Bachelor Degree Project

UNIVERSITY OF SKÖVDE

A COMPARATIVE STUDY OF FACIAL RECOGNITION TECHNIQUES

With focus on low computational power

Bachelor Degree Project in Information Technology
Basic level 30 ECTS
Spring term 2019

Timmy Schenkel, Oliver Ringhage, Nicklas Branding

Supervisor: András Márki
Examiner: Juhee Bae

Abstract

Facial recognition is an increasingly popular security measure in scenarios with low computational power, such as phones and Raspberry Pi's. There are many facial recognition techniques available. The aim is to compare three such techniques in both performance and time metrics.

An experiment was conducted to compare the facial recognition techniques Convolutional Neural Network (CNN), Eigenface with the classifiers K-Nearest Neighbors (KNN) and support vector machines (SVM) and Fisherface with the classifiers KNN and SVM under the same conditions with a limited version of the LFW dataset. The Python libraries scikit-learn and OpenCV as well as the CNN implementation FaceNet were used.

The results show that FaceNet is the best technique in all metrics except for prediction time. FaceNet achieved an F-score of 100% while the OpenCV implementation of Eigenface using SVM scored the worst at 15.5%. The technique with the lowest prediction time was the scikit-learn implementation of Fisherface with SVM.

Keywords: Machine Learning, Facial Recognition, Low Computational Power

Table of Contents

1 Introduction

The use of facial recognition is becoming increasingly popular as a security measure in low computational power systems such as phones. One difficulty with using facial recognition as a security measure is the ability to identify an individual in an image correctly within an acceptable time frame.

Facial recognition is used to automatically label an individual in an image by learning from a dataset that contains labeled images of people. The three facial recognition techniques used in this thesis are Eigenface, Fisherface and Convolutional Neural Network (CNN).

The aim is to evaluate already existing facial recognition techniques and classifiers in order to see if they can be used in a limited computational power system. The work is motivated by the lack of comparisons between facial recognition techniques when the techniques are utilized in limited computational power systems. The few comparisons that have been made uses accuracy as the only metric which can give misleading results. Hence it makes it difficult to know which technique would be most suitable for a specific application.

The thesis answers the following three research questions:

1. Is there a significant difference in performance between three different common facial recognition techniques paired with two different classifiers?
2. Is there a significant difference in training time between three different common facial recognition techniques paired with two different classifiers?
3. Is there a significant difference in prediction time between three different common facial recognition techniques paired with two different classifiers?

The chosen methodology for this thesis is an experiment. Statistical tests are used to analyze the numerical data gathered from the experiment in order to prove whether there is a statistical significance between the facial recognition techniques.

This thesis measures and compares accuracy as well as recall, precision, F-score, fallout, prediction time and training time. ANOVA tests are used to prove whether the results have a significant difference or not. If there is a significant difference a Tukey post hoc test is used to see which techniques differ significantly.

2 Background

This chapter presents important concepts and theories needed to understand the problem area. Theories describing the techniques and important concepts such as machine learning, facial recognition, artificial neural networks, subspaces and classifiers related to the techniques must be explored in detail.

2.1 Machine Learning

Machine learning is the concept that machines can perform specific tasks without instructions but rather based upon identifying patterns and receiving rewards (Russel & Norvig 2010). Training time for the machine learning algorithms is when the model is built out of the data. When a model has been created, it can be used to make decisions on any future data. There are three main types of machine learning algorithms, supervised, unsupervised and reinforcement learning.

In supervised learning, all data must be labeled with what the data contains (Russel & Norvig 2010). Each input data must be matched with some output data. Two types of supervised learning exist, namely classification and regression. If the output is a finite set of values, it becomes a classification problem. An example would be trying to determine if a picture contains either a cat or a dog. If the set of output is infinite, it becomes a regression problem. An example would be trying to determine the temperature by looking at the weather.

In unsupervised learning there are no labels which the algorithm can train on, so the algorithm must try to find patterns in the data without any help (Russel & Norvig 2010). The most common task of unsupervised learning is clustering, which is to detect useful clusters in some input data. An example of clustering would be that an algorithm may identify a set of articles being in the same category, even though it was not explicitly told to make splits in the data on categories.

The third type of machine learning is reinforcement learning (Russel & Norvig 2010). This type will either reward or punish the algorithm depending on the choices it makes. The goal is to get as many rewards as possible. An example of reinforcement learning is the program AlphaGo developed by Google Deepmind which learned to play the game Go. This program was used to beat the Go grandmaster Lee Sedol in 2016 (Wang et al. 2016).

Training on the models is performed on a dataset which is split up into two parts, the validation set and the training set, preferably randomly (Russel & Norvig 2010). The majority of the dataset should be trained upon. Just doing a random split on the dataset has some problems, however. The validation could just by chance be a bad reflection of the rest of the data which will grant a poor estimate of the actual accuracy. A better way to test is by doing cross-validation. In cross-validation, the data is split up into k equal subsets which will be used in k rounds of learning. In each round $\frac{1}{k}$ of the data is the validation set and the rest is the training set. This will take k times longer to compute, but the estimates from the model are statistically likely to be accurate. Five and ten are common values for k.

2.2 Artificial Neural Networks

An Artificial Neural Network (ANN) can be described as a directed graph that consists of interconnected processing elements known as neurons or nodes (Svozil, Kvasnicka & Pospichal 1997; Yao 1999). Each node in the network receives the output from the nodes that are connected to the given node as input and performs a transfer function such as Heaviside, sigmoid or Gaussian function. The node then outputs the result to all nodes that the given node is connected to.

Many different types of ANNs exist but the most common one is multi-layer feed-forward (MLF) neural networks (Svozil, Kvasnicka & Pospichal 1997). An MLF neural network contains three types of layers that the nodes are divided into. The first layer is the input layer, the last layer is the output layer and all layers between these two are called hidden layers. These layers are fully connected, which means that every node in a specific layer is connected to all nodes in the next layer.

The degree of importance for a connection between two nodes in the network is evaluated by using a real number called the weight coefficient (Yao 1999). MLF neural networks are trained using supervised learning by comparing the actual and desired output of the ANN. The training process tries to minimize an error function such as the total mean square error between the actual and desired output summed over all available data. The weight coefficients are then iteratively adjusted using a gradient descent-based optimization algorithm such as backpropagation in order to minimize the error.

2.2.1 Convolutional Neural Networks

An MLF requires a lot of parameters since each layer is fully connected which means that they require a lot of computational power (LeCun, Bengio & Hinton 2015; Gogul & Kumar 2017). Convolutional Neural Networks (CNNs) requires less computational power by splitting the work up into smaller layers that each perform smaller tasks. CNNs process arrays of data by taking advantage of natural signals using the four key ideas local connections, shared weights, pooling and many layers.

A typical CNNs architecture can be described as a series of stages where the first two or three stages are composed of convolutional and pooling layers (LeCun, Bengio & Hinton 2015). These stages are followed by more convolutional and fully connected layers. The data from the previous layer is used by a convolutional layer to detect local conjunctions of features and a pooling layer merges semantically similar features from the previous layer into one.

A convolutional layer organizes the nodes in feature maps where each node in a feature map is connected to local patches of the previous layers feature map (LeCun, Bengio & Hinton 2015). This connection is created using a filter bank containing a set of weights where all nodes in a given feature map share the same filter bank but different feature maps in a specific layer have their own filter bank. These feature maps perform a filtering operation known as convolution which is the reason behind the name.

A pooling layer merges semantically similar features by coarse-graining the position of each feature since the relative positions of features can vary (LeCun, Bengio & Hinton 2015). Each node in a pooling layer usually computes the maximum of a local patch within one or a few feature maps. The pooling layer reduces the dimension of the representation and creates an

invariance to small shifts and distortions by shifting the input for neighboring nodes by more than one row or column.

CNNs have over the years improved significantly in performance and are used in many state of the art applications (Parkhi, Vedaldi, Zisserman & others 2015; Amos, Ludwiczuk & Satyanarayanan 2016). The largest factor in the performance of a CNN is the number of images it can train upon. DeepFace is a CNN implementation which has trained on 4.4 million images and shows some good results. FaceNet is another CNN implementation which makes use of one of two different models which has trained on 200 million and 3.3 million images respectively. OpenFace is also a CNN implementation which has trained on 500 thousand images.

2.3 Facial Recognition

Facial recognition is a field within computer vision which identifies and recognizes faces within images or videos (Mehta & Tomar 2016). Facial recognition is used widely in many commercial products, such as Facebook which uses facial recognition to automatically tag people in images. Mastercard is using facial recognition as a payment method, which they refer to as Selfie Pay. Facial recognition has also been used to automatically take attendance at a school. There are two distinct components to facial recognition which are detection and recognition. Detection is to detect and find a face within an image and recognition is the task of recognizing the face in the picture as a specific individual.

Every face is unique in the same way that a fingerprint is unique, even when comparing the faces of identical twins (Rodavia, Bernaldez & Ballita 2017). This implies that the accuracy of a facial recognition system should be able to reach the accuracy levels of a fingerprint scanner. Achieving a good balance between the computational speed and accuracy of a facial recognition technique is a huge challenge and needs further research. The system must be accurate while still being fast enough to not be a nuisance.

The facial recognition technique Sparse Representation Classification is implemented with the intention of being executed on a system with limited computational power, such as a smartphone or a Raspberry Pi (Shen, Yang, Wei, Chou & Hu 2017).

There are many ways to have security in a building, such as RFID cards or some biometric authentication technique (Mehta & Tomar 2016). Biometric authentication methods used today, for example the fingerprint scanner or retina scanner, requires some interaction from the user and takes an unnecessary amount of time to achieve. Facial recognition has the advantage of being unobtrusive and very convenient to use.

2.3.1 Subspaces

Performing facial recognition per-pixel is a computationally expensive task, with the reason being that an image with 250x250 pixels would mean that there is 62500 pixels that needs to be matched to 62500 other pixels (Shakhnarovich & Moghaddam 2004). Sirovich and Kirby (1987) realized that faces are still recognizable even after doing dimensional reductions, making the facial recognition computationally cheaper.

A subspace is created by doing dimensional reductions on multiple images making each image become a point in the subspace. The subspace can be used for facial recognition by performing a dimensional reduction of a new image and then plotting it in a subspace, and then using a

classification algorithm to see which class this new image belongs to. The choice of dimensions affects the time it takes to compute and the performance (Shakhnarovich & Moghaddam 2004). Figure 1 shows a two-dimensional subspace, while a real subspace would be multiple dimensions which is difficult to show on a graph. There is one point in the example for Tony Blair which would most likely be misclassified as Hugo Chavez.

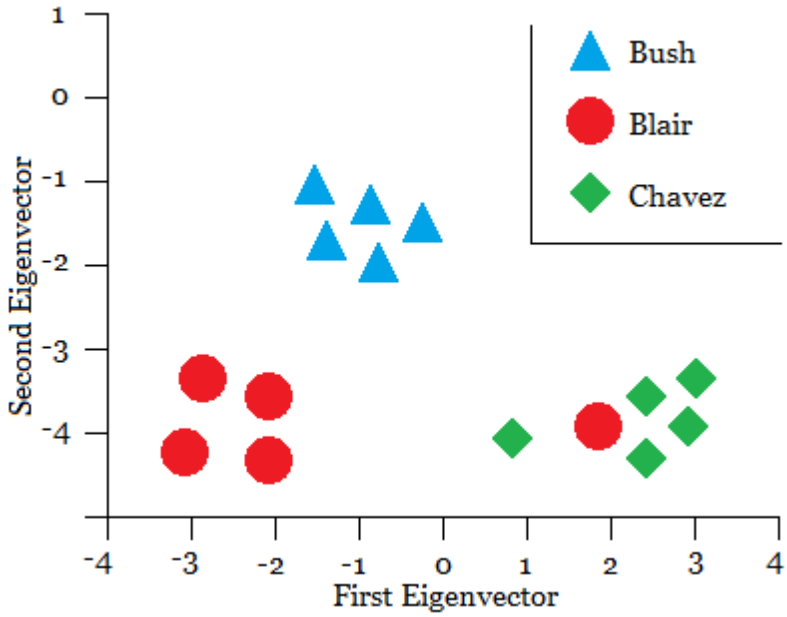

Figure 1 – A two-dimensional plot of a subspace with three different individuals with five images each

2.3.2 Eigenface

The Eigenface technique was invented by Sirovich & Kirby (1987) and was first used for facial recognition by Turk & Pentland (1991). Principal Component Analysis (PCA) is used on a facial image to extract the important features needed for identification and encode it to an image. This encoded version of the image is the Eigenface which can be used to compare with other Eigenfaces for facial recognition (Turk & Pentland 1991).

