Master's thesis

Master of Science in Computer Engineering

Real-time face recognition using one-shot learning
A deep learning and machine learning project

Alex Darborg

Mittuniversitetet
MID SWEDEN UNIVERSITY

MID SWEDEN UNIVERSITY
Department of Information and Communication Systems

Examiner Tingting Zhang, tingting.zhang@miun.se
Supervisor: Johannes Lindén, johannes.linden@miun.se
Author: Alex Darborg, alda1502@student.miun.se
Degree programme: Master of Science in Computer Engineering, 120 credits
Main field of study: Computer Engineering
Semester, year: VT, 2020

Abstract

Face recognition is often described as the process of identifying and verifying people in a photograph by their face. Researchers have recently given this field increased attention, continuously improving the underlying models. The objective of this study is to implement a real-time face recognition system using one-shot learning. "One shot" means learning from one or few training samples. This paper evaluates different methods to solve this problem. Convolutional neural networks are known to require large datasets to reach an acceptable accuracy. This project proposes a method to solve this problem by reducing the number of training instances to one and still achieving an accuracy close to 100%, utilizing the concept of transfer learning.

Keywords: *Face recognition, one-shot learning, machine learning, deep learning, face expression, Inception-Resnet-v1, Squeezenet, web service*

Acknowledgements

I would like to thank Knightec for giving me the opportunity to conduct this project. A special thank you goes out to André Dankert for providing me with frequent feedback and guidance throughout the project, always happy to share his knowledge and experience.

I would also like to thank my supervisor, Johannes Linden at Mid Sweden University for his helpful advice throughout the project.

Table of Contents

Terminology

Abbreviations	Description
CNN	Convolutional Neural Network
Mtcnn	Multi-Task Cascaded Convolutional Networks
SVM	Support Vector Machine
k-NN	k-Nearest Neighbors
GPU	Graphics Processing Unit
CPU	Central Processing Unit
TP	True Positive
TN	True Negative
FP	False Positive
FN	False Negative
FPS	Frame Per Second
PCA	Principal Component Analysis

1 Introduction

This thesis is divided in seven parts and will present a how to implement real-time face recognition using one-shot learning. This chapter will go through the background and problem motivation for the thesis, followed by its goal and scope.

1.1 Background and problem motivation

The field of face recognition has been a topic of study since the 1960s. It has stayed relevant both due to the practical importance of the topic and the theoretical interest from cognitive scientists. Face recognition aims to verify or identify the identity of an individual using their face, either from a single image or from a video stream. Face recognition systems are used in many contexts, such as within security and healthcare where it is used to accurately track patient medication consumption and support pain management procedures.

Researchers have recently given this field an increased attention, conducting a multitude of studies and continuously improving the models that already exists. Convolutional Neural Networks (CNNs) are commonly used in computer vision and significantly improves the state of art in many applications. One of the most important ingredients is the availability of large quantities of training data.

Building a face recognizer can be challenging, especially when the dataset is limited, as oftentimes is the scenario in real world applications. One of the major challenges when the dataset is limited is that an individual's face may look different if various lightning, but also that different persons may have similar looking faces. Suppose a mobile ID unlocking recognizer should be developed. It would not be feasible to require that the person should upload millions of images for building the face recognizer system. In this scenario one-shot learning, where the algorithm learns by using only one or few training samples, would be a suitable technique.

A face recognizer comprises two main components where the first one is a face detection algorithm. The task of face detection is to find a face in a single image or in a video stream. The second step is to verify or identify the identity. There are several techniques in order to accomplish this. [1][2][3]

In this paper a face recognizer will be implemented using one-shot learning. Several suitable deep learning algorithms will be presented and evaluated, and in the final system a face recognition system will be implemented on a Jetson Nano together with a web service.

1.2 Overall aim

The goal of this project is to implement a real-time face recognition system using one-shot learning. This as it proceeds from the assumption that there is only one image available to learn from. Algorithms should be evaluated based on accuracy and time required to train.

The system should be easy to interact with and keep up to date. Therefore, a web service will also be implemented. This facilitates when a user wants to add more people to the system, such as when a company hires new employees. Since it is a real-time face recognition system it is important to focus on the speed of the system as well. The goal of all algorithms is to find a person in the video frame and classify it in real-time.

It should be possible to apply the system in various contexts. One example could be to install the system in the entrance of a company, where it should be able to recognize employees entering the doorway. Other contexts where the system could be used include security applications or autonomous cars where the system recognizes the person behind the steering wheel and adapts the driving settings accordingly.

1.3 Concrete and verifiable goals

The goal of this study is to evaluate and select different algorithms to solve the following requirements:

- Research and find different approaches in order to solve the one-shot learning problem
- Implement a face recognition system with only one sample per class

- Implement a suitable face detection algorithm
- Implement algorithms that can classify the age, gender, face landmarks and facial expressions of a person
- Implement a webserver with all algorithms included
- Possibility to add a new user to the system and re-train the network
- Implement the entire system on a Jetson Nano and optimize it.

This thesis will use various programming languages, such as Python for backend and HTML, CSS and Javascript for frontend.

1.4 Scope

This thesis has several limitations in scope. There are multiple solutions to implement a face recognition system. In this thesis the scope is to solve the one-shot learning problem, meaning that there will only be one sample per class to train the algorithm. Therefore, solutions that require large quantities of data, such as implementing a convolutional neural network from scratch, are outside the scope of this study.

1.5 Outline

In the next chapter of this thesis, the theoretical framework is presented, along with a short section of related works. Subsequently follows chapter 3, which describes the methods that are used to fulfil the study's goals. Thereafter follows chapter 4 which presents the final implementation of the system. Chapter 5 presents the results obtained by the algorithms and the web server that is implemented, and chapter 6 discusses the results. Lastly, chapter 7 describes the conclusions and discusses ethical aspects and presents suggestions of future research.

1.6 Detailed problem statement

This system is divided in three parts: face detection, face recognition and face classification. The face detection detects the face in the image, which is to be used for further analysis. The purpose of the face recognition algorithm is to produce feature maps which is a representation of the face. These maps contain important information about the user's face, such as the width between the eyes. Then, a supervised learning algorithm will be used to classify the feature maps and provide the user with a predicted name.

Evaluation is an important part of this project since multiple networks are to be compared. It will be important to measure how well they perform. Commonly known accuracy functions will be implemented, namely F1-Score, Precision and Recall.

2 Theory

This chapter describes the key concepts and the theory behind the thesis. This section will cover important methods that are used during the project followed by different algorithms that are used and evaluated. The basics of deep learning and machine learning will not be presented here since it is assumed to be familiar concepts for the reader.

The chapter begins with an introduction of the methods that has been used followed by a briefly description of convolutional neural networks (CNNs) and common architectures of them. Then the chapter continues with a description of face detection, face recognition and image classification.

2.1 One-shot learning

This is an object categorization problem and usually machine learning algorithms require training on hundreds or thousands of samples which results in a large dataset. However, one-shot learning aims to learn information from one or few samples. Learning from few samples remains a key challenge in machine learning. Therefore the task is challenging since there might be limited instances per class which means limited number of training instances and in some cases only one image for each of them. This challenge naturally exists in many real scenarios. [4][5]

2.2 Transfer learning

Transfer learning (TL) is an approach that focuses on storing knowledge gained while solving one problem and applying it to a different but related problem. For instance, suppose that a system should recognize human faces in an image. By utilizing models that already have been trained on millions of faces, often referred to as pre-trained models, related problems can be solved without the need of large quantities of data. [6]

2.3 Face Detection

Face detection is a technique to detect and find a face in an image or a video stream. There are different methods to accomplish the tasks. Either a convolutional neural network (CNN) can be implemented from scratch, which will require huge amounts of data, or a pre-trained model that already has been trained on millions of faces can be used. The later approach is an effective method when the dataset is small or if the problem that is going to be solved is related to the problem of the pre-trained model. All algorithms have different sizes. Therefore, some algorithms are more suitable for webpages and mobile phones, for instance.

