Master of Science Thesis in Electrical Engineering
Department of Electrical Engineering, Linköping University, 2016

Face Recognition with Preprocessing and Neural Networks

David Habrman

Master of Science Thesis in Electrical Engineering

Face Recognition with Preprocessing and Neural Networks

David Habrman

LiTH-ISY-EX--16/4953--SE

Supervisor: **Giulia Meneghetti**
ISY, Linköping University
Fredrik Noring
Combitech

Examiner: **Per-Erik Forssén**
ISY, Linköping University

Computer Vision Laboratory
Department of Electrical Engineering
Linköping University
SE-581 83 Linköping, Sweden

Abstract

Face recognition is the problem of identifying individuals in images. This thesis evaluates two methods used to determine if pairs of face images belong to the same individual or not. The first method is a combination of principal component analysis and a neural network and the second method is based on state-of-the-art convolutional neural networks. They are trained and evaluated using two different data sets. The first set contains many images with large variations in, for example, illumination and facial expression. The second consists of fewer images with small variations.

Principal component analysis allowed the use of smaller networks. The largest network has 1.7 million parameters compared to the 7 million used in the convolutional network. The use of smaller networks lowered the training time and evaluation time significantly. Principal component analysis proved to be well suited for the data set with small variations outperforming the convolutional network which need larger data sets to avoid overfitting. The reduction in data dimensionality, however, led to difficulties classifying the data set with large variations. The generous amount of images in this set allowed the convolutional method to reach higher accuracies than the principal component method.

Acknowledgments

I would like to take this opportunity to thank Combitech AB for the possibility to perform my master thesis. Special thanks to my supervisor at Combitech, Fredrik Noring, for technical help and support. Additionally I would like to thank my supervisor at ISY, Giulia Meneghetti, for help with the report. Finally, thanks to my examiner Per-Erik Forssén for all the help and encouragement.

Linköping, June 2016
David Habrman

Contents

Notation

ABBREVIATIONS

Abbreviation	Description
CNN	Convolutional neural network
LFW	Labeled faces in the wild
PCA	Principal component analysis
ReLU	Rectified linear unit
ROC	Receiver operating characteristic

1 Introduction

1.1 Motivation

Inside a computer, images are represented by matrices with numbers corresponding to color. The process of categorizing images into one or more classes is called image classification and is a difficult problem for a machine. There are several solutions to this, which span from the simple nearest neighbor classification to more complex neural network approaches. Face recognition is an image classification problem and can be solved with neural networks.

It is trivial for most people to recognize someone they have seen before, but how the brain processes signals from the eyes is still unknown. This makes face recognition an interesting problem. How can we make a computer interpret images like a human? What are the important features and how to process them?

Face recognition can be used in several applications, among them this thesis focus on surveillance, identification in login systems and personalized technology. Schneier [19] discusses how the field of study has moved forward, allowing real-time applications. Car movements, in the USA, are already recorded and monitored. For Schneier this is a sign that the advances in face recognition will lead to the same monitoring of people, violating their privacy. Beside the ethical problems related to this technology, there are advantages. Cameras can, for example, be used to verify identity when using person specific cards, such as credit cards or buss passes.

Convolutional neural networks (CNN) are the state-of-the-art method for face recognition. They require large data sets as well as long time to train and that is why this thesis aims to develop a new light weight method.

1.2 Problem Formulation

The best face recognition methods are based on CNN and require a long time to train. The thesis tackles this issue by aiming to develop a light weight method for face recognition and compare it to a state-of-the-art system based on CNN. Two approaches are implemented and training time, evaluation time and, of course, accuracy is measured and compared on the same data sets. Additionally sensitivity to environmental differences such as illumination, head orientation and facial expressions are evaluated.

The light weight method is based on a preprocessing step, using principal component analysis (PCA), followed by a standard neural network. It aims to lower the number of trainable parameters and thereby the training time.

1.3 Limitations

1. The network sizes are limited to decrease training time. This is due to the thesis spanning over 20 weeks. It allows for testing of different network configurations with different parameters.
2. There are many possible preprocessing steps, however, only PCA is used to limit the scope of the thesis.
3. The number of method configurations is limited and the CNN method only uses one configuration due to time constraints.
4. Only two data sets are used due to time limitations.
5. The number of images in the data sets are limited. The data sets that are free to use do not contain as many images as, for example, Googles data set.

1.4 Related Work

This thesis explores two methods for face recognition. The first is a CNN method, mainly based on FaceNet [20]. FaceNet proposes the use of inception modules, enabling larger networks, and triplet loss making it one of the top performers at the Labeled Faces in the Wild (LFW) benchmark.

In 1991 Turk and Pentland [23] proposed the use of eigenfaces (a form of PCA) for face recognition. The second method uses eigenfaces in a preprocessing step as input to a neural network. This approach employs the same preprocessing as [12] and [1] but with a different classifier. Instead of the ones proposed in the papers, it uses only one neural network with 1-3 layers and triplet loss. This implies that also this method is influenced by FaceNet.

Triplet loss makes feasible the training of a generic face recognition network that learns a mapping from an image to a point in a feature space. Images are classi-

fied as same or different inside the space by thresholding the distance between points. This allows the user to add new identities without the need to retrain the network. It was Chopra, Hadsell and LeCun [3] who first proposed this type of learning approach.

Since face recognition should be independent of the background it is reasonable to crop images to only contain faces. In order to automate this process a face detection algorithm is needed. Haar Cascades can be used for fast detection, as described in [24]. The paper introduces the image representation called "integral image" as well as smart feature selection. It uses more complex classifiers in a cascade structure to increase speed. After the face detection images can easily be cropped. The methods used in this thesis are evaluated on cropped images.

1.5 Thesis Outline

Chapter 2 introduces theory and concepts used in the thesis, such as PCA, different networks and network components. Chapter 3 follows, outlining the used methods. Chapter 4 describes the acquired results of the different stages. Finally chapter 5 contains conclusions, discussion and future work.

2
Theory

2.1 Neural Networks

Neural networks can be used to classify signals and images. The network parameters need to be tuned to make the network evolve into a good classifier. This process is called training and uses a data set already classified, the training set. Part of the known data is withheld during training and used to test the network accuracy, this is called test set. After training unlabeled signals and images can be used as input to the network to produce a classified output.

2.1.1 Motivation and History

Neural networks are inspired by the brain's biological neural network. It consists of approximately 86 billion neurons [7] connected to each other by synapses forming a network. Figure 2.1 illustrates a biological neuron which receives signals from other neurons through the dendrites. These transport the signal to the cell body which takes all signals into consideration and produces an output which is sent through the axon to other neurons. At the ends of each axon there are synapses connecting the neurons to each other. Figure 2.2 illustrates an artificial neuron model. The signals from other neurons, x, are transferred to the dendrites by synapses, w, producing an input signal, wx, to the cell body. The output, y, is calculated using an activation function and sent through the axon to the next artificial neuron.

