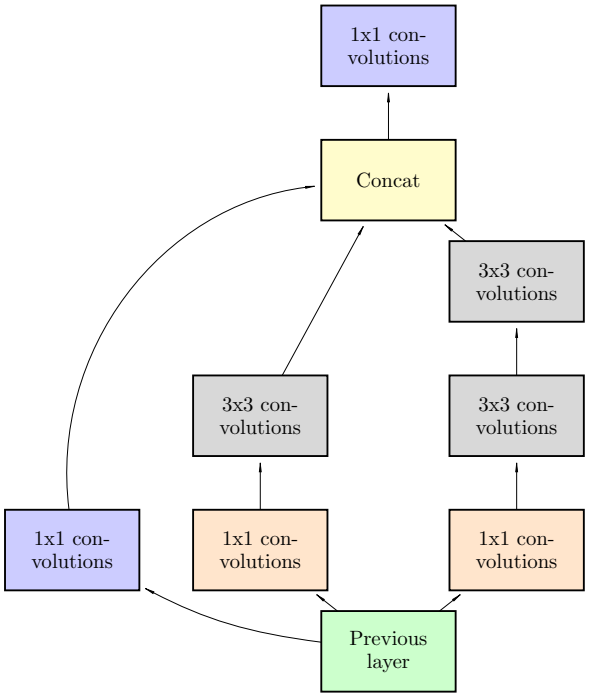

Figure A: Inception Resnet A module

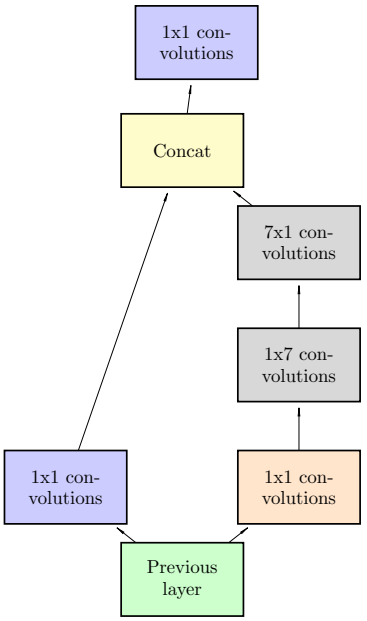

Figure B: Inception Resnet B module

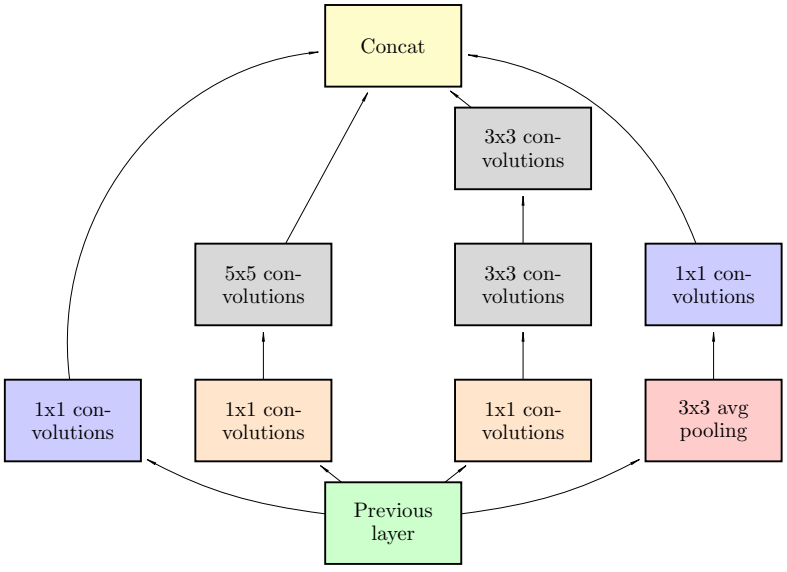

Figure C: Inception 5a module

Figure 3.5: Inception Resnet modules used in the Inception Resnet V2 by He et al. [27]. It is slightly modified to make it suitable for image size 160 by 160 in comparison with the original 224 by 224.

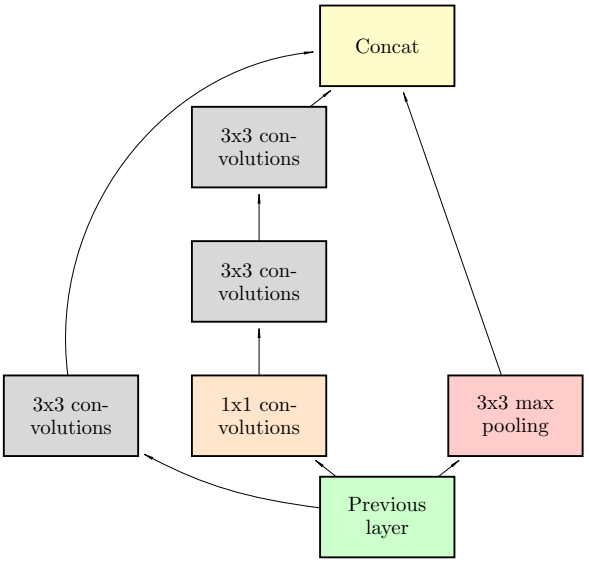

Figure A: Inception Resnet 6a module

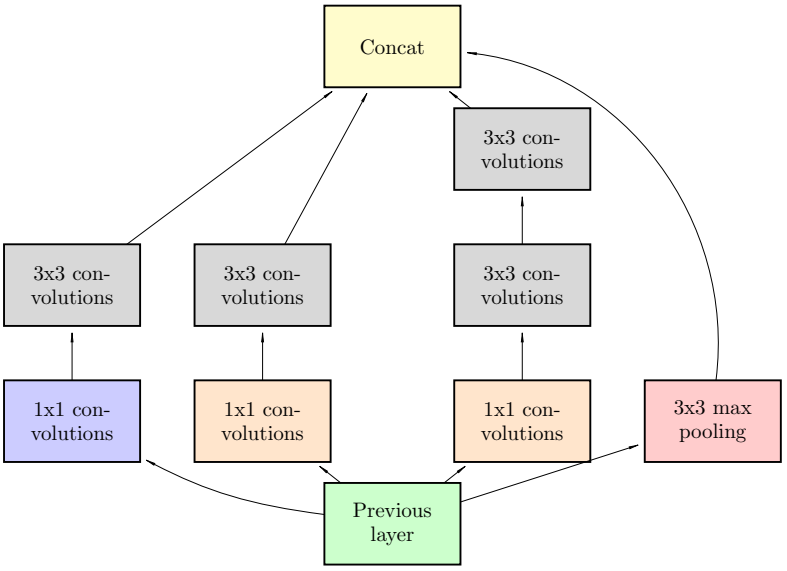

Figure B: Inception Resnet 7a module

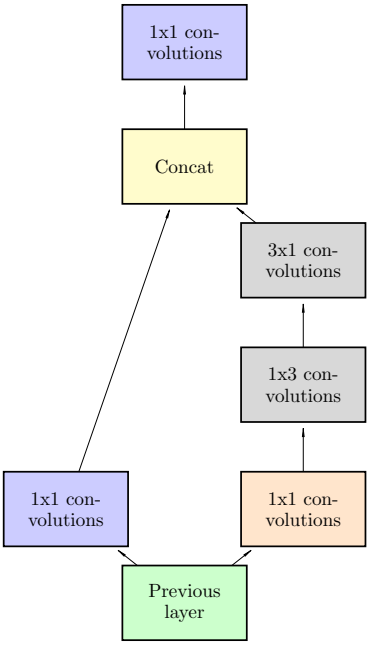

Figure C: Inception Resnet C module

Figure 3.9: Inception Resnet modules used in the Inception Resnet V2 by He et al. [27]. It is slightly modified to make it suitable for image size 160 by 160 in comparison with the original 224 by 224.

3.1.3 Loss functions

A combination of categorical cross entropy and center loss is used as the loss function to optimize the architectures. In the next two subsections the two loss functions are explained.

Categorical cross-entropy

The cross-entropy is a loss function which is used to quantify the difference between two probability distributions. This difference is calculated for every point in the dataset or batch. The formula to calculate the probability difference is given in equation 3.1. An example is defined as (x, y), where x is

the input value and y is the true value. $\hat{y}$ is the predicted value by the network, N is the number of examples and C is the number of class labels.

$$\mathcal{L}_{cross}(y, \hat{y}) = -\sum_{t=1}^{N}\sum_{i=1}^{C} y_i^t \cdot log(\hat{y}_i^t) \tag{3.1}$$

As an example, y has three classes with the following distributions: $[0, 1, 0]$. Which means that y has value 1 for the second class. Furthermore x has the predicted values $\hat{y}$: $[0.1, 0.5, 0.4]$. The $\mathcal{L}_{cross}(y, \hat{y})$ for point x is calculated as follows: $\mathcal{L}_{cross}(y, \hat{y}) = -((0*log(0.1))+(1*log(0.5))+(0*log(0.4))) = -log(0.5) \approx 0.30$. We looked at $N = 1$ but if there are more examples then they are summed to get the total loss.

Center loss

Instead of using only the categorical cross-entropy loss, Wen et al. proposed in 2016 an extra loss function called *center loss* [80]. The idea is to improve the discriminative power of the deeply learned features by minimizing the intra-class variations. The center loss function is formulated as follows:

$$\mathcal{L}_{center}(y, \hat{y}) = \frac{1}{2}\sum_{t=1}^{N}\sum_{i=1}^{C}(\hat{y}_i^t - c_{y_i}^t)^2, \tag{3.2}$$

with c_{y_i} is the y_ith class center of the features and N is the number of examples. The idea is that for the training dataset, the average features for each class in each iteration are calculated. Wen et al. noticed that this did not obtain the intended result. To solve this problem, Wen et al. made two modifications. The first modification is to update the centers based on a mini-batch instead of the entire dataset. For the second modification Wen et al. introduced one new variable α and the δ-function. α is used to control the learning rates of the centers and the δ-function is a boolean that returns 1 if the condition is true and 0 if the condition is false. The update function of the class center is defined as follows:

$$\Delta c_j(y, \hat{y}) = \frac{\sum_{i=1}^{N} \delta(y_i = j) \cdot (c_j - \hat{y}_i)}{1 + \sum_{i=1}^{N} \delta(y_i = j)}. \tag{3.3}$$

The new center of each class is defined as:

$$c_j^{t+1} = c_j^t - \alpha \cdot \Delta c_j^t, \tag{3.4}$$

with $\alpha \in [0, 1]$. For the total loss function Wen et al. introduce λ to balance the two loss functions. The final function is given in equation 3.5. If λ is set to 0 then the original categorical cross-entropy function is used.

$$\mathcal{L} = \mathcal{L}_{cross} + \lambda\mathcal{L}_{center} \tag{3.5}$$

3.2 Evaluation methods

In the experiments a couple of evaluation methods are used. In this section we describe which evaluation methods are used and how they are calculated. The *accuracy*, *ROC curve*, *area under the curve*, *identification rate* and *paired t-test* are the five evaluation methods. First we will start with the accuracy, then an explanation of the ROC-curve and at last the area under the curve (AUC) is explained.

3.2.1 Accuracy

To explain the accuracy, we introduce the confusion matrix. The confusion matrix is an approach that can be used to determine how many predicted values ($\hat{y}$) are the actual values (y). The confusion matrix is schematically visualized in table 3.3. For a binary classifier often a threshold is used that determines if the predicted value is positive or negative. This is done by using a boolean function that if the predicted

value is above the threshold then it is a positive value. If it is below the threshold then it is a negative value.

Table 3.3: Overview of a confusion matrix. The actual value is in this research specified as y and the prediction value is $\hat{y}$.

		Actual value (y)	
		Positive value	Negative value
Predicted value ($\hat{y}$)	Positive value	True positive (TP)	False positive (FP)
	Negative value	False negative (FN)	True negative (TN)

The confusion matrix is used to derive the accuracy. The accuracy is calculated by adding the true positive with the true negative and then dividing it over all samples. It is multiplied by 100% to make it percentages. The equation is as follows:

$$accuracy = \frac{TP + TN}{TP + FP + FN + TN} * 100\%. \tag{3.6}$$

The *true positive rate* (TPR) and *false positive rate* (FPR) can also be calculated from the confusion matrix:

$$TPR = \frac{TP}{TP + FN}, \tag{3.7}$$

$$FPR = \frac{FP}{FP + TN}. \tag{3.8}$$

The TPR is also known as the *sensitivity* and the FPR is known as *fall-out*. From the FPR, the true negative rate (TNR) can also be calculated by $TNR = 1 - FPR$. This is also known as the *specificity*. The false negative rate (FNR) can be calculated by using the TPR: $FNR = 1 - TPR$.

3.2.2 ROC-curve

The second metric is called *receiver operating characteristic curve* or shortly ROC-curve. This is a method to evaluate a binary classifier by making a graphical plot. Figure 3.10 shows an example of a ROC-curve. The *true positive rate* (TPR) and *false positive rate* (FPR) are plotted against each other using the threshold or cut-off point. This threshold determines if a predicted value is positive or negative. If the predicted value is higher than the threshold then it is positive and if it is lower than the threshold then it is negative. So if the threshold is set low then there more positive predicted values than negative predicted values. This results in a cut-off with high TPR and high FPR. If the threshold is set high then there are more negative predicted values than positive predicted values. This results in a cut-off point with a low TPR and a low FPR. The ROC-curve shows the cut-off points for all the TPR and FPR combinaties. The TPR is set on the y-axis and the FPR is set on the x-axis. The area under the curve (AUC) is one of the evaluation metrics that can be used in the ROC curve. In figure 3.10 the AUC is specified in the blue area. It is the area between the ROC curve and the x-axis.

The accuracy also can be calculated using the TPR and FPR: $accuracy = \frac{TPR+(1-FPR)}{2} * 100\%$. If there is only a ROC-curve, then this formula can be used to calculate the accuracy. From this ROC-curve, the max value of the accuracy is calculated by going over all cut-off points. The max value of the acccuracy shows the threshold or cut-off point that gives the best TPR and FPR combination and therefore the best accuracy.

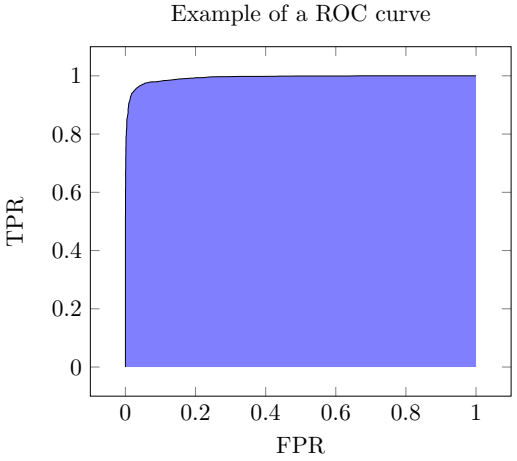

Figure 3.10: An example of the receiver operating characteristic curve (ROC), where the area under the curve is shown by the blue area. The y-axis is the true positive rate (TPR) and the x-axis is the false positive rate (FPR).

3.2.3 Identification rate

We use the identification rate as a measurement in an identification task. This is a 1:N matching problem, where N is the number of examples. The confusion matrix is again used for calculating the identification rate. But now only the positive actual value, positive predicted value and negative actual value are used. If we using this terminology again then the following equation defines the identification rate:

$$\text{identification rate} = \frac{TP}{TP + FN} * 100\%. \tag{3.9}$$

3.2.4 Significance test (paired t-test)

We use the *paired t-test* for testing the significance of results between two models. This test is used in statistics to compare population means where you have two samples in which observations in one sample can be paired with observations in the other sample. In our case, the accuracies on the models are the populations and we want to check if there is a significant difference between these accuracies. Like in other statistical tests, the t-test has a null hypothesis and an alternative hypothesis. The null hypothesis of the t-test assumes that the true difference between the mean of the paired samples is zero. The alternative hypothesis of the t-test assumes that the true difference between the means of the paired samples is not equal to zero.