Face detection can be challenging in unconstrained environments due to various poses, illuminations and occlusions. A photography that contains human faces can easily be spotted by humans. However, for computers this is a challenging task. But, studies have shown that deep learning can achieve good results in this area. [4][5]

2.4 Face Recognition

A face recognition system is capable of identifying or verifying a person from an image or video frame. There are different methods of face recognition systems but in general, they work by comparing images from a given image within a database. All images in the database are known by the model. Euclidean distance is a common function that can be used in order to classify a given image from the images in the database. A face recognition system can be used in various contexts. For instance, they are commonly used in security applications. They can also be used in autonomous cars where the purpose is to predict the person behind the steering wheel and change all car settings for them. [7]

2.5 Convolutional Neural Network

A common and a well-known method for image classification is called convolutional neural network (CNN). There are various CNNs and they all have the same underlying structure but they can have different architectures. The underlying structure is the same for all CNNs. All of the CNNs uses three types of layers, convolutional, pooling, fully connected layers and output layer. There are multiple kernels in the convolutional layers and they have the responsibilities to calculate feature representations of the image. The feature representation is called feature maps and the feature maps contain values of the face. Thereafter a pooling layer aims to reduce the resolution of the feature maps. Usually there are multiple convolutional and pooling layers. To summarize, the first convolutional layer spots edges in an image and this is called low-level features. Thereafter, the upcoming convolutional layer is responsible for extracting more abstract features. The convolutional and pooling layers are often followed by fully connected layers. The task of the fully connected layers is to create a communication line between neurons. The final layer of the CNN is known as the output layer. Softmax function is one example of a function that can be used in the last layer. Figure 2.1 illustrates a CNN architecture. [10][11]

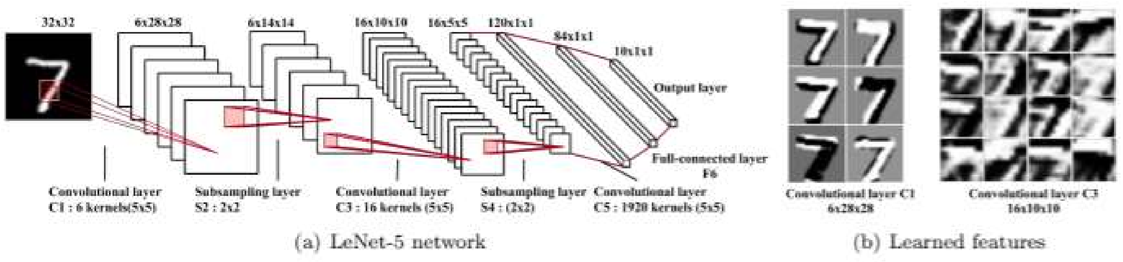

32x32
6x28x28
6x14x14
16x10x10
16x5x5
120x1x1
84x1x1
10x1x1
Output layer
Full-connected layer F6
Convolutional layer C1 : 6 kernels(5x5)
Subsampling layer S2 : 2x2
Convolutional layer C3 : 16 kernels (5x5)
Subsampling layer S4 : (2x2)
Convolutional layer C5 : 1920 kernels (5x5)
(a) LeNet-5 network
Convolutional layer C1 6x28x28
Convolutional layer C3 16x10x10
(b) Learned features

Figure 2.1: CNN architecture. [11]

2.5.1 Common CNN Architectures

Section 2.5 presents an introduction of the convolutional neural networks. As mentioned, there are various CNNs and they all have different architectures. The following section will present a list of common CNN architectures.

InceptionResnetv1 is a convolutional neural network and a hybrid Inception module. The computational cost in InceptionResnetv1 is similar to Inception-v3. The computational cost describes how much processing power the network needs. However, in InceptionResnetv1 a residual connection is added on the output of the convolutional operations. This means that the output from the Inception module is added as an input to the residual connection. For this to work the dimensions need to be the same for the input and output after the convolutional operations. Therefore, a 1x1 convolutional is added to the original convolutional. In InceptionResnetv1 the pooling operations from the Inception module is replaced with the residual connection. [12]

Inception-v3 is a convolutional neural network consisting of a 42-layer deep network. A pre-trained model that has been trained on millions of images, on the ImageNet database, is available for use. Compared to the previous Inception architectures this is more efficient since it has fewer parameters. This architecture becomes the first runner up for image classification in the ImageNet Large Scale Visual Recognition Competition (ILSVRC) 2015. There are two earlier versions namely, Inception-v1 and Inception-v2. It got a low error rate and performed good during the competition. ImageNet consists of 15 million labeled high-resolution images with 22.000 categories. [13]

Resnet18 is a convolutional neural network. This CNN has multiple variants of different sizes for instance, Resnet18, Resnet34, Resnet50, Resnet101 and Resnet152. The networks can be chosen after the size of the dataset. For instance, if the dataset is small a network with fewer layers may be more suitable. [14]

Alexnet is a convolutional neural network and is introduced in ImageNet Classification with Deep Convolutional Neural Networks. Alexnet was the first successfully convolutional neural network according to the ImageNet dataset. The network has eight layers where five layers are convolutional and three fully-connected layers. [15]

Squeezenet is a convolutional neural network that is described in the paper AlexNet-level accuracy with 50% fewer parameters and < 0.5MB model size. This network has 18 layers. Squeezenet is a network that is suitable for small datasets. For instance, a dataset that contains two classes, ants and bees. [16]

Densenet121 is a convolutional neural network and is described in the paper Density Connected Convolutional Networks. Densenet121 is a 121 layered deep convolutional network. There are different networks with this architecture to choose from. [17]

2.6 Multi-Task Cascade Convolutional Neural Network

Multi-Task Cascade Convolutional Neural Network is a state-of-art face detection algorithm that is called Mtcnn. The network is able to simultaneously propose bounding boxes, five-point facial landmarks and detection probabilities. This model is a deep cascaded multi-task framework which exploits the inherent correlation between them in order to boost up the performance. The network breaks down the task into three stages namely, P-Net, R-Net and O-Net. P-Net produces candidate window by a shallow convolutional network. R-Net has the objective to reject as many non-face windows as possible. O-Net uses a more complex network to further refine the output of R-Net. In particular, the model has a cascaded structure with three stages of carefully designed deep convolutional networks that classify face and landmark locations. The network achieved superior accuracy in the challenging FDDB and WIDER FACE benchmark for face detection. It keeps real time performance which make it possible to use the network in a real time system. Figure 2.2 shows the structure of the network. [9]

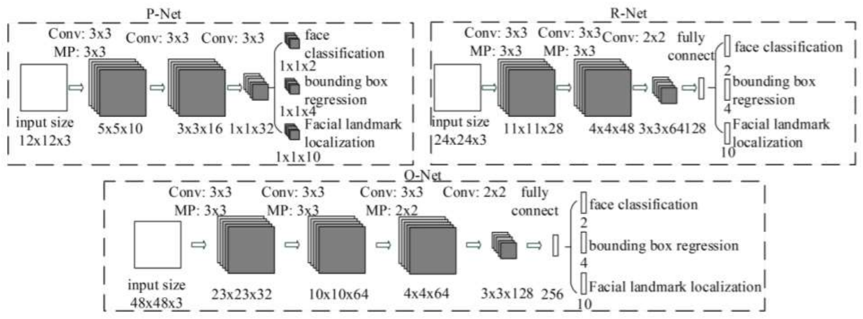


Figure 2.2: The architecture of Mtcnn. [9]

2.7 FaceNet

FaceNet provides a unified embedding for face recognition, verification and clustering tasks. Another commonly used term to describe embedding is feature map which corresponds to a vector representation of information on the face. It maps each face image into a Euclidean space such that the distance in that space corresponds to face similarities. More specifically, an image of person X will be placed closer to all the other images of person X and person Y will have a large distance to person X. The main difference between Facenet and other networks is that it learns the mapping from the images and creates embeddings. To summarize, the created embeddings can be used directly for face recognition, verification and clustering using for instance, Support Vector Machine (SVM) or K-Nearest Neighbors (K-NN). Facenet uses Triplet loss function to learn, which is further described in section 2.7.1. [18]

2.7.1 Triplet Loss

The functionality of triplet loss is that it minimizes the distance between an anchor and a positive where the positive represents the same person in the dataset. Thus, it maximizes the distance between the anchor and a negative where the negative represents a different identity. Thereby, an image of an anchor (person A) should be closer to the positive images (all images of person A) and have larger distances to any negative images (all other images). Figure 2.3 shows an illustration of the Triplet Loss. [18]

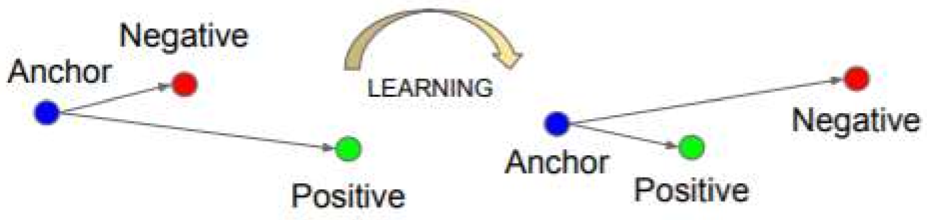


Figure 2.3: Triplet loss. [18]

2.8 Siamese Neural Network

Siamese Neural Network is an artificial neural network that uses the same weights while working in tandem on two different input vectors to compute comparable output vectors. Therefore it is also called a twin neural network. The architecture of the network uses two input images which are forwarded to the neural networks. The model then produces two feature maps, in other words two vectors, which have the representations of the faces. Thereafter, these vectors are computed with a distance function in order to calculate the similarities between the two feature maps. While the network is training the idea is to minimize the distance function for similar classes and maximize the distance between un-correlated classes. [19]

2.9 OpenCV Haar Cascades

Different from a convolutional neural network, Haar Cascade is a classifier but somewhat related to convolutional neural networks. A Haar Feature is similar to a kernel that CNN uses. Thus, in a CNN the values of the kernel are determined by the training while Haar-Feature is manually determined. Figure 2.4 illustrates Haar-Features.