PCA is an unsupervised technique which is used to find patterns in the data (Turk & Pentland 1991). Such patterns in the images are called features, which can be human features such as the nose and eyes, but also other things such as the illumination in the picture. The number of features will determine both the accuracy and the time it takes to encode.

The major strength of Eigenface is that it is a simple technique that is fast to compute which makes Eigenface suitable for limited computational power systems, such as phones (Shen et al. 2017). Eigenface has some problems with accuracy, especially when it gets challenged by low levels of illumination and different angles, but it is efficient enough to be ran on a smartphone.

2.3.3 Fisherface

While Eigenface can extract and encode all relevant information efficiently, it does not give the best accuracy. The reason for this is that Eigenface also extracts some non-facial features, such as illumination, which means that two different images may be seen as the same, given that the illumination conditions are similar enough. Fisherface is an extension of Eigenface

with an added layer of Linear Discriminant Analysis (LDA). The facial features becomes the important factor rather than non-human features, such as illumination, by using LDA (Belhumeur, Hespanha & Kriegman 1997; Liu, Huang, Lu & Ma 2002). Therefore LDA is more useful for classification of facial features than for example PCA (Sanmoy, Acharya & Bhuva 2018).

The subspace that was created by PCA is then used by LDA to extract fisherfaces. This is done by assuming LDA to be normally distributed, which helps to calculate the discriminant scores used to classify a face to a class label (Belhumeur, Hespanha & Kriegman 1997; Sanmoy, Acharya & Bhuva 2018). To calculate the discriminant scores a Bayes optimal solution is used. If the number of classes exceeds two the more suitable discriminant rule, minimizing within-class differences and maximizing between-class distance, is used (McLachlan 2012). The results from a discriminant rule is called an eigenvalue, meaning it is the value of how good the discriminant rule predicted for a given class label. From the eigenvalues and discriminant scores one can extract the fisherfaces, as the first eigenvalue is associated with the first score, which can be interpreted to a group of faces that have been classified to the first class label (Belhumeur, Hespanha & Kriegman 1997).

2.4 Classifiers

Eigenface and Fisherface does not recognize faces by themselves since they are two ways of putting faces into a subspace in two different ways. A classifier is used for the actual recognition. This chapter presents common algorithms used to classify pictures in the techniques Fisherface and Eigenface and a short description detailing how the algorithms in this case classifies images.

2.4.1 K-Nearest Neighbor

The classification algorithm K-nearest neighbor (KNN) is used in facial recognition to classify different faces in an image.

The algorithm uses a subspace which contains reference and query points that are used to both represent and help classify images to a label name (Garcia, Debreuve & Barlaud 2008). The query point is usually represented in a circle with a plus sign, and the reference points are represented as normal points shown in Figure 2. From these points a subset of query points and reference point can be defined. With the defined subset, one can derive a k-value with the help of reference points. The number of reference points that the query point is connected to is determined by the k-value. The last connected reference point will have the starting point for the distance calculation. The two most general distance calculations are Euclidean and Manhattan distance. From the distance calculations a classification from multiple images are transformed. The value of k should be higher than 1, to make a good use of KNN.

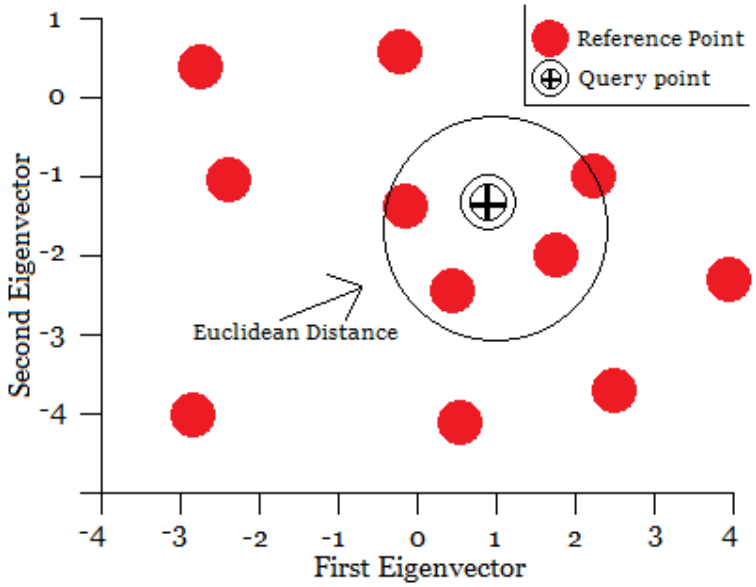

Figure 2 - Illustration of KNN (after Garcia, Debreuve & Barlaud 2008, p. 3)

Figure 2 is an example of KNN, where the query point and reference points are represented respectively as a plus sign and as points representing different images of faces with the calculated Euclidean distance. The query points inside the circle will be classified as the same face.

2.4.2 Support Vector Machine

The classification algorithm support vector machine (SVM) is one of the most influential algorithms in supervised learning (Zhang et al. 2016). The algorithm has been applied to numerous different areas in recognition, like identifying brain disorder such as Alzheimer's disease (Zhang, Wang & Dong 2014), spatiotemporal activity (Behmann, Hendriksen, Müller, Büscher & Plümer 2016) and multiple sclerosis (Zhang et al. 2016).

SVMs use a given set of image inputs from a subspace to classify images to a class label (Dougherty 2012). The linear SVM is a method which finds the most optimal hyperplane in the subspace by maximizing the margin and the perpendicular distance relative to the support vectors. The support vectors are the points closest to the hyperplane, which are used in maximizing the margin and help to calculate the perpendicular distance from the hyperplane. The word perpendicular is used to describe the similarities between the support vectors. Meaning if there exists perpendicular attributes, the support vectors are not similar, meaning the classes are different from each other.

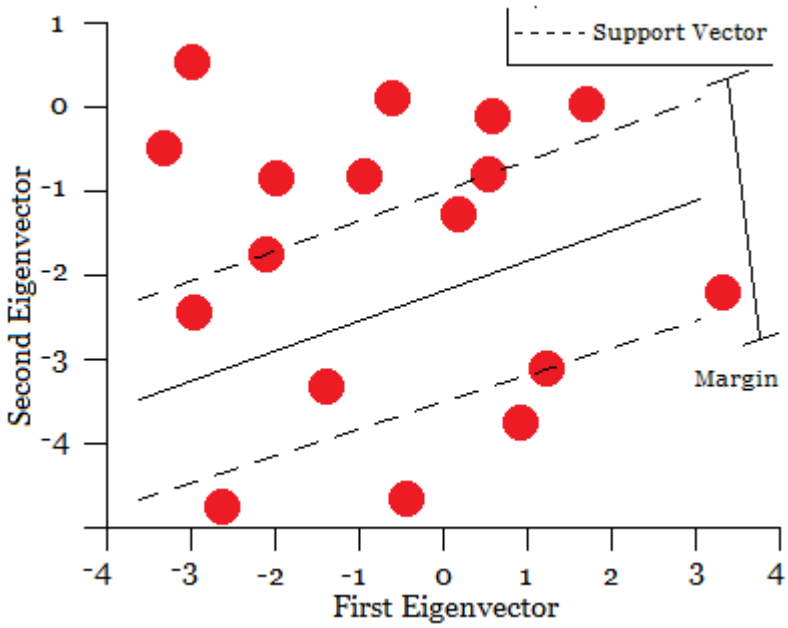

Figure 3 – Illustration of SVM (after Dougherty 2012, p. 118)

Figure 3 represents an example of SVM, with a calculated margin between the two support vectors. The margin is the classifier to separate the two classes.

There are multiple different SVM versions that can be used for facial recognition classification (Hsu, Chang & Lin 2008). Three examples of these SVM versions are the radial based function (RBF), polynomial and the linear version. These SVM versions have an impact on the accuracy and training time. The linear SVM shows to have the optimal accuracy and training time in relation to the other SVM versions when used in facial recognition.

2.5 Libraries & Implementations

A code library is a collection of implementations of behaviors, which are used in this thesis.

2.5.1 Scikit-learn

Pedregosa et al. (2011) introduced the open sourced machine learning library scikit-learn. This library includes a lot of implementations for classification and dimensional reductions. Implementations for the classifiers SVM and KNN are included as well as the dimensional reduction algorithms PCA and LDA. Eigenface and Fisherface can be implemented using the PCA and LDA implementations, respectively, and then be combined with SVM and KNN to classify faces.

2.5.2 OpenCV

OpenCV[1] is an open source computer vision and machine learning library which has many implementations and modules of different algorithms. The face module includes both Eigenface and Fisherface using SVM to classify, but neither facial recognition technique is implemented using KNN to classify.

2.5.3 FaceNet

FaceNet is a unified system for facial recognition that is based on learning a Euclidean embedding per image using a CNN that consist of 22 layers. The CNN used by FaceNet has been trained to directly optimize the embedding itself in order to replace the intermediate bottleneck layer that has been used by previous deep learning approaches (Schroff, Kalenichenko & Philbin 2015). The authors of FaceNet strongly recommends using their model which has been pre-trained on the *VGGFace2*[2] dataset which consists of 3.3 million images for the best results.

2.5.4 OpenFace

OpenFace is a real-time facial recognition system that adapts depending on context where the main design consideration was mobile scenarios (Amos, Ludwiczuk & Satyanarayanan 2016). OpenFace is built around the CNN architecture of FaceNet and the goal was to achieve high accuracy while also achieving low training and prediction times.

[1] OpenCV, accessed 18 February 2019, <https://opencv.org>
[2] VGGFace2, <http://www.robots.ox.ac.uk/~vgg/data/vgg_face2>

3 Problem

In this chapter the aim of the thesis is presented, followed by a motivation on why this is an important aim. This is followed by three research questions and the objectives that must be solved in order to achieve the aim. Hypotheses for the thesis are presented and the appropriate method is also discussed.

3.1 Aim

The aim is to evaluate already existing facial recognition techniques and classifiers in order to see if they can be used in a limited computational power system.

3.2 Motivation

The motivation for this thesis project is to give guidelines for choosing the appropriate facial recognition technique in a limited computational power system. There are lots of studies creating new and cutting-edge facial recognition techniques, but there is a lack of comparisons between the already created, well known and used facial recognition techniques. The few comparisons that exist only compares a few techniques at the time or only uses accuracy, which can be misleading. Parkhi et al. (2015) compares Fisherface with CNN and Santosh Dadi & Mohan (2015) compares Eigenface with Fisherface but there is a lack in comparisons between all three techniques on the same dataset and on limited computational power systems and using more than one performance metric. Amos et al. (2016) compares Eigenface, Fisherface and CNN, but measures only the accuracy metric. This thesis measures and compares five performance metrics, including recall and F-score, as well as the training and prediction times, which was not measured nor compared in the comparisons mentioned. Time metrics are usually a part of performance, but these have been separated in this thesis.