Even though neural networks have grown more popular in recent years they are not a new concept. In 1943 McCulloch and Pitts published a paper [14] which proves that even simple networks can model any arithmetic or logical function. This marked the beginning of neurocomputing. Thanks to more powerful com-

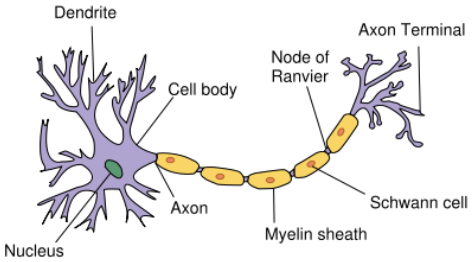

Figure 2.1: *An illustration of a biological neuron. Signals from other neurons propagates through the dendrites to the nucleus. It produces the output which is sent through the axon to connected neurons. Image derived from Public Domain work [25].*

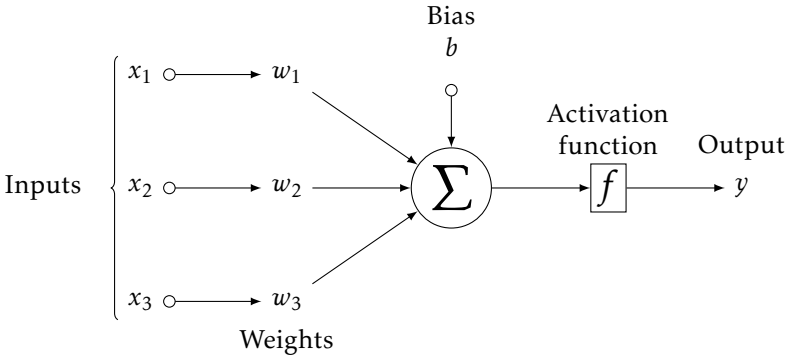

Figure 2.2: *Model of a neural network node. Signals from nodes in the previous layer, x, are multiplied by weights, w, summed up and a bias, b, is added. The summed signal is mapped through an activation function to produce the node output, y.*

puters the first network simulations were performed in the 1950's and in 1987 the first open conference on neural networks was held and the International Neural Network Society was formed [15]. Further more it was during the 1980's that convolutional neural networks were developed.

Today neural networks are widely used both by famous companies, such as Google and Facebook, and start up projects, such as SmartPlate. SmartPlate uses neural networks to identify types of food to calculate the number of calories in a meal.

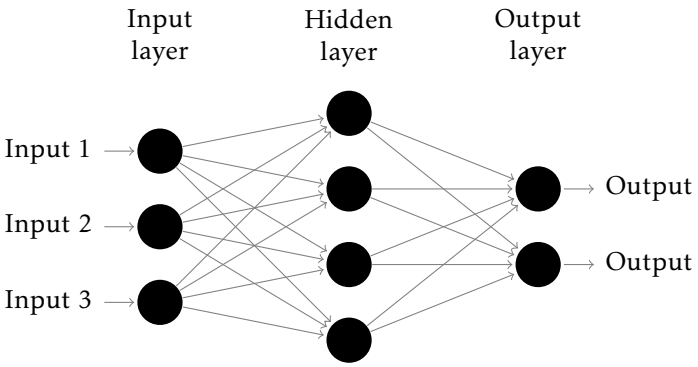

Figure 2.3: *A neural network with three layers*

2.1.2 Neural Network Architecture

Neural networks consist of layers with nodes. Figure 2.3 illustrates a basic neural network with three layers and figure 2.2 illustrates a single node. Each layer is connected to the next with weights, w, which are multiplied to the output of the nodes in the previous layer, x. Each node additionally has a bias, b. The input layer has three nodes that are fully connected to the four nodes in the hidden layer which in turn are fully connected to two output nodes. Layers between input and output layers are called hidden layers and these transform the input to something that the output layer can use.

Each connection gives the node a signal. These signals are added to each other to produce a sum, z, as in equation 2.1 where n is the number of connections to the previous layer. The sum is then passed through an activation function, such as ReLU or Logistic, which produces the node output, y. Equations 2.2 and 2.3 demonstrate how the node output is calculated with ReLU and Logistic activation functions respectively.

$$z = \sum_{i=1}^{n} w_i * x_i + b \tag{2.1}$$

$$y = \max(0, z) \tag{2.2}$$

$$y = \frac{1}{1 + e^{-z}} \tag{2.3}$$

The network architecture is important and needs to be designed to fit its purpose. The number of input nodes needs to be the same as the dimension of the data that should be classified. Further more the number of hidden layers and the

width (number of nodes) is important. More and wider hidden layers allow the network to represent more complicated functions. This does, however, increase training and evaluation time, since there are more parameters to train. Furthermore this means that the network improves the fitting of the data. This can lead to overfitting which means that the network can classify the training samples well (including outliers) while not generalize to unseen data. Several methods, such as dropout techniques, have been studied to overcome this issue.

Output

The output layer design is important and depends on the used loss function. The loss function is used during training to update network variables. One way to design the output layer is to have the same number of output nodes as the number of classes representing the input data. The expected output would be a one-hot vector (a vector where one element is one and all others are zero). The network in figure 2.3, for example, can only classify data into two classes.

Another possibility is to have an arbitrary number of output nodes e.g. 128. In this case the network maps the input to a point in a 128-dimensional feature space. The loss can then be measured by calculating the L_2-distance between output vectors.

2.1.3 Data and Loss Function

Neural networks need to be trained using enough training samples to enable it to reach high accuracy and generalize to unseen data well. The training set needs to contain enough variance to be a good representation of the general case. When using faces, for example, two different individuals are not a good enough representation of the general population. Images of 200 individuals would, however, contain a larger variety of people, achieving a more general representation of the population.

During training, the data should be split into a training set, used to train the network parameters, and a test set, used to test the network accuracy. This is done to obtain an estimate of how it would perform in a practical application when the data is unknown. If the accuracy on the training set is much higher than on the test set it is a sign of an overfitted network. There are two major ways to split the data and they depend on the quantity of images per identity belonging to the data set and what type of loss is used.

Cross-entropy [10] loss uses vectors, with one element for each class, where the largest element determines the classification. This means that the network needs to be trained using images of all identities which it should be able to recognize. Furthermore it needs to be retrained if an identity is added. At least one image per person is needed in both training and test set, however, to achieve good results it requires more images. The first way to split the data is to use a percentage of the images for each person for training and the rest for testing.

Triplet loss allows training of a generic network that can classify any person without the need to retrain the network. It makes the network learn a mapping from

input to a generic feature space. When using triplet loss one can split the data set to contain images of all individuals in both training and test set too, however some data sets, such as LFW [13], have just one image per person. In this case we need to use triplet loss and another way to split the data. The split uses a percentage of the identities for training and the rest for testing. In this case all images of one identity are used for either training or testing.

Cross Entropy

To measure the network error a loss function is needed. The total loss, L, is the average of all individual losses, L_i, and is calculated as in equation 2.4 where N is the number of samples used to calculate the loss. Each individual loss is calculated with the cross-entropy function in equation 2.5 where y is the network output vector, i is the index of the correct class and n is the number of output nodes (classes). If the fraction in equation 2.5 is included in the computations of all outputs in the output layer, it is called a Softmax layer. It normalizes all outputs and this makes it possible to see the output as class probabilities. The computations in equation 2.6 demonstrate an example of how a Softmax layer calculates the output. The first vector in equation 2.6 is the vector generated by the layer activation function.

$$L = \frac{1}{N}\sum_{i=1}^{N} L_i \tag{2.4}$$

$$L_i = -\log\left(\frac{e^{y_i}}{\sum_{j=1}^{n} e^{y_j}}\right) \tag{2.5}$$

$$\begin{pmatrix} -2.85 \\ 0.86 \\ 0.28 \end{pmatrix} \xrightarrow{\exp} \begin{pmatrix} 0.058 \\ 2.36 \\ 1.32 \end{pmatrix} \xrightarrow{\text{normalize}} \begin{pmatrix} 0.016 \\ 0.631 \\ 0.353 \end{pmatrix} \tag{2.6}$$

Triplet Loss

Triplet loss was introduced by Google in "FaceNet: A Unified Embedding for Face Recognition and Clustering" [20] and can be used in networks that map data to a point in a d-dimensional space. It is based on measuring the distance between these points and obtains its name from using triplets (three points), one anchor point, x^a, one positive point, x^p, and one negative point, x^n. The anchor and positive point belong to the same class while the negative point should belong to a different class. The total loss tries to minimize the distance between the anchor and the positive point while maximizing the distance between the anchor and the negative point. It is defined by equation 2.7 where β is a margin that is enforced between positive and negative pairs. Figure 2.4 illustrates training with triplet loss.