To formulate it in our terminology, the null hypothesis assumes that the true difference between the means of the accuracies of two different models is zero. The alternative hypothesis assumes that the true difference between the means of the accuracies of two different models is not equal to zero. For the evaluation of the significance, we will use two variables: the t-distribution and the p-value. The significance is determined by looking at the value for the p-value. Statistical significance is determined by looking at the p-value. The p-value gives the probability of observing the test results under the null hypothesis. The lower the p-value, the lower the probability of obtaining a result like the one that was observed if the null hypothesis was true. Thus, a low p-value indicates decreased support for the null hypothesis.

Chapter 4

Experiments

4.1 Implementation details

To answer the research questions, which are defined in section 1.3, we evaluate the pre-trained models on different face recognition tasks. There are a couple of steps that these recognition tasks have in common. Figure 4.1 shows the experimental setup to evaluate which pre-trained model obtains the best results on the face recognition tasks. In the first step, there are five different datasets chosen that are used for pre-training the model. This is described in more detail in subsection 4.1.1.

In the second step, all the datasets are preprocessed. The methods are used in the preprocessing, are described in subsection 4.1.3. In the third step, all the models are pre-trained with the different datasets and the weights are saved. The settings for the pre-training are described in subsection 4.1.4. In the fourth step, the CASIA dataset is used to train the pre-trained models. This dataset is described in subsection 4.1.2. The training is always done on the layers of the last n blocks.

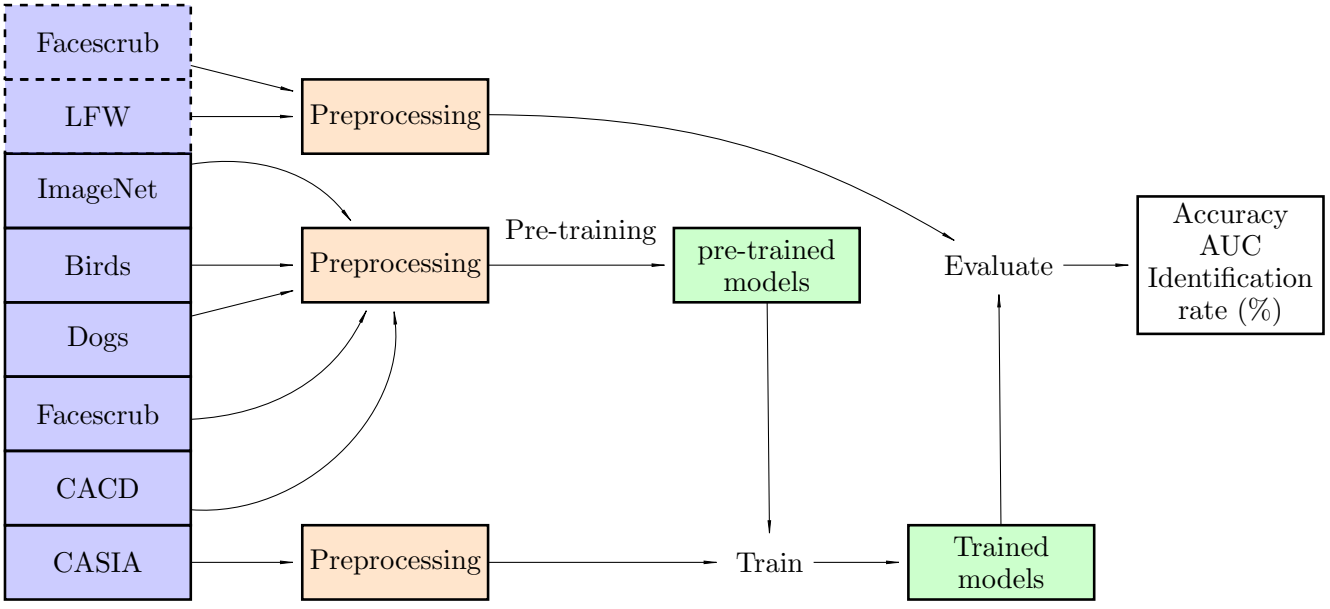

Figure 4.1: Experimental setup of the evaluation on the LFW and Facescrub dataset using various datasets for pre-training. In the first step the dataset is chosen. The second step is to preprocess the dataset to a usable format. The preprocess phase is the same for the pre-training, training and evaluation phases. The third step is to pre-train the models with the selected datasets and to train the last n blocks with the CASIA dataset. The last step is to evaluate the models with the three recognition tasks.

4.1.1 Datasets for pre-training

For the pre-training of the models five different datasets are used. An overview of the datasets is shown in table 4.1. The first dataset is Cross-Age Celebrity Dataset (CACD) that was made public by Chen et al. [14] and contains 163,446 images with 2,000 classes. This dataset is mostly used for training and testing on cross-age faces of celebrities. Cross-age means that the faces of the celebrities ranges over several years. The second dataset is called Facescrub and was made available by Ng et al. in 2014 [50]. It consists of 530 classes with 106,863 images. The third dataset is the well-known ImageNet dataset that has 1,000 different classes of different objects with 1,200,000 images [20]. The Stanford Dogs is a dataset that contains images of 120 different dogs [34]. This dataset was made public by Khosla et al. and contains 20,580 images. The last dataset contains 6,033 images of 200 different types of birds. This is called Caltech Birds 200 and was made public by Welinder et al. in 2010 [79].

This collection of datasets varies in size, statistics per classes and type of images. ImageNet, Facescrub and CACD are considered big datasets because they have more than 100k images. The Dogs and Birds dataset are considered small datasets. The statistics per class also varies. ImageNet, Facescrub and Dogs have more images per class compared with CACD and Birds. The type of images is the same for the Facescrub and CACD dataset. These are the datasets with face images. It must be noticed that ImageNet dataset does have faces in the dataset. Horizontal flipping and random cropping are used as data augmentation when pre-training the models.

Table 4.1: An overview of the datasets, used for pre-training, training or validation. The datasets vary in the number of images, classes, types and year.

Dataset	# of images	Classes				Type	Year
		Number	Mean	Max	Min		
Cross-Age Celebrity [14]	163,446	2,000	81	139	22	Faces	2015
ImageNet [20]	1.2M	1,000	1,278	1,300	102	Objects	2012
Stanford Dogs [34]	20,580	120	171	252	148	Dogs	2011
Caltech Birds 200 [79]	6,033	200	30	39	20	Birds	2010
Facescrub [50]	106,863	530	180	314	56	Faces	2014
LFW [29]	13,233	5.749	2	530	1	Faces	2016
CASIA WebFace [81]	494,414	10,575	42	769	2	Faces	2014

4.1.2 Dataset for training

The CASIA WebFace dataset is used for training the last n blocks of layers [81]. This dataset was gathered by Yi et al. in 2014. It has web collected images from celebrities. As is shown in table 4.1, the dataset consists of 494,414 images that are divided over 10,575 different person classes. From the dataset the images that also exist in the validation datasets are removed. If this dataset is compared with other datasets (200M [59], 8.5M [26] and 4.4M [73] images), that are shown in table 2.2, then it can be considered a small dataset.

4.1.3 Preprocessing

In the preprocessing phase, there are three steps. The first step is to detect the faces of the images. This is done by a method with a multi cascaded convolutional neural network proposed by Zhang et al. [83]. This method uses five landmarks (two eyes, two mouth corners and a nose) to detect the faces. If there are no faces detected in the image then the image is discarded from the dataset. The second step is cropping the images to 160x160 RGB images, to make all the images the same size. The last step of the preprocess phase is the normalization of the pixels in the images. Normalization is done by going over all values in the images and for every value, subtract the mean of all images with that value and then divide this value by the standard deviation of all values. This results in images that have zero mean and unit variance.

4.1.4 Detailed settings in CNNs

Table 4.2: Parameter settings for the pre-training and the training of the Inception V1 and the Inception Resnet V2. The parameters for the RMSprop can be found in section 2.2. The parameters that are needed for the center loss are explained in subsection 3.1.3. The parameters for the batch normalization are described in subsection 2.4.

Type	Parameter	Symbol	Inception V1	Inception Resnet V2
RMSprop	Decay	β	0.9	0.9
RMSprop	Momentum	γ	0.9	0.9
RMSprop	Epsilon	ϵ	0.0001	0.0001
RMSprop	Learning rate	η	50:0.1, 65:0.01, 80:0.001	90:0.1, 105:0.01, 120:0.001
Center loss	Center loss factor	λ	1e-4	1e-4
Center loss	Center loss alfa	α	0.9	0.9
Batch normalization	Decay	Γ	0.995	0.995
Batch normalization	Epsilon	ε	0.001	0.001
Network	Weight Decay	NA	5e-5	5e-5
Network	Batch size	NA	48	48
Network	Epoch	NA	120	80
Network	Early stop threshold	NA	0.001	0.001

The Inception V1 and Inception Resnet V2 are implemented in the python library Tensorflow [1] with the TF-slim module. For the pre-training and training of the models the same parameter setting are used for the Inception V1 and the Inception Resnet V2. The parameters are shown in table 4.2. It is not the scope of this research to find the optimal parameter settings for the pre-training and training of the models and therefore the same settings were taken that were used by Schroff et al. in their FaceNet architecture [59]. One thing that must be noted is that Schroff et al. were using the triplet loss, where we chose to use the center loss, which is explained in subsection 3.1.3.

The learning rate of the Inception V1 model starts at 0.1, is decayed to 0.01 after 50 epochs and is set to 0.001 after 65 epochs. The Inception Resnet V2 has a different learning rate schedule. It starts also on 0.1, is decayed to 0.01 after 90 epochs and is set to 0.001 after 105 epochs. This learning rate schedule is often used in research with convolutional neural networks. Setting the learning rate high at the start speeds up the learning of the models. The lowering of the learning rate in the last epochs is used for fine-tuning of the weights, which results in better performances. Early stopping is used for the pre-training and training of the models. If during the last five epochs the loss function has not changed more than the threshold it will stop training. The threshold for the Inception V1 and Inception Residual network V2 is set to 0.001.

4.2 Experimental setup

4.2.1 Labeled faces in the wild

The first two recognition tasks are done on the famous labeled faces in the wild dataset (LFW) benchmark [29]. This dataset contains 13,233 images that are collected from the web, from 5,749 different identities. In the following two subsection the experimental setup are described for the two recognition tasks.

Verification

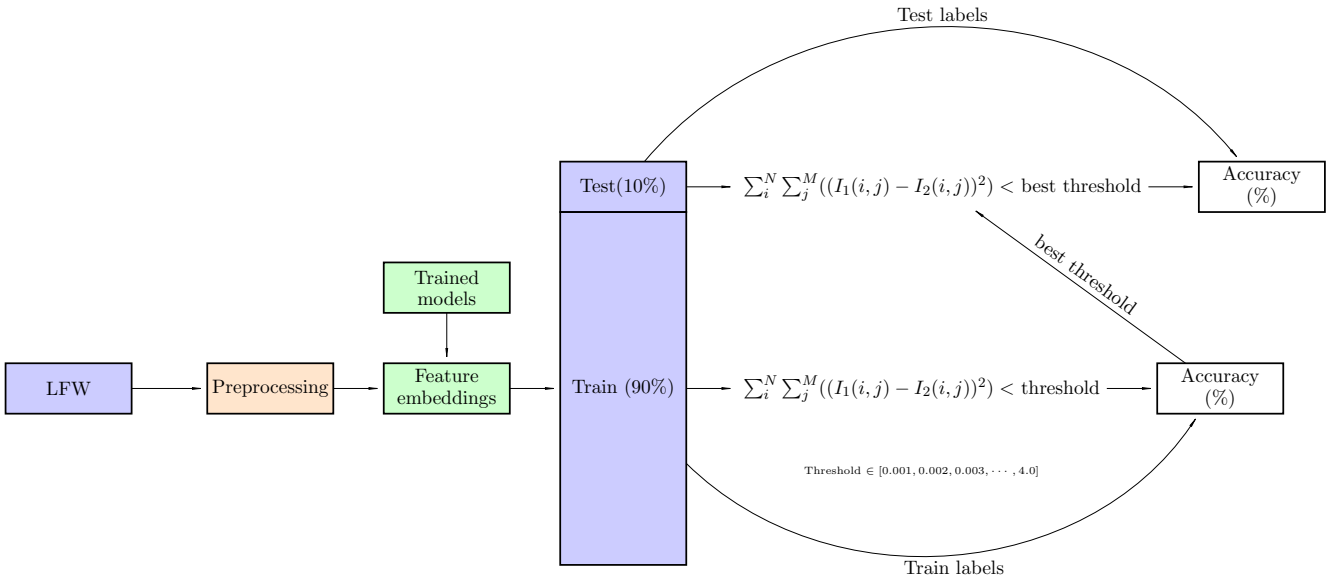

Figure 4.2: Schematic overview of the creation of the verification approach from the LFW dataset. The steps are described in section 4.2.1. The distance between the feature embeddings is calculated by equation 4.1.

The first validation approach is a verification task on LFW. As described before, verification is a 1:1 matching problem. Figure 4.2 shows a schematic overview of the steps that are done in the verification experiment. The process of the verification experiment uses these steps:

1. Get all the image pairs from the LFW site[1].
2. Preprocess all the images according to subsection 4.1.3.
3. Extract from all the image pairs the feature embeddings by using the convolutional neural network (trained models).
4. Randomly split the extracted feature embeddings 10 times in 90% training data and 10% test datas.
5. Use the training dataset to get the best threshold. This is done by calculating the squared difference between the two extracted features embeddings as is shown in equation 4.1. First substract all pixels in the first feature embedding from the second feature embedding. Secondly calculate squared difference for all pixels and sum the squared difference of all pixels to obtain one distance value.

$$distance = \sum_{i}^{N} \sum_{j}^{M} (I_1(i,j) - I_2(i,j))^2 \tag{4.1}$$

   The embedding pair is seen as equal when the distance is smaller than the threshold. Else the embedding pair is seen as different. To find the best threshold we use a range from 0 until 4 with steps of 0.001.
6. The result is compared with the actual train label. The accuracy is calculated by summing all the correct predictions and dividing this by the number of examples.
7. The best threshold is the value for which the accuracy is the highest.

[1] http://vis-www.cs.umass.edu/lfw/pairs.txt

8. This threshold is used to get the accuracy of the test set. Firstly, the distance between the test embeddings pair is calculated according to equation 4.1. Secondly, the test embeddings pair is seen as equal when the distance is lower than the threshold. This is compared with the actual test label. The accuracy is number of correct predictions on the test dataset divided by the number of examples in the test dataset.

For the comparison the *Unrestricted, Labeled Outside Data Results* protocol is followed. This means that there are no restrictions on how the evaluation is done and an own dataset can be used to train the model. However we choose to use the CASIA dataset for training the model. The metrics that are used to evaluate the task are the accuracy, ROC-curve and the AUC. The calculation of the metrics can be found in section 3.2.

Identification

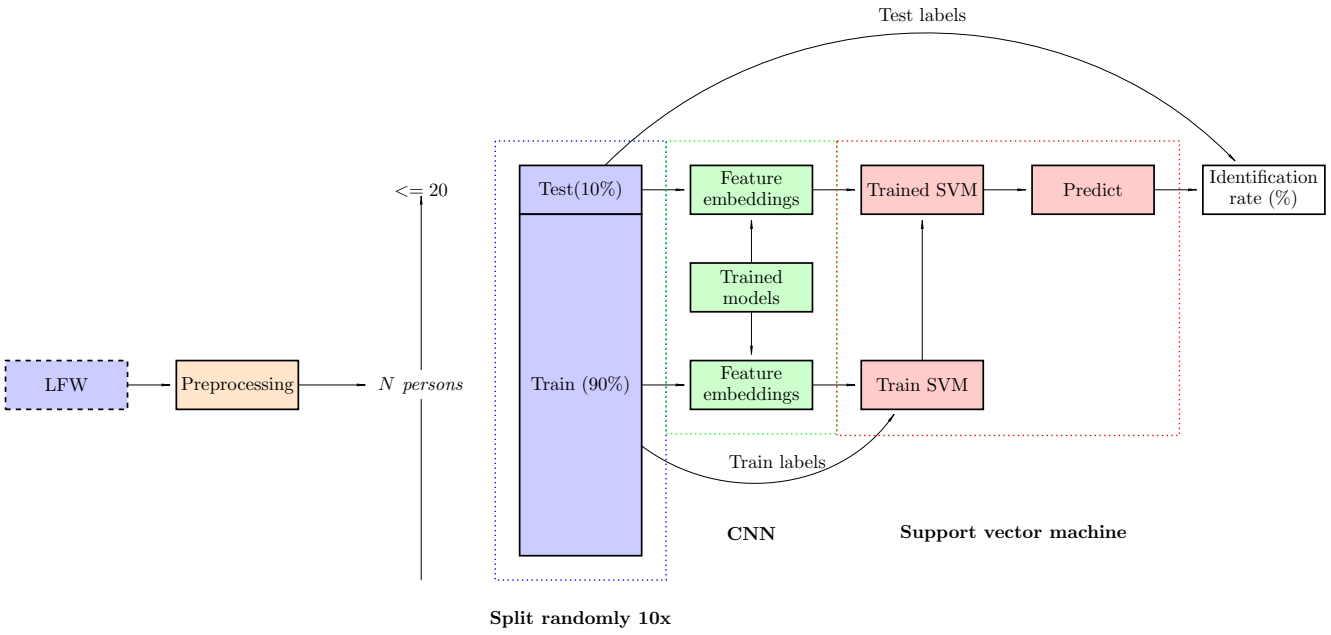

Figure 4.3: Schematic overview of the creation of the identification approach for the LFW dataset. This approach is proposed by Amos et al. [4]. The first step is to sort the dataset, then split it 10x randomly in 90% training data and 10% testing data. In the second step the feature embeddings are extracted by the convolutional neural network. In the third step a support vector machine is trained with the training data. With this trained support vector machine, the test dataset is predicted and the accuracy is calculated by comparing the predicted values with the actual values.