For a security system using facial recognition as an extra layer of security, it is important to balance low computational time and prediction ability which could be implemented at a reasonable cost using a low computational power system, such as a phone or Raspberry Pi.

3.3 Research Questions

1. Is there a significant difference in performance between three different common facial recognition techniques paired with two different classifiers?
2. Is there a significant difference in training time between three different common facial recognition techniques paired with two different classifiers?
3. Is there a significant difference in prediction time between three different common facial recognition techniques paired with two different classifiers?

3.4 Objectives

1. Research the related problem area (All)
2. Implement the programs (All)
3. Train and test Fisherface (Nicklas)
4. Train and test CNN (Timmy)
5. Train and test Eigenface (Oliver)
6. Analyze and compare on performance metrics (All)
7. Analyze and compare on training time (All)
8. Analyze and compare on prediction time (All)

3.5 Hypotheses

A null hypothesis and alternative hypothesis will be used to answer each research question.

- Null and alternative hypothesis for Research Question 1:
  - h_0: Eigenface, Fisherface and CNN will perform similarly when comparing performance in predicting faces.
  - h_1: Eigenface, Fisherface and CNN will not perform similarly when comparing performance in predicting faces.
- Null and alternative hypothesis for Research Question 2:
  - h_2: Eigenface, Fisherface and CNN will perform similarly when comparing training time.
  - h_3: Eigenface, Fisherface and CNN will not perform similarly when comparing training time.
- Null and alternative hypothesis for Research Question 3:
  - h_4: Eigenface, Fisherface and CNN will perform similarly when comparing prediction time.
  - h_5: Eigenface, Fisherface and CNN will not perform similarly when comparing prediction time.

The results that the authors expect are the hypotheses h_1, h_3 and h_5. The hypotheses are based on (Belhumeur, Hespanha & Kriegman 1997; Amos, Ludwiczuk & Satyanarayanan 2016), which compares the techniques on the metric accuracy. Amos et al. (2016) present results that shows an accuracy of 100% for the CNN implementation OpenFace while Eigenface is at 25% when comparing on the same dataset. Belhumeur et al. (1997) shows that Eigenface has huge problems when testing on datasets with varying illumination conditions, while Fisherface handles it well.

Comparing the results from (Schroff, Kalenichenko & Philbin 2015) and (Shen et al. 2017) shows that CNNs require more time and resources to train than the simpler techniques Eigenface and Fisherface.

4 Method

This chapter presents the chosen methodology, metrics and environment used to perform the comparisons along with the techniques that will be compared. The statistical methods used for the comparisons are presented and some validity threats and alternative methods are also discussed.

4.1 Experiment

The appropriate problem, results and conclusions for the research questions is dependent on conducting multiple benchmarks for different techniques, which is best suitable with an experiment. Conducting an experiment gives the ability to control the environment to make good changes of variables for the different techniques to be more suitable for the benchmarks (Berndtsson, Hansson, Olsson & Lundell 2008). Two crucial aspects in experiment characteristics are the ability to evaluate the accuracy of a given technique and the ability to validate the given technique's measures. The strength in experiments is the ability to claim each of the null hypotheses to be rejected or failed to be rejected (Wohlin et al. 2012).

4.2 Selection of Techniques & Metrics

Facial recognition can be performed in multiple different ways and several different techniques has been created for this purpose. The techniques which were selected for the comparison in this thesis are Eigenface, Fisherface and CNN. Both Eigenface and Fisherface are techniques that only aid the process of facial recognition and is combined with a classifier in order to complete the facial recognition process. Two popular classifiers within image recognition, SVM and KNN, are therefore used in conjunction with Eigenface and Fisherface. The techniques compared in this thesis are CNN, Eigenface combined with SVM, Eigenface combined with KNN, Fisherface combined with SVM and Fisherface combined with KNN.

One experiment is conducted for each technique in order to gather data that can be used to compare the techniques. Each experiment gathers data using three measurements, with the first one being *actual and predicted*, which is a pair that contains the actual name and the predicted name of the individual in an image. From the results of *actual and predicted*, a confusion matrix is used to calculate the true positives (TP), false positives (FP), true negatives (TN) and false negatives (FN). Table 1 shows an example of how these values are calculated for one individual but a confusion matrix containing all individuals is used to calculate the values.

Table 1 – Confusion matrix with one individual

		Actual Name	
		Tony Blair	Not Tony Blair
Predicted Name	Tony Blair	True Positive (TP)	False Positive (FP)
	Not Tony Blair	False Negative (FN)	True Negative (TN)

The outcome where the technique correctly labels an image as containing a specific individual is a TP and the outcome where the technique correctly labels an image as not containing a specific individual is a TN. The outcome where the technique incorrectly labels an image as containing a specific individual is a FP, and the outcome where the technique incorrectly labels an image as not containing a specific individual is a FN. The values can be used to calculate accuracy, fallout, recall, precision and F-score which are the performance metrics used to answer the first research question.

The second metric is *training time*, the time it takes to train on the entire dataset, which is used to answer the second research question. The third metric is the time it takes to predict all test images in the data set, the *prediction time*, which is used to answer the third research question. The second and third metrics are measured by taking timestamps in Python before and after the performed action.

4.3 Comparison of Results

A one-way ANOVA test is used to see if there is a significant difference between multiple groups regarding one independent variable (Wiley & Pace 2015). The seven ANOVA tests that will be performed in this thesis uses the different facial recognition techniques as the groups and the different metrics as the independent variables.

The result from the ANOVA test will either reject or fail to reject the null hypothesis. If the null hypothesis is rejected, the hypothesis is false with a given significance, while if it is failed to be rejected, nothing can be said of the outcome (Wohlin et al. 2012).

An ANOVA test can only prove that there is a significant difference between different groups, but not which groups differ. A Tukey post hoc test is used when the ANOVA test has proven that there is a significant difference between the groups in order to see which groups differ and how significant the difference is (Wiley & Pace 2015).

All statistical tests performed in this thesis will use a significance level of 0.05 in order to achieve 95% confidence. This value was chosen because it is a commonly used significance level when performing statistical tests.

4.4 Environment

This thesis defines low computational power as a system that has limited memory and processing power which represents lower-end devices such as phones and single-board computers such as Raspberry Pi's. A virtual machine emulating a Raspberry Pi 3 Model B with a clean install of Ubuntu is used as the low computational power system during each experiment. The reason for conducting each experiment on a clean install of Ubuntu is that the data collection should be as fair as possible.

The selection of dataset is limited since the experiments are performed on a system with limited computational power. This is not a limitation in itself but most systems with limited computational power also has limited disk space which is an important factor when selecting a dataset.

4.5 Validity Threats

Handling validity threats for research projects is an important step to increase the quality of a research project. Furthermore, validity threats fundamentally affect the results and conclusions, which in the end can be applied outside of the study. Therefore, having a strong relationship between measured data and the intended measured data is important to minimize unreliability to results and conclusions which leads to a more generalizable study.

Wohlin et al. (2012) identifies four different categories of validity threats, *Conclusion*, *Internal*, *Construct* and *External Validity*.

Interaction of Setting and Treatment is one type of threat against *External Validity* where the experimental setting or material is not representative of, for example, the industrial practice (Wohlin et al. 2012). One threat against this validity is the selection of a dataset. An uneven distribution of nationality, gender or other important features in the selected dataset could lead to misrepresenting results where the implementation is better at verifying people of a certain nationality or gender.

Instrumentation is a threat against *Internal Validity*, which states that the design of artifacts used can have an effect on the experiment (Wohlin et al. 2012). The structure of the different libraries used in this thesis are unknown which results in a loss of control.

Low Statistical Power is a threat to *Conclusion Validity*, which states that having a low statistical power increases the risk of an erroneous conclusion (Wohlin et al. 2012). This can lead to the inability to reject the erroneous hypothesis.

4.6 Alternative Methods

This thesis could have been constructed with two other methods, case study and implementation.

Case studies are an excellent tool for comparative studies, but they require a population from which statistical samples can be drawn (Wohlin et al. 2012). The intention in a case study is to try to generalize the results by finding similarly characteristic cases and try to find similarities in the analyzes. This is something that is possible for the thesis as the results will give relevance to similar cases, which can be applicable to find value in the research area. One downside with case studies is that the results become hard to generalize.

The method implementation could be used to conduct this thesis by implementing the different techniques and then comparing them as it would give full control over the environment used for testing, but this could lead to a lot of problems (Wohlin et al. 2012). It is completely possible to both build and compare the techniques, but since there are already existing method implementations, creating a new method is deemed unnecessary. A lot of validity threats which would be applicable are avoided by doing an experiment instead of implementation.

The greatest advantage that an implementation would give is full control of the environment (Wohlin et al. 2012). This does not outweigh all cons however, since it can be very difficult to implement something and to prove that it works as it should. If a technique performs badly, it would be impossible to determine if it is because of the technique, or the implementation (Wohlin et al. 2012).

5 Related Work

A comparison was done of the techniques which are available in the open source framework *FaceRecLib*[3] (Sadhya, Gautam & Singh 2017). The techniques are used over several different facial datasets which all has different properties, such as angle of the face and illumination. They conclude that Eigenface gives the worst accuracy compared to the seven other techniques. The only measured metric in the paper is accuracy, and not computational time, which is going to be a focus point in this thesis. Since Fisherface and CNN is not available in FaceRecLib, these are not explored nor compared in their study. This comparison will be examined in this thesis.

Shen et al. (2017) implemented their own facial recognition technique which they later compared with six facial recognition techniques, including Eigenface and Fisherface. They concluded that while their newly implemented technique has an increased accuracy of 8-17%, it takes at least two times as long in computational time when comparing to Eigenface and Fisherface. Eigenface and Fisherface has the same accuracy, but Eigenface is much faster to compute.

In the last decades several comparisons with different datasets and techniques have been studied (Belhumeur, Hespanha & Kriegman 1997; Delac, Grgic & Grgic 2006). The study from (Santosh Dadi & Mohan 2015) compared the techniques Eigenface and Fisherface with different datasets. The ORL dataset, which contains a set of face images, is used as well as a proposed enhancement of said dataset. The experimental setup consisted of converting 400 images of the ORL dataset into PNG, GIF and TIFF images, and compared Fisherface and Eigenface. The results were put together in a table, where the findings concluded that Fisherface has the best accuracy of 90.07% with the enhanced dataset converted to TIFF.

Belhumeur et al. (1997) does a comparison between facial recognition techniques which include Fisherface and Eigenface. Two datasets called Harvard Robotics Laboratory and Yale were used. The focus of the comparison was on different levels of illumination. The experiment was conducted by extracting 330 images of the same five individuals, 66 for each individual, and created five subsets with different illuminations conditions. They then compared Eigenface against Fisherface with table of relative performance of the many conducted experiments (interpolating, extrapolating, glasses and multiple PCA:s). The performance of the techniques Eigenface and Fisherface concluded that Fisherface was the best facial recognition technique and that if Eigenface removes the first three PCA:s the performance of the technique will improve significantly (Belhumeur, Hespanha & Kriegman 1997).

Gunawan et al. (2017) performed an experiment with Eigenface on a Raspberry Pi. They concluded that the recognition rate was up to 90% on the three tested individuals. The experiment was conducted with a camera setup on the Raspberry Pi and a test of three individuals with ten iterations for evaluation. The results found that nine out of ten tests were successful in verifying the correct user. Fisherface was considered for the experiment, but the computational requirements of Fisherface were too high for the Raspberry Pi to handle.