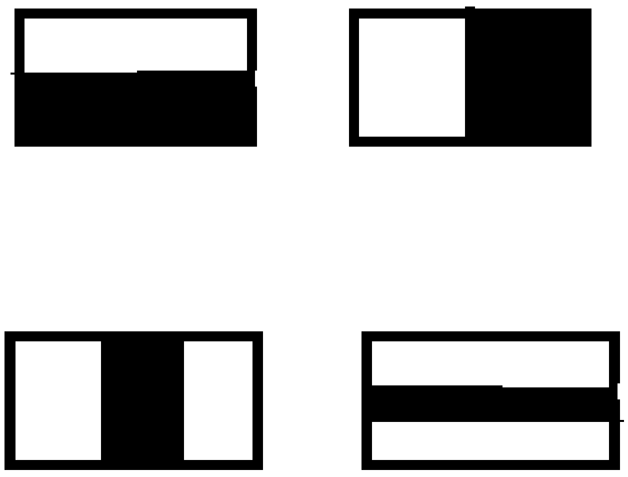


Figure 2.4: Line and edge features.

This approach makes it effective in detecting face since Haar-Feature are good at detecting edges and lines. [20]

2.10 OpenCV DNN

This is a module in the Opencv which is described in section 3.5. It is possible to use a pre-trained model from Tensorflow which also is described in section 3.5. However, this is a deep neural network which can be used for inference using a pre-trained model. OpenCV DNN has support for several frameworks such as Caffe, Tensorflow, Darknet and PyTorch. It is possible to create various applications with this module including face detection and object detection. [21]

2.11 Support Vector Machine

Support Vector Machine, commonly denoted SVM, is a supervised learning model that learns from data. This model can be used for classification and regression. SVM uses a linear separating hyperplane which separates the data into two classes, see figure 2.5. [21]

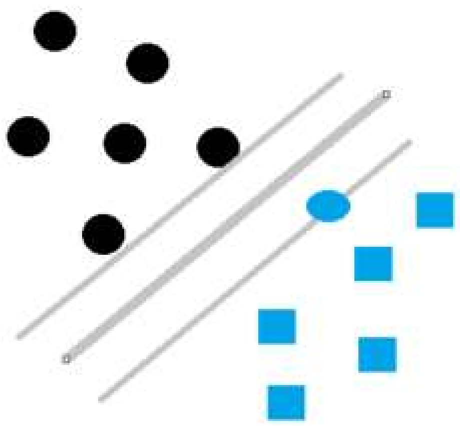


Figure 2.5: Support Vector Machine, two classes that are linear separable.

The figure above illustrates two classes, the grey dots belongs to one class and the blue squares belongs to another class. There is a hyperplane in the middle that separates these two classes. It is important to find an optimal hyperplane that can do this separation. Often SVM uses kernels and there are different kernels that can be applied depending on the dataset and the context of the application. Generally in machine learning a kernel refers to a kernel trick. It is a method which uses a linear classifier to solve non-linear problems. Figure 2.6 illustrates this and Table 2.1 shows two common kernels.

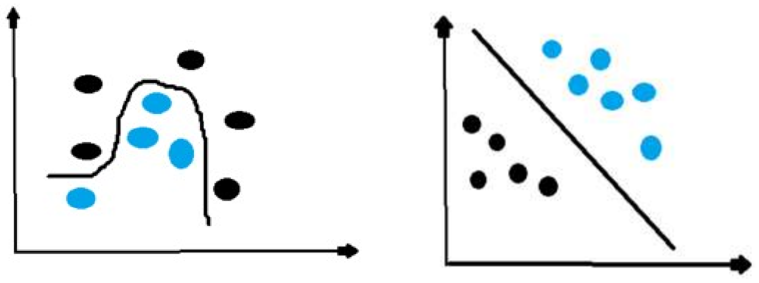


Figure 2.6: (From left) Non-linear separable data and separable data.

Kernel functions	Formula
Linear	$k(x_i, x_j) = x_i, x_j$
Polynomial	$k(x_i, x_j) = k(\gamma x_i, x_j + r)^d, \gamma > 0$

Table 2.1: Common kernel functions.

2.12 Gaussian Naïve Bayes

Gaussian Naïve Bayes (GNB) is a supervised learning algorithm which produces a linear classifier. GNB is divided into three parts. First, when it handles real-time data with continuous distribution it assumes that the data is generated with a normal distribution. Second, Multinomial Naïve Bayes can be applied in multinomial distribution which means that the features are represented as frequencies. Third, if the features are independent or Boolean the feature are generated through a Bernoulli process therefore a Bernoulli Naïve Bayes classifier can be applied. See equation 2.12 for the linear classifier. [22]

$$P(A_i|B) = \frac{P(A_i)P(B|A_i)}{\sum_{j=1}^{n} P(A_j)P(B|A_i)} \tag{2.12.1}$$

2.13 K-Nearest Neighbors

K-Nearest Neighbors is a supervised learning algorithm which is used for regression and classification problems. In the scenario of a classification problem the output is a class membership. Thus, an object is classified by a vote of its neighbors. For instance, if k = 1 the object is assigned to the class that are nearest the neighbor. However, if it is a regression problem, the output is the property value for the object. The value for the object is the average values of k nearest neighbors. [19] Equation 2.13 shows the formula for Euclidean distance. The method for calculating the distance between the points is denoted as,

$$\begin{aligned} & d(p,q) = d(q,p) \\ = \quad & \sqrt{(q_1 - p_1)^2 + (q_2 - p_2) + \cdots + (q_n - p_n)^2} \\ = \quad & \sqrt{\sum_{i=1}^{n} (q_i - p_i)^2} \end{aligned} \tag{2.13.1}$$

2.14 Javascript algorithms

Age and Gender Recognition Model is a model that predicts age and gender in an image or a video stream. It employs a feature extraction layer, an age regression layer and a gender classifier. The size of this model is 420kb and it is similar to an Xception architecture. [21]

Face Expression Recognition Model is a lightweight model it is fast and provides good accuracy. The size of this model is 310kb and it employs depth wise separable convolutions and densely connected blocks. [21]

Face Recognition Model has an architecture similar to ResNet-34 which is a convolutional neural network. It is implemented to compute a face descriptor and the output is a feature vector with 128 values. [21]

68 Point Face Landmark Detection Models is a lightweight landmark detector. The detector is fast and accurate. The model has various sizes to choose from. For instance, the default model has a size of 350kb and the tiny model is only 89kb. [21]

Tiny Face Detector is a real time face detector and has good performance. It is a mobile and web friendly model. The model size is roughly 190 KB. [21]

2.15 Related works

In [25], Yandong Guo and lei Zhang presents the problem of one-shot learning in a face recognition context. Their goal is to build a large-scale face recognition that is able to recognize several different identities. The used approach is a novel super vision signal namely, underrepresented-classes promotion (UP) loss term. This technique aligns the norms of the weight vectors in order address the problem of unbalanced data. The new loss term (UP) is efficient since it promotes the un-represented classes in the learn model. This improves the performance of a face recognition system.