The second approach uses the same dataset but this time it is an identification task that is based on a method proposed by Amos et al. [4]. An overview of the creation of the identification task is shown in figure 4.3. The steps for creating this identification task are as follows: the LFW dataset is sorted, where the first person has the most images and the second person has the second most images until a specified number N. The second step is that from the N persons, we take randomly 20 images. If the person does not have 20 images then all the images are taken. The third step is to randomly split the dataset 10 times with 90% of the images as training data and 10% of the images as test data. These three steps are visualized as the blue part in figure 4.3.

To evaluate the models, the green part of figure 4.3 is the most important. In this part, the feature embeddings are extracted from the train and test dataset by the trained models. These feature embeddings are used to train a support vector machine. Amos et al. already optimized the support vector machine and to get results comparable to theirs, we use the same settings as they proposed for the support vector machine [4]. In the last step, the predicted labels are compared with the actual labels, which results in the identification rate.

In the approach proposed by Amos et al., they only took until 100 persons but in our evaluation, the validation is made a bit more difficult by taking more steps with different number of persons: {10, 25, 50, 100, 125, 150, 175, 200, 225, 250, 300, 400, 500} [4]. It must be noticed that there is an overlap between the higher steps in this identification task. This is because there are not many persons with more than 20 images in the LFW dataset. Table 4.3 gives an overview of the number of images for each number of persons. The metric to evaluate this task is the identification rate. How the identification rate is calculated is shown in section 3.2.

Table 4.3: Schematic overview of the resulting number of images for the training and the testing dataset with the corresponding number of persons.

# of persons	# of training images	# of test images
10	180 +- 0	20 +- 0
25	450 +- 0	50 +- 0
50	900 +- 0	100 +- 0
100	1681 +- 23	187 +- 23
125	1972 +- 43	219 +- 43
150	2215 +- 64	246 +- 64
175	2425 +- 84	269 +- 84
200	2613 +- 105	290 +- 105
225	2786 +- 126	310 +- 126
250	2943 +- 145	327 +- 145
300	3218 +- 184	358 +- 184
400	3678 +- 255	409 +- 255
500	4059 +- 322	451 +- 322

4.2.2 MegaFace challenge 1

To validate the models on a more difficult task, we chose to use the first challenge of the MegaFace competition [33]. The MegaFace competition is an international competition where a lot of different companies compete with each other. This challenge is called *unrestricted recognition with varying numbers of distractors*. The evaluation dataset is the Facescrub dataset that was proposed by Ng et al. [50]. This contains 106,863 images with 530 classes. The organisers of the challenge also give a distractor dataset. This consist of 1M images of different persons. In this thesis, we only use the verification task of this challenge. The verification approach on the Facescrub dataset is explained in [33]. To evaluate the verification, all the pairs between the Facescrub dataset and the distractor dataset are computed. In this thesis only 10, 100 and 1000 distractors are used.

The evaluation is reported as an ROC-curve. The metric to evaluate this task is the accuracy that is calculated by the ROC-curve, which is shown in section 3.2. We only use the ROC-curve for this challenge. We are finding the max value for all combinations of the true positive rate with the false positive rate of the ROC-curve. The script to achieve these results is given by the organizers of the competition and is available on their site[2]. Note: the Facescrub pre-trained models cannot be used to evaluate the research question because the Facescrub is used for the evaluation of the models.

4.2.3 Experiments with swapping first blocks

One of the sub-questions of this thesis is defined as: *Is there a difference in results when swapping the first blocks of the pre-trained models?* This question was based on the principle that the first blocks with layers of the model have universal features. This means that for the different pre-trained models, the first layers should not have an influence on the results. In this experiment, we swap the weights of the first layers of the two best pre-trained models with the weights of the two worst pre-trained models. An overview of the experiment is shown in figure 4.4. The evaluation is done on the verification task and identification task of the labeled faces in the wild benchmark.

[2] http://megaface.cs.washington.edu

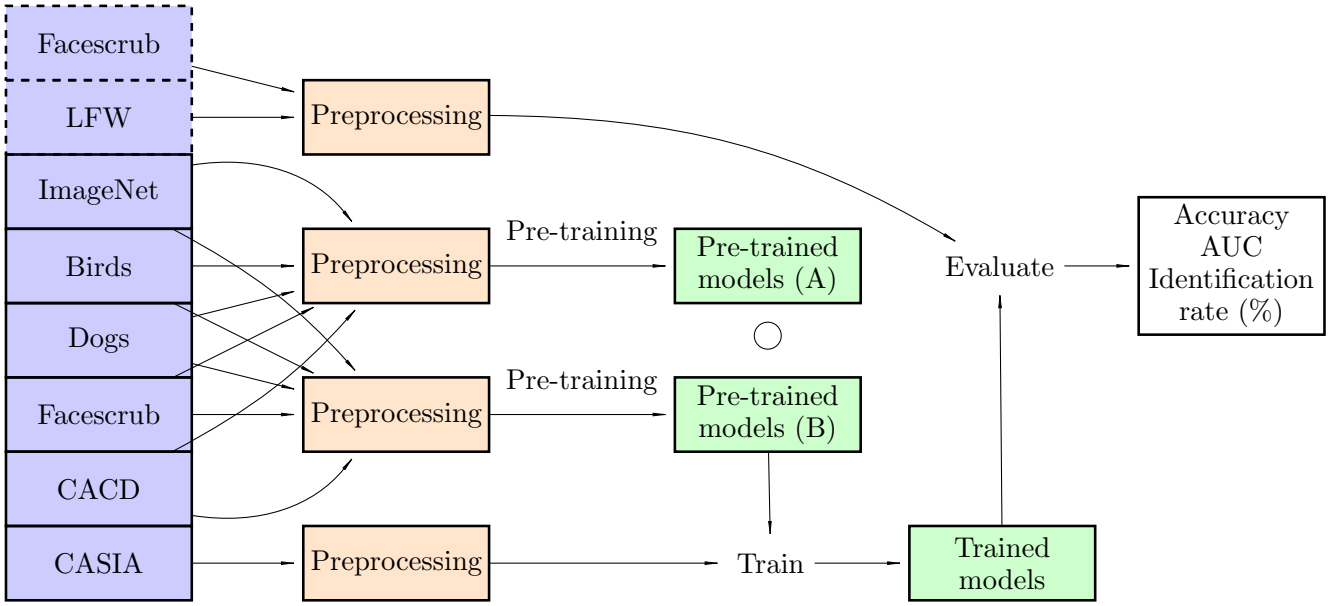

Figure 4.4: Experimental setup for preprocessing, pre-training, training and the evaluation of swapping the first layers. The difference with the experimental setup in figure 4.1, is that the layers of the first n blocks are swapped between the pre-trained models. The pre-trained models (A) use the weights of the first n blocks and pre-trained models (B) use the weights of the weights of the n blocks after (A). The preprocess phase is the same for the pre-training, training and evaluation phases.

Chapter 5

Results

The results in this chapter are used to try to answer the two main research question of this thesis. In section 5.1 the results are presented to answer the first question *Do models pre-trained on face datasets achieve better results in comparison with non-face datasets on face recognition?* The results to answer the second research question are given in section 5.2. The second research question is defined as *Do the first few layers have influence on the results in face recognition?*

5.1 Results with models pre-trained on face datasets and pre-trained on non-face datasets

In this section the results for the first research question are presented. These results are focused on comparing the models pre-trained on face datasets and models pre-trained on non-face datasets. In this section we compare the models that are pre-trained and the model fully trained on the CASIA dataset. The first research question is separated into the following sub-questions:

1. Do models pre-trained on face datasets perform better in comparison with non-face datasets per n frozen layers?
2. Are models pre-trained on face datasets more robust than models pre-trained on non-face datasets?

In this section we present the results to answer the first two sub-questions. The models are evaluated using three different evaluation methods which are described in subsections 4.2.1 and 4.2.2.

5.1.1 Results per n blocks of frozen pre-trained layers

In this subsection we present the results using the three evaluation methods of models pre-trained on face datasets and on non-face datasets. A block is defined as a number of pre-trained frozen layers. As an example: if block 2-5 is mentioned then it means that block 1 is pre-trained with a dataset and the other blocks are trained with the CASIA dataset. If block 3-5 is shown then it means that blocks 1 and 2 are pre-trained with a dataset and the rest of the blocks are trained with the CASIA dataset. Block 1-5 and block 1-8 specify that the model is fully trained with the CASIA dataset and there is no pre-training. In this subsection, we investigate the comparison of the different pre-trained models per block of pre-trained frozen layers.

Verification. The results using the Inception V1 are presented in table 5.1 and the ROC-curves are shown in figure 5.1. The metrics in the table for the verification task are the accuracy and the AUC. In the table and figure the results are split by a number of blocks pre-trained on the particular dataset. In the ROC-curve is shown that for block 2-5, the models pre-trained on CACD and Facescrub outperform the models pre-trained on ImageNet, Dogs and Birds. The results in this block are close to each

Table 5.1: The results for verification and identification using Inception V1. The evaluation metrics are the accuracy, the area under the curve (AUC) and the identification rate. The results of Inception V1 are divided into 4 blocks. In block 1-5 is the model fully trained with the CASIA dataset. The other results are from the pre-trained models that have n blocks as the pre-trained model and the rest is trained with the CASIA dataset. For example block 2-5 has the weights from block 1 of the pre-trained model and blocks 2, 3, 4 and 5 are trained with the CASIA dataset.

Blocks	Dataset	Accuracy (%)	AUC	Identification rate (%)
1-5	CASIA	95.3+-0.9	0.990	93.6+-8.9
2-5	Birds	95.3+-1.0	0.990	91.4+-10.0
2-5	CACD	96.8+-0.7	0.995	95.4+-6.1
2-5	Dogs	94.4+-0.9	0.987	90.4+-10.0
2-5	Facescrub	97.5+-1.0	0.996	96.0+-5.4
2-5	ImageNet	95.5+-0.9	0.990	90.5+-11.3
3-5	Birds	91.8+-1.0	0.975	81.8+-14.2
3-5	CACD	96.5+-0.8	0.995	95.8+-5.5
3-5	Dogs	80.9+-1.0	0.892	36.7+-14.2
3-5	Facescrub	95.9+-0.7	0.993	91.8+-9.8
3-5	ImageNet	94.5+-0.9	0.988	90.8+-10.2
4-5	Birds	87.4+-2.2	0.946	72.2+-16.7
4-5	CACD	96.1+-0.8	0.993	94.4+-6.5
4-5	Dogs	50.0+-0.0	0.500	0.7+-0.5
4-5	Facescrub	94.6+-0.7	0.988	92.5+-9.0
4-5	ImageNet	88.6+-1.5	0.956	77.2+-17.0

other, which is not surprising because only block 1 is used for pre-training and the rest of the network is trained with the CASIA dataset. The results in block 2-5 in table 5.1 give a more precise view. Models pre-trained on Facescrub perform the best, closely followed by the model pre-trained on CACD. The difference in accuracy between the models pre-trained on face datasets and on non-face datasets is ~ 2.0 and the difference in AUC is ~ 0.006. It is interesting to see that the model pre-trained on ImageNet is achieving similar results as the models pre-trained on Dogs and Birds. Another interesting fact is the comparison between models fully trained on the CASIA dataset (block 1-5) and the pre-trained models. This shows that models pre-trained on face datasets obtain better results than the model fully trained on the CASIA dataset. The models pre-trained on non-face datasets achieve similar results as the model fully trained on the CASIA dataset.

The ROC-curve of block 3-5 shown in figure 5.1, shows that models pre-trained on CACD and Facescrub obtain the best results, followed by the model pre-trained on ImageNet. The model pre-trained on Birds has a better performance than the modes pre-trained on Dogs. The results are also presented in table 5.1. Here, it is shown that the difference between the best and the worst performing model is 15.6% in accuracy and 0.103 in AUC. The model pre-trained on ImageNet has a better performance in comparison with the models pre-trained on Dogs and Birds. The models pre-trained on face datasets obtains better results than the model that is fully trained with the CASIA dataset.

The results of block 4-5 are shown in figure 5.1 and presented in table 5.1. ROC-curve shows that the models pre-trained on face datasets outperforms the models pre-trained on non-face datasets. The model pre-trained on CACD is the best performing model in comparison with the other models. The models pre-trained on Birds and ImageNet are comparable. The model pre-trained on Dogs obtains a performance that is comparable with chance. The difference between the pre-trained models is more precisely presented in table 5.1. There is shown that the smallest difference between models pre-trained on face datasets and on non-face datasets is $\sim 6.0\%$.

The results using the Inception Resnet V2 are presented in table 5.2 and the ROC-curves are shown

Table 5.2: The results for verification and identification for Inception Residual Network V2. The evaluation metrics are the accuracy, the area under the curve (AUC) and identification rate. The results of the Inception Residual Network V2 are divided into 7 blocks. In block 1-8 is the model fully trained with the CASIA dataset. The other results are from the pre-trained models that have n blocks as the pre-trained model and the rest is trained with the CASIA dataset. For example block 2-8 has the weights from block 1 of the pre-trained model and blocks 2, 3, 4, 5, 6, 7 and 8 trained are with the CASIA dataset.

Blocks	**Dataset**	**Accuracy (%)**	**AUC**	**Identification rate (%)**
1-8	CASIA	98.5+-0.4	0.998	98.8 +- 1.9
2-8	Birds	95.2+-1.5	0.991	93.2+-8.8
2-8	CACD	96.6+-0.9	0.994	95.7+-6.7
2-8	Dogs	95.6+-0.9	0.990	93.3+-9.2
2-8	Facescrub	96.5+-0.8	0.994	95.4+-7.0
2-8	ImageNet	95.5+-1.1	0.991	94.4+-8.5
3-8	Birds	94.4+-0.7	0.988	92.3+-9.3
3-8	CACD	97.5+-0.8	0.996	96.4+-5.2
3-8	Dogs	94.9+-1.3	0.988	93.6+-9.2
3-8	Facescrub	96.4+-0.8	0.994	95.7+-6.4
3-8	ImageNet	95.7+-1.0	0.992	94.2+-9.1
4-8	Birds	94.4+-1.0	0.988	92.8+-9.6
4-8	CACD	96.8+-0.8	0.994	96.1+-5.6
4-8	Dogs	94.4+-0.9	0.988	92.1+-9.8
4-8	Facescrub	96.5+-1.0	0.994	95.9+-6.1
4-8	ImageNet	95.5+-1.0	0.991	93.8+-9.0
5-8	Birds	95.6+-0.8	0.974	93.5 +-8.2
5-8	CACD	97.4+-0.9	0.996	97.4 +- 3.7
5-8	Dogs	96.5+-0.6	0.981	95.0 +- 6.7
5-8	Facescrub	97.5+-0.8	0.994	97.3 +- 3.8
5-8	ImageNet	96.3+-0.9	0.980	95.3 +- 6.7
6-8	Birds	93.9+-0.9	0.983	89.4 +- 10.5
6-8	CACD	97.5+-0.7	0.996	97.0 +- 4.5
6-8	Dogs	94.8+-1.1	0.988	92.4 +-8.8
6-8	Facescrub	96.8+-0.9	0.995	96.3 +- 5.0
6-8	ImageNet	95.0+-1.0	0.989	92.3 +- 9.7
7-8	Birds	91.6+-1.5	0.991	82.7 +- 12.5
7-8	CACD	97.1+-0.8	0.997	96.2 +- 4.9
7-8	Dogs	92.8+-1.0	0.994	87.9 +- 12.5
7-8	Facescrub	96.8+-1.0	0.996	96.1 +- 5.1
7-8	ImageNet	92.3+-1.2	0.994	85.8 +- 12.9

in figure 5.2. The metrics for the verification task are the accuracy and the AUC. All ROC-curves show that the models fully trained on CASIA outperform all the models that are pre-trained. The ROC-curves also show that for blocks 2-8, 3-8, 4-8 and 5-8, the results are close to each other. Looking closely at these ROC-curves then it reveals that the models pre-trained on face datasets perform better than the models pre-trained on non-face dataset. The differences are presented better in table 5.2. The models pre-trained on face datasets perform in accuracy $\sim 1.0\%$ and in AUC ~ 0.003 better than models pre-trained on non-face datasets for all these four blocks.