A comparison between seven different facial recognition techniques was conducted on two different datasets by Parkhi et al. (2015). Their conclusion was that FaceNet with alignment had the best accuracy and Fisherface had the worst accuracy. Their study only focused on the

[3] FaceRecLib, <https://github.com/idiap/facereclib>

metric accuracy whereas this thesis will focus on metrics around computational time as well. The facial recognition technique Eigenface will be explored in this thesis but was not explored in their comparison.

An implementation of the existing facial recognition technique CNN called OpenFace was created by Amos et al. (2016) where the focus was on mobile scenarios. A comparison was conducted to see how accurate OpenFace was compared to existing techniques such as Fisherface and Eigenface. The results presented shows that OpenFace was the best technique and Eigenface the worst when comparing accuracy and training time. Another aspect to the results is that OpenFace had the highest prediction time while Fisherface is the technique with the lowest prediction time. The only metric used for performance by Amos et al. (2016) is accuracy while this thesis will use five different metrics for performance.

6 Implementation

This chapter presents the dataset that was used in the experiment, how the experiments were conducted, how the results were compared and the specifications of the used machine.

6.1 Dataset

The Labeled Faces in the Wild (LFW) dataset used in this thesis was created by the University of Massachusetts. The dataset consists of more than 13000 images of faces which all are labeled with the name of the individual (Huang, Ramesh, Berg & Learned-Miller 2007). Every technique used in the experiment needs labels for the training phase, since they are supervised techniques.

Out of the LFW dataset the ten individuals with the most images are selected to build a limited version of the dataset. Each of these ten individuals have 40 images intended for training and four images for testing. This gives a total of 440 images that are split into 400 training images and 40 testing images in total. This split of the images was done randomly. One more random split has been done to group these images into ten parts for 10-fold cross-validation, which will be used to mitigate the possibility of the dataset getting a misrepresenting split.

6.2 Training & Testing

Eigenface and Fisherface has been tested in the open source libraries OpenCV and scikit-learn. CNN will be tested with the implementation called FaceNet. These libraries and implementations are written in Python.

OpenCV has implementations for both Eigenface and Fisherface using SVM to classify. Scikit-learn has been used to implement Eigenface with both KNN and SVM as well as Fisherface with both KNN and SVM. The CNN implementations FaceNet and OpenFace were intended to be used for the comparison but OpenFace could not be implemented which resulted in that FaceNet was the only CNN implementation used.

6.3 Configurations

Table 2 presents all changed settings for Eigenface and Fisherface. The default values were used for all parameters that are not presented below. The values for the changed settings was taken from the scikit-learn documentation.

Table 2 – Settings used for facial recognition techniques

Module	Setting	Description
Library: sklearn Package: decomposition Class: PCA	n_components = 150	The number of components that will be used for the facial recognition. A higher number should in theory result in both better predictions and longer prediction times. The default value is: None.
	whiten = True	Removes noise and improves the predictive accuracy. The default value is: False.
Library: sklearn Package: lda Class: LDA	n_components = 150	The number of components for dimensionality reduction. A higher number should in theory result in both better predictions and longer predication times. The default value is: None.
Library: cv Package: face Class: EigenFaceRecognizer	num_components = 150	This setting has the same effect as the previously mentioned sklearn.decomposition.PCA n_components setting. The default value is: 0.
Library: cv Package: face Class: FisherFaceRecognizer	num_components = 150	This setting has the same effect as the previously mentioned sklearn.lda.LDA n_components setting. The default value is: 0.

Table 3 presents all settings for the classifiers used. The default values are used if nothing else is specified. As discussed in chapter 2.4.1 - K-Nearest Neighbor, the choice of k will be a significant factor in the result. All numbers within the range of 1 to 100 was tested to find the best k value for each of the models which uses KNN, and the best k was used to gather the results from the experiments. The range of 1 to 100 is arbitrary, but the authors of scikit-learn mentions that 25 should be sufficient. The k-value which gave the best result was 13 for the Eigenface implementation and 17 for the Fisherface implementation.

Using the value *scale* for the gamma setting means that it will automatically be set to an appropriate value.

Table 3 – Settings used for the classifiers

Module	**Setting**	**Description**
Library: sklearn Package: neighbors Class: KNeighborsClassifier	k = 13 for Eigenface	k is the number of neighbors used to classify a point to a specific class. The default value is: 5.
	k = 17 for Fisherface	
Library: sklearn Package: svm Class: SVC	gamma = 'scale'	The gamma parameter defines how far the influence of a single training example reaches, with low values meaning 'far' and high values meaning 'close'. The default value is: "auto_deprecated".
	kernel = 'linear'	An SVM can work in many ways and for these experiments a linear type was chosen for the speed advantage over other kernel types. The default value is: "rbf".
	class_weight = 'balanced'	The dataset used in the experiments has the same amount of testing and training images which would make the class weights balanced. The default value is: None.

The settings that were changed for the techniques Eigenface and Fisherface are presented in Table 2 and the changed settings for the classifiers KNN and SVM are presented in Table 3. There are no presented settings for FaceNet because there were no configurable parameters.

6.4 Specifications

The intention was to conduct the experiments on a Raspberry Pi 3 Model B which ended up being broken. The decision was then made to conduct the experiments on a virtual machine that was hosted by a laptop with the specifications shown in Table 4 in order to emulate the low computational power of the Raspberry Pi.

The virtual machines were created with as similar processing power to a Raspberry Pi as possible, except for the amount of RAM. FaceNet was the only technique that were unable to complete the training phase using 1GB of RAM and the decision was made to double the amount of memory to 2GB of RAM when testing all techniques.

Three virtual machines were created with the same operating system as the laptop, with one library respectively. All tests were performed on the CPU and under the same conditions, meaning with a plugged-in charger and no other programs running in the background.

Table 4 - Specifications

Specification	**ASUS 550LN**
Operating System	Ubuntu 18.04 LTS
CPU	Intel i5-4200U @ 1.6GHz
GPU	GeForce 840M
RAM	6GB DDR3 1600MHz
Storage	HDD 5400RPM

7 Results

The data used in the results are taken from seven different models. Four models were created from the library scikit-learn, two from the library OpenCV and one from FaceNet.

7.1 Presentation of Results

Table 5 specifies all used techniques and their names in the coming figures and tables. More detailed results for each individual and performance metric is presented in Appendix A - Performance per Individual.

Table 5 – All used techniques

Name	Library	Technique
FaceNet (FN)	FaceNet (Implementation)	CNN
CV_Eigen	OpenCV	Eigenface with SVM
CV_Fisher	OpenCV	Fisherface with SVM
S_Eigen_K	scikit-learn	Eigenface with KNN
S_Eigen_S	scikit-learn	Eigenface with SVM
S_Fisher_K	scikit-learn	Fisherface with KNN
S_Fisher_S	scikit-learn	Fisherface with SVM

7.1.1 Accuracy

All tested techniques achieve a high accuracy of above 80%, as presented in Figure 4, with *FaceNet* reaching 100%. This metric is often the only presented measurement by papers in the area of facial recognition which can give misrepresented results. The reason for why accuracy can be misrepresented is that it is calculated by taking all correct predictions, TPs and TNs, and dividing that value by the total population, which mathematically is: $\frac{TP+TN}{Total\ Population}$. The limited version of the LFW dataset used in this thesis contains ten classes, individuals, with four images each resulting in a population of 40. If only one out of the four test images for Tony Blair gets classified correctly this would give a TP value of 1 and a TN value of 36 which gives an accuracy of $\frac{37}{40}$, 92.5%, even though Tony Blair only got correctly classified one out of four times.

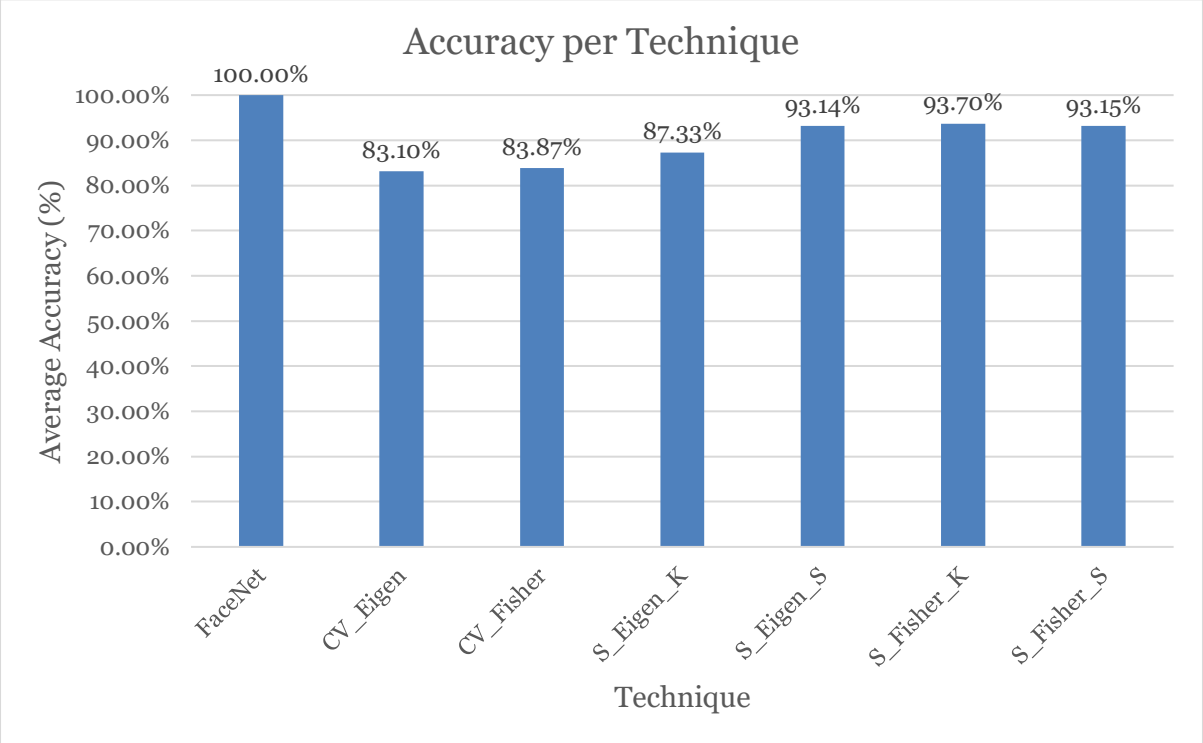

Figure 4 – Graph over the average accuracy, measured in percentage

The one-way ANOVA test shows that there is a significant difference between the techniques which means that the null hypothesis, h_o, is rejected since p is lower than the significance level ($F_{(6,63)} = 309$, $p = .0001$). These results only show that there is a significant difference between the techniques. A Tukey post hoc test presented in Table 6, was then used to see which techniques differ from each other.

Green with diagonal lines shows that the technique on the row is significantly better, with a p-value of less than 0.05, than the technique on the column. Red shows that the technique on the row is significantly worse than the technique on the column. Yellow shows the techniques which are not significantly different and the p-value.

Table 6 – Results from Tukey post hoc on Accuracy

	FN	CV_Eigen	CV_Fisher	S_Eigen_K	S_Eigen_S	S_Fisher_K	S_Fisher_S
FN							
CV_Eigen			**0.696**				
CV_Fisher		**0.696**					
S_Eigen_K							
S_Eigen_S						**0.909**	**1**
S_Fisher_K					**0.909**		**0.919**
S_Fisher_S					**1**	**0.919**	

7.1.2 Recall

Figure 5 shows that *FaceNet* has the best possible recall of 100% which is 31.5 percent points better than the scikit-learn implementation of Fisherface using KNN which performed second best at 68.50%. The recall is calculated by taking all TPs and divide it by the number of test images for the respective class, P, which mathematically is: $\frac{TP}{P}$. If only one out of the four test images of Tony Blair get classified correctly and the remaining three got classified incorrectly, this would give a TP value of 1, which gives a recall of $\frac{1}{4}$, 25%. This gives a better representation of the performance of the technique than accuracy, since it only cares about the TPs, rather than the TNs. A high recall does imply that the technique performs well and has a high rate of correct predictions.