$$L = \sum_{i=1}^{N} \max\{0, \|x_i^a - x_i^p\|_2^2 - \|x_i^a - x_i^n\|_2^2 + \beta\} \tag{2.7}$$

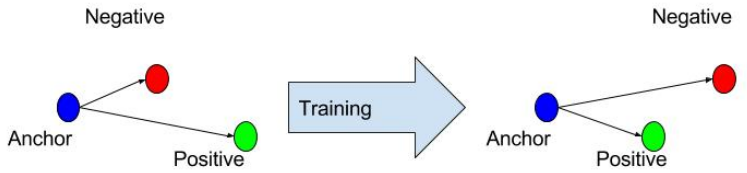

Figure 2.4: *Illustration of training with triplet loss. Three points are chosen, an anchor, a positive (belonging to the same class as the anchor) and a negative point (belonging to a different class than the other two). Triplet loss aims to minimize the L_2-distance between the anchor and the positive point while maximizing the distance between the anchor and the negative point.*

2.1.4 Training

Training aims to minimize the loss function by changing network parameters. This is done with backpropagation, which consists of two phases: propagation and weight update. Propagation pushes training input through the network and then backpropagates the loss function derivatives. The weight update step then uses equation 2.8, where α is the learning rate, v^t is the current variable at iteration t and L is the total loss, to update all network weights and biases.

$$v^{t+1} = v^t - \alpha * \frac{\partial L}{\partial v} \tag{2.8}$$

Dropout

Dropout [17] is a way to avoid overfitting the network during training. It keeps nodes active with the probability p and deactivated with probability $1 - p$. Figure 2.5 illustrates how dropout is applied during training with $p = 0.5$. Which nodes are active change from batch to batch and dropout is turned off during testing to utilize the complete network. This makes the network architecture change between batches which helps to avoid complex co-adaptation of the network nodes which is common, especially when using a small data set. Co-adaptation means that a node depends on the activation of specific nodes in the previous layer while discarding the rest. This is avoided by randomly deactivating inputs, forcing a node to take more inputs into consideration.

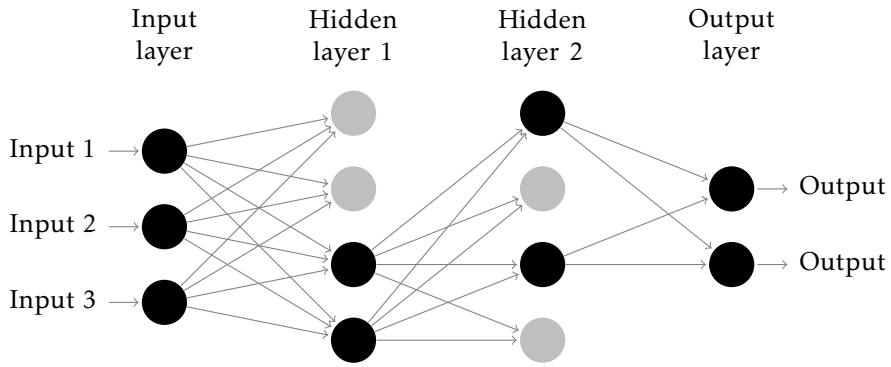

Figure 2.5: *A neural network with dropout during training. The activation probability is set to* 0.5 *and the black nodes are active while the gray are deactivated. The activations are changed between each batch during training and dropout is turned off (all nodes are active) during testing.*

Cross-Validation

Using triplet loss results in a network that can determine if two images belong to the same category or not depending on their distance in the feature space. A threshold is used to make this decision and it needs to be determined. The data is split into training set, validation set and test set. The training set is used to generate losses and to train network weights and biases. The validation set is used to calculate network accuracy after each training phase to control how the training is progressing and to determine the model threshold.

A training phase consists of a number of batches and one batch consists a number of training samples. For example, one batch can consist of 90 training samples and one training phase can be set to 1000 batches. When all batches have been used to train the network a training phase is done. Finally the test set is used to evaluate the model.

When using small data sets this split can result in small training sets. This can, to some degree, be improved by the use of cross-validation [21]. It splits the training set in a number of folds. One fold is used for validation while the others are used for training. After each training phase the validation fold is changed to another, figure 2.6 illustrates how the data is split using 4 folds. Fold 1-3 are used for training and fold 4 for validation during the first training phase. During the second step fold 2-4 are used for training while fold 1 is used for validation and so on.

Figure 2.6: *An illustration of a possible splitting of the data for cross-validation. Training set is split into a number of folds, in this case 4. The blue folds are used for training while the yellow one is used for validation. The fold used for validation is changed after each training phase.*

2.2 Principal Component Analysis

Principal component analysis is a method to reduce data dimensionality by finding a basis of eigenvectors. This can be done on face images by reshaping them to vectors. The resulting basis vectors can later be reshaped to images with face-like appearance which is the reason they are called eigenfaces.

The images are reshaped to vectors and the mean of all image vectors, Φ in equation 2.9 where v is an image vector and N is the number of images, is subtracted from each vector. The vectors are then normalized and finally placed into the columns of a data matrix, A. The eigenfaces are the eigenvectors of the covariance matrix, $C = AA^T$, corresponding to the largest eigenvalues. The eigenfaces are inserted in the columns of a basis matrix, B, which is used in equation 2.10 to find the coordinates for each image relative to this basis. The number of eigenfaces used determines the new data dimensionality. Figure 2.7 illustrates the first eigenface from calculations on the data set faces94 [8].

$$\Phi = \frac{1}{N}\sum_{i=1}^{N} v_i \tag{2.9}$$

$$W = B^T[v - \Phi] \tag{2.10}$$

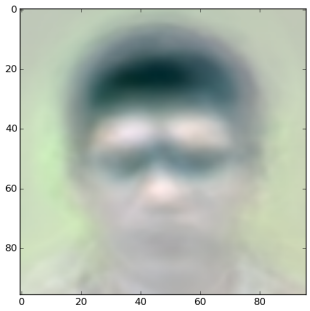

Figure 2.7: *When performing PCA on face images the eigenvectors can be reshaped to face-like images called eigenfaces. The illustration is the eigenface corresponding to the largest eigenvalue from calculations on faces94 [8].*

2.3 Convolutional Neural Networks

Regular neural networks use vectors as input and have fully connected layers, which means that each element is connected to all nodes in the next layer. Images, then, need to be reshaped to vectors and the number of weights, between the input layer and one node in the hidden layer, is the same as the number of pixels. Larger images would, therefore, result in an unreasonable amount of weights. Convolutional neural networks (CNN:s) use images as input and convolution kernels, made up out of weights, as connections to the next layer. This means that weights are shared on different spatial positions and the needed amount of weights are lowered. These weights are arranged in 3D volumes and transforms the input image into an output of node activations.

2.3.1 Layers

CNN:s consist of three different types of layers: convolutional, max pooling and fully connected layers. Convolutional layers have weights arranged in kernels that are convolved with the input. Each pixel in the 3D convolution output is mapped through an activation function, such as ReLU in equation 2.2, to produce a 3D output volume of node activations. The use of convolution can both keep and decrease the size of the input image. Usually convolutional layers keep the size which is accomplished by the use of zero padding. It adds zeros around the image before the convolution and then keeps only the central part of the result with the same size as the input.

Max pooling layers are used to downsample the 3D volume. A max pooling layer takes an image region and preserves only the highest value inside the region. It can be seen as a convolution kernel, with a specific stride, extracting max values instead of calculating dot products. A stride of 1 would preserve the volume size while a stride of 2 would divide the width and height dimension by two.

The fully connected layers are exactly the same as the ones in a regular neural network. Each node in the previous layer has a connection to each node in the fully connected layer. Such layers are usually placed at the end of a CNN.