The ROC-curves of blocks 6-8 and 7-8 show that the models pre-trained on face datasets obtains better results than the models pre-trained on non-face datasets. This ROC-curve also shows that the worst performing model is the model pre-trained on Birds. The models pre-trained on ImageNet and Dogs have similar results. These differences in results are more clearly presented in table 5.2. The models pre-trained on the CACD dataset obtain the best results in blocks 6-8 and 7-8 with respectively 97.5% and 97.1% followed by the model pre-trained on Facescrum. The differences in accuracy between the model pre-trained on face datasets and on non-face datasets are $\sim 2\%$ for block 6-8 and $\sim 4\%$ for block 7-8.

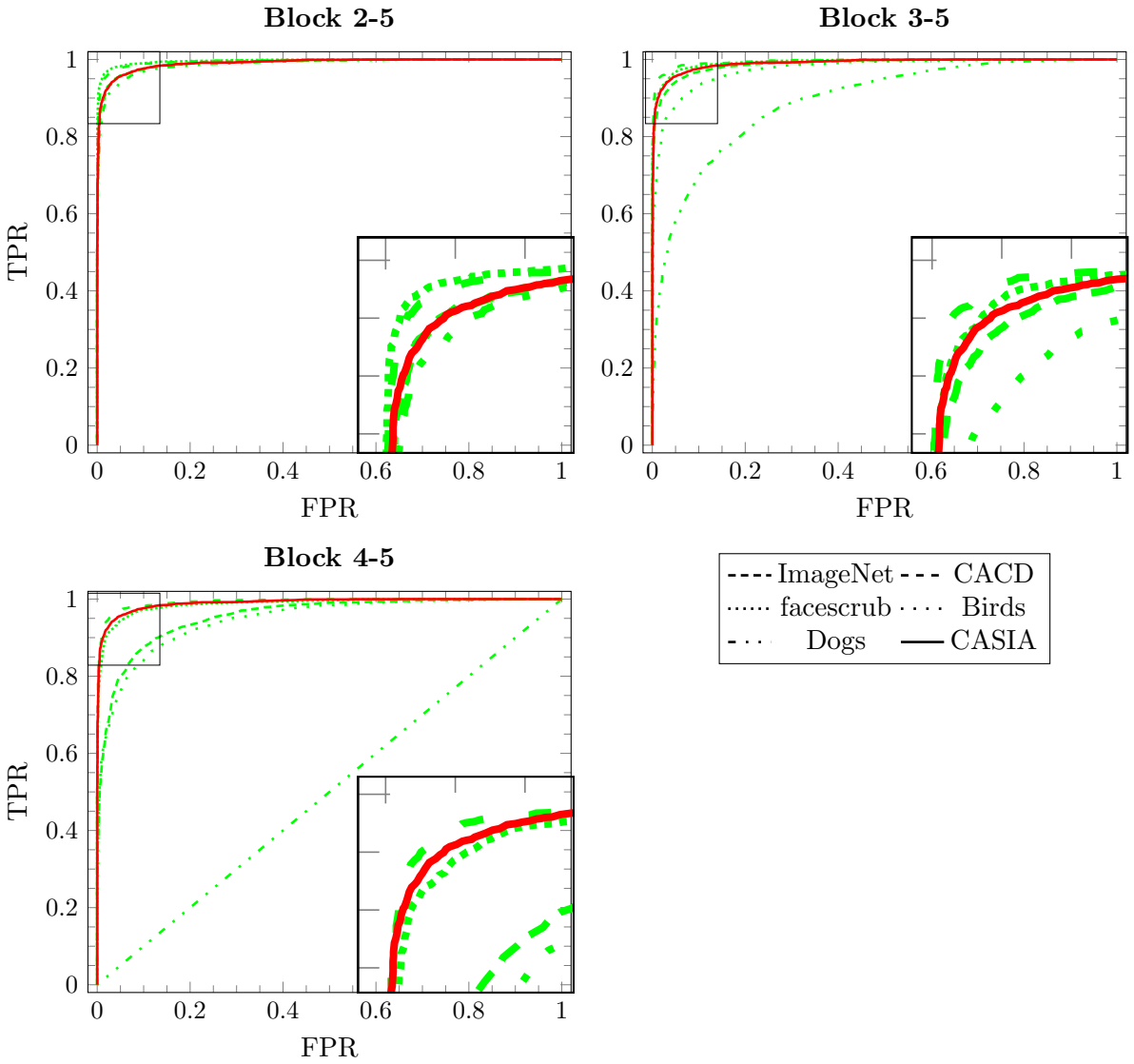

Figure 5.1: ROC curves for the verification task on the labeled faces in the wild dataset with the Inception V1 model. The different pre-trained models are specific with the different line styles. The results on the Inception V1 model with blocks 1-5, 2-5, 3-5 and 4-5. Block 3-5 means that blocks 1 and 2 are frozen pre-trained layers and blocks 3, 4 and 5 are layers that are trained by the CASIA dataset.

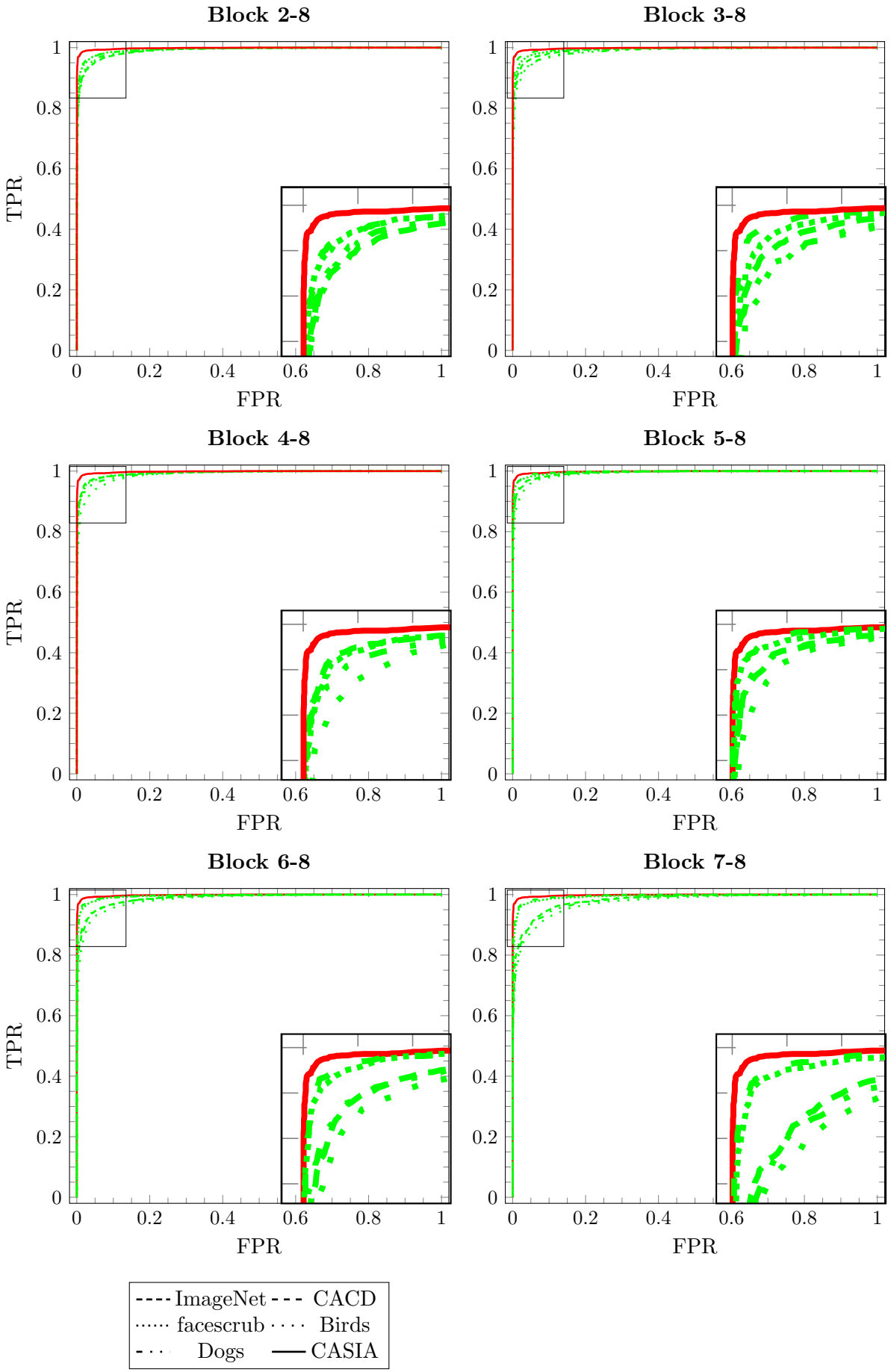

Figure 5.2: ROC curves for the verification task on the labeled faces in the wild dataset with the Inception Residual Network V2. The different pre-trained models are specific with the different line styles. The results on the Inception Residual Network v2 with blocks 1-8, 2-8, 3-8, 4-8, 5-8, 6-8 and 7-8. Block 3-8 means that blocks 1 and 2 are the blocks with frozen pre-trained layers and blocks 3, 4, 5, 6, 7 and 8 are trained with the CASIA dataset.

Identification. The results on the identification task using the Inception V1 are shown in figure 5.3. Table 5.1 presents the identification rate (%) with 500 persons. The plot for block 2-5 shows the same pattern as for the verification task. The models pre-trained on face datasets outperforms the models pre-trained on non-face datasets. The plot shows that when the task becomes more difficult, then the differences in results is increased. One interesting result is that the model pre-trained on Birds performs better than the model pre-trained on ImageNet and Dogs. These results are more precisely presented in table 5.1. It shows that the models pre-trained on face datasets obtain $\sim 5\%$ better identification rates than models pre-trained on non-face datasets.

The plot for block 3-5 shows that the model pre-trained on CACD performs best, followed by the model fully trained with CASIA. The models pre-trained on Facescrub and ImageNet have comparable results. The model pre-trained on Birds performs worse than these three models but is still performing better than the model pre-trained on the Dogs dataset. This is the same pattern as we already noticed in the verification tasks, only the difference in performance between the model pre-trained on Birds and Dogs becomes more visible. Table 5.1 presents more clearly these results. The model pre-trained on CACD performs $\sim 4.0\%$ better in comparison with the next best performing model. The difference in identification rate between models pre-trained on Facescrub and ImageNet is only $\sim 1.0\%$.

Block 4-5 shown in figure 5.3 has the same pattern as in the verification task. The model pre-trained on CACD outperforms all the other pre-trained models. This is followed by the model pre-trained on Facescrum, which has a better identification rate than models pre-trained on ImageNet, Birds and Dogs. The model pre-trained on Dogs is performing badly, which is an indication that this model did not capture any features that can be used for face recognition. Looking at block 4-5 of table 5.1, the results give a more precise insight. The model pre-trained on CACD has the best performance with an identification rate of 94.4%, compared with the results of models pre-trained on Facescrum (92.5%), ImageNet(77.2%), Birds (72.2%) and Dogs (0.7%). The difference in this block between the model pre-trained on Facescrum and ImageNet is more than 15%, which shows the difference again that we did not noticed in block 3-5.

The results using the Inception Resnet V2 are presented in table 5.2 and the identification plots are shown in figure 5.4. The identification plots reveal that the model fully trained on CASIA outperforms all the models that are pre-trained. This is the same pattern as we noticed in the verification task. In all blocks, the identification plots show that models pre-trained on CACD performs better than the other pre-trained models. In blocks 2-8, 3-8 and 4-8 the models pre-trained with ImageNet perform better than the models pre-trained on Dogs and Birds. In blocks 5-8, 6-8 and 7-8 the models pre-trained on Dogs perform better than the models pre-trained on ImageNet. The models pre-trained on Birds perform better at block 4-8 than the models pre-trained on Dogs. At the rest of the blocks, the models pre-trained on Birds has the worst performance.

Table 5.2 shows the identification rate for different blocks using the Inception Resnet V2. The model fully trained on CASIA performs the best with an identification rate of 98.8%, which is $\sim 1.5\%$ more than any other pre-trained model. In every block the difference in identification rate between the models pre-trained on face datasets and on non-face datasets shows $\sim 2.0\%$ or more. Especially in the block 7-8 the difference in performance between models pre-trained on face datasets and on non-face datasets is clear ($\sim 9.0\%$). In the identification approach, we see the same pattern as with the verification approach. The models pre-trained on face datasets outperform the models pre-trained on non-face datasets. Also, the difference is bigger when there are more blocks of layers used for pre-training and less for training.

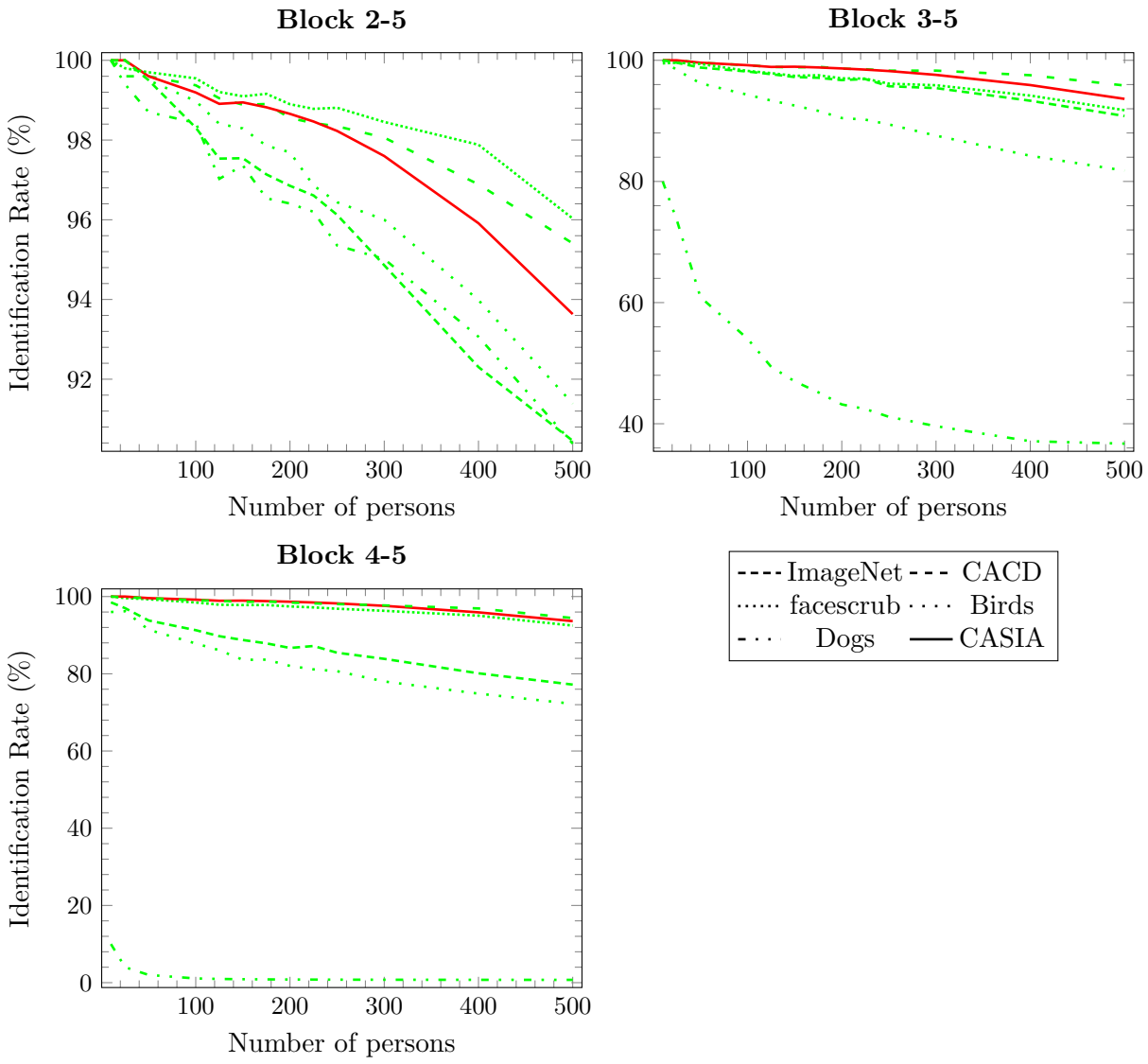

Figure 5.3: Results for the identification task on the labeled faces in the wild for the Inception V1. The y-axis is the identification rate in % and the x-axis is the number of persons. The line styles specify the dataset that is used.