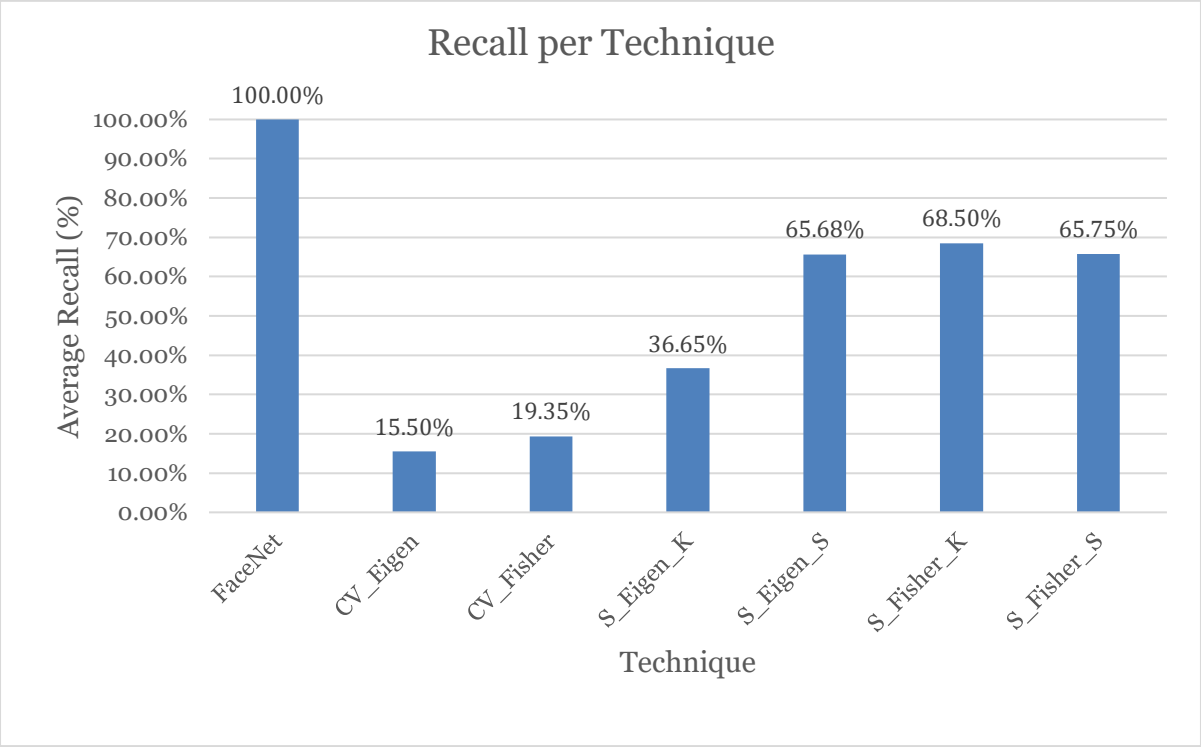

Figure 5 - Graph over the average recall, measured in percentage

The one-way ANOVA test shows that there is a significant difference between the techniques which means that the null hypothesis, h_0, is rejected ($F_{(6,63)} = 309$, $p = .0001$). A Tukey post hoc was performed and the results are presented in Table 7.

Table 7 - Results from Tukey post hoc on Recall

	FN	CV_Eigen	CV_Fisher	S_Eigen_K	S_Eigen_S	S_Fisher_K	S_Fisher_S
FN							
CV_Eigen			**0.702**				
CV_Fisher		**0.702**					
S_Eigen_K							
S_Eigen_S						**0.909**	**1**
S_Fisher_K					**0.909**		**0.919**
S_Fisher_S					**1**	**0.919**	

7.1.3 Precision

Figure 6 shows that *FaceNet* achieves the best possible precision of 100%. The second best technique is the scikit-learn implementation of Fisherface using KNN once again with a precision of 70.44%. These results show that FaceNet outperforms all other techniques with at least 29 percent points regarding this metric as well. Precision is calculated by dividing the TPs with all positive predictions, whether correct or not, which mathematically is: $\frac{TP}{TP+FP}$. Following the same example as before, Tony Blair only gets one image predicted correctly, and the rest incorrectly. If an image from Hugo Chavez would be classified as Tony Blair, this would give 1 TP and 1 FP for Tony Blair. This would give a precision of $\frac{1}{2}$, 50%, which gives a better representation of the technique's performance than accuracy.

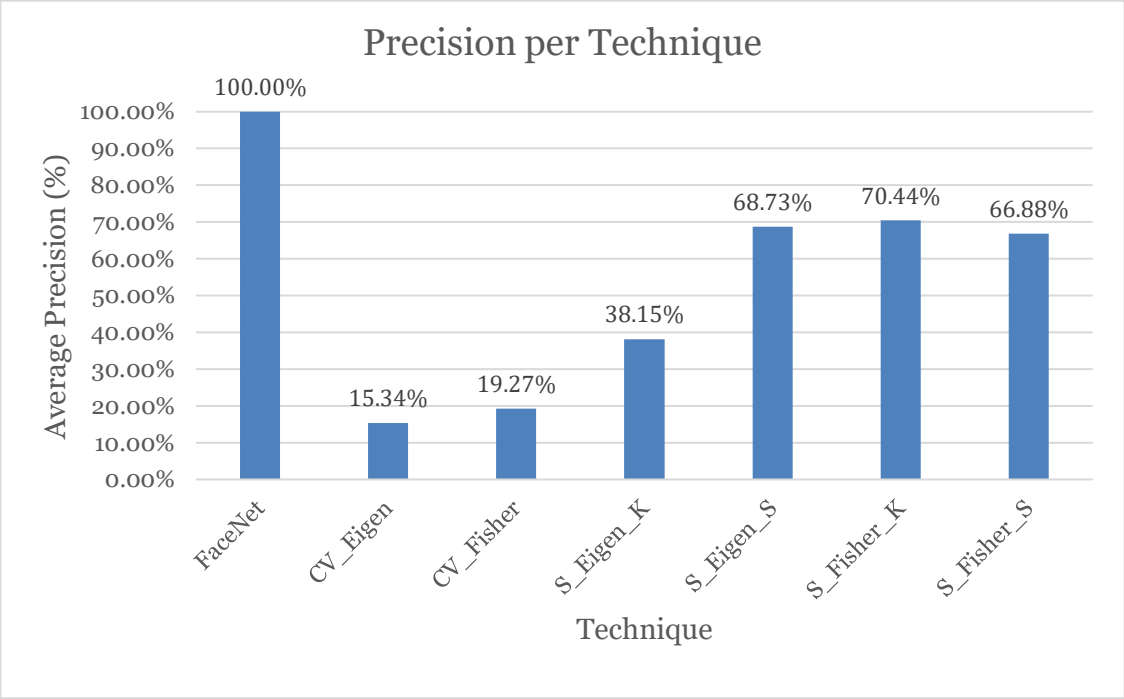

Figure 6 - Graph over the average precision, measured in percentage

The one-way ANOVA test shows that there is a significant difference between the techniques which means that the null hypothesis, h_0, is rejected ($F_{(6,63)} = 250$, $p = .0001$). A Tukey post hoc was performed and the results are presented in Table 8.

Table 8 - Results from Tukey post hoc on Precision

	FN	CV_Eigen	CV_Fisher	S_Eigen_K	S_Eigen_S	S_Fisher_K	S_Fisher_S
FN							
CV_Eigen			**0.787**				
CV_Fisher		**0.787**					
S_Eigen_K							
S_Eigen_S						**0.996**	**0.994**
S_Fisher_K					**0.996**		**0.854**
S_Fisher_S					**0.994**	**0.854**	

7.1.4 F-score

The graph in Figure 7 shows the average F-score of the different techniques. F-score is arguably the best metric to use to see which technique performs the best, both in doing correct predictions and not doing incorrect predictions. The mathematical formula for F-score is: $\frac{Precision * Recall}{Precision + Recall} * 2$. Using the same example as above, this would give a F-score of 33%. *FaceNet* is the best with 100% with over 32 percent points to the next best technique which is the scikit-learn implementation of Fisherface using KNN once again with a F-score of 67.23%.

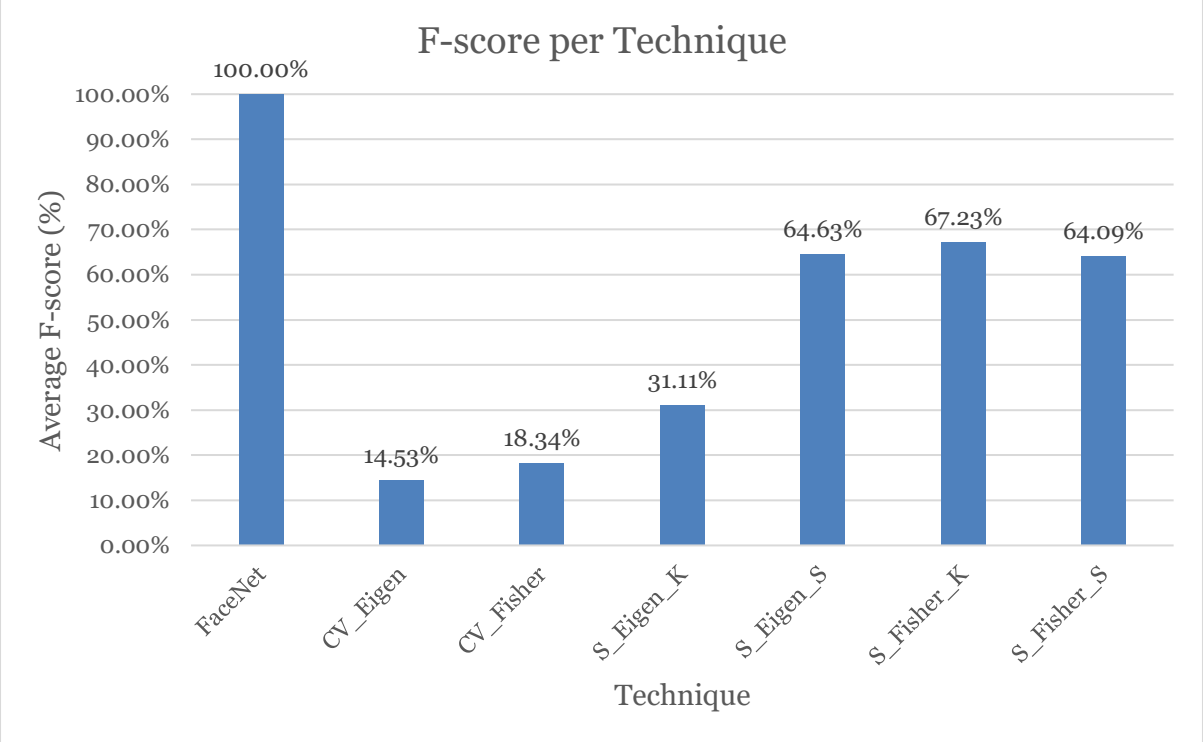

Figure 7 - Graph over the average F-score, measured in percentage

The one-way ANOVA test shows that there is a significant difference between the techniques which means that the null hypothesis, h_0, is rejected ($F(_{6,63}) = 93$, $p = .0001$). A Tukey post hoc was performed and the results are presented in Table 9.