In their development of the one-shot learning phase they used a dataset consisting of 21.000 classes which they use in order to train a classifier. The technique is a multinomial logistic regression based on a 34-layer residual network. The results of this study shows 99% for underrepresented classes and 99.8% for normal classes. [25]

Another study on face recognition is presented by Zhao et al. In the paper they present a face recognition method based on Principal Component Analysis (PCA) and Linear Discriminant Analysis (LDA). The technique consists of two steps where they first use PCA in order to project the original vector of the face image to a face subspace. Then they use LDA which act as a linear classifier. The combination of these techniques improves the capability of classifying classes from a model that has been trained on a small dataset. [26]

In [27] they present a face recognition system using convolutional neural network and principal component analysis (CNN-PCA). Edy et al describe their system of a face recognition and they use a hybrid feature extraction method (CNN-PCA). The idea is to combine these two techniques to get a better feature extractor method which leads to a more accurate model. The idea of their system is to produce a system that is reliable and powerful to identify human faces in real-time. The results of their study shows that their method has an accurate data processing and high accuracy. They present that adding a CNN to PCA increases the accuracy. In 50 objects they get 98% accuracy instead of 96% when using PCA only. [27]

Another research is from [28] where they implement a face recognition system. They first start with pre-processing the images by performing noise removal and hole filling. After this they use viola jones algorithm in order to extract faces from the image. They compute the SURF features of the extracted face in order to use feature matching to recognize identities. M-estimator Sample Consensus (MSAC) is used in order to remove outliers. In this study they get an accuracy of 95.9% in Graz 01 dataset. [28]

3 Methodology

This chapter presents the methodologies used in this study. After research, multiple suitable algorithms were found. However, all of them are not evaluated. This chapter describes the algorithms that are evaluated and implemented. Chapter 3.1 presents the datamining methodologies. Thereafter follows the structure of the dataset. It then continues with Chapter 3.3 and its sub chapters presents the algorithms that are used and evaluated. Thereafter follows chapter 3.4, which describes the evaluation metrics. It describes how the project is evaluated and it also describes common methods in order to evaluate the models. Chapter 3.5 presents all libraries that are used during the project. Lastly, chapter 3.6 goes through the hardware that is used.

Agile methodologies are used during the project and small releases are presented throughout the project in order to establish the direction and the next releases. This project uses a sprint of a seven days interval. However, some sprints use a longer interval since unexpected delays occurred. The project uses a backlog that is updated on a daily basis and a scrum board is used as well. Before starting a new backlog, each ticket is estimated.

3.1 Datamining Method

This section presents the datamining methods which are used to overcome the challenges of producing a face recognition system with one-shot learning. Each paragraph represents the steps taken to process and solve the challenges.

Problem identification is a way to research and identify the problem with different techniques that currently are used. For instance, convolutional neural networks are powerful and can be very accurate. The goal is to implement a face recognition system with one-shot learning therefore building a convolutional neural network from scratch is not an alternative. However, existing pre-trained convolutional neural network models is a solution for this project. The final solution for this project is presented in chapter 4.

Literature study is a method to collect as much relevant information about the topic as possible, to understand and take previous learnings into account.

Understanding of the business is important since the final solution is given to the company and therefore the final system need to be formed by their guidelines.

Understanding the data means to analyze the data and understand it. This is important as this will be used as basis for the preparation of the data.

Preparing the data means to prepare it for the networks for instance, when then input image is forwarded to the face recognition algorithm it needs to be cropped and in a right format (png, jpg o jpeg).

Modelling all networks are performed on the data that was mentioned in the previous step.

Evaluation is in this study the process of comparing multiple networks and discarding some of them, due to their low accuracy, overfitting or too skewed results.

Deployment is the final step, when the networks have been evaluated correctly with good results.

3.2 Dataset structure

Several networks are evaluated during the project and they all have identical data structures. All images are cropped and resized before they are put in separate folders specifically created for each class. Figure 3.1 presents the structure of the dataset.

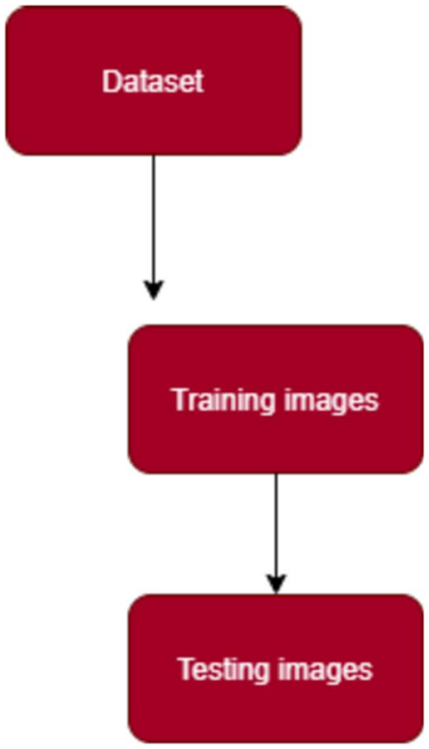


Figure 3.1: Folder structure.

3.3 Choice of algorithms

In order to achieve the objective of this study, three main methods are required, namely face detection, face recognizer and face classification. Section 3.3.1, 3.3.2, 3.3.3 presents this further.

3.3.1 Face detection

To evaluate which face detection algorithms work best, several face detection algorithms are implemented on a specific video clip, results are registered and compared. The video clip contains several different faces and lasts around one minute.

In order to find face detection algorithms to compare and evaluate, a study of various papers was conducted. Facenet, Mtcnn, Opencv_Dnn and Opencv_Haar all showed reasonable results. Thus, these four algorithms were chosen to be evaluated for the face detection.

3.3.2 Face recognition

Several face recognizer networks are evaluated, namely InceptionResnetv1, Resnet18, Alexnet, VGG11, Squeezenet, Densenet121 and Inceptionv3. These seven pieces are all deep learning networks which are evaluated under the same conditions. They are all evaluated using the same datasets which are presented in section 4.1. One reasoning why these networks are chosen to be evaluated is that previous papers present good results of them. Another reasoning why InceptionResnetv1 and Squeezenet are selected is because they are suitable for small datasets. [7][8][9][10][11]

3.3.3 Image classification

The last part is image classification and the reasoning why image classification is needed in this project is that the face recognizer networks that are mentioned in section 3.3.2 produce feature maps. Each feature map is mapped to a face and contains information about each identity. More precisely, person A has a feature map and person B has another feature map.

The purpose of using feature maps is to classify them, more precisely get the name of the person in the image. This can be achieved with a classification algorithm or to use Euclidean distances function.

The classification algorithms that are chosen are SVM, k-NN and Gaussian Naïve Bayes. Euclidean distance will also be evaluated. [16][17][18]

3.3.4 Web server

One of the goals in this paper is to create the functionality to upload new individuals to the system, which can be required if a new employee has been hired at the company. Therefore, a web server is created which gives the functionality to add a new user to the system when required. Several algorithms from each part for example, one algorithm from section 3.3.1, one algorithm from section 3.3.2 and one algorithm from section 3.3.3 are implemented and this is presented in section 4. Moreover, several other algorithms are implemented such as, face landmarks, gender, age and face expression. [20]

3.4 Evaluation

This project is divided into 3 parts, face detection, face recognition and face classification. The first part is evaluated based on research and commonly known challenges. Face recognition models and the face classification algorithms are evaluated with commonly known methods which is described in section 3.4.2.

3.4.1 Evaluation of Images

The task of image classification can be accomplished in various ways. One approach is to predict each sample in other words each image. There are multiple performance metrics that can be used during the evaluation of a classification task. The approaches that are used in this project is presented in section 3.4.2. When predicting a label, it can result in four outcomes, namely True Positive (TP), True Negative (TN), False Positive (FP) and False Negative (FN). Table 3.1 presents the outcomes.

Actual Class	Ant	Bee
Ant	True Positive (TP) Correct predicted as ant	False Negative (FN) Incorrectly predicted as bee
Bee	False Positive (FP) Incorrectly predicted as ant	True Negative (TN) Correctly predicted as bee

Table 3.1: Illustrates the different outcomes.

3.4.2 Evaluation metrics

It is interesting to measure and visualize the networks that are described in section 2.5.1. There are different methods in order to evaluate a model but common approaches are Recall, F1-Score, Precision. All of these methods are important to understand. Since the last example gives an accuracy but it does not give information about if the model predicted all samples wrong in one class and all correct for the remaining classes even if it gives an accuracy. However, F1-Score is a measure that can be interesting to look at. Below is a list of the different approaches that are included in this project.

F1 – score: [30] also known as **F-measure**, is a measure of a test's accuracy. This method considers precision p and recalls r of the test in order to calculate the score. Number of correct positive results divided by number of all positive results is denoted as p. Thus, r is denoted as, number of correct positive results divided by the number of all relevant samples.