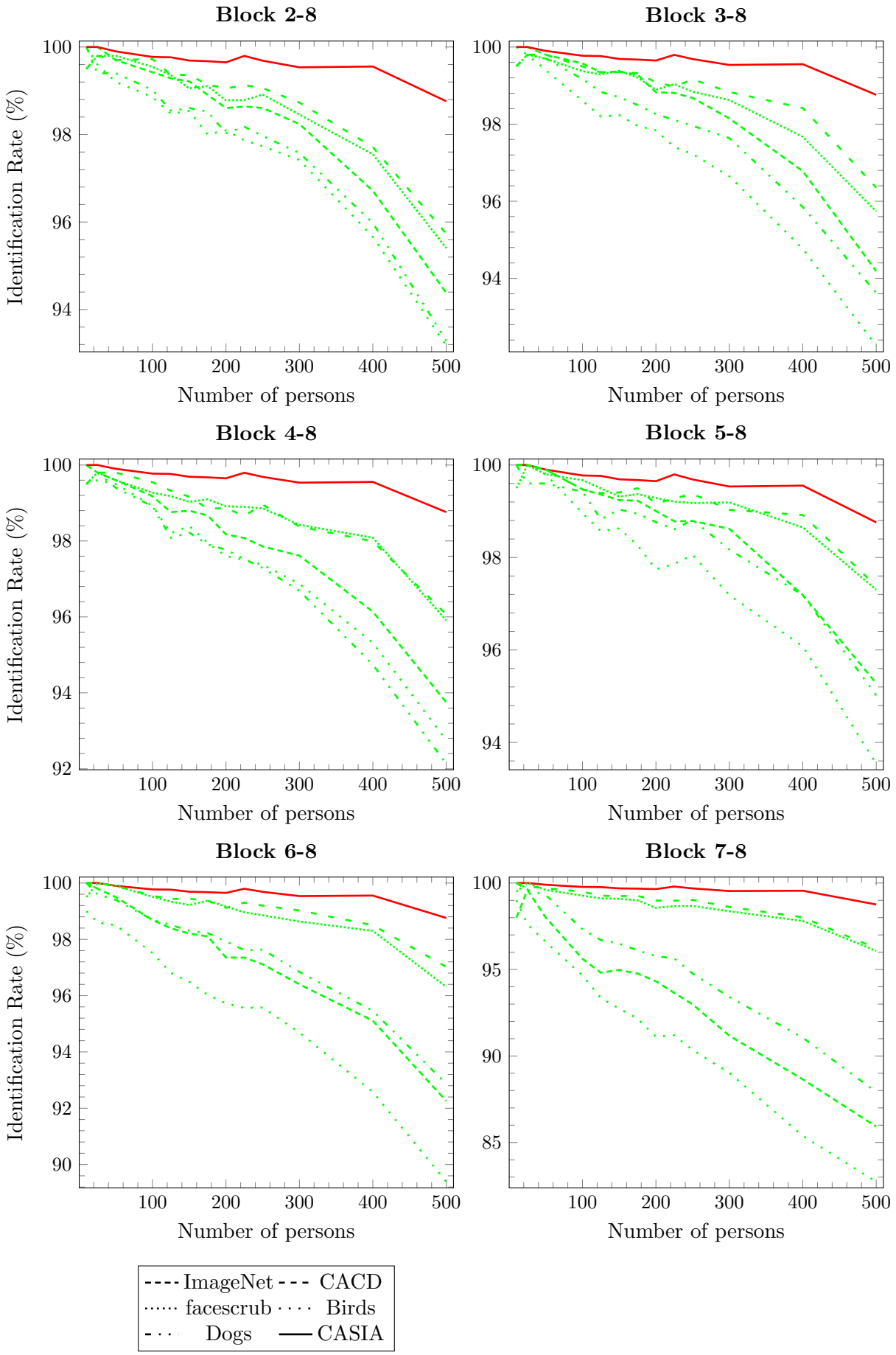

Figure 5.4: Results for the identification task on the labeled faces in the wild for the Inception Residual Network V2. The y-axis is the identification rate in % and the x-axis is the number of persons. The line styles specify the dataset that is used and the colors show which n of blocks is used of the pre-trained models.

MegaFace challenge 1. Table 5.3 shows the results for the MegaFace challenge 1 using the Inception V1 model. The results are split by the number of distractors that are used, namely $\{10, 100, 1000\}$. The results are also split by the number of blocks that are used for pre-training and training. The results for the models pre-trained with Facescrub are excluded from the table because the evaluation dataset is also the Facescrub. The accuracies in the table show that for block 2-5, there is not a clear difference between the pre-trained models. Only with 100 distractors the models pre-trained with Dogs perform worse and the same holds for the model pre-trained on Birds with 1000 distractors. For block 3-5 the pattern, is the same as we already noticed in the verification and identification task. The models pre-trained on CACD perform better than the models pre-trained on Birds, Dogs and ImageNet. There is no difference between the results of the models pre-trained on non-face datasets. The same pattern is shown in the results for block 4-5. The models pre-trained on CACD outperform the models pre-trained on Birds, Dogs and ImageNet. There is also a difference in which dataset is used for pre-training with non-face datasets. The models pre-trained on Dogs perform worse with an accuracy that is similar to chance (50%) and models pre-trained on ImageNet outperform the models pre-trained on Birds. For 100 distractors, the difference between models pre-trained on CACD and Birds is not significant.

The results using the Inception Resnet V2 are presented in table 5.4. Again the results on the model pre-trained on Facescrub are excluded from the evaluation. The results are not as confident as with the verification and the identification with LFW. Most of the times the models pre-trained on the CACD outperforms the models pre-trained on non-face datasets but a couple of times the models pre-trained on Birds, Dogs or ImageNet outperform CACD. From these results, we cannot conclude that the models pre-trained on face datasets outperform the models pre-trained on non-face datasets. Another interesting fact is that the model fully trained on CASIA is not the best model to use. There are a couple of pre-trained models that outperform the fully trained model. We did not have this finding in the verification and identification on LFW.

Another interesting fact presented in tables 5.3 and 5.4, is that the accuracy is sometimes higher using more number of distractors. We do not have a clear reason for this fact but there are three possible reasons. The verification task on the MegaFace challenge 1 is done by comparing all the images of the testset with the distractors. By using more distractors, it can happen that more images from the testset are correctly recognized and therefore it can result in a higher accuracy. The second reason can be that the images in the distractors set are randomly selected. Therefore it can happen that the lower sets (10,100) has more difficult images in the distractor set than the 1000 number of distractors. The last reason can be that the accuracy is not the best metric to evaluate this challenge. The accuracy is derived from the max value of all TPR and FPR combinations.

Table 5.3: The accuracy in % on the MegaFace challenge 1. This is done for the distractors set $\{10, 100, 1000\}$. The results are from using the Inception V1 models. The results are split into four blocks: 1-5, 2-5, 3-5 and 4-5. One example of a block is 2-5. This block has only the first block used in pre-training and the rest is trained with the CASIA dataset. *The results on models pre-trained on Facescrub cannot be used because the model is evaluated with the Facescrub dataset.

Block	**Dataset**	**10**	**100**	**1000**
1-5	CASIA	95.4	97.6	93.5
2-5	Birds	94.0	96.1	88.7
2-5	CACD	95.2	95.2	93.9
2-5	Dogs	93.4	86.0	94.4
*2-5	Facescrub	94.6	96.2	96.0
2-5	ImageNet	95.7	96.0	98.2
3-5	Birds	89.0	89.8	88.7
3-5	CACD	96.4	95.0	92.5
3-5	Dogs	75.6	83.2	85.2
*3-5	Facescrub	94.8	89.3	92.3
3-5	ImageNet	89.3	85.2	82.9
4-5	Birds	81.4	87.4	80.7
4-5	CACD	94.5	88.2	94.5
4-5	Dogs	50.0	50.0	50.0
*4-5	Facescrub	95.2	94.0	92.7
4-5	ImageNet	88.2	85.3	85.2

Table 5.4: The accuracy in % on the MegaFace challenge 1. This is done for the distractors set $\{10, 100, 1000\}$. The results are from using the Inception Resnet V2 models. The results are split into seven blocks: 1-8, 2-8, 3-8, 4-8, 5-8, 6-8 and 7-8. One example of a block is 2-8. This block has only the first block used in pre-training and the rest is trained with the CASIA dataset. *The results on models pre-trained on Facescrub cannot be used because the model is evaluated with the Facescrub dataset.

Block	**Dataset**	**10**	**100**	**1000**
1-8	CASIA	96.3	90.2	95.0
2-8	Birds	95.3	87.2	94.7
2-8	CACD	98.1	97.1	95.3
2-8	Dogs	95.4	90.4	85.8
*2-8	Facescrub	96.4	96.5	94.7
2-8	ImageNet	91.8	88.6	93.1
3-8	Birds	95.2	91.5	94.1
3-8	CACD	95.8	96.7	94.1
3-8	Dogs	96.8	91.2	88.4
*3-8	Facescrub	93.5	90.9	96.4
3-8	ImageNet	94.3	94.8	94.7
4-8	Birds	96.6	89.6	94.8
4-8	CACD	91.8	88.7	90.5
4-8	Dogs	95.6	89.6	93.4
*4-8	Facescrub	94.6	95.6	93.3
4-8	ImageNet	93.5	92.5	91.1
5-8	Birds	92.7	97.6	92.7
5-8	CACD	93.8	94.3	93.7
5-8	Dogs	95.8	94.4	97.1
*5-8	Facescrub	96.0	90.2	95.0
5-8	ImageNet	89.1	95.8	90.1
6-8	Birds	94.0	88.7	85.0
6-8	CACD	95.0	94.7	91.7
6-8	Dogs	89.6	85.2	91.8
*6-8	Facescrub	94.4	96.5	94.8
6-8	ImageNet	89.2	89.8	90.8
7-8	Birds	89.0	92.1	85.2
7-8	CACD	97.0	94.2	93.4
7-8	Dogs	92.5	93.0	87.2
*7-8	Facescrub	94.3	96.2	87.4
7-8	ImageNet	90.8	91.2	88.1

5.1.2 Robustness

The second sub-question: *Are models pre-trained on face datasets more robust than models pre-trained on non-face datasets?* To investigate the robustness of the pre-trained models, we introduce a new variable called drop percentage (δB_{ij}). This variable calculates the difference in results (B_{ij}) between block j (B_j) and block i (B_i) in the same pre-trained model. The equation is formulated in 5.1:

$$\delta B_{ij} = \frac{(B_j - B_i)}{B_i} * 100\%. \tag{5.1}$$

Table 5.5: δB_{ij} results using the Inception V1 (left) and Inception Resnet V2 (right) for the verification task on LFW. The accuracy of tables 5.1 and 5.2 are filled in for B_j and B_i. δB_{ij} is calculated according to equation 5.1.

Dataset	δB_{23} (%)	δB_{24} (%)
Birds	-3.67	-8.29
CACD	-0.31	-0.72
Dogs	-14.30	-47.03
Facescrub	-1.60	-2.97
ImageNet	- 1.05	-7.23

Dataset	δB_{23} (%)	δB_{24} (%)	δB_{25} (%)	δB_{26} (%)	δB_{27} (%)
Birds	-0.8	-0.8	0.4	-1.3	-3.8
CACD	0.9	0.2	0.8	0.9	0.5
Dogs	-0.7	-1.2	0.9	-0.8	-2.9
Facescrub	-0.1	0.0	1.0	0.3	0.3
ImageNet	0.2	0.0	0.8	-0.5	-3.3

Verification. The results of δB_{23} and δB_{24} using the Inception v1 for the verification task are presented in table 5.5 (left). The table shows that the drop percentages between block 2 and 3 are the biggest for models pre-trained on the Dogs (-14.30%) and Birds (-3.67%) dataset. These are followed by the models pre-trained on Facescrub (-1.60%), ImageNet (-1.05%) and CACD (-0.31%). It is interesting that the models pre-trained on ImageNet have a smaller drop percentage than the models pre-trained on Facescrub. The difference is the best visible at δB_{24} using the Inception V1 model. The pre-trained model on CACD has a drop percentage of 0.72% and models pre-trained on Facescrub has a drop of -2.97%. This is low in comparison with the -7.33%, -8.29% and -47.03% using models pre-trained on non-face datasets. The results show that the models pre-trained on the face datasets are more robust than the models pre-trained on non-face datasets.

The drop percentage between block 2 and the other blocks using the Inception Resnet V2 are presented in table 5.5 (right). The results show that the drop percentage between block 2 and blocks 3, 4 and 5 are between -1.0% and 1.0%. This indicates that the Inception Resnet V2 with blocks 2-8, 3-8, 4-8 and 5-8 are robust for all the pre-trained models. In δB_{26} the pre-trained model on Birds has an accuracy drop of 1.3%, which can already be an indication that there is a difference between the pre-trained models. This difference is best visible for δB_{27}. The models pre-trained on face datasets obtain a δB_{27} between -1.0% and 1.0% and the models pre-trained on non-face datasets obtain a drop percentage more than 2.9%. It is important to notice that the differences in δ_{B27} between the Birds, Dogs and ImageNet pre-trained models are negligible.

Table 5.6: δB_{ij} results using the Inception V1 (left) and Inception Residual Network V2 (right) for the identification task on LFW. The identification rate of tables 5.1 and 5.2 are filled in for B_j and B_i. δB_{ij} is calculated according to equation 5.1.

Dataset	δB_{23} (%)	δB_{24} (%)
Birds	-10.5	-21.0
CACD	0.4	-1.0
Dogs	-59.4	-99.2
Facescrub	-4.4	-3.7
ImageNet	0.3	-14.7

Dataset	δB_{23} (%)	δB_{24} (%)	δB_{25} (%)	δB_{26} (%)	δB_{27} (%)
Birds	-0.9	-0.4	0.3	-4.0	-11.2
CACD	0.7	0.4	1.8	1.3	0.5
Dogs	0.3	-1.2	1.8	-0.9	-5.7
Facescrub	0.3	0.5	2.0	0.9	0.7
ImageNet	-0.2	0.6	0.9	-2.0	-9.0

Identification. For the identification task the drop percentages of the identification rate are calculated and presented in table 5.6. δB_{23} shows us that the models pre-trained on Birds and Dogs are less robust than the models pre-trained on CACD, Facescrub and ImageNet. δB_{23} of the Facescrub pre-trained

models would suggest that this is less robust than the CACD and ImageNet pre-trained models. But if we look at the results of δB_{24} then we see that the models pre-trained on face datasets are more robust than the models pre-trained on non-face datasets. The models pre-trained on Dogs obtained the worst drop percentage on δB_{23} and δB_{24}. This drop percentages on the identification task show the same pattern as we already noticed for the verification task.