Inception Module

Inception modules were introduced by Google in "Going Deeper with Convolutions" [4] and are used to increase the network size without uncontrolled increase in computational complexity. They are combinations of convolutional and max pooling layers and are based on the idea that images should be processed at several different scales. The module sends the input from the previous layer on four different paths and the combination of the path outputs is the module output. The first is a 1×1 convolution layer, the second is a 1×1 followed by a 3×3 convolution layer, the third is a 1×1 followed by a 5×5 convolution layer and the last path is a 3×3 max pooling layer followed by a 1×1 convolution layer. Figure 2.8 illustrates an inception module.

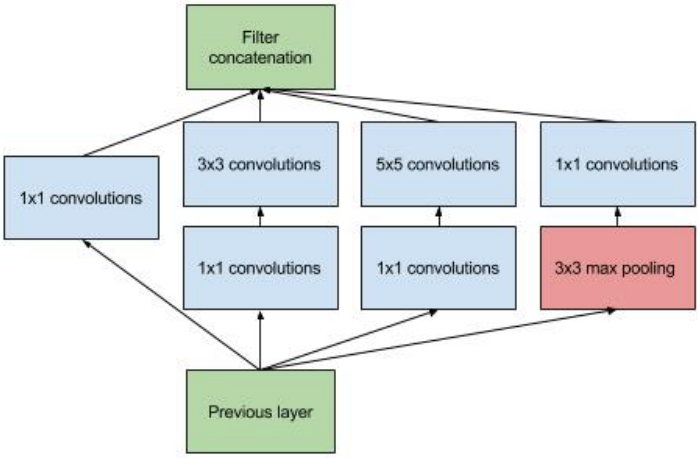

Figure 2.8: *An illustration of an inception module. The green boxes are connections to another layer or module. The blue ones are convolution layers and the red one is a max pooling layer, all with specified kernel size. Each module contains four "paths" for the images and the output from all paths are concatenated to form the module output.*

Layer	Size-in	Size-Out	Kernel	Stride
conv	28×28×1	28×28×32	5×5×1	1
pool	28×28×32	14×14×32	2×2×1	2
conv	14×14×32	14×14×64	5×5×32	2
pool	14×14×64	7×7×64	2×2×1	2
fc	7×7×64	1×1×1024		
Softmax	1×1×1024	1×1×10		

Table 2.1: *A simple CNN architecture. The size columns describe the input and output dimensions, $width \times height \times channels$. The kernel column specifies kernel $width \times height \times channels$ and the stride is only in space dimension. All convolution and max pooling layers use zero padding making the stride determine the output size.*

2.3.2 CNN Architecture

CNN:s usually contain repetitions of convolutional layer(s) followed by max pooling. The final layers are one or two fully connected layer(s) followed by a Softmax layer. Table 2.1 demonstrates a simple CNN architecture. It is designed for gray images with 28 pixels in both width and height. Zero padding is used in all the convolutional and max pooling layers making the stride determine the output size.

2.4 Region of Interest

Images usually contain a region of interest. When performing face recognition it is the face. The background should not be part of the information used for classification, a person should be recognized regardless of the environment. Images can be cropped to only contain faces, however, in order to automate this process a robust face detector is needed. Haar Cascades can be used for this purpose, as described in [24]. After detection the images can easily be cropped to only contain the region of interest.

2.4.1 Haar Cascades

Viola and Jones propose a fast and robust face detection algorithm in [24]. They propose the use of three kinds of features, all illustrated in figure 2.9. Each feature is obtained by computing the difference between the sum of the image pixels in the white regions and the sum of the image pixels in the black region. This is then repeated while sliding the region over the image, just like a convolutional kernel. These calculations take time but this is solved with a smart image representation, called "integral image", simplifying an image region sum to a calculation involving only four array references. Equation 2.11, where I_r is the integral image and I is the original image, demonstrates the definition.

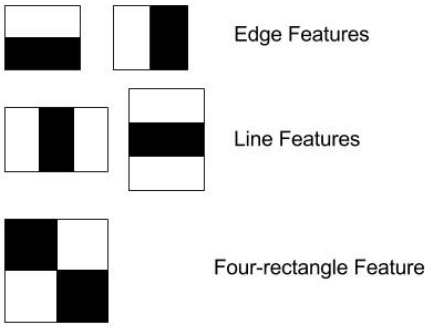

Figure 2.9: *An illustration of features used for face detection. The features detect bright regions (the white rectangles) bordering to dark regions (the black rectangles).*

$$I_r(x, y) = \sum_{x' \leq x, y' \leq y} I(x', y') \tag{2.11}$$

Figure 2.10 illustrates two features in practice. They can be moved all over the image but most placements are irrelevant. The first feature is based on the fact that the region around the eyes is darker than the one around the cheeks and nose. The same feature placed in the mouth region would be irrelevant since it would not give any useful information due to the fact that this area has a different structure. Machine learning is used to select a small set of features and to train the final classifier.

The cascade part of the algorithm is a fast rejection of regions not containing faces. The features are grouped into different stages of classifiers. The first stage may contain 1 feature, the second 10, the third 25 and so on. If a region fails a stage it is discarded and no more calculations are needed, it needs to pass all stages to be classified as a face.

Figure 2.10: *An illustration of two features in practice. The image in the middle illustrates how an edge feature can be used to detect the dark region around the eyes relative to the bright region on the cheeks. Line features can instead be used to detect the dark region around the eyes relative to the bright region around the nose as illustrated in the image to the right. The image to the left is the original image. The face image is part of the data set Faces94 [8].*

3
Method

3.1 Literature Search

A pilot study was performed to find related work and understand the latest algorithms in the field. Since the choice of the data is important to neural networks, it additionally involved the search for proper data sets. IEEE Xplore [11] and arXiv [2] were used for retrieving state-of-the-art papers. The data sets suggested in these articles are used as a baseline for the evaluation of the neural networks.

3.2 Implementation

This thesis evaluates and compares two different methods for face recognition. The first is a CNN, which is based on Googles FaceNet [20], and the other is a regular neural network with preprocessing, in the form of PCA, and triplet loss. Both implementations are described in sections 3.2.2 and 3.2.3 and the results are presented in section 4. All images are resized to 96×96 pixels and normalized according to equation 3.1, where I is the original image, Φ the mean pixel value and σ is the pixel standard deviation, before they are used.

$$I_n = \frac{I - \Phi}{\sigma} \tag{3.1}$$

All data sets were cropped around the faces, with the method described in section 2.4, since FaceNet [20] observed an increase in accuracy when only using the face region. All code is written in Python version 2.7.10. The face detection used

to crop the images, used OpenCV [18] while the networks and PCA were implemented with the use of TensorFlow [22], compiled with GPU support. All tests and training were run on Ubuntu 15.10 with Intel core i7-6700k and NVIDIA GeForce GTX 980 Ti.

3.2.1 Data and Training

The data set is split into training, validation and test sets. The training set is used to calculate the loss and to train the network. The validation set is used to evaluate the network after each training phase. It is used to determine when the network has converged. Finally, the test set is used to evaluate the network accuracy.

Each training phase consists of 1000 batches containing 90 training samples used to calculate the loss and update network parameters. The validation step takes place after each training phase. It creates all possible positive pairs and the same number of random negative pairs from the validation set. A positive pair is two images belonging to the same identity while the images in a negative pair belong to different identities. It continues by calculating the feature point corresponding to the images and then tries to classify the positive pairs as same and the negative pairs as different using a range of thresholds. The used range is 0 to 2 with a step length of 0.01. After each validation step the best threshold, T_b, is used to update the model threshold, T, with equation 3.2.

$$T_n = 0.75T_{n-1} + 0.25T_b \tag{3.2}$$

3.2.2 CNN Implementation

CNN:s are the leading method for face recognition on the data set Labeled Faces in the Wild [13]. According to the website [13], FaceNet [20] achieves a impressive 99,63% accuracy. The CNN implementation in this thesis is based on the network [20] and uses inception modules as well as triplet loss and backpropagation.