Table 9 - Results from Tukey post hoc on F-score

	FN	CV_Eigen	CV_Fisher	S_Eigen_K	S_Eigen_S	S_Fisher_K	S_Fisher_S
FN							
CV_Eigen			**0.98**	**0.372**			
CV_Fisher		**0.98**		**0.067**			
S_Eigen_K		**0.372**	**0.067**				
S_Eigen_S						**0.997**	**1**
S_Fisher_K					**0.997**		**0.991**
S_Fisher_S					**1**	**0.991**	

7.1.5 Fallout

The graph in Figure 8 below shows the average fallout of the different techniques. Fallout is the FP rate, meaning the probability of a false alarm, or an incorrect prediction, which mathematically is: $\frac{FP}{FP+TN}$. The optimal of this is 0%. The best fallout is achieved by FaceNet with the optimal of 0% and the worst by the OpenCV implementation of Eigenface with 9.39%.

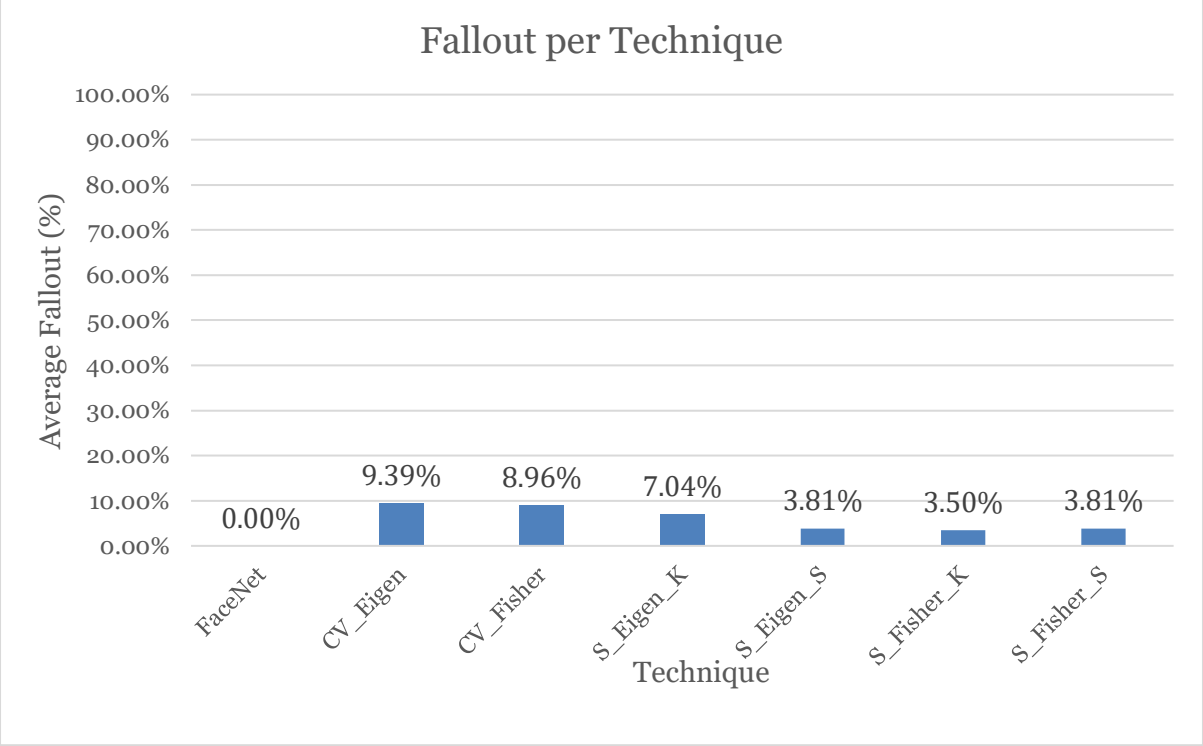

Figure 8 - Graph over the average fallout, measured in percentage

The one-way ANOVA test shows that there is a significant difference between the techniques which means that the null hypothesis, h_0, is rejected ($F_{(6,63)} = 310$, $p = .0001$). A Tukey post hoc was performed and the results are presented in Table 10.

Table 10 - Results from Tukey post hoc on Fallout

	FN	CV_Eigen	CV_Fisher	S_Eigen_K	S_Eigen_S	S_Fisher_K	S_Fisher_S
FN							
CV_Eigen			**0.704**				
CV_Fisher		**0.704**					
S_Eigen_K							
S_Eigen_S						**0.909**	**1**
S_Fisher_K					**0.909**		**0.92**
S_Fisher_S					**1**	**0.92**	

7.1.6 Training Time

Training time is the time it takes to train the model for a specific technique on the 400 training images in the dataset. The results are shown in Table 11. Green indicates the technique which had the shortest training time and red indicates the technique with the longest training time.

Table 11 - Table showing the models with the best and worst training times

Models	**Average Training Time (sec)**
FaceNet	**0.24**
opencv_eigen_svm	3.68
opencv_fisher_svm	**4.19**
scikit_eigen_knn	0.38
scikit_eigen_svm	0.43
scikit_fisher_knn	1.02
scikit_fisher_svm	0.93

The one-way ANOVA test shows that there is a significant difference between the techniques which means that the null hypothesis, h_2, is rejected ($F(_{6,63}) = 5060$, $p = .0001$). A Tukey post hoc was performed and the results are presented in Table 12.

Table 12 - Results from Tukey post hoc on Training Time

	FN	CV_Eigen	CV_Fisher	S_Eigen_K	S_Eigen_S	S_Fisher_K	S_Fisher_S
FN							
CV_Eigen							
CV_Fisher							
S_Eigen_K					**0.832**		
S_Eigen_S				**0.832**			
S_Fisher_K							**0.158**
S_Fisher_S						**0.158**	

7.1.7 Prediction Time

Prediction time is the time it takes to predict all 40 testing images in the dataset. The results are shown in Table 13. Green indicates the technique which had the shortest prediction time and red indicates the technique which had the longest prediction time.

Table 13 - Table showing the models with the best and worst prediction times

Models	**Average Prediction Time (sec)**
FaceNet	0.0067
opencv_eigen_svm	**0.0982**
opencv_fisher_svm	0.0075
scikit_eigen_knn	0.0070
scikit_eigen_svm	0.0034
scikit_fisher_knn	0.0029
scikit_fisher_svm	**0.0007**

The one-way ANOVA test shows that there is a significant difference between the techniques which means that the null hypothesis, h_4, is rejected ($F_{(6,63)} = 427$, $p = .0001$). A Tukey post hoc was performed and the results are presented in Table 14.

Table 14 - Results from Tukey post hoc on Prediction Time

	FN	CV_Eigen	CV_Fisher	S_Eigen_K	S_Eigen_S	S_Fisher_K	S_Fisher_S
FN			**1**	**1**	**0.83**	**0.193**	**0.697**
CV_Eigen							
CV_Fisher	**1**			**1**	**0.627**	**0.091**	**0.427**
S_Eigen_K	**1**		**1**		**0.755**	**0.144**	**0.606**
S_Eigen_S	**0.83**		**0.627**	**0.755**		**0.92**	**1**
S_Fisher_K	**0.193**		**0.091**	**0.144**	**0.920**		**0.974**
S_Fisher_S	**0.697**		**0.472**	**0.606**	**1**	**0.974**	

7.2 Analysis

This chapter contains an analysis of the results and answers the research questions. Objective 6 focuses on the performance metrics, Objective 7 on the training time and Objective 8 on the prediction time.

7.2.1 Objective 6 – Performance Metrics

Most papers in the area of facial recognition uses accuracy as the only metric for performance, which can give the impression that high accuracy equals good performance, even though this often is not the case. The average accuracy of the OpenCV implementation of Eigenface using SVM to classify shown in Figure 4 is 83.1%, even though every other performance metric shows that the technique is not really usable since the technique does a lot of incorrect predictions. Metrics such as recall, precision and F-score should therefore be used in order to display more accurate results that represent the techniques actual performance. *FaceNet* is the overall best in every single performance metric.

The most surprising aspect about the results is the difference between the OpenCV and scikit implementations of the same technique. The results show that the scikit implementations perform significantly better in all metrics except for prediction time compared to the OpenCV counterparts even though they are different implementations of the same technique. Comparing the F-score between the OpenCV implementation of Fisherface with the SVM classifier and the scikit implementation of the same technique shows that the scikit implementation has an F-score of 64.09% compared to OpenCVs 18.34%, which gives the scikit implementation 3.5 times better performance. One possible reason for this could be the default values that are used in the implementations. If these differ greatly, it would explain the differences, but this needs to be explored further.

Another surprising aspect of the results is the difference in performance comparing Fisherface with the classifiers KNN and SVM using the scikit implementation. The results show that Fisherface with KNN performs better than Fisherface with SVM regarding recall, precision and F-score. One reason could be that Fisherface with SVM uses the default settings, while

Fisherface with KNN was tested using different k-values to find the optimal k-value, but this should be explored further.

The results from the ANOVA tests that were performed on each performance metric shows that there is a significant difference between the techniques and the null hypothesis, h_0, could be rejected with 95% confidence which translates to that the alternative hypothesis, h_1, is accepted and the answer to the first research question is that there is a difference in performance between the techniques. A Tukey post hoc test shows that FaceNet performs significantly better compared to all other techniques in every performance metric, with a significance level of 0.05.

7.2.2 Objective 7 – Training Time

FaceNet achieves the fastest training time. The reason for this is that the training time only measures the time to optimize a pre-trained model for the dataset used in this thesis by training the top layer. The pre-trained model used for these experiments were trained on a dataset which contains over three million images of over 9000 individuals and trained for 30 hours on a machine with a Nvidia Pascal Titan X GPU which has a lot more computational power than the machine used for the experiments performed in this thesis. To train a model on a dataset of that size is not feasible on a low computational power system which would require access to more resources and better hardware during the training phase, but a pre-trained model can be used as they are publicly available.

The OpenCV implementations takes considerably longer time to train than the scikit-learn implementations of the same techniques. A comparison of both Eigenface with SVM implementations shows that the OpenCV version takes 8.8 times longer than the scikit-learn version of the same technique. Another comparison shows that the OpenCV implementation of Fisherface with SVM takes 4.5 times longer to train than the scikit-learn counterpart. A possible explanation for this immense difference is that the scikit-learn implementations are much more optimized than the OpenCV version. The library scikit-learn starts by looking at the data to find the optimal settings for performing the dimensional reductions in order to get the best results which is not possible in OpenCV.

An increase in training time can be seen when comparing the two different Eigenface implementations in scikit-learn, where the only difference is the classifier. This increase was expected for the Fisherface implementation as well, but this is not the case. This contradicts what most papers state about the time difference when comparing KNN with SVM.

An interesting observation of the training time results is that the classifiers for the Fisherface and Eigenface techniques in scikit-learn did not show any significant difference, meaning that the choice of classifier is not of importance. This is probably due to the small amount of training data used in this thesis. A significant difference may be proven when using a larger dataset.

The results from the ANOVA test on training time shows that there is a significant difference between the techniques and the null hypothesis, h_2, could be rejected with 95% confidence which translates to that the alternative hypothesis, h_3, is accepted and the answer to the second research question is that there is a difference in training time between the techniques. A Tukey post hoc test shows that FaceNet has a significantly shorter training time when compared to all other techniques, with a significance level of 0.05.

7.2.3 Objective 8 – Prediction Time

The results from Table 13 shows that the scikit-learn implementation of Fisherface with SVM achieves the fastest prediction time with a time that is four times faster than the same implementation with KNN. This contradicts what most papers state about the time difference when comparing KNN with SVM which is that KNN should be fast to predict. The reason for this may be that there is a small amount of prediction images. SVM may be faster to predict on small datasets compared to KNN. Even though there is a four times difference comparing SVM and KNN in combination with Fisherface, it was not proven as a significant difference by the Tukey post hoc test with a significance level of 0.05.