The drop percentages using the Inception Residual Network V2 on the identification task shows the same pattern as on the verification task. Table 5.6 shows that the drop percentages of δB_{23}, δB_{24} and δB_{25} are minimal. With values between 2.0% and -2.0%. δB_{26} shows already the first difference between models pre-trained on face datasets and on non-face datasets. But the difference between these pre-trained models is clearly shown in the drop percentages of δB_{27}. The drop percentages of models pre-trained on non-face dataset are 5.7% or higher which are high compared with the drop percentages of the models pre-trained on face datasets. An interesting finding is that the models pre-trained on Dogs are more robust on this task than the model pre-trained on ImageNet. The models pre-trained on Birds is the least robust on this task and the two models pre-trained on face datasets obtain around the same drop rates and therefore are equally robust.

Table 5.7: δB_{ij} results using the Inception V1 (left) and Inception Resnet V2 (right) for the accuracy on the MegaFace challenge 1. The results of tables 5.3 and 5.4 are filled in for B_j and B_i. δB_{ij} is calculated according to equation 5.1.

Distractors	**Dataset**	δB_{23}	δB_{24}
10	Birds	-5.3	-13.4
	CACD	1.2	-0.7
	Dogs	-19.0	-46.4
	ImageNet	-6.7	-7.8
100	Birds	-6.5	-9.0
	CACD	-0.2	-7.4
	Dogs	-3.3	-41.9
	ImageNet	-11.3	-11.1
1000	Birds	0.0	-9.0
	CACD	-0.8	0.6
	Dogs	-9.7	-47.0
	ImageNet	-15.6	-13.2

Distractors	**Dataset**	δB_{23}	δB_{24}	δB_{25}	δB_{26}	δB_{27}
10	Birds	0.1	1.3	-2.7	-1.3	-6.6
	CACD	-2.3	-6.4	-4.4	-3.2	-1.1
	Dogs	1.5	0.2	0.4	-6.0	-3.0
	ImageNet	2.7	1.9	-2.9	-2.8	-1.0
100	Birds	4.9	2.8	11.9	1.7	5.6
	CACD	-0.4	-8.6	-2.9	-2.5	-3.0
	Dogs	0.8	-0.8	4.4	-5,7	2.9
	ImageNet	7.0	4.4	8.1	1.4	2.9
1000	Birds	-0.6	0.1	-2.1	-10.2	-10.0
	CACD	-1.3	-5.0	-1.7	-3.7	-2.0
	Dogs	3.0	8.7	13.2	7.0	1.6
	ImageNet	1.7	-2.1	-3.2	-2.5	-5.4

MegaFace challenge 1. The results of the δB_{ij} on the MegaFace challenge 1 can be found in table 5.7. The left table shows the results using the Inception V1 and on the right the results using the Inception Resnet V2. The results are split by the number of distractors in the set $\{10, 100, 1000\}$ and the dataset that is used for pre-training. The results for the models that are pre-trained with Facescrub are excluded from the results because the evaluation dataset is also Facescrub.

From the results using Inception V1 with 10 and 100 distractors, we see that the models pre-trained with CACD are the most robust for δB_{23} and δB_{24}. The difference is not big in the δB_{24} for 100 distractors but still the drop percentage difference between the models pre-trained on CACD and Birds is $\sim 2\%$. The pattern that models pre-trained on face datasets are more robust than on non-face datasets. This is a pattern that we also see in this challenge for 10 and 100 distractors. For the δB_{23} for the 1000 distractors, we do not see this pattern. The models pre-trained on Birds are equally robust as the models pre-trained on CACD. But in δB_{24}, we see the pattern again. It is surprising that within the models pre-trained on non-face datasets, there is a lot of variation between the drop in percentages. In some cases, the models pre-trained on Birds are more robust. In other cases, the models pre-trained on Dogs are more robust and in some cases the models pre-trained on ImageNet are more robust. The δB_{24} for the model pre-trained on Dogs always peforms badly.

The right of table 5.7 presents the results on the MegaFace challenge 1 using the Inception Resnet V2. All results for the different δB_{ij} and each number of distractors show no clear pattern. Therefore it will not be used in the discussion.

5.2 Universal features

Universal features are the features of the first layers of the architecture. These are often represented as Gabor filters or color blobs. In this section, an investigation is described to see if these universal features have an influence on the results. The research question defined in the introduction is separated into two sub-questions:

1. Is there a difference in results when only using the first layers of the pre-trained models?
2. Is there a difference in results when swapping the first layers of the pre-trained models?

5.2.1 Layers of the first blocks

For the first sub-question is important to look at the results of block 2-5 using the Inception V1 and block 2-8 using the Inception Resnet V2. This is done for the verification and identification task of the LFW benchmark. Block 1 of the Inception V1 architecture consists of three convolutional layers out of a total of 22 trainable layers. Block 1 from the Inception Resnet V2 has only five convolutional layers of in total 134 trainable layers. Therefore the layers in block 1 are for both models defined as the universal layers.

Table 5.8: Paired t-test performed on the results of blocks 1-5 and 2-5 using the Inception V1 on the verification tasks, which is shown as the accuracy in table 5.1. The first value is the t-distribution and the second value is the p-value of this t-distribution. A positive t-distribution means that the dataset in the row has a higher accuracy than the dataset in the column header.

	Birds	**CACD**	**Dogs**	**Facescrub**	**ImageNet**	**CASIA**
Birds	x	-3.91 (0.001)	2.04 (0.06)	-4.70 (0.0002)	-0.50 (0.6)	0.0 (1.0)
CACD	3.91 (0.001)	x	6.49 (4.2e-06)	-1.66 (0.1)	3.69 (0.002)	4.05 (0.0007)
Dogs	-2.04 (0.06)	-6.49 (4.2e-06)	x	-6.09 (1.9e-06)	-2.70 (0.01)	-2.09 (0.05)
Facescrub	4.73 (0.0002)	1.66 (0.1)	6.89 (1.9e-06)	x	4.55 (0.0002)	4.85 (0.0001)
ImageNet	0.50 (0.6)	-3.6 (0.002)	2.70 (0.01)	-4.55 (0.0002)	x	0.52 (0.6)
CASIA	0.0 (1.0)	-4.05 (0.0007)	2.09 (0.05)	-4.85 (0.0001)	-0.52 (0.61)	x

Table 5.9: Paired t-test performed on the results of blocks 1-8 and 2-8 using the Inception Resnet V2 on the verification tasks, which is shown as the accuracy in table 5.2. The first value is the t-distribution and the second value is the p-value of this t-distribution. A positive t-distribution means that the dataset in the row has a higher accuracy than the dataset in the column header.

	Birds	**CACD**	**Dogs**	**Facescrub**	**ImageNet**	**CASIA**
Birds	x	-2.33 (0.03)	-0.68 (0.5)	-2.04 (0.06)	-0.29 (0.8)	-6.30 (6.2e-06)
CACD	2.33 (0.03)	x	2.21 (0.04)	0.43 (0.67)	3.20 (0.005)	-5.68 (2.18-05)
Dogs	0.68 (0.5)	-2.21 (0.04)	x	-1.84 (0.08)	1.20 (0.3)	-8.46 (1.1e-07)
Facescrub	2.04 (0.06)	-0.43 (0.67)	1.84 (0.08)	x	2.89 (0.01)	-6.50 (3.9e-06)
ImageNet	0.29 (0.8)	-3.20 (0.005)	-1.19 (0.2)	-2.89 (0.01)	x	-8.68 (7.5e-08)
CASIA	6.30 (6.2e-06)	5.68 (2.2e-05)	8.47 (1.1e-07)	6.52 (3.9e-06)	8.68 (7.5e-08)	x

Verification. Results of block 2-5 using the Inception V1 on the verification task were presented in the previous subsection in table 5.1. The results were as follows: models pre-trained on Facescrub obtained the best performances with 97.5%, followed by the model pre-trained on CACD (96.8%). The models pre-trained on ImageNet, Dogs and Birds have a 2.5% lower accuracy than the other two pre-trained models. It is interesting that all the pre-trained models obtained better or equal compared to the models trained on block 1-5. The Facescrub pre-trained model achieves a 2.2 % better accuracy and 0.006 better AUC. The only difference between the models trained on block 1-5 and block 2-5 is the layers of the first block.

Table 5.8 shows the significance of the accuracies differences between the different models. The first value is the t-distribution and the second value is the p-value, which we use to determine the significance of the differences in results. If the p-value is smaller than 0.05 then the null hypothesis of the t-test can be rejected and the alternative hypothesis can be accepted. The rejection of the null hypothesis means that the difference between the two results are significant and the acceptance of the null hypothesis means that there is no significant difference. For block 2-5 the difference between the results on models pre-trained on Facescrub and CACD is not significant ($0.1 > 0.05$) but differences between the results on models pre-trained on face datasets and non-face datasets are significant. All the p-values are smaller than 0.05 between models pre-trained on CACD, Facescrub and ImageNet, Dogs, Birds. Given that the differences in results are significant, we can conclude that it makes a difference which pre-trained layers are used for block 2-5.

The difference in results in the first block between the different pre-trained models of the Inception Resnet V2 is less than for the Inception V1. Table 5.2 shows that the best performing pre-trained model (CACD) is only 1.4% better in accuracy than the worst performing pre-trained model (Birds). The model trained on block 1-8 obtains the best result on this recognition task. It outperforms the best performing model in block 2-8 with 1.9% in accuracy and 0.004 in AUC. Apparently, the model that is fully trained (block 1-8) on the CASIA dataset extracted the best features for the face verification task in comparison with the pre-trained models. The layers in the first block of the Inception Resnet V2 makes a difference for the performance.

Table 5.9 shows the results of the t-test for the results for blocks 1-8 and 2-8 using the Inception Resnet V2. The differences between results on the models pre-trained on CACD and Birds (0.03), Dogs (0.04) and ImageNet (0.005) are significant. The differences between the results on the model pre-trained on Facescrum and Birds (0.06) and Dogs (0.08) are not significant but between ImageNet, it is significant (0.01). Also the differences between results on the models trained on blocks 1-8 and 2-8 are significant, because the p-values are all smaller than 0.05. We can conclude that there is a difference using pre-trained model for the first block of layers.

Table 5.10: Paired t-test performed on the results of blocks 1-5 and 2-5 using Inception V1 on the identification task, which is shown as the identification rate in table 5.1. The first value is the t-distribution and the second value is the p-value of this t-distribution. A positive t-distribution means that the dataset in the row has a higher identification rate than the dataset in the column header.

	Birds	**CACD**	**Dogs**	**Facescrub**	**ImageNet**	**CASIA**
Birds	x	-1.03 (0.3)	0.21 (0.8)	-1.22 (0.2)	0.18 (0.9)	-0.50 (0.6)
CACD	1.03 (0.3)	x	1.29 (0.21)	-0.23 (0.8)	1.16 (0.3)	0.50 (0.6)
Dogs	-0.21 (0.8)	-1.29 (0.21)	x	-1.49 (0.2)	-0.01 (1.0)	-0.73 (0.5)
Facescrub	1.22 (0.2)	0.23 (0.8)	1.49 (0.2)	x	1.33 (0.2)	0.70 (0.5)
ImageNet	-0.18 (0.9)	-1.16 (0.3)	0.01 (1.0)	-1.33 (0.2)	x	-0.66 (0.5)
CASIA	0.50 (0.6)	-0.50 (0.6)	0.73 (0.5)	-0.70 (0.5)	0.66 (0.5)	x

Table 5.11: Paired t-test performed on the results of blocks 1-8 and 2-8 using Inception Resnet V2 on the identification task, which is shown as the identification rate in table 5.2. The first value is the t-distribution and the second value is the p-value of this t-distribution. A positive t-distribution means that the dataset in the row has a higher identification rate than the dataset in the column header.

	Birds	**CACD**	**Dogs**	**Facescrub**	**ImageNet**	**CASIA**
Birds	x	-0.69 (0.5)	-0.03 (1.0)	-0.60 (0.6)	-0.30 (0.8)	-1.87 (0.08)
CACD	0.69 (0.5)	x	0.64 (0.5)	0.10 (0.92)	0.37 (0.7)	-1.30 (0.2)
Dogs	0.03 (1.0)	-0.64 (0.5)	x	-0.55 (0.60)	-0.26 (0.80)	-1.75 (0.1)
Facescrub	0.60 (0.6)	-0.1 (0.9)	0.55 (0.6)	x	0.28 (0.8)	-1.38 (0.2)
ImageNet	0.30 (0.8)	-0.38 (0.7)	0.26 (0.8)	-0.28 (0.8)	x	-1.52 (0.2)
CASIA	1.87 (0.08)	1.30 (0.2)	1.75 (0.1)	1.38 (0.2)	1.52 (0.2)	x

Identification. The results using the Inception V1 on the identification task of the LFW dataset are presented in table 5.1 and the significance of the results are in table 5.10. The significance table

shows that the results, only using block 2-8 is not significant for the identification task. All the results of the p-values on the identification are above 0.05, which means that the null hypothesis is accepted and therefore the differences between the results are not significant.

The significance of the results for the identification test using Inception ResnetV2 are presented in table 5.11. This table shows that all the comparisons between the models pre-trained with different datasets, the null hypothesis cannot be rejected. All the p-values are above the 0.05, which means that the null hypothesis for all the comparisons are accepted and therefore all the differences between the results for the identification task are not significant.

5.2.2 Swapping the first blocks

In this section, we investigate if putting the first few layers of the best performing pre-trained model in the worst performing pre-trained model can improve the results. To explain: the first n blocks are taken from the best performing pre-trained models, then n blocks from the worst performing pre-trained models are taken and the last blocks are trained with the CASIA dataset. Table 5.1 shows that the differences in block 4-5 and 6-8 are significant. Swapping certain blocks from a dataset A pre-trained model with a dataset B pre-trained model should not have an influence if the layers are universal. This is investigated in this section on two evaluation datasets for the sub-question: *is there a difference in results when swapping the first layers of the pre-trained models?* The experimental setup is described in subsection 4.2.3. We evaluate the verification task and identification task on the LFW dataset. The idea is to boost the performance of the two worst performing pre-trained models (Birds, Dogs) with the first n blocks of the best performing pre-trained models (Facescrub, CACD).

Verification. The results on the LFW verification task using the Inception V1 with swapping layers are presented in table 5.12. A row in column *blocks* in the table shows which blocks belongs to a dataset A and which blocks belongs to a dataset B. For example, with blocks: 1 - 2, 3, 4 is meant that block 1 is used from the model pre-trained on a dataset A and blocks 2, 3 and 4 are used from the model pre-trained on a dataset B. Table 5.12 shows that swapping of the first block has not improved the results for the Birds pre-trained model. The accuracy and AUC are less than the results of 4-5 (blocks: 1, 2, 3, 4).This could suggest that using only the first block of the model pre-trained on CACD or Facescrub is not enough to improve the results.

Swapping of the first two blocks has a little improvement in the accuracy and AUC for the model pre-trained on Birds. This is the case for the model pre-trained on CACD (88.4% compared to 87.4%) and Facescrub (88.2% compared to 87.4%). From these results, we cannot conclude that universal features improve the results because the performance between the swapped models is not enough. The results of the model pre-trained on Dogs reveal that the pre-trained model did not capture facial features. Swapping the first two blocks on the pre-trained model obtains the similar results as the original pre-trained model with block 4-5 using Dogs. This indicates that the weights of the third and/or fourth block of the pre-trained model on Dogs do not contain usable features for these face recognition tasks.