3.2.3 Neural Network with Preprocessing Implementation

This method uses a neural network, with triplet loss, that uses preprocessed image data as input. The training set is used to calculate a basis, B, and a mean image, Φ, as described in section 2.2. All images are projected onto a subspace, using equation 2.10, reshaping the input images to vectors with the same length as the number of eigenvectors used in the basis.

The projected images are used as input to a neural network with 1-3 hidden layers and a softmax output layer with 64 nodes when using Faces94 and 128 nodes when using FaceScrub. Triplet loss and backpropagation are used as loss function and to update the variables. Network parameters and results are presented in section 4.

3.3 Evaluation

Evaluation takes place when the training has converged and the model is finished. It is reminiscent of the validation step but creates all possible positive as well as negative pairs from the test set. It calculates the feature points for all images and then the true positive rate, TPR, and false positive rate, FPR, according to equation 3.3 and 3.4. P_{pos} are the positive pairs, P_{neg} are the negative pairs, t is a distance threshold and $D(y_i, y_i)$ is the L_2-distance between y_i and y_j.

$$TPR(t) = \frac{|\{(y_i, y_j) \in P_{pos}, \text{ with } D(y_i, y_j) \leq t\}|}{|P_{pos}|} \tag{3.3}$$

$$FPR(t) = \frac{|\{(y_i, y_j) \in P_{neg}, \text{ with } D(y_i, y_j) \leq t\}|}{|P_{neg}|} \tag{3.4}$$

TPR and FPR describe how many of the positive and negative pairs are classified as same. By calculating the rates using different thresholds and plotting TPR against FPR one obtains receiver operating characteristic (ROC) curves such as the one in figure 3.1. The ROC illustrates how good the clustering of points (in the feature space) belonging to the same class is. Random guessing would result in the green curve and this is the approximate result of an untrained neural network. Training improves the clustering resulting in an increase in TPR and decrease in FPR giving the blue curve. The rates at the model threshold, T, are used to calculate the model accuracy according to equation 3.5.

$$A_{cc} = \frac{TPR(T) + (1 - FPR(T))}{2} \tag{3.5}$$

Evaluation was performed by running the implementations described in section 3.2, with varying variables and network architectures, on different data sets. The data sets, parameters and results are described in section 4.

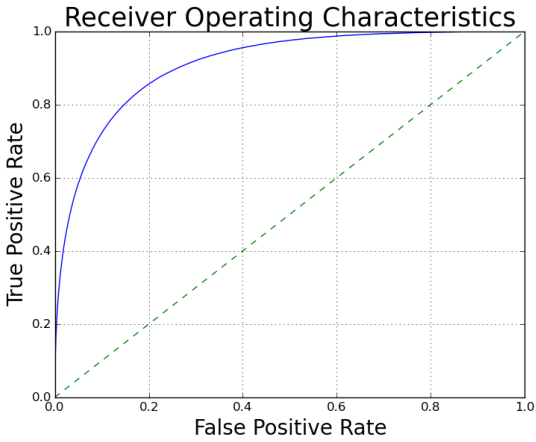

Figure 3.1: *An example ROC. TPR and FPR are calculated using different model thresholds. TPR is then plotted against FPR. The goal is to maximize TPR while minimizing FPR. With random guessing one would receive the green line. Trained neural networks can improve the results giving the blue line.*

4
Result

4.1 Literature Search

The pilot study resulted in the findings of the related work described in section 1.4. Furthermore it led to the selection of two data sets: Faces94 [8] and FaceScrub [16].

Faces94 consists of 152 individuals with 20 images each for a total of 3040 images. All subjects are seated in front of a green background on a fixed distance from the camera. They were reading a text while the images were taken. This limits the variations to small changes in the facial expression and the pose, making the images a good representation of a login system. Figure 4.1 illustrates examples of images from Faces94 and figure 4.2 illustrates the cropped versions.

The FaceScrub images are acquired by following the instructions received after filling out the form on the website [9]. This results in a data set containing 530 individuals, 265 male and 265 female, with a total of 67346 images. These images have been collected from the Internet and are not pure face images, as can be seen in figure 4.3. Therefore the face detection described in section 2.4 is used to crop the images to only contain the faces. The cropped result of the images in figure 4.3 are presented in figure 4.4. This fails for some images resulting in a data set of 530 individuals but with only 65084 images. The data set contains large variations in lighting, age, poses, hair style, and face expressions which create a challenging problem for image classification.

The data sets were chosen because of their differences. One is smaller with controlled environment, while the other is larger with uncontrolled conditions. This enables an analysis of the strengths of the approaches tested.

Figure 4.1: *Images from Faces94 [8], illustrating smaller variations in facial expression.*

Figure 4.2: *Cropped images from Faces94 [8], illustrating smaller variations in facial expression.*

Figure 4.3: *Images of Al Pacino from FaceScrub [9], illustrating variations in lighting, hair style, age, pose and facial expression which create a challenging problem for image classification.*

The pilot study finally resulted in method and implementation sketches. The CNN sketch is greatly influenced by FaceNet [20] while the preprocessing method uses a combination of principal component analysis combined with a neural network and triplet loss (from FaceNet).

Figure 4.4: *Cropped images of Al Pacino from FaceScrub [9], illustrating variations in lighting, hair style, age, pose and facial expression which create a challenging problem for image classification.*

4.2 Implementation Details

The implementations are described in section 3.2, however, details such as parameters and network architectures were left out and are explained in sections 4.2.1 and 4.2.2. The data sets used were cropped versions of FaceScrub and Faces94. The implementations use 90% of the identities for training and 10% for testing. This split is used to obtain a perception of how the methods would perform in a practical application, e.g. a login system, where new identities could be added without the need to retrain the network. The identities used for training are once again split into 10 folds used for cross-validation to increase the size of the training set. The triplet loss margin, β in equation 2.7, is set to 0.2 for all models except for the CNN trained on Faces94 which uses a margin of 0.5, a batch size of 90 (30 triplets) is used and one training phase is set to 1000 batches.

4.2.1 CNN Implementation

The CNN method uses color images with size 96×96×3 as input and maps each image to a point in a 128-dimensional feature space. The complete architecture is specified in table 4.1 and table 4.2 describes the inception modules in detail. The topology of an inception module is illustrated in figure 4.5. The network consists of 18 layers (if counting inception modules as single layers) and almost 7 million trainable parameters.

Type	Output size	Kernel	Stride	# Param
conv1	48×48×64	7×7×3	2	9K
Max pool1	24×24×64	3×3×1	2	
conv2	24×24×64	1×1×64	1	4K
conv3	24×24×192	3×3×64	1	111K
Max pool2	12×12×192	3×3×1	2	
inception 1a	12×12×256			164K
inception 1b	12×12×320			228K
inception 1c	6×6×640			398K
inception 2a	6×6×640			545K
inception 2b	6×6×640			595K
inception 2c	6×6×640			654K
inception 2d	6×6×640			722K
inception 2e	3×3×1024			717K
inception 3a	3×3×896			1.386M
inception 3b	3×3×896			1.295M
Average pool	1×1×896			
FC	1×128			115K
L2 normalize	1×128			

Table 4.1: *The architecture of the CNN method. The network takes images of size 96×96×3 as input and maps each image to a point in a 128-dimensional feature space. The first column specifies the layer/module type, the second specifies the kernel size, the third specifies the stride in space dimension and the last specifies the number of trainable parameters. The networks contains a total of 6.943 million trainable parameters.*