The difference between the OpenCV and scikit-learn implementations of the techniques is also present when comparing prediction time as displayed in Table 13. The most substantial difference is the prediction time for OpenCV with Eigenface which is 29 times longer than the prediction time for the scikit-learn counterpart. This difference in prediction time between OpenCV and scikit-learn is similar to the difference in training time between the implementations and the possible reasons for these differences may be the same. One reason for this difference was mentioned in chapter 7.2.2 - Objective 7.

The results from the ANOVA test on prediction time shows that there is a significant difference between the techniques and the null hypothesis, h_4, could be rejected with 95% confidence which translates to that the alternative hypothesis, h_5, is accepted and the answer to the third research question is that there is a difference in prediction time between the techniques. A Tukey post hoc test shows that the OpenCV implementation of Eigenface using SVM to classify is significantly worse compared to all other techniques. No significant difference could be proven between the other techniques.

8 Discussion

This chapter includes a summary of the entire thesis, followed by a comparison to the related work, validity threats, ethics and potential future works.

8.1 Summary

The aim of this thesis was to evaluate already existing facial recognition techniques and classifiers to see if they can be used in a limited computational power system. The motivation for this was that there is a lack of comparisons between existing techniques, especially when it comes to comparing time metrics.

The three techniques used in this thesis were Eigenface, Fisherface and CNN and the classifiers were SVM and KNN. In order to test these techniques, the libraries OpenCV and scikit-learn was used, as well as FaceNet. Every technique has been tested using the same dataset with 10-fold cross-validation and ten seeded runs per fold.

The best technique when comparing the performance metrics is FaceNet. Three out of the four scikit-learn techniques could be used in some system where the accuracy is not of utmost importance, such as a face tagging system, where mis-predictions could happen without doing any significant damage. ANOVA tests, with a significance level of 0.05, proves that there is a significant difference in performance between the techniques on all metrics. A Tukey post hoc test, with a significance level of 0.05, shows that FaceNet performs significantly better on all performance metrics.

FaceNet achieves the best training time out of all the techniques in the experiment when using a pre-trained model. An ANOVA test proves that there is a significant difference in training time between all techniques. A Tukey post hoc test shows that FaceNet has a significantly shorter training time than all other techniques.

An ANOVA test proves that there is a significant difference in the prediction time between all techniques. A Tukey post hoc test shows that the Eigenface with SVM implementation from OpenCV has a significantly worse prediction time comparing with all other techniques. No significant difference can be proven between the other techniques on the prediction time. Even though no significant difference can be proven the scikit-learn implementation of Fisherface using SVM has a prediction time which is 8.6 faster than FaceNet and needs to be considered if the prediction times are more important than the performance. More measurements on larger datasets could result in a significant difference in prediction time between the two techniques.

The guideline is to use the scikit-learn implementation of Fisherface using KNN as the classifier when the training phase must be conducted on a limited computational power system with very limited amounts of RAM. The reason is that the best performing technique, FaceNet, were unable to train on a system with 1GB RAM as mentioned in chapter 6.4 - Specifications. The scikit-learn implementation of Fisherface using SVM as the classifier is the recommended technique for scenarios where the prediction time is of higher priority than the performance. FaceNet is the recommended technique in all other scenarios since it had the best performance across all metrics.

8.2 Conclusion

This chapter will compare the results from earlier works and mention some validity threats that must be explained to fully understand the results, as well as some ethical considerations.

8.2.1 Comparisons to Previous Work

Shen et al. (2017) compares the OpenCV implementations of Eigenface and Fisherface and finds that there is no significant difference in accuracy between the techniques, which is in line with the results from this thesis. Their results show that Fisherface has a prediction time which is 125% worse than Eigenface, while the results from this thesis shows that Fisherface has a prediction time which is 1306% better than Eigenface. One possible reason could be that Shen et al. (2017) uses 100 components while the experiments conducted in this thesis used 150 components, as presented in Table 2. The results presented by Shen et al. (2017) and this thesis shows that the OpenCV implementations of Eigenface and Fisherface are comparable to each other but neither of these results are comparable to the results of their scikit-learn counterparts presented in this thesis, as mentioned in chapter 7.2 - Analysis.

Amos et al. (2016) compares OpenFace to Eigenface and Fisherface on the metrics accuracy, training time and prediction time. Comparing the presented accuracy with the results presented in this thesis shows a considerable difference. Amos et al. (2016) used the OpenCV implementation of Eigenface and Fisherface to present an accuracy of 25% for Eigenface and 75% for Fisherface whereas the results presented in this thesis is 83.10% for Eigenface and 83.87% for Fisherface. The training time for Eigenface and Fisherface in this thesis has them in the opposite order compared to the results of Amos et al. (2016) but the prediction times are in line with the results from this thesis. Details such as chosen settings were not presented by Amos et al. (2016) which makes it difficult to pinpoint the reason for why the results differ, but one reason could be the number of components used.

Parkhi et al. (2015) compares FaceNet to Fisherface on multiple datasets using two metrics which is accuracy and Equal Error Rate, which is not measured in this thesis. FaceNet achieves an average accuracy of 97% and Fisherface using SVM 88.5%. The details on how the Fisherface and SVM is implemented very sparse which makes it difficult to determine why their results are quite good compared to FaceNet.

Santosh Dadi & Mohan (2015) concludes that Fisherface has a significantly higher accuracy of 87.8% compared to Eigenface with an accuracy of 71.6%. Their results differ from the results presented in this thesis which shows that a significant difference between the techniques could not be proven when comparing the implementations from the same library. A possible reason is that the dataset used by Santosh Dadi & Mohan (2015) could have more varying illumination conditions compared to the dataset used in this thesis, which Fisherface handles better than Eigenface. They implement the techniques using a different library, but do not mention any of the settings used which makes it difficult to compare the results.

The experiments in this thesis uses a KNN with the default distance calculation which is the Euclidean distance. Delac et al. (2006) compares Eigenface with Fisherface using KNN as the classifier and multiple different ways to measure the distance between points. The results from this thesis shows that Eigenface with KNN has a significantly worse accuracy than Fisherface with KNN, while the results from Delac et al. (2006) shows that they are quite similar at 37.7% and 38.2% respectively. Delac et al. (2006) uses a larger dataset with 255 individuals,

compared to ten in this thesis, with varying illumination conditions, which could explain the difference.

8.2.2 Validity

The threat of *Interaction of Setting and Treatment* was mitigated by not having a skewed dataset. In order to avoid a skewed dataset, the same number of pictures per individual were randomly selected to build the limited version of the dataset. The pictures were not manually checked, and therefore the pictures may contain obstructions, such as hands or hair. There may also very well be a chance that all pictures are perfect, so that the results are a misrepresentation of how the techniques would perform. This was taken into consideration but not mitigated by the limited version of the LFW dataset used in this thesis because it would require a dataset that would reflect real-world application usage, which could be difficult to define.

The threat of *Instrumentation* as identified in chapter 4.5 - Validity Threats has been mitigated somewhat by using two different implementations of Eigenface and Fisherface. If only the OpenCV implementations would have been used, the results would have been very different and be a misrepresentation of the techniques. Some control is lost by using libraries however, but this is something that must be accepted for comparative studies to exist. Two CNN implementations were planned but only one was implemented, which results in that it was not mitigated for the CNN technique. There is however a chance that the settings chosen for the techniques may unintentionally be suboptimal due to the lack of experience from the authors.

The threat to *Low Statistical Power* was mitigated by doing ten seeded runs per fold in the cross-validation, resulting in 100 measures per seed. Giving a total of 1000 measures for ten folds per technique. Given the magnitude of the measures, the decision was made to take out the averages of each fold, resulting in ten samples per technique. This was later added to the rest of the sample size per technique giving a total sample size of 70 for the ANOVA tests which according to the central limit theorem is sufficient to get a good sample size.

Seen from the performance metrics FaceNet is significantly better in every category when comparing to the other techniques. The time measurements do however not include the publicly available pre-trained model from the creator of FaceNet. The pre-trained model has trained on about three million images, which is something that the technique requires to function properly. Eigenface and Fisherface cannot handle training on three million images, since the size of the Eigenface and Fisherface models would be huge, at around a couple of terabytes, and would take about two weeks to build. This is from an experiment done by the authors where an Eigenface model was built using 1000 images, which created a model with a size of about two gigabytes.

The alternative to using a pre-trained model would have been to train a new model using the same dataset as the other techniques was trained upon, which would be an unfair comparison for the CNN technique since it requires loads of training to function properly. Both alternatives were bad alternatives, but since the pre-trained model is publicly available it seemed like that comparison would be more interesting than to train the CNN technique incorrectly.

8.2.3 Ethics

Wohlin et al. (2012) mentions four key principles in ethics, *informed consent*, *scientific value*, *confidentiality* and *beneficence*.

Informed consent is the first principle which is that any human subjects involved in the study should have access to enough information about the thesis to know if they want to participate or not, and that they can withdraw at any time. This does not apply to this thesis since no human participants were included but this would have been applicable if a new dataset was built using individuals who are not already in a publicly available dataset.

The second principle says that the study should have enough *scientific value* to motivate subjects to expose themselves to the risks involved with the study. A lot of studies have been conducted in the area of facial recognition in order to create new techniques but there is a lack of comparison between the existing techniques. This thesis adds to the research in the area by comparing different facial recognition techniques that already exist.

Confidentiality is the third principle which is that all sensitive data should be handled correctly. This thesis does not make use of any confidential data since everything used is open source or publicly available, which makes the principle inapplicable.

Beneficence is the fourth principle which says that the benefits should outweigh the risks and harms associated with the study. The risks of making a comparative study in facial recognition is that it could be used for malicious intent, such as illegal monitoring of people, but it could also be used for positive purposes, like helping law enforcement to identify thieves. The techniques used in this thesis already exist and are publicly available which results in the risk of usage for malicious intent whether this thesis was conducted or not.

Previous works have been used in this thesis and an ethical aspect is to give credit to the creators of the previous works. Credit has been given to the creators throughout this thesis and everything else is created by the authors of this thesis.

8.3 Future Work

The dataset used in the experiments is a limited version of the LFW dataset. This limited version was built using the ten individuals who has the most images. The dataset used is therefore limited when it comes to both gender and ethnicities. The next step would be to build a balanced dataset, which includes people of all ethnicities.

The idea from the beginning was to have more than one CNN implementation, but because of limited time and resources, the second CNN implementation OpenFace was not carried out. This is something that will need to be explored further to compare FaceNet to other CNN implementations. This should be done in order to mitigate the validity threat Instrumentation which was discussed in chapter 4.5 - Validity Threats.

This comparative thesis only compares three different techniques. Three techniques are quite a small number for a real comparative study. There are many techniques used in the field of facial recognition, and there may be a technique which outperforms FaceNet in every metric.

Even though it is probably not feasible, it could be interesting to see if the results would be the same if Eigenface and Fisherface was trained on the dataset with three million images. This

would be interesting since it would be the fairest comparison between the techniques, rather than using a pre-trained model, or training the CNN in a way that CNN was not built for.

Bibliography

Amos, B., Ludwiczuk, B. & Satyanarayanan, M. (2016). *OpenFace: A general-purpose face recognition library with mobile applications*.

Behmann, J., Hendriksen, K., Müller, U., Büscher, W. & Plümer, L. (2016). Support Vector machine and duration-aware conditional random field for identification of spatio-temporal activity patterns by combined indoor positioning and heart rate sensors. *GeoInformatica*, 20(4), 693–714.

Belhumeur, P. N., Hespanha, J. P. & Kriegman, D. J. (1997). Eigenfaces vs. Fisherfaces: recognition using class specific linear projection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7), 711–720.

Berndtsson, M., Hansson, J., Olsson, B. & Lundell, B. (2008). *Thesis Projects: A Guide for Students in Computer Science and Information Systems*. London: Springer London.