Thereby, F1-score is the harmonic mean of the precision and recall that will be listed below. It reaches its best value at 1.

$$F1\ score = \frac{2TP}{2TP + FP + FN} \tag{3.4.1}$$

Precision: [30] describes the accuracy of classified classes. In order words it is the ratio between correct classified classes and the total number of predictions.

$$Precision = \frac{TP}{TP + FP} \tag{3.4.2}$$

Recall: [30] the ratio between correct classified samples of a class and the total number of instances of that class.

$$Recall = \frac{TP}{TP+F} \tag{3.4.3}$$

3.5 Libraries

During the project different libraries have been used in order to implement machine learning algorithms and the web server. Most of the algorithms are Python libraries that are commonly used in Machine learning. However, not only machine learning libraries are used. For instance, a library called Split-folders is used in order to split the data into a certain percentage. The libraries will be presented in the section below.

Tensorflow [30] is a Python open source artificial intelligence library. It is well documented and it allows developers to create large-scale neural network with multiple layers. It can be run on a CPU as well as GPU. This library is used to load the inception models.

Numpy [31] is a library for Python, adding support for large, multi-dimensional arrays and matrices. This library is used to make array operations.

Matplotlib [32] is a Python library for creating static, animated and interactive visualizations in Python. It is a good tool for machine learning. This library is used to sketch the accuracy from the machine learning algorithms.

PyTorch [33] is a deep learning library similar as Tensorflow. This library can be run on both CPU and GPU as well. The library is used to create the dataset in this project such as, cropping and resize the images.

OpenCV [34] is an open source computer vision and machine learning library. It is a huge library that contains more than 2500 optimized machine learning algorithms. It supports C++ and Python, for instance. This library has been widely used in this project for reading images, resizing images, convert images to RGB, saving images and is also used to make the live face recognition system work in real-time.

Jupyter Notebook [35] is a web-based interactive development environment for Jupyter Notebook, code and data. This environment is used to implement and test the software.

SKlearn [36] is an open source machine learning algorithm that supports functionalities such as multi-dimensional arrays. This library also has support for high-level mathematical functions in the arrays. It also supports machine learning algorithms.

Flask [37] is a micro web framework, flask is a Python class datatype. Thus, it is used to create instances of web application or web applications. It is a simple and a good tool that is well documented on the web. This library is used for creating the webpages.

Glob [38] is used to find all the pathnames matching a specified pattern according to the rules used by the Unix shell.

Albumentations [39] is a python open source library that makes it possible to boost the dataset. This library has been used in this project in order to boost the dataset by increasing the amount of data samples.

Pytube [40] is a lightweight dependency free Python library for downloading online videos. It is used to download an online video stream. The face recognition system uses this video stream in order to classify the persons in the video stream.

Split-folders [41] is a python library that split folders into training, validation and test. This library is used to split the folders and put images in the subfolders for training and testing. This library is used to create the datasets that have been tested in machine learning networks with different amounts of images per class.

3.6 Hardware

The final system is implemented on the Jetson Nano but during the project different algorithms have been trained and evaluated on different machines. Jetson Nano has a GPU and the final system is implemented in order to run on the GPU.

Term	Specification
Graphical Processing Unit	128-core NVIDIA Maxwell™
Central Processing Unit	Quad-core ARM® A57
Memory	4GB 64-bit LPDDR4
Connectivity	Gigabit Ethernet
OS Support	Linux for Tegra®

Table 3.2: Specifications for Jetson Nano

Term	Specification
Graphical Processing Unit	Intel® Xeon® 2.0GHz, 38.5MB cache
Central Processing Unit	Tesla T4 16GB
Memory	13 GB
Disk storage	69 GB HDD

Table 3.3: Hardware used for face recognition.

The entire system is implemented on a Jetson Nano. To optimize the system, GPU programming has been used. This is the process of programming in such way that the GPU takes care of the code which speeds up the time for all calculations compare to a CPU. Figure 3.1 illustrates the hardware.

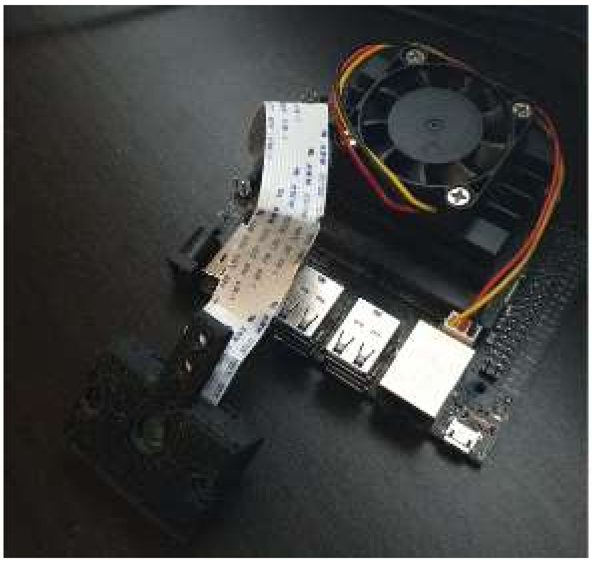


Figure 3.1: Jetson Nano with an external camera.

4 Implementation

This section presents the final results of the system and how they are implemented. All the networks and algorithms and how they are implemented are presented. This section starts with an overview of the entire system, see figure 4.1.

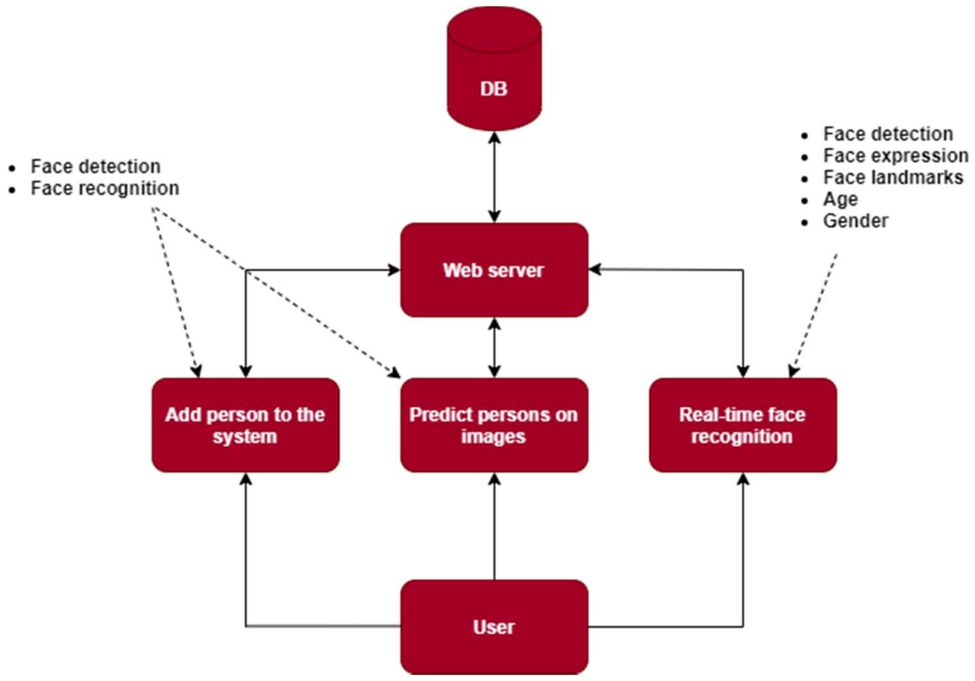


Figure 4.1: Overview of the face recognition system

Add person to the system: a person adds an image of its face to the system. The image will be processed.

Face Detection: when the images are added into the system the face detection algorithm will try to find a face in the image. If a face is found it will then be cropped and saved in a database or be used for a prediction. The face detection algorithm is presented in section 4.2.

Face recognizer: when a face is found and cropped this phase will be the following step. Face recognizer will output a feature map representation of the individual face and save it for uses in the future.

Predict image: during this step, the classification algorithm will predict all images (all feature maps) and output the name of the individual.

Live face recognition: Several algorithms are implemented for live-face recognition such as, face detection, face expression, age, gender and face landmarks.