Module	Kernel Stride	#1×1	[#1×1,#3×3]	[#1×1,#5×5]	#1×1
inception 1a	1	64	[96,128]	[16,32]	32
inception 1b	1	64	[96,128]	[32,64]	64
inception 1c	2	0	[128,256]	[32,64]	0
inception 2a	1	256	[96,192]	[32,64]	128
inception 2b	1	224	[112,224]	[32,64]	128
inception 2c	1	192	[128,256]	[32,64]	128
inception 2d	1	160	[144,288]	[32,64]	128
inception 2e	2	0	[160,256]	[64,128]	0
inception 3a	1	384	[192,384]	[0,0]	128
inception 3b	1	384	[192,384]	[0,0]	128

Table 4.2: *The table specifies the details of the inception modules used in the CNN method. The kernel stride column specifies the stride used by the 3×3 and 5×5 convolutional layers as well as the max pooling layer. All other strides are set to 1. The rest of the columns specify the number of kernels used in each layer inside each signal path. Observe that max pooling is performed in the last path even though the last 1×1 column is 0.*

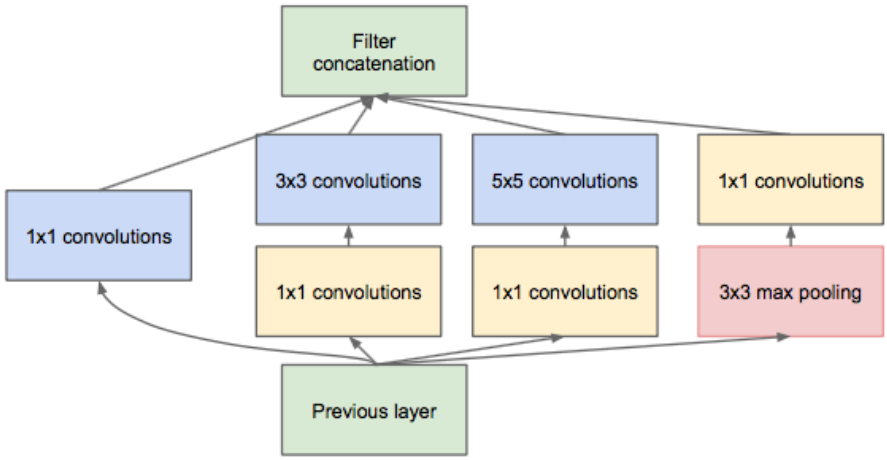

Figure 4.5: *The topology of an inception module.*

4.2.2 Neural Network with Preprocessing Implementation

Basis size, number of hidden layers and nodes in the hidden layers are varied in this method. The number of layers used range from 1 to 3 and the width ranges from 128 to 1024 nodes. The number of eigenfaces used in the basis is varied between 5 and 1000. The exact parameters and results on different data sets are presented in section 4.3.2.

4.3 Evaluation

The evaluation step is described in section 3.3 and is performed on the test sets of cropped versions of FaceScrub and Faces94.

4.3.1 CNN Evaluation

The initial learning rate, α in equation 2.8, is set to 0.1 and decreased each 200 training phase. The decay in triplet loss (2.7) and increase in validation accuracies are illustrated in figure 4.6. The ROC on the test sets of Faces94 and FaceScrub are illustrated in figure 4.7 and the model accuracies with trained thresholds are presented in table 4.3.

The CNN network contains 7 million trainable parameters leading to long training time. Using FaceScrub, the network was trained for 250 training phases (58 hours) before the validation accuracy stopped increasing. The large networks also led to the long evaluation time of 0.017 seconds.

The network performs well on Faces94 reaching 96.29% while it has some problems with the large variations in FaceScrub reaching 82.94%. The shorter training

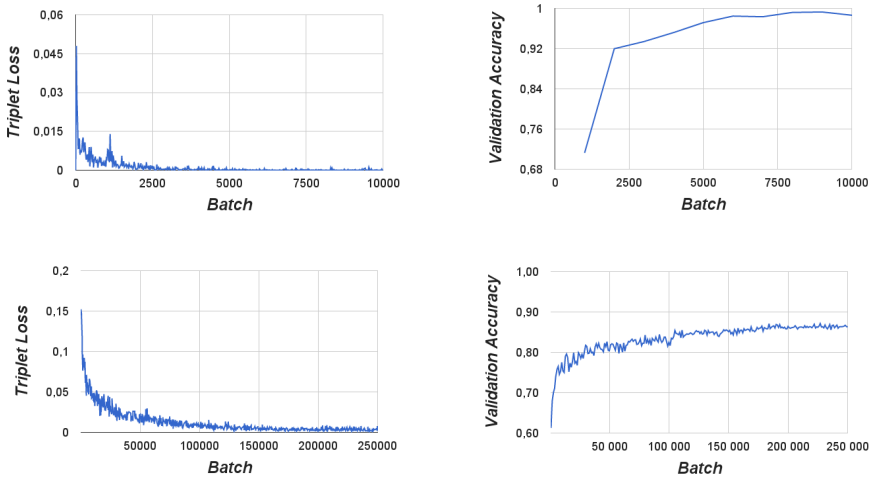

Figure 4.6: *The two top plots illustrate the decay in triplet loss and increase in validation accuracy when training the CNN method on Faces94. The lower plots illustrate the same thing while training on FaceScrub.*

Data set	Threshold	Accuracy	Training	Eval time [s]
Faces94	0.8282	96.29%	[10,2]	0.017
FaceScrub	1.048	82.94%	[250, 58]	0.017

Table 4.3: *CNN accuracies on the two data sets. The training column specifies the number of training phases used as well as training duration in hours, [#phases, duration]. Eval time specifies the average number of seconds to calculate one image feature vector.*

time of only 10 training phases (2 hours) on Faces94 can be explained by the small number of images and the small variations in the data set in combination with the large network making the identities easily separated. However, figure 4.8 illustrates the ROC after training the network on FaceScrub and then testing on Faces94. This model reaches an accuracy of 97.05% on Faces94 which is higher than the 96.29% reached when training and testing on Faces94. The large amount of images together with the large variations in FaceScrub allows the network to learn robust features resulting in a better, more general, face recognition model.

4.3.2 Neural Network with Preprocessing Evaluation

The ROC:s on the test sets of Faces94 and FaceScrub corresponding to the top performing network configurations are illustrated in figure 4.7. Model accuracies along with model parameters and thresholds are presented in tables 4.4 and 4.5. The initial learning rate, α in equation 2.8, is adapted to each network depending on size and data set. It ranges from 0.0001 to 0.1. When using FaceScrub, a subset

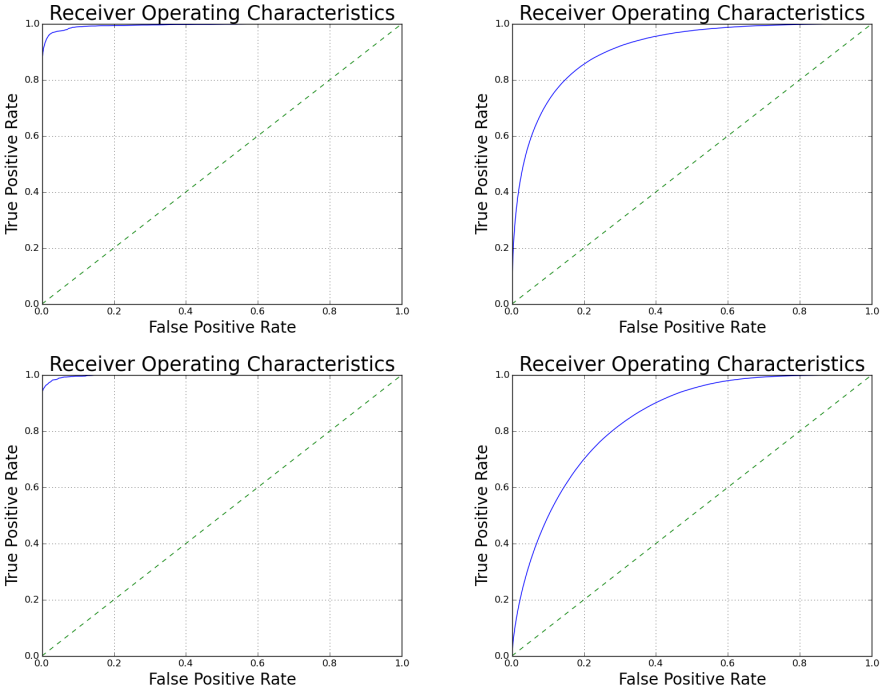

Figure 4.7: *The ROC on the test sets of Faces94 (left) and FaceScrub (right) after training the CNN (top) and preprocessing (bottom) method on the corresponding training sets.*

of 20 random images of each identity is used to calculate the basis. This is done to reduce the size of the data matrix to [9540×9540].