Table 5.13 shows the results with swapping layers using the Inception Resnet V2 pre-trained models. Swapping the first block has a small influence on the accuracy of the model pre-trained on Birds. The model pre-trained on Birds with the swapped weights of the model pre-trained on CACD improved with $\sim 2.1\%$ accuracy and 0.009 AUC. The swapped weights of the model pre-trained on Facescrub improved $\sim 1.5\%$ accuracy and 0.006 AUC compared with the model pre-trained on Birds with block 6-8. The improvement of the model pre-trained on Dogs with swapping the first block is negligible to the pre-trained model on Dogs with block 6-8. Adding blocks 2 and 3 to the swapped weights obtains a better improvement, compared with the model pre-trained on Birds and Dogs. Swapping blocks 1 and 2 to the model pre-trained on Birds and Dogs obtained an increase of $\sim 4-5\%$. The best result is obtained when swapping blocks 1,2 and 3 with an increase of $\sim 5-6\%$. These results suggest that swapping only the first block is not enough to get a significant increase. But the features from blocks 2 and/or block 3 obtained the worst performing pre-trained model an increase in performance.

Identification. On the identification task, we notice the same pattern as for the verification task using the Inception V1. The results in table 5.12 present that the performances with the model pre-trained on Dogs on block 4-5 are still worse. The results show that swapping the first block on the

model pre-trained on Birds does not improve the model compared with the block 4-5 model pre-trained on Birds. The difference between using swapping the model pre-trained on CACD or Facescrub is negligible. However swapping the first two blocks improves the identification rate using layers of the model pre-trained on CACD and Facescrub. The increase of the performance is 4.3% and 4.6%. This gives the same results as the verification task.

Table 5.13 reveals the results using the Inception Resnet V2. The results are a little different from the verification task. The results on the identification task shows that swapping only the first block of the pre-trained models does not improve the model in comparison with the 6-8 models pre-trained on Dogs and Birds. Table 5.13 shows that swapping the first two and three blocks increases the identification rate. The results for the models pre-trained on Birds have an increase of $\sim 5.5\%$ when swapping the first two blocks and $\sim 6.5\%$ when swapping the first three blocks. The increase on the models pre-trained on Dogs is a little lower with $\sim 1.0\%$ and $\sim 3.0\%$.

Table 5.12: Results of the verification and identification task on the LFW benchmark with swapping layers of the Inception V1 model. The column **Blocks** defines which blocks are taken from a dataset A pre-trained model and from a dataset B pre-trained model. The rest of the blocks are trained with the CASIA dataset. The metrics used, are the accuracy, the area under the curve (AUC) and identification rate. For example, 1 - 2, 3, 4 (CACD, Birds) is the model where the first block is the CACD pre-trained moded and blocks 2, 3 and 4 are from the Birds dataset. Blocks 5 and 6 are trained with the CASIA dataset.

Blocks	Dataset (A)	Dataset (B)	Accuracy (%)	AUC	Identification rate (%)
1 - 2,3,4	CACD	Birds	86.6 +-1.6	0.942	69.9 +-16.0
1 - 2,3,4	CACD	Dogs	50.0 +- 0.0	0.500	0.7 +- 0.6
1 - 2,3,4	Facescrub	Birds	86.1 +- 1.6	0.938	68.0 +- 17.0
1 - 2,3,4	Facescrub	Dogs	50.0 +-0.0	0.500	0.8 +-0.5
1,2 - 3,4	CACD	Birds	88.4+-1.5	0.953	76.8+-17.0
1,2 - 3,4	CACD	Dogs	50.0+-0.0	0.500	0.7+-0.6
1,2 - 3,4	Facescrub	Birds	88.2+-1.8	0.952	76.5+-17.0
1,2 - 3,4	Facescrub	Dogs	50.0+-0.0	0.5	0.7+-0.6
1,2,3,4	Birds	Birds	87.4+-2.2	0.946	72.2+-16.7
1,2,3,4	Dogs	Dogs	50.0+-0.0	0.5	0.7+-0.5

Table 5.13: Results of the verification and identification task on the LFW benchmark with swapping layers of the Inception Resnet V2 model. The column **Blocks** defines which blocks are taken from a dataset A pre-trained model and from a dataset B pre-trained model. The rest of the blocks are trained with the CASIA dataset. The metrics used, are the accuracy, the area under the curve (AUC) and identification rate. For example, 1 - 2, 3, 4, 5, 6 (CACD, Birds) is the model where the first block is the CACD pre-trained moded and blocks 2, 3, 4, 5, 6 are from the Birds dataset. Blocks 7 and 8 are trained with the CASIA dataset.

Blocks	Dataset (A)	Dataset (B)	Accuracy (%)	AUC	Identification rate (%)
1 - 2,3,4,5,6	CACD	Birds	93.7+- 1.0	0.982	88.1 +- 12.1
1 - 2,3,4,5,6	CACD	Dogs	93.1 +- 1.1	0.982	86.7 +- 12.2
1 - 2,3,4,5,6	Facescrub	Birds	93.2 +-1.2	0.979	88.2 +-12.4
1 - 2,3,4,5,6	Facescrub	Dogs	93.3 +- 1.1	0.982	87.9 +- 12.3
1,2 - 3,4,5,6	CACD	Birds	95.8+-1.1	0.992	94.0+-7.2
1,2 - 3,4,5,6	CACD	Dogs	96.0+-0.7	0.993	93.6+-7.6
1,2 - 3,4,5,6	Facescrub	Birds	95.9+-1.1	0.991	93.8+- 7.7
1,2 - 3,4,5,6	Facescrub	Dogs	95.9+-0.9	0.992	93.5+-7.8
1,2,3 - 4,5,6	CACD	Birds	96.5+-1.2	0.994	95.9 +-5.3
1,2,3 - 4,5,6	CACD	Dogs	96.8+-0.8	0.995	96.0 +-5.5
1,2,3 - 4,5,6	Facescrub	Birds	96.2+-1.1	0.993	95.5 +- 6.2
1,2,3 - 4,5,6	Facescrub	Dogs	96.3+-0.7	0.994	95.2 +- 6.6
1,2,3,4,5,6	Birds	Birds	91.6+-1.5	0.973	89.4+-10.5
1,2,3,4,5,6	Dogs	Dogs	92.8+-1.0	0.981	92.4+-8.8

Chapter 6

Discussion and Conclusion

The discussion and conclusion are described in this chapter, with in section 6.1 answers to the two research questions of this thesis and in section 6.2 the conclusion.

6.1 Discussion

The two research questions of this thesis are: *Do models pre-trained on face datasets achieve better results in comparison with non-face datasets on face recognition?* and *Do first few layers have an influence on the results in face recognition?*. In the following subsections 6.1.1 and 6.1.2 the answers are given to these research questions.

6.1.1 Comparison of results with models pre-trained on face datasets and pre-trained on non-face datasets

To answer the first research question of this thesis, we looked at two different approaches. For the first approach, we investigated the results on models pre-trained on face datasets and on non-face datasets. These results were split by the number of blocks with pre-trained frozen layers. It was done on three different face recognition tasks and two different architectures.

The results using the Inception V1 for all the three face recognition tasks showed that the model pre-trained on face datasets achieved better results than models pre-trained on non-face datasets. This is an extension of the results of the research by Yosinski et al. and a contradiction of the results that Huh et al. achieved [30, 82]. The results using the Inception V1 showed that using more layers for pre-training made the difference in results between the different pre-trained models bigger. The models pre-trained on face datasets obtain better results than the models pre-trained on non-face datasets.

The results between the two models pre-trained on face datasets are comparable and therefore the difference is negligible. This is surprising because the CACD dataset is $\sim 70K$ images and $\sim 1.5K$ classes bigger than the Facescrub dataset. Huh et al. showed in their research that using more images and classes for pretraining and training, increases the performance of the model. This effect is not visible when we used two different face datasets for pre-training. This has may be to do with the difference between the size of the datasets. The difference in size between the Facescrum and CACD is not even a factor 10. Using the Inception V1, we noticed this effect if we look at the results of the models pre-trained on non-face datasets. The models pre-trained on ImageNet outperforms the models pre-trained on Birds and Dogs dataset. The ImageNet dataset has 60x more images than the Dogs dataset and 200x more images than the Birds dataset. This pattern is not visible between the results of models pre-trained with Dogs and Birds, in which the Dogs dataset is bigger than the Birds dataset. The reason may be that the number of images in the ImageNet surpassed a limit in which the difference becomes visible, which the Dogs and Birds dataset does not surpass. If the size of datasets for pre-training are influencing the results. This should be investigated more to get a better insight.

The results on the two face recognition tasks using the Inception Resnet V2 also showed that the models pre-trained with face datasets perform better than models pre-trained on non-face datasets. In the results of the MegaFace challenge 1, we did not find this pattern. The results of the challenge are not consistent and therefore we can not make any conclusions. The results of the tasks on the LFW benchmark also showed that using more pre-trained layers resulted in bigger differences between models pre-trained on face datasets and on non-face datasets. The performances of the models pre-trained on the face datasets show that both models obtain comparable results and therefore the difference in results is negligible. Also, the results using the models pre-trained on non-face datasets is comparable and therefore this difference is negligible. This is interesting because we did find this for the Inception V1 model so there is a difference between the two models. It is even more interesting that models pre-trained on a 200x smaller dataset achieve the same results as models pre-trained on the ImageNet dataset. Which is a contradiction with the results that Huh et al. found in their research, in which they stated that pre-training on more images results in better performances [30]. The question is: Why we should spent more time on pre-training on big datasets if the same features can be taken from smaller datasets?

The results using the Inception V1 showed that the models pre-trained on face datasets outperform the model that is fully trained on the CASIA dataset which is interesting if we look at the size of the datasets. The CASIA dataset has 5x more images than the Facescrub dataset and 3x more images than the CACD dataset. So the layers of the models that are fully trained with the CASIA dataset should have learned more facial features because it got more data in comparison with the model that is pre-trained with Facescrub or CACD. In the results using the Inception Resnet V2 this is visible. The model fully trained on the CASIA dataset outperforms all the pre-trained models. For now we can not make conclusion based on two models. To make a conclusion based on the different types of models, we need to investigate more different type of models or more of the same type of models.

For the second approach, we investigated the robustness of models pre-trained on face datasets and on non-face datasets. The drop percentage using the Inception V1 showed that the models pre-trained on face datasets are more robust than the models pre-trained on non-face datasets. This is not the case in δB_{23} between the model pre-trained on Facescrub and ImageNet, where the model pre-trained on ImageNet is more robust. But the δB_{24} showed that models pre-trained on face datasets are more robust. Within models pre-trained on face datasets there is also a difference in robustness. The model pre-trained on CACD has $\sim$ 2.0% and $\sim$ 2.7% more δB_{24} than the model pre-trained on Facescrub. Within the models pre-trained on non-face datasets, the models pre-trained on Dogs is the least robust, followed by Birds and then ImageNet dataset.

The robustness of the Inception Resnet V2 between the pre-trained models showed that the δB_{23}, δB_{24} and δB_{25} are robust for all the pre-trained models. The differences in these drop percentages were negligible. The drop percentage δB_{27} has the biggest difference between the pre-trained models. The models pre-trained on face datasets are more robust than the models pre-trained on non-face datasets. The difference between the results using the two models pre-trained on face datasets is negligible. This is not the case with the models pre-trained on non-face datasets. The models pre-trained on Dogs is more robust than the models pre-trained on ImageNet dataset, which is a surprising result. The model pre-trained on the Birds dataset is the least robust model.

In the two approaches the models pre-trained on face datasets perform better than the models pre-trained on non-face datasets. The difference is more visible when more used for pre-training (blocks 4-5 and 7-8). This is an indication that the facial features are better captured with the models pre-trained on face datasets in comparison with the models pre-trained on non-face datasets. The training of the last blocks of the models pre-trained with non-face datasets could not capture all the facial features that are needed to get the same results as the models pre-trained on face datasets. With these findings we can answer the first main research question of this thesis.

We showed in two ways that the models pre-trained on face datasets obtained better results and are more robust than models pre-trained on non-face datasets. For sub-questions the answer is yes and therefore the answer for the first research question also yes. Using pre-trained Inception V1 models, we noticed that a difference in dataset size can make a difference in the performance. The models pre-trained on ImageNet obtained better results than the models pre-trained on Dogs and Birds. However we did not find this result using the Inception Resnet V2. This can be an indication that the features captured

from a model pre-trained on non-face datasets can already be derived from a small dataset like the Birds dataset. Therefore pre-training on big datasets like ImageNet might not be needed.

6.1.2 Universal features

The second research question of this thesis is: *Do the first few layers have an influence on the results in face recognition?* This research question is separated into two sub-questions and the two approaches were investigated in the previous chapter. In the first approach the results of blocks 1-5, 2-5, 1-8 and 2-8 from the pre-trained models were compared.

In subsection 5.2.1 the results of the first blocks were compared in accuracy, identification rate and significance. It is shown that there is a significant difference between the results using pre-trained layers in the first block. Using the Inception V1 and the Inception Resnet V2, it was shown that the models pre-trained on face datasets had a slightly better performance than the non-face datasets. This gives an indication that the first few layers of the models are influencing the results. This is in contradiction with the results found by a lot of researchers (for example Yosinski et al. [82]). They argued that the first few layers of a model are often Gabor filters and/or color blobs and can be used to transfer the knowledge of that pre-trained model to another not yet trained model. Our results have however shown that the first block of pre-trained layers already influences the results significantly.

It was said by for example Huh et al. that it is interesting to pre-train the models with the ImageNet dataset because this dataset consists of many different image classes [30]. This implies that the first few layers captures universal features of different image classes and it would improve the models that are used for transfer learning. This means that the first few layers of the models pre-trained with ImageNet can be used for another target objective. In our results, we saw the opposite results on face recognition. It is still better to use a face dataset for pre-training if the target objective is face recognition. For the identification task the results were not significant and therefore we will not discuss them. The answer to the sub-question belonging to this approach is: yes. The first pre-trained layers have an influence on the results on face recognition. Therefore a face dataset should be used to pre-train the layers if the target objective is face recognition.

Huh et al. also argued that using a bigger dataset for pre-training would improve the performance of the target objective [30]. This argument has not been investigated in this thesis. The Facescrub and CACD dataset are smaller than the CASIA dataset but for future research, it would be interesting if it makes a difference. This might also increase the overall performance using the Inception Resnet V2 because the first few layers are then pre-trained with a bigger dataset and therefore capture more facial features in the first few layers. If the rest then is trained with the CASIA dataset, it could improve the overall performance. But this is still guessing and needs to be investigated more.

The second approach was to swap the first layers of the best performing pre-trained models (CACD, Facescrub) with the worst performing pre-trained models (Dogs, Birds) and training the last blocks of layers with the CASIA dataset. As far as known by us, the swapping of layers from one pre-trained model into another model has not been done before. Veit et al. and Greff et al. removed and shuffled nodes in the model, which is also a form of modifying the model [25, 77]. They showed that this did not influence the residual models like the Inception Resnet V2 but it did influence the non-residual models, like the Inception V1 model. If we look at the results of swapping the layers in subsection 5.2.2, then we also see that the Inception V1 model was not improving using the first blocks of layers from the best performing pre-trained model. The results using the Inception Resnet V2 did show an improvement when using the first two or three blocks of layers from the best performing pre-trained model.

This might be an indication of the *unrolled iterative estimation* view that Greff et al. proposed. In this view, they say that in every layer the model is building features upon each other. This means that the next layer is building upon the feature map of the previous layer. This is different than the common representation view in which every layer of the model is responsible for a certain level of representation of the data. If we follow the unrolled iterative estimation view then it makes sense that the Inception Resnet V2 performs better by swapping because the model is building up from the features that are extracted from the best performing pre-trained model. If this is true then the first blocks of pre-trained layers have influence on face recognition because it improves the results and this is what we saw in the

experiments in subsection 5.2.2. The universal features have an influence on the results of the Inception Resnet V2, so it makes a difference which pre-trained model is used for the universal features. To answer the sub-question: *Is there a difference in performance when swapping the first layers of the pre-trained models?* Only for the results with the Inception Residual Network V2, we noticed an improvement when swapping the first few layers of the best performing pre-trained model. In the results of the Inception V1 we did not notice any improvement.

The two approaches showed that the first few layers make a difference in transfer learning. Therefore the second main research question: *Do the first few layers have an influence on the results in face recognition?* is answered with yes. However, we did see that this was the case for all the experiments. The identification tasks did not result in a significant difference between the results. Swapping of the layers using the Inception V1 did not improve the model.