4.1 Enhancing the dataset

The dataset is divided into 7 various datasets where each dataset will be evaluated for instance *one-shot 30 classes* means that the dataset consists of 30 classes where every class has one instance. There are several datasets and they are presented below:

- 1 shot 30 classes
- 2 shot 30 classes
- 5 shot 30 classes
- 1 shot 50 classes
- 2 shot 50 classes
- 5 shot 50 classes
- 50 shot 50 classes

All of the datasets above are evaluated using the same method, described in section 3.4.2. The main reason for evaluating the different datasets is to see how the size of the training data can affect the performance. All images are cropped to 160 x 160 pixels. The last evaluation test will first use augmentation in order to increase the size of the dataset. For instance, it will augment the one-shot 50 classes to 50-shot 50 classes in order to see if the accuracy will improve, and if so, how much. Figure 4.2 illustrates the augmentation phase.

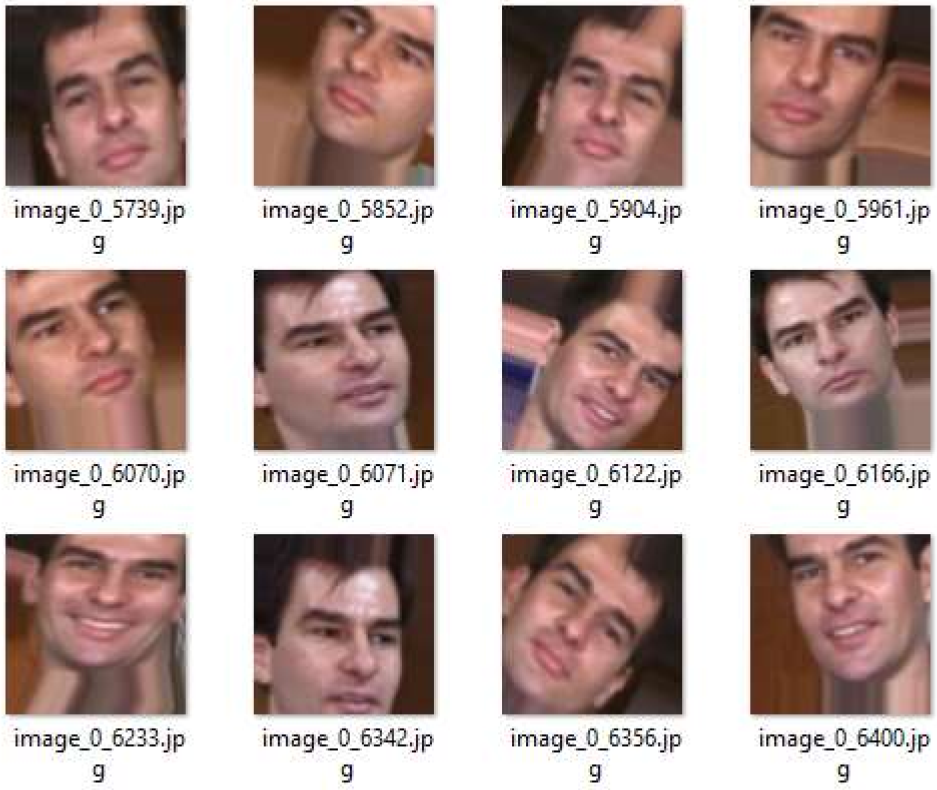
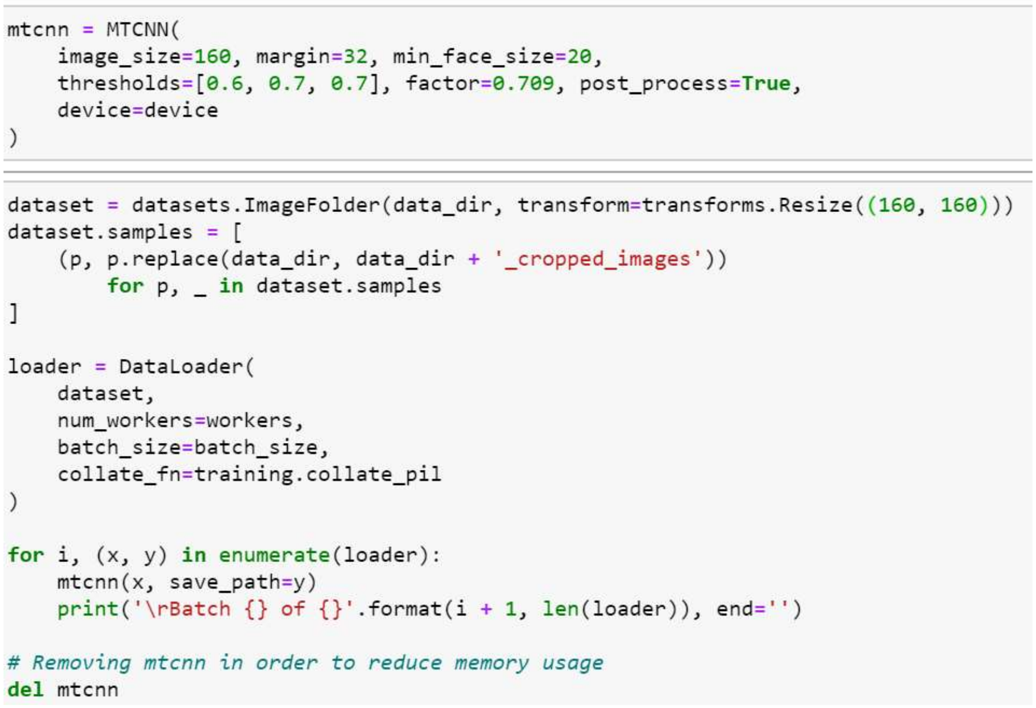


Figure 4.2: Enhancing the dataset with augmentation.

4.1.1 Cropping function

Before the dataset can be forwarded to the network the dataset needs to be pre-processed. This means that all images in the dataset need to be cropped and also contain a face since it is a face recognition system which will be implemented. It works by allowing the face detection algorithm to spot a face in an image. Thereafter, the face detection algorithm provides 4 points which gives information about the upper left, upper right, down left and down right corners of the face. These points are then used to crop the image, using a cropping function.

Figure 4.3 presents the cropping function. The function iterates through the folder, as it crops and saves the new pictures into a new folder. In this project the images are resized to 160 x 160 pixels.



```python
mtcnn = MTCNN(
    image_size=160, margin=32, min_face_size=20,
    thresholds=[0.6, 0.7, 0.7], factor=0.709, post_process=True,
    device=device
)
```

```python
dataset = datasets.ImageFolder(data_dir, transform=transforms.Resize((160, 160)))
dataset.samples = [
    (p, p.replace(data_dir, data_dir + '_cropped_images'))
        for p, _ in dataset.samples
]

loader = DataLoader(
    dataset,
    num_workers=workers,
    batch_size=batch_size,
    collate_fn=training.collate_pil
)

for i, (x, y) in enumerate(loader):
    mtcnn(x, save_path=y)
    print('\rBatch {} of {}'.format(i + 1, len(loader)), end='')

# Removing mtcnn in order to reduce memory usage
del mtcnn
```

Figure 4.3: Function that creates two folders (train,val) and crops the images.

4.2 Face detection - Mtcnn

Four face detection algorithms are evaluated in this project. All face detection algorithms have the responsibility to find a face in a single image or in a video stream. However, one of the four evaluated face detection algorithms which are presented in section 3.3.1 are used in the final system.

Mtcnn is selected as this algorithm performs well, is accurate and speedy. The entire project is implemented on a Jetson Nano which has a GPU. Mtcnn performs also well on the Jetson Nano which makes it possible to use the algorithm in real time. Figure 4.4 illustrates how the face detection algorithm spots a face before the image has been cropped. The execution time and accuracy are presented in section 5.3.