PCA alone proved to work well on Faces94 as demonstrated in table 4.4. Using 200 eigenvectors in the basis gave the highest accuracy of 96.77% with a evaluation time of 0.0004 seconds. Both increasing the basis size to 1000 and decreasing it to 5 gave lower accuracies of 96.06% and 94.76%. Using 200 eigenvectors and combining it with a neural network, with 1 layer with 128 nodes, increased the accuracy to 97.0%. Additionally, it increased evaluation time to 0.0008 and it required 30 training phases (1.5 hours).

Table 4.5 demonstrates the results when training the preprocessing method with FaceScrub. PCA alone does not perform as well as it did on Faces94, reaching only 58.74% using a basis with 1000 eigenvectors. This is only a marginal increase from the 50% achieved by guessing. Decreasing the number of eigenvectors to 200 decreases the result slightly to 58.64%. The evaluation time using the two sizes was 0.0005 and 0.0004 seconds respectively. By using PCA with 1000 eigenvectors in combination with a neural network, with 3 layers with 1024, 512 and 256 nodes each, the accuracy increased to 76.11%. The large network proved

Threshold	Accuracy	Basis size	# Nodes	Training	Eval time [s]
0.34	94.76%	5	N/A	N/A	0.0003
0.84	96.77%	200	N/A	N/A	0.0004
0.87	96.06%	1000	N/A	N/A	0.0005
0.6325	**97.00%**	200	[128]	[30, 1.5]	0.0008

Table 4.4: *Preprocessing implementation details and accuracies on Faces94. Basis size specifies the number of eigenvectors used in the basis. The node column specifies the number of nodes in each hidden layer. The training column specifies how many training phases the network was trained for as well as training duration in hours, [#phases, duration]. Finally eval time specifies the average number of seconds to calculate one image feature vector. The network uses an output layer with 64 nodes giving a total of 34 thousand parameters.*

Threshold	Accuracy	Basis size	# Nodes	Training	Eval time [s]
1.6773	58.65%	200	N/A	N/A	0.0004
1.7355	58.74%	1000	N/A	N/A	0.0005
0.4035	74.68%	200	[256]	[60,8]	0.0006
0.4364	75.20%	1000	[256]	[60,15]	0.0009
0.4537	**76.11%**	1000	[1024, 512, 256]	[160, 37]	0.0026

Table 4.5: *Preprocessing implementation details and accuracies on FaceScrub. Basis size specifies the number of eigenvectors used in the basis. The node column specifies the number of nodes in each hidden layer. The training column specifies how many training phases the network was trained for as well as training duration in hours, [#phases, duration]. Finally eval time specifies the average number of seconds to calculate one image feature vector. All networks use an output layer with 128 nodes. The smallest network has 84 thousand trainable parameters while the largest has 1.7 million parameters.*

to increase the evaluation time to 0.0026 seconds and it took 160 training phases (37 hours) to train the network. Using a smaller network with only 1 layer with 256 nodes the evaluation time was decreased to 0.0009 seconds and the training to only 60 training phases (15 hours). Additionally the smaller network had an accuracy of 75.20%, only a small decrease compared to the much larger network. Furthermore, using a basis with 200 eigenvectors decreased the evaluation time to 0.0006 seconds. While the number of training phases needed to train the networks was still 60 the total training time was almost cut in half to 8 hours.

The large difference in accuracy on Faces94 and Facescrub depends on PCA. Dimensionality reduction means loss of information which may be needed to separate classes. When using Faces94 this information is largely contained in the eigenvectors corresponding to the largest eigenvalues. However, when using Face-

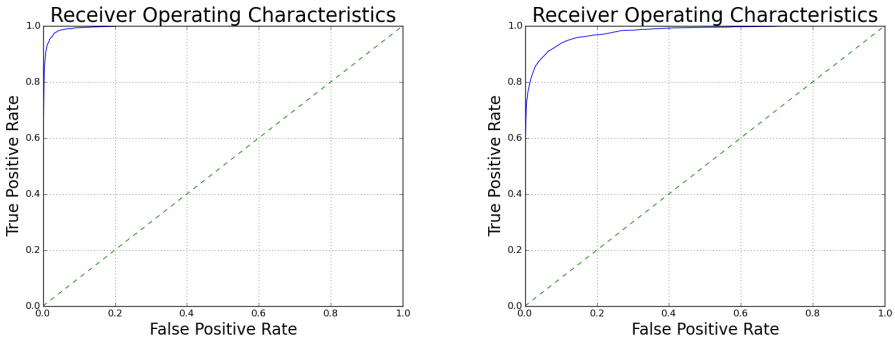

Figure 4.8: *The ROC on the test set of Faces94 after training the CNN (left) and preprocessing (right) method on the FaceScrub training set.*

Scrub more information is contained amongst the smaller components. This results in a lower accuracy. Tables 4.4 and 4.5 prove that adding a neural network to PCA is useful when using data which PCA alone cannot classify very well while it only increases the accuracy marginally when PCA already separates the data.

Figure 4.8 illustrates the ROC of the method using the largest network trained on FaceScrub and tested on Faces94. The result demonstrates how important it is to choose a training set which is a good representation of the test set when using this method. The model reaches only 60.08% accuracy on Faces94.

5

Conclusion and Future Work

5.1 Discussion

There are general limitations to face recognition such as, variations in pose, lighting, hair style, makeup and image quality. Additionally there is the need for a database to store images in order to recognize people. A limitation in login system identification is, for example, the possibility of hacking the access by simply printing an image of an authorized person and holding it in front of the camera. This means that it has to be used in combination with other verification methods such as an access card or code. Another limitation occurs when using it on surveillance footage. Obtaining good images of faces is not always easy. People could be wearing hats, covering their faces, or be looking away from the camera.

5.2 Final Conclusions

The CNN network contains almost 7 million trainable parameters compared to the 1.7 million used in the largest network in the preprocessing method. This is one of the big disadvantages of CNN since the larger number of parameters led to much longer training time and evaluation time. The results in chapter 4 demonstrates how the network performs well on Faces94, just as the preprocessing method. The preprocessing method demonstrates its strengths when using Faces94. The controlled conditions make the images of one individual much alike and this proved to be well suited for PCA as demonstrated in section 4.3.2. The preprocessing method reaches a higher accuracy than the CNN and the ROC is also better.

FaceScrub differs from Faces94 in two key aspects, the large variations and the number of images. The variations meant that both methods had difficulties classifying the data compared to Faces94. The large number of images, however, allowed the CNN network to converge leading to a higher accuracy than the preprocessing method. The problem when using the preprocessing method on FaceScrub is PCA. It is not well suited for the large variations in the data set and has difficulties separating identities. However, table 4.5 demonstrates that adding a neural network can significantly increase the accuracy compared to PCA alone.

Using the CNN model trained on FaceScrub and evaluating it using Faces94 results in the highest accuracy on the test set. The large training set allowed the network to learn robust features making it a good face recognition model for different types of face images. The same can not be said for the best preprocessing model trained on FaceScrub. This indicates that it is important to choose a training set that contains the same type of images as the test set when using the preprocessing method while the CNN can learn a general and robust model using a broad training set with many images and large variations.