6.2 Conclusion

Pre-trained models are often used without the complete understanding how to use them for transfer learning. In this thesis, we investigated and described certain aspects of pre-trained models that influence the performance on face recognition with two different architectures. We showed in the first part that using face datasets for pre-training increases the performance on face recognition in comparison with, for example, the models pre-trained on the famous ImageNet dataset. We also showed that models pre-trained on face datasets are more robust than the non-face datasets. Another component of transfer learning is the use of universal features, which are features that can be used for every model. In the second part of the thesis, we investigated these universal features for face recognition. In the verification task it was shown that there was a significant difference between results using the pre-trained universal features which indicate that these features might not be universal. In this part, we also experimented with swapping the first few layers of the pre-trained model and the Inception Resnet V2 increased in performance. In this thesis, transfer learning has been investigated a bit more, but there are still some aspects that need to be investigated deeper to get a complete view.

Bibliography

[1] Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., Corrado, G. S., Davis, A., Dean, J., Devin, M., Ghemawat, S., Goodfellow, I., Harp, A., Irving, G., Isard, M., Jia, Y., Jozefowicz, R., Kaiser, L., Kudlur, M., Levenberg, J., Mané, D., Monga, R., Moore, S., Murray, D., Olah, C., Schuster, M., Shlens, J., Steiner, B., Sutskever, I., Talwar, K., Tucker, P., Vanhoucke, V., Vasudevan, V., Viégas, F., Vinyals, O., Warden, P., Wattenberg, M., Wicke, M., Yu, Y., and Zheng, X. (2015). Tensorflow: Large-scale machine learning on heterogeneous systems. Software available from tensorflow.org.

[2] Abate, A. F., Nappi, M., Riccio, D., and Sabatino, G. (2007). 2d and 3d face recognition: A survey. *Pattern Recognition Letters*, 28(14):1885–1906.

[3] Ahonen, T., Hadid, A., and Pietikainen, M. (2006). Face description with local binary patterns: Application to face recognition. *IEEE transactions on pattern analysis and machine intelligence*, 28(12):2037–2041.

[4] Amos, B., Ludwiczuk, B., and Satyanarayanan, M. (2016). Openface: A general-purpose face recognition library with mobile applications. Technical report, CMU-CS-16-118, CMU School of Computer Science.

[5] Arora, S., Bhaskara, A., Ge, R., and Ma, T. (2014). Provable bounds for learning some deep representations. In *International Conference on Machine Learning*, pages 584–592.

[6] Bartlett, M. S., Movellan, J. R., and Sejnowski, T. J. (2002). Face recognition by independent component analysis. *IEEE Transactions on neural networks*, 13(6):1450–1464.

[7] Bengio, Y. (2012). Deep learning of representations for unsupervised and transfer learning. In *Proceedings of ICML Workshop on Unsupervised and Transfer Learning*, pages 17–36.

[8] Bengio, Y., Goodfellow, I. J., and Courville, A. (2015). Deep learning. *An MIT Press book in preparation. Draft chapters available at http://www. iro. umontreal. ca/ bengioy/dlbook.*

[9] Bottou, L. (2010). Large-scale machine learning with stochastic gradient descent. In *Proceedings of COMPSTAT'2010*, pages 177–186. Springer.

[10] Bowyer, K. W., Chang, K., and Flynn, P. (2006). A survey of approaches and challenges in 3d and multi-modal 3d+ 2d face recognition. *Computer vision and image understanding*, 101(1):1–15.

[11] Carreira, J., Agrawal, P., Fragkiadaki, K., and Malik, J. (2016). Human pose estimation with iterative error feedback. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4733–4742.

[12] Cauchy, A. (1847). Méthode générale pour la résolution des systemes déquations simultanées. *Comp. Rend. Sci. Paris*, 25(1847):536–538.

[13] Cevikalp, H., Neamtu, M., Wilkes, M., and Barkana, A. (2005). Discriminative common vectors for face recognition. *IEEE Transactions on pattern analysis and machine intelligence*, 27(1):4–13.

[14] Chen, B.-C., Chen, C.-S., and Hsu, W. H. (2015). Face recognition and retrieval using cross-age reference coding with cross-age celebrity dataset. *IEEE Transactions on Multimedia*, 17(6):804–815.

[15] Chopra, S., Hadsell, R., and LeCun, Y. (2005). Learning a similarity metric discriminatively, with application to face verification. In *Computer Vision and Pattern Recognition. CVPR 2005. IEEE Computer Society Conference on*, volume 1, pages 539–546.

[16] Cinbis, R. G., Verbeek, J., and Schmid, C. (2011). Unsupervised metric learning for face identification in tv video. In *2011 International Conference on Computer Vision*, pages 1559–1566.

[17] Cireşan, D. C., Meier, U., and Schmidhuber, J. (2012). Transfer learning for Latin and Chinese characters with deep neural networks. In *Neural Networks (IJCNN), The 2012 International Joint Conference on*, pages 1–6.

[18] Cottrell, G. W. and Fleming, M. (1990). Face recognition using unsupervised feature extraction. In *Proc. of the Int. Neural Network Conf*, pages 322–325.

[19] Dalal, N. and Triggs, B. (2005). Histograms of oriented gradients for human detection. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 886–893.

[20] Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*, pages 248–255.

[21] Donahue, J., Anne Hendricks, L., Guadarrama, S., Rohrbach, M., Venugopalan, S., Saenko, K., and Darrell, T. (2015). Long-term recurrent convolutional networks for visual recognition and description. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2625–2634.

[22] Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., Tzeng, E., and Darrell, T. (2014). Decaf: A deep convolutional activation feature for generic visual recognition. In *International conference on machine learning*, pages 647–655.

[23] Erhan, D., Szegedy, C., Toshev, A., and Anguelov, D. (2014). Scalable object detection using deep neural networks. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2147–2154.

[24] Girshick, R., Donahue, J., Darrell, T., and Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587.

[25] Greff, K., Srivastava, R. K., and Schmidhuber, J. (2016). Highway and residual networks learn unrolled iterative estimation. *arXiv preprint arXiv:1612.07771*.

[26] Guo, Y., Zhang, L., Hu, Y., He, X., and Gao, J. (2016). Ms-celeb-1M: challenge of recognizing one million celebrities in the real world. *Electronic Imaging*, 2016(11):1–6.

[27] He, K., Zhang, X., Ren, S., and Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778.

[28] Hinton, G. E., Srivastava, N., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. R. (2012). Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*.

[29] Huang, G. B., Ramesh, M., Berg, T., and Learned-Miller, E. (2007). Labeled faces in the wild: A database for studying face recognition in unconstrained environments. Technical Report 07-49, University of Massachusetts, Amherst.

[30] Huh, M., Agrawal, P., and Efros, A. A. (2016). What makes Imagenet good for transfer learning? *arXiv preprint arXiv:1608.08614*.

[31] Ioffe, S. and Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International Conference on Machine Learning*, pages 448–456.

[32] Karpathy, A. and Fei-Fei, L. (2015). Deep visual-semantic alignments for generating image descriptions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 3128–3137.

[33] Kemelmacher-Shlizerman, I., Seitz, S. M., Miller, D., and Brossard, E. (2016). The MegaFace benchmark: 1 million faces for recognition at scale. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4873–4882.

[34] Khosla, A., Jayadevaprakash, N., Yao, B., and Fei-Fei, L. (2011). Novel dataset for fine-grained image categorization. In *First Workshop on Fine-Grained Visual Categorization, IEEE Conference on Computer Vision and Pattern Recognition*, Colorado Springs, CO.

[35] Kirby, M. and Sirovich, L. (1990). Application of the Karhunen-Loeve procedure for the characterization of human faces. *IEEE Transactions on Pattern analysis and Machine intelligence*, 12(1):103–108.

[36] Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

[37] Lawrence, S., Giles, C. L., Tsoi, A. C., and Back, A. D. (1997). Face recognition: A convolutional neural-network approach. *IEEE transactions on neural networks*, 8(1):98–113.

[38] LeCun, Y. and Bengio, Y. (1995). Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10):1995.

[39] Lin, M., Chen, Q., and Yan, S. (2013). Network in network. *arXiv preprint arXiv:1312.4400*.

[40] Lin, S.-H., Kung, S.-Y., and Lin, L.-J. (1997). Face recognition/detection by probabilistic decision-based neural network. *IEEE transactions on neural networks*, 8(1):114–132.

[41] Lipton, Z. C., Berkowitz, J., and Elkan, C. (2015). A critical review of recurrent neural networks for sequence learning. *arXiv preprint arXiv:1506.00019*.

[42] Liu, C. and Wechsler, H. (2002). Gabor feature based classification using the enhanced Fisher linear discriminant model for face recognition. *IEEE Transactions on Image processing*, 11(4):467–476.

[43] Liu, Z., Luo, P., Wang, X., and Tang, X. (2015). Deep learning face attributes in the wild. In *Proceedings of International Conference on Computer Vision (ICCV)*.

[44] Lowe, D. G. (2004). Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110.

[45] Lu, J., Plataniotis, K. N., and Venetsanopoulos, A. N. (2003). Face recognition using LDA-based algorithms. *IEEE Transactions on Neural networks*, 14(1):195–200.

[46] Martínez, A. M. and Kak, A. C. (2001). PCA versus LDA. *IEEE transactions on pattern analysis and machine intelligence*, 23(2):228–233.

[47] Nagi, J., Ducatelle, F., Di Caro, G. A., Cireşan, D., Meier, U., Giusti, A., Nagi, F., Schmidhuber, J., and Gambardella, L. M. (2011). Max-pooling convolutional neural networks for vision-based hand gesture recognition. In *Signal and Image Processing Applications (ICSIPA), 2011 IEEE International Conference on*, pages 342–347.

[48] Nair, V. and Hinton, G. E. (2010). Rectified linear units improve restricted Boltzmann machines. In *Proceedings of the 27th International Conference on Machine Learning (ICML-10)*, pages 807–814.

[49] Nech, A. and Kemelmacher-Shlizerman, I. (2017). Level Playing Field For Million Scale Face Recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*.

[50] Ng, H.-W. and Winkler, S. (2014). A data-driven approach to cleaning large face datasets. In *Image Processing (ICIP), IEEE International Conference on*, pages 343–347.

[51] Oquab, M., Bottou, L., Laptev, I., and Sivic, J. (2014). Learning and transferring mid-level image representations using convolutional neural networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1717–1724.

[52] Osler, T. J. (1970). *Leibniz Rule, the Chain Rule, and Taylor's Theorem for Fractional Derivatives*. PhD thesis, New York University, Graduate School of Arts and Science.

[53] Pan, S. J. and Yang, Q. (2010). A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, 22(10):1345–1359.

[54] Parkhi, O. M., Simonyan, K., Vedaldi, A., and Zisserman, A. (2014). A compact and discriminative face track descriptor. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1693–1700.

[55] Phillips, P. J., Beveridge, J. R., Draper, B. A., Givens, G., O'Toole, A. J., Bolme, D. S., Dunlop, J., Lui, Y. M., Sahibzada, H., and Weimer, S. (2011). An introduction to the good, the bad, & the ugly face recognition challenge problem. In *Automatic Face & Gesture Recognition and Workshops (FG 2011), IEEE International Conference on*, pages 346–353.

[56] Qian, N. (1999). On the momentum term in gradient descent learning algorithms. *Neural networks*, 12(1):145–151.

[57] Rothe, R., Timofte, R., and Gool, L. V. (2015). Dex: Deep expectation of apparent age from a single image. In *IEEE International Conference on Computer Vision Workshops (ICCVW)*.

[58] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1985). Learning internal representations by error propagation. Technical report, DTIC Document.

[59] Schroff, F., Kalenichenko, D., and Philbin, J. (2015). Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 815–823.

[60] Sharif Razavian, A., Azizpour, H., Sullivan, J., and Carlsson, S. (2014). CNN features off-the-shelf: an astounding baseline for recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition workshops*, pages 806–813.

[61] Simonyan, K. and Zisserman, A. (2014a). Two-stream convolutional networks for action recognition in videos. In *Advances in neural information processing systems*, pages 568–576.

[62] Simonyan, K. and Zisserman, A. (2014b). Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*.

[63] Sivic, J., Everingham, M., and Zisserman, A. (2005). Person spotting: video shot retrieval for face sets. In *International Conference on Image and Video Retrieval*, pages 226–236. Springer.

[64] Sivic, J., Everingham, M., and Zisserman, A. (2009). Who are you? -learning person specific classifiers from video. In *Computer Vision and Pattern Recognition. IEEE Conference on*, pages 1145–1152.

[65] Srivastava, N., Hinton, G. E., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. *Journal of machine learning research*, 15(1):1929–1958.

[66] Sun, Y., Chen, Y., Wang, X., and Tang, X. (2014a). Deep learning face representation by joint identification-verification. In *Advances in neural information processing systems*, pages 1988–1996.

[67] Sun, Y., Wang, X., and Tang, X. (2014b). Deep learning face representation from predicting 10,000 classes. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1891–1898.

[68] Sun, Y., Wang, X., and Tang, X. (2015). Deeply learned face representations are sparse, selective, and robust. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2892–2900.

[69] Sutton, R. S. (1986). Two problems with backpropagation and other steepest-descent learning procedures for networks. In *Proc. 8th annual conf. cognitive science society*, pages 823–831. Erlbaum.

[70] Svozil, D., Kvasnicka, V., and Pospichal, J. (1997). Introduction to multi-layer feed-forward neural networks. *Chemometrics and intelligent laboratory systems*, 39(1):43–62.

[71] Szegedy, C., Ioffe, S., Vanhoucke, V., and Alemi, A. A. (2017). Inception-v4, inception-resnet and the impact of residual connections on learning. In *AAAI*, pages 4278–4284.

[72] Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., and Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–9.

[73] Taigman, Y., Yang, M., Ranzato, M., and Wolf, L. (2014). Deepface: Closing the gap to human-level performance in face verification. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1701–1708.

[74] Tang, Y., Peng, L., Xu, Q., Wang, Y., and Furuhata, A. (2016). CNN based transfer learning for historical Chinese character recognition. In *Document Analysis Systems (DAS) 12th IAPR Workshop on*, pages 25–29.

[75] Tieleman, T. and Hinton, G. (2012). Lecture 6.5-RMSPROP: Divide the gradient by a running average of its recent magnitude. *COURSERA: Neural networks for machine learning*, 4(2).

[76] Van de Wolfshaar, J., Karaaba, M. F., and Wiering, M. A. (2015). Deep convolutional neural networks and support vector machines for gender recognition. In *Computational Intelligence, IEEE Symposium Series on*, pages 188–195.

[77] Veit, A., Wilber, M., and Belongie, S. (2016). Residual networks are exponential ensembles of relatively shallow networks. *arXiv preprint arXiv:1605.06431*, 1.

[78] Weinzaepfel, P., Revaud, J., Harchaoui, Z., and Schmid, C. (2013). Deepflow: Large displacement optical flow with deep matching. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 1385–1392.

[79] Welinder, P., Branson, S., Mita, T., Wah, C., Schroff, F., Belongie, S., and Perona, P. (2010). Caltech-UCSD Birds 200. Technical Report CNS-TR-2010-001, California Institute of Technology.

[80] Wen, Y., Zhang, K., Li, Z., and Qiao, Y. (2016). A discriminative feature learning approach for deep face recognition. In *European Conference on Computer Vision*, pages 499–515. Springer.

[81] Yi, D., Lei, Z., Liao, S., and Li, S. Z. (2014). Learning face representation from scratch. *arXiv preprint arXiv:1411.7923*.

[82] Yosinski, J., Clune, J., Bengio, Y., and Lipson, H. (2014). How transferable are features in deep neural networks? In *Advances in neural information processing systems*, pages 3320–3328.

[83] Zhang, K., Zhang, Z., Li, Z., and Qiao, Y. (2016). Joint face detection and alignment using multitask cascaded convolutional networks. *IEEE Signal Processing Letters*, 23(10):1499–1503.

[84] Zhao, W., Chellappa, R., Phillips, P. J., and Rosenfeld, A. (2003). Face recognition: A literature survey. *ACM computing surveys (CSUR)*, 35(4):399–458.

[85] Zhu, Z., Luo, P., Wang, X., and Tang, X. (2014). Recover canonical-view faces in the wild with deep neural networks. *arXiv preprint arXiv:1404.3543*.