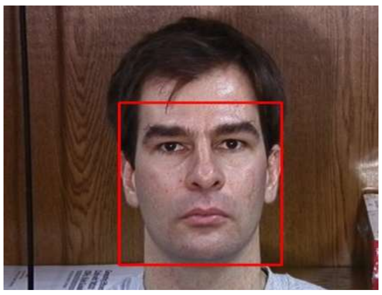
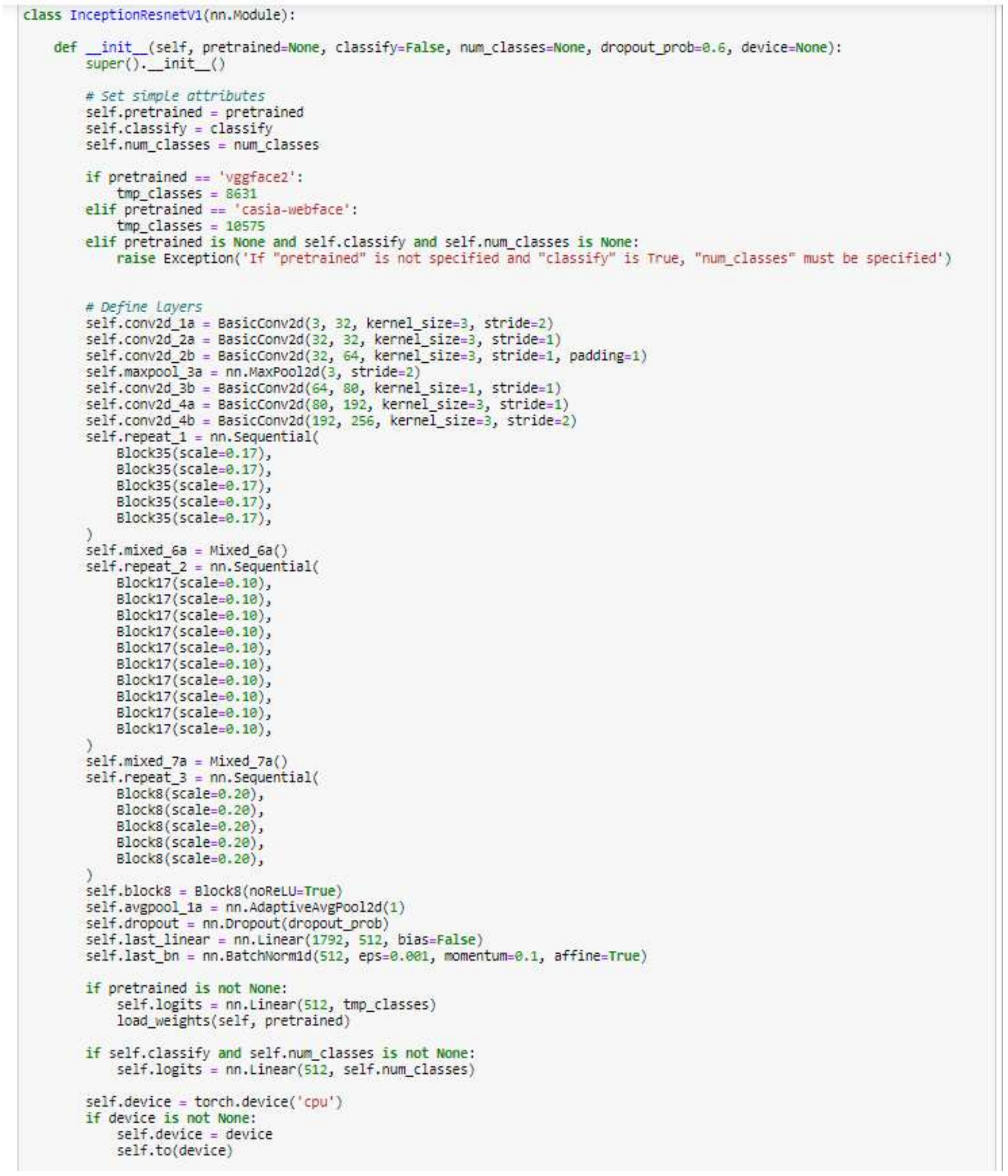


Figure 4.4: Mtcnn (Face detection).

4.2.1 Face recognizer – InceptionResnetv1

Several networks are evaluated in order to find an appropriate algorithm for face recognizer. Namely InceptionResnetv1, Resnet18, Alexnet, Squeezenet, VGG11, Densenet121 and Incpetionv3. The choice was InceptionResnetv1 which is described in chapter 6.1. However, a Tensorflow implementation is done and it has similar architecture as Facenet but with one adjustment. Instead of using triplet loss, which is used in Facenet, InceptionResnetv1 uses softmax. The pre-trained model has been trained on VGG-Face2 dataset. Thereafter on my own dataset. The idea of this face recognizer is not to use it as a classification, meaning to predict the outcomes. Instead, the face recognizer will produce feature maps. The training phase is done using the GPU. Figure 4.5 shows the architecture of InceptionResnetv1.



```
class InceptionResnetV1(nn.Module):

    def __init__(self, pretrained=None, classify=False, num_classes=None, dropout_prob=0.6, device=None):
        super().__init__()

        # Set simple attributes
        self.pretrained = pretrained
        self.classify = classify
        self.num_classes = num_classes

        if pretrained == 'vggface2':
            tmp_classes = 8631
        elif pretrained == 'casia-webface':
            tmp_classes = 10575
        elif pretrained is None and self.classify and self.num_classes is None:
            raise Exception('If "pretrained" is not specified and "classify" is True, "num_classes" must be specified')


        # Define layers
        self.conv2d_1a = BasicConv2d(3, 32, kernel_size=3, stride=2)
        self.conv2d_2a = BasicConv2d(32, 32, kernel_size=3, stride=1)
        self.conv2d_2b = BasicConv2d(32, 64, kernel_size=3, stride=1, padding=1)
        self.maxpool_3a = nn.MaxPool2d(3, stride=2)
        self.conv2d_3b = BasicConv2d(64, 80, kernel_size=1, stride=1)
        self.conv2d_4a = BasicConv2d(80, 192, kernel_size=3, stride=1)
        self.conv2d_4b = BasicConv2d(192, 256, kernel_size=3, stride=2)
        self.repeat_1 = nn.Sequential(
            Block35(scale=0.17),
            Block35(scale=0.17),
            Block35(scale=0.17),
            Block35(scale=0.17),
            Block35(scale=0.17),
        )
        self.mixed_6a = Mixed_6a()
        self.repeat_2 = nn.Sequential(
            Block17(scale=0.10),
            Block17(scale=0.10),
            Block17(scale=0.10),
            Block17(scale=0.10),
            Block17(scale=0.10),
            Block17(scale=0.10),
            Block17(scale=0.10),
            Block17(scale=0.10),
            Block17(scale=0.10),
            Block17(scale=0.10),
        )
        self.mixed_7a = Mixed_7a()
        self.repeat_3 = nn.Sequential(
            Block8(scale=0.20),
            Block8(scale=0.20),
            Block8(scale=0.20),
            Block8(scale=0.20),
            Block8(scale=0.20),
        )
        self.block8 = Block8(noReLU=True)
        self.avgpool_1a = nn.AdaptiveAvgPool2d(1)
        self.dropout = nn.Dropout(dropout_prob)
        self.last_linear = nn.Linear(1792, 512, bias=False)
        self.last_bn = nn.BatchNorm1d(512, eps=0.001, momentum=0.1, affine=True)

        if pretrained is not None:
            self.logits = nn.Linear(512, tmp_classes)
            load_weights(self, pretrained)

        if self.classify and self.num_classes is not None:
            self.logits = nn.Linear(512, self.num_classes)

        self.device = torch.device('cpu')
        if device is not None:
            self.device = device
            self.to(device)
```

Figure 4.5: (InceptionResnetv1) A piece of the architecture.

Two types of models can be used as can be seen in the top of the code, namely vggface2 or casia-webface. Depending on which model that is chosen it is either trained on vggface2 or casia-webface, and this is an option that can be chosen.

However, in this project vggface2 is selected. Thereafter some adjustments are needed, for instance the last layer of the network needs to be changed in order to put my own dataset to it. The input of the network is an image of size 160 x 160 pixels, and the output of this network is a feature map meaning a vector containing information about the face, more precisely a face representation. Thereafter the idea is to implement a classification algorithm that can classify these feature maps, it is fine to use Euclidean distance as well. But, in this project SVM is chosen to classify the feature maps which is described in section 4.2.2. Table 4.1. describes the parameters used in this project.

Parameters	Values
Architecture	IncpetionResnetv1
Batch size	10
Input size	160x160

Table 4.1: Parameters for InceptionResnetv1.

4.2.2 Image classification - SVM

This is the last part that is needed in order to classify or identify an identity from a single image or a video stream. There are several classification algorithms such as k-NN, SVM, Random Forest and a lot more. This project uses SVM. The classifier is implemented with the Python library SKlearn which is described in section 3.3. The parameters of the SVM is presented in table 4.2.

Parameters	Values
Classifier	SVM
Kernel	linear
gamma	auto

Table 4.2: Parameters for SVM.

4.2.3 Webserver

One goal of the project was to have the ability to add a new user to the system. The system should dynamically re-train the network when a new user is added to the system. More specifically, InceptionResnetv1 will train on the new image. Thus, a web server is implemented. It is possible to access the web server from other networks and access it with a mobile phone as well. For instance, a person that is located in another country can access the web server and add its face to the system if wanted. The project uses several programming languages in order to accomplish the goals for instance, HTML, CSS, Javascript and Python.