The preprocessing method proved to have a shorter training time and evaluation time than the CNN method. It reached a higher accuracy than CNN when using Faces94 while performing worse when using FaceScrub. This proves it has difficulties, compared to CNN, handling large variations in illumination, head orientation, facial expression and age. The highest accuracy on Faces94 was, however, reached by the CNN model trained on FaceScrub indicating that the CNN method is more robust given a large data set with variations.

5.3 Future work

The algorithms developed in this thesis can be used for any image classification problem, one only has to change the data set and retrain the networks. It would be interesting to see how the two approaches work on other data sets than Faces94 and FaceScrub, e.g. CIFAR-100 [5] which is a general classification problem.

There are many possible preprocessing steps, some of which are mentioned by Ciregan, Meier and Schmidhuber in [6], and these can be used in combination with both CNN and neural networks. PCA was chosen because of the reduction in data dimension making it possible to use smaller networks to reduce training time. Other preprocessing steps may be suitable to replace the first layer(s) of the CNN network.

As stated in the limitations section 1.3, this thesis spans over 20 weeks. This led to limitations in network size and in the number of training runs. Future work should include the evaluation of larger networks as well as more parameter tuning of, for example, learning rate, triplet loss margin and basis size. In addition a smaller CNN trained on Faces94 could be evaluated. This would reduce the training and evaluation time. Furthermore the limited time led to the use of eval-

uation time instead of FLOPS. Evaluation time is highly dependent on hardware while FLOPS is not, making it a better measurement of method complexity.

According to the FaceNet paper [20] training on larger data sets improves network accuracy. They compare accuracies after training on 2.6-265 million images. The accuracy increases when using up to 26 million images and then levels out. This can be put in perspective to the data sets used in this thesis containing only 3040 and 65084 images. Finding/creating a larger data set is an important part of the future work. To increase the accuracy further, face alignment could be used. The same paper [20] proves that alignment compared to only cropping increases the accuracy.

Finally both algorithms result in general face recognition models. These can be used in practical applications such as login systems or surveillance. Future work could very well include the development of such a system.

Bibliography

[1] M. Agarwal, H. Agrawal, N. Jain, and M. Kumar. Face recognition using principle component analysis, eigenface and neural network. In *Signal Acquisition and Processing, 2010. ICSAP '10. International Conference on*, pages 310–314, Feb 2010. doi: 10.1109/ICSAP.2010.51. Cited on page 2.

[2] arXiv. arxiv. `http://arxiv.org/`, 2016. Accessed: 2016-03-29. Cited on page 19.

[3] S. Chopra, R. Hadsell, and Y. LeCun. Learning a similarity metric discriminatively, with application to face verification. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 539–546 vol. 1, June 2005. doi: 10.1109/CVPR.2005.202. Cited on page 3.

[4] S. Christian, L. Wei, J. Yangqing, S. Pierre, R. Scott, A. Dragomir, E. Dumitru, V. Vincent, and R. Andrew. Going deeper with convolutions. In *Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on*, pages 1–9, June 2015. doi: 10.1109/CVPR.2015.7298594. Cited on page 14.

[5] CIFAR. Cifar. `https://www.cs.toronto.edu/~kriz/cifar.html`, 2016. Accessed: 2016-05-09. Cited on page 34.

[6] D. Ciregan, U. Meier, and J. Schmidhuber. Multi-column deep neural networks for image classification. In *Computer Vision and Pattern Recognition (CVPR), 2012 IEEE Conference on*, pages 3642–3649, June 2012. doi: 10.1109/CVPR.2012.6248110. Cited on page 34.

[7] CS231n. Cs231n convolutional neural networks for visual recognition. `http://cs231n.github.io/`, 2016. Accessed: 2016-03-22. Cited on page 5.

[8] faces94. Collection of facial images: Faces94. `http://cswww.essex.ac.uk/mv/allfaces/faces94.html`, 2016. Accessed: 2016-03-23. Cited on pages 12, 13, 17, 23, and 24.

[9] FaceScrub. Facescrub. http://vintage.winklerbros.net/facescrub.html, 2016. Accessed: 2016-04-27. Cited on pages 23, 24, and 25.

[10] I. J. Good. Some terminology and notation in information theory. *Proceedings of the IEE - Part C: Monographs*, 103(3):200–204, March 1956. ISSN 0369-8904. doi: 10.1049/pi-c.1956.0024. Cited on page 8.

[11] IEEE Xplore. Ieee xplore. http://ieeexplore.ieee.org/, 2016. Accessed: 2016-03-29. Cited on page 19.

[12] V. P. Kshirsagar, M. R. Baviskar, and M. E. Gaikwad. Face recognition using eigenfaces. In *Computer Research and Development (ICCRD), 2011 3rd International Conference on*, volume 2, pages 302–306, March 2011. doi: 10.1109/ICCRD.2011.5764137. Cited on page 2.

[13] LFW. Labeled faces in the wild. http://vis-www.cs.umass.edu/lfw/, 2016. Accessed: 2016-01-04. Cited on pages 9 and 20.

[14] W. S. McCulloch and W. Pitts. A logical calculus of the idea immanent in nervous activity. *Bulletin of Mathematical Biophysics*, 5:115–133, 1943. Cited on page 5.

[15] Y. Neha, Y. Anupam, and K. Manoj. *An Introduction to Neural Network Methods for Differential Equations*. Springer Netherlands, first edition, 2015. Cited on page 6.

[16] H. W. Ng and S. Winkler. A data-driven approach to cleaning large face datasets. In *2014 IEEE International Conference on Image Processing (ICIP)*, pages 343–347, Oct 2014. doi: 10.1109/ICIP.2014.7025068. Cited on page 23.

[17] S. Nitish, H. Geoffrey, K. Alex, S. Ilya, and S. Ruslan. Dropout: A simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.*, 15(1): 1929–1958, January 2014. ISSN 1532-4435. URL http://dl.acm.org/citation.cfm?id=2627435.2670313. Cited on page 10.

[18] OpenCV. Opencv python face detection. http://opencv-python-tutroals.readthedocs.org/en/latest/py_tutorials/py_objdetect/py_face_detection/py_face_detection.html, 2016. Accessed: 2016-03-30. Cited on page 20.

[19] B. Schneier. Automatic face recognition and surveillance, October 2015. URL https://www.schneier.com/blog/archives/2015/10/automatic_face_.html. Accessed: 2016-03-14. Cited on page 1.

[20] F. Schroff, D. Kalenichenko, and J. Philbin. Facenet: A unified embedding for face recognition and clustering. In *Computer Vision and Pattern Recognition (CVPR), 2015 IEEE Conference on*, pages 815–823, June 2015. doi: 10.1109/CVPR.2015.7298682. Cited on pages 2, 9, 19, 20, 24, and 35.

[21] M. Stone. Cross-validatory choice and assessment of statistical predictions. *Journal of the Royal Statistical Society B*, 36(2):111–147, 1974. Cited on page 11.

[22] TensorFlow. Tensorflow. `https://www.tensorflow.org/`, 2016. Accessed: 2016-03-24. Cited on page 20.

[23] M. Turk and A. Pentland. Eigenfaces for recognition. *Journal of Cognitive Neuroscience*, 3(1):71–86, Jan 1991. ISSN 0898-929X. doi: 10.1162/jocn.1991.3.1.71. Cited on page 2.

[24] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1, pages I–511–I–518 vol.1, 2001. doi: 10.1109/CVPR.2001.990517. Cited on pages 3 and 15.

[25] wikimedia commons. wikimedia commons. `https://commons.wikimedia.org/wiki/File:Neuron.svg`, 2016. Accessed: 2016-05-13. Cited on page 6.