Delac, K., Grgic, M. & Grgic, S. (2006). Independent Comparative Study of PCA, ICA, and LDA on the FERET Data Set. *Int J Imaging Syst Technol*, 15, 252–260.

Dougherty, G. (2012). *Pattern Recognition An introduction*. Heidelberg Dordrecht London: Springer New York.

Garcia, V., Debreuve, E. & Barlaud, M. (2008). Fast k nearest neighbor search using GPU. *In*: *2008 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*. IEEE, 1–6.

Gogul, I. & Kumar, V. S. (2017). Flower species recognition system using convolution neural networks and transfer learning. *In*: *2017 4th International Conference on Signal Processing, Communication and Networking, ICSCN 2017*.

Gunawan, T., Gani, M., Kartiwi, M., Hamdan, M., Gani, H., Diyana, F. & Rahman, A. (2017). Development of Face Recognition on Raspberry Pi for Security Enhancement of Smart Home System. *Indonesian Journal of Electrical Engineering and Informatics (IJEEI)*, 5(4), 317–325.

Hsu, C.-W., Chang, C.-C. & Lin, C.-J. (2008). *A Practical Guide to Support Vector Classification*. BJU international.

Huang, G. B., Ramesh, M., Berg, T. & Learned-Miller, E. (2007). *Labeled Faces in the Wild: A Database for Studying Face Recognition in Unconstrained Environments*.

LeCun, Y., Bengio, Y. & Hinton, G. (2015). Deep Learning. *Nature*, 521(7553), 436–444.

Liu, Q., Huang, R., Lu, H. & Ma, S. (2002). Face recognition using kernel based fisher discriminant analysis. *In*: *Proceedings - 5th IEEE International Conference on Automatic Face Gesture Recognition, FGR 2002*.

McLachlan, G. J. (2012). Discriminant analysis. *Wiley Interdisciplinary Reviews: Computational Statistics*, 4(5), 421–431.

Mehta, P. & Tomar, P. (2016). An Efficient Attendance Management Sytem based on Face Recognition using Matlab and Raspberry Pi 2. *International Journal of Engineering Technology Science and Research*.

Parkhi, O. M., Vedaldi, A., Zisserman, A. & others (2015). Deep face recognition. *In*: *bmvc*. 6.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M.,

Müller, A., Nothman, J., Louppe, G., Prettenhofer, P., Weiss, R., Dubourg, V., Vanderplas, J., Cournapeau, D., Brucher, M. & Perrot, M. (2011). *Scikit-learn: Machine Learning in Python*. Journal of Machine Learning Research.

Rodavia, M. R. D., Bernaldez, O. & Ballita, M. (2017). Web and mobile based facial recognition security system using Eigenfaces algorithm. *In*: *Proceedings of 2016 IEEE International Conference on Teaching, Assessment and Learning for Engineering, TALE 2016*.

Russel, S. & Norvig, P. (2010). *Artificial intelligence—a modern approach 3rd Edition*. The Knowledge Engineering Review.

Sadhya, D., Gautam, A. & Singh, S. K. (2017). Performance comparison of some face recognition algorithms on multi-covariate facial databases. *In*: *2017 4th International Conference on Image Information Processing, ICIIP 2017*.

Sanmoy, P., Acharya, S. & Bhuva, K. (2018). *A Review on Facial Recognition Algorithm; Their Application in Retail Stores*. Academy of Marketing Studies Journal.

Santosh Dadi, H. & Mohan, P. G. K. (2015). *Performance Evaluation of Eigen faces and Fisher faces with different pre-processed Data sets Denoising of Speckle noise in SAR Images View project image processing techniques View project*. International Journal of Advanced Research in Computer Engineering & Technology (IJARCET).

Schroff, F., Kalenichenko, D. & Philbin, J. (2015). FaceNet: A Unified Embedding for Face Recognition and Clustering. *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Shakhnarovich, G. & Moghaddam, B. (2004). Face Recognition in Subspaces. *In*: *Handbook of Face Recognition*. New York: Springer-Verlag, 141–168.

Shen, Y., Yang, M., Wei, B., Chou, C. T. & Hu, W. (2017). Learn to Recognise: Exploring Priors of Sparse Face Recognition on Smartphones. *IEEE Transactions on Mobile Computing*.

Sirovich, L. & Kirby, M. (1987). Low-dimensional procedure for the characterization of human faces. *Journal of the Optical Society of America A*.

Svozil, D., Kvasnicka, V. & Pospichal, J. (1997). Introduction to multi-layer feed-forward neural networks. *Chemometrics and Intelligent Laboratory Systems*, 39(1), 43–62.

Turk, M. & Pentland, A. (1991). Eigenfaces for recognition. *Journal of Cognitive Neuroscience*.

Wang, F. Y., Zhang, J. J., Zheng, X., Wang, X., Yuan, Y., Dai, X., Zhang, J. & Yang, L. (2016). Where does AlphaGo go: From church-turing thesis to AlphaGo thesis and beyond. *IEEE/CAA Journal of Automatica Sinica*.

Wiley, J. F. & Pace, L. A. (2015). *Beginning R*. Berkeley, CA: Apress.

Wohlin, C., Runeson, P., Höst, M., Ohlsson, M. C., Regnell, B. & Wesslén, A. (2012). *Experimentation in Software Engineering*. Revised Ve. New York: Springer.

Yao, X. (1999). Evolving artificial neural networks. *Proceedings of the IEEE*, 87(9), 1423–1447.

Zhang, Y., Lu, S., Zhou, X., Yang, M., Wu, L., Liu, B., Phillips, P. & Wang, S. (2016). Comparison of machine learning methods for stationary wavelet entropy-based multiple sclerosis detection: decision tree, K nearest neighbors, and support vector machine. *SIMULATION*, 92(9), 861–871.

Zhang, Y., Wang, S. & Dong, Z. (2014). *Classification of Alzheimer Disease Based on*

Structural Magnetic Resonance Imaging by Kernel Support Vector Machine Decision Tree. Progress In Electromagnetics Research.

Appendix A - Performance per Individual

Table 1 – Performance results for each technique on an individual basis

		Average Performance (Percent in decimal form)				
Technique	Individual	Accuracy	Precision	F-score	Fallout	Recall
CV_Eigen	Ariel_Sharon	0.81	0.13	0.13	0.11	0.15
	Colin_Powell	0.82	0.10	0.09	0.10	0.10
	Donald_Rumsfeld	0.84	0.15	0.11	0.08	0.10
	George_W_Bush	0.83	0.17	0.15	0.09	0.15
	Gerhard_Schroeder	0.83	0.13	0.11	0.09	0.11
	Hugo_Chavez	0.82	0.09	0.10	0.11	0.13
	Jean_Chretien	0.84	0.19	0.18	0.09	0.20
	John_Ashcroft	0.82	0.10	0.10	0.10	0.12
	Junichiro_Koizumi	0.88	0.41	0.38	0.06	0.38
	Tony_Blair	0.81	0.09	0.09	0.11	0.11
CV_Fisher	Ariel_Sharon	0.82	0.18	0.16	0.10	0.16
	Colin_Powell	0.82	0.12	0.11	0.10	0.11
	Donald_Rumsfeld	0.85	0.19	0.17	0.08	0.16
	George_W_Bush	0.84	0.19	0.18	0.09	0.18
	Gerhard_Schroeder	0.86	0.31	0.24	0.07	0.23
	Hugo_Chavez	0.82	0.10	0.11	0.10	0.12
	Jean_Chretien	0.85	0.26	0.24	0.08	0.24
	John_Ashcroft	0.84	0.12	0.13	0.09	0.15
	Junichiro_Koizumi	0.88	0.38	0.43	0.08	0.51
	Tony_Blair	0.81	0.08	0.08	0.11	0.08
FaceNet	Ariel_Sharon	1.00	1.00	1.00	0.00	1.00
	Colin_Powell	1.00	1.00	1.00	0.00	1.00
	Donald_Rumsfeld	1.00	1.00	1.00	0.00	1.00
	George_W_Bush	1.00	1.00	1.00	0.00	1.00
	Gerhard_Schroeder	1.00	1.00	1.00	0.00	1.00
	Hugo_Chavez	1.00	1.00	1.00	0.00	1.00
	Jean_Chretien	1.00	1.00	1.00	0.00	1.00
	John_Ashcroft	1.00	1.00	1.00	0.00	1.00
	Junichiro_Koizumi	1.00	1.00	1.00	0.00	1.00
	Tony_Blair	1.00	1.00	1.00	0.00	1.00
S_Eigen_K	Ariel_Sharon	0.91	0.52	0.27	0.01	0.19
	Colin_Powell	0.89	0.49	0.46	0.07	0.48
	Donald_Rumsfeld	0.92	0.63	0.39	0.01	0.31
	George_W_Bush	0.91	0.57	0.40	0.03	0.33
	Gerhard_Schroeder	0.90	0.11	0.06	0.00	0.05
	Hugo_Chavez	0.88	0.42	0.39	0.07	0.39
	Jean_Chretien	0.89	0.41	0.27	0.03	0.23
	John_Ashcroft	0.81	0.31	0.43	0.18	0.72
	Junichiro_Koizumi	0.72	0.27	0.42	0.30	0.96
	Tony_Blair	0.90	0.08	0.03	0.00	0.02

S_Eigen_S	Ariel_Sharon	0.94	0.76	0.69	0.03	0.67
	Colin_Powell	0.92	0.57	0.57	0.05	0.59
	Donald_Rumsfeld	0.92	0.65	0.63	0.05	0.68
	George_W_Bush	0.94	0.77	0.70	0.03	0.68
	Gerhard_Schroeder	0.89	0.47	0.47	0.07	0.53
	Hugo_Chavez	0.95	0.79	0.68	0.02	0.64
	Jean_Chretien	0.93	0.65	0.66	0.04	0.72
	John_Ashcroft	0.94	0.73	0.69	0.04	0.71
	Junichiro_Koizumi	0.98	0.91	0.89	0.01	0.89
	Tony_Blair	0.91	0.59	0.50	0.04	0.47
S_Fisher_K	Ariel_Sharon	0.95	0.74	0.72	0.03	0.73
	Colin_Powell	0.94	0.70	0.71	0.04	0.75
	Donald_Rumsfeld	0.93	0.66	0.67	0.04	0.70
	George_W_Bush	0.95	0.80	0.70	0.02	0.68
	Gerhard_Schroeder	0.90	0.48	0.48	0.06	0.53
	Hugo_Chavez	0.95	0.83	0.67	0.01	0.60
	Jean_Chretien	0.95	0.74	0.72	0.03	0.75
	John_Ashcroft	0.94	0.72	0.76	0.04	0.83
	Junichiro_Koizumi	0.97	0.85	0.84	0.02	0.85
	Tony_Blair	0.90	0.53	0.46	0.05	0.45
S_Fisher_S	Ariel_Sharon	0.94	0.72	0.68	0.03	0.68
	Colin_Powell	0.93	0.61	0.67	0.06	0.75
	Donald_Rumsfeld	0.94	0.70	0.70	0.04	0.73
	George_W_Bush	0.94	0.72	0.65	0.03	0.65
	Gerhard_Schroeder	0.90	0.51	0.45	0.06	0.48
	Hugo_Chavez	0.93	0.68	0.62	0.03	0.60
	Jean_Chretien	0.93	0.68	0.67	0.04	0.70
	John_Ashcroft	0.95	0.70	0.69	0.03	0.70
	Junichiro_Koizumi	0.96	0.81	0.81	0.02	0.83
	Tony_Blair	0.91	0.56	0.47	0.05	0.48
Total Result		**0.91**	**0.54**	**0.51**	**0.05**	**0.53**