4.2.4 Face expressions

The web server does not only include a face detection and a face recognizer network. Also, face expression, face landmark, age, gender, tiny face detector which are implemented for uses of live-face recognition. These networks does not utilize the embeddings that are created with InceptionResnetv1 for face recognition, since the embeddings from InceptionResnetv1 are only responsible for creating embeddings for image classification. Each of these algorithms are implemented with Javascript and they are created by different CNNs architectures. Face expression, face landmark, age, gender, tiny face detector is described further below.

The Age and Gender Recognition Model accomplishes the task of predicting the age and gender of a person a single image or a video stream. It employs a feature extraction layer, an age regression layer and a gender classifier. The size of this model is 420kb and it has an architecture similar to Xception.

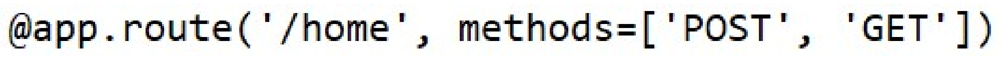
The Face Expression Recognition Model is fast and provides good accuracy. The size of the model is 310kb and it employs depth wise separable convolutions and densely connected blocks.

The 68 Point Face Landmark Detection Models is a lightweight landmark detector. The detector is fast and accurate. The model has various sizes to choose between for instance, the default model has a size of only 350kb and the tiny model is only 89kb.

The Tiny Face Detector is a real time face detector and has good performance. It is mobile and web friendly model. The model size is roughly 190 KB.

4.3 Routes of the web server

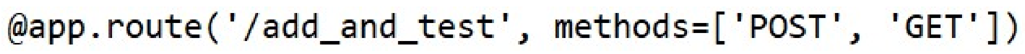
The purpose of the web server is to make the algorithms available for the users which are located on different networks. They have access to all functionalities such as, add a new user to the system, predict with a face and start real-time face recognition. Thus, the web server contains several pages and in order to access them, several routes are created. For instance, figure 4.4 illustrates the first route or entry point.



```
@app.route('/home', methods=['POST', 'GET'])
```

Figure 4.4: First entry point

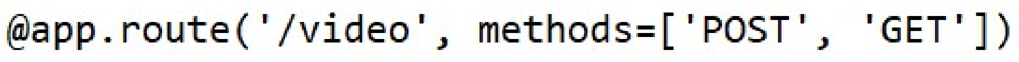
Chapter 5.5 presents the content for this entry point further. From the first entry point the user is able to navigate through the web server. For instance */add_and_test* in figure 4.5, is another entry point which means that the user has navigated to a new web page in the web server.



```
@app.route('/add_and_test', methods=['POST', 'GET'])
```

Figure 4.5: Second entry point

This entry point has several options to choose between for instance, the user can start a real-time face recognition, or the user can add a new face to the system. The last entry point is illustrated in figure 4.6. One face detection algorithm and a face recognizer network, namely Mtcnn and InceptionResnetv1 are implemented on the video in order to illustrate the system.



```
@app.route('/video', methods=['POST', 'GET'])
```

Figure 4.6: Navigates to the video page.

5 Results

This chapter present the results of all important pieces during the project. It starts by presenting one-shot learning of InceptionResnetv1 which is the network that is implemented in the final system. Thereafter, it shows the results for the competitors to InceptionResnetv1, meaning the other face recognizer networks which are not implemented in the final system. Then follows a presentation of the results, in terms of accuracies and execution times for several face detection algorithms. The execution time is the time it takes for the algorithms to process the video. Thereafter follows a presentation of the FPS of the face detection and face recognizer, followed by a presentation of the web server.

5.1 One-shot learning results of InceptionResnetv1

One-shot learning and few-shot learning are evaluated with several networks. The final system uses InceptionResnetv1 as the face recognizer. Thus, this chapter presents the results of InceptionResnetv1 with various datasets, namely one-shot 30 classes, two-shot 30 classes, five-shot 30 classes, one-shot 50 classes, two-shot 50 classes and five-shot 50 classes. The tables below present the results.

Iterations	300	500	1000	2000	4000
Number of classes	30	30	30	30	30
Number of training images	30	30	30	30	30
Test Accuracy	88.7%	98.0%	98.4%	98.8%	99.3%
Training time	~4m	~10m	~19m	~39m	~86m

Table 5.1. One-shot 30 classes (InceptionResnetv1).

```
Training complete in 86m 26s
Best val Acc: 0.993333
```

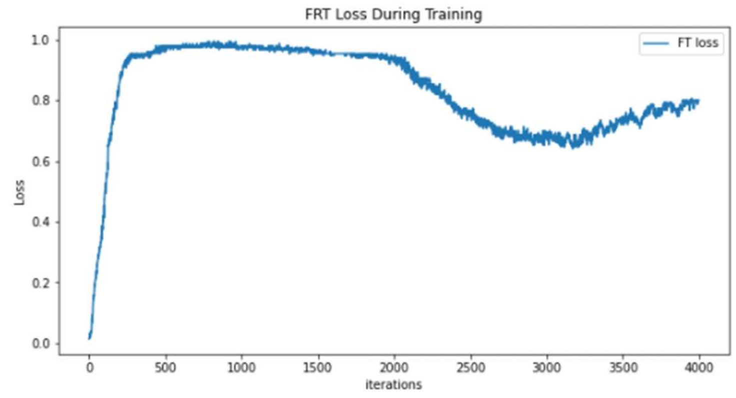


Figure 5.1: Accuracy of one-shot 30 classes using 4000 iterations (InceptionResnetv1).

Iterations	300	500	1000	2000	4000
Number of classes	30	30	30	30	30
Number of training images	60	60	60	60	60
Test Accuracy	100%	100%	100%	100%	100%
Training time	~6m	~13m	~21m	~43m	~91m

Table 5.2. Two-shot 30 classes (InceptionResnetv1).

```
Training complete in 21m 17s
Best val Acc: 1.000000
```

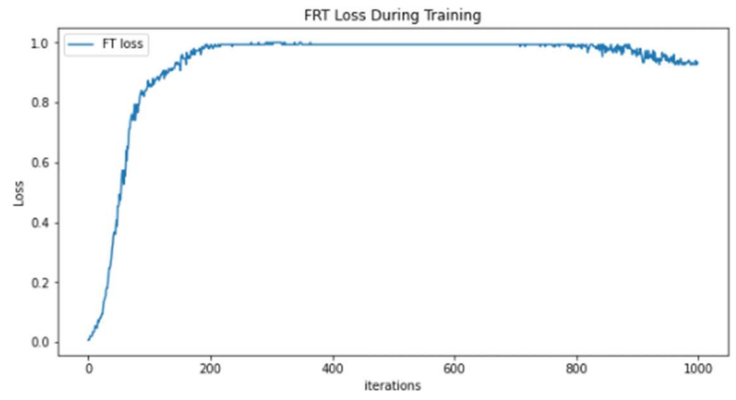
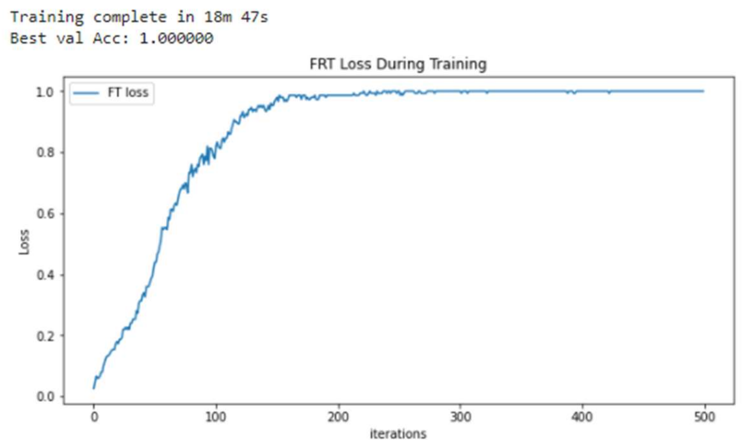


Figure 5.2: Accuracy of two-shot 30 classes using 1000 iterations (InceptionResnetv1).

Iterations	300	500	1000	2000	4000
Number of classes	30	30	30	30	30
Number of training images	150	150	150	150	150
Test Accuracy	97%	100%	100%	100%	100%
Training time	~8m	~19m	~28m	~49m	~101m

Table 5.3. Five-shot 30 classes (InceptionResnetv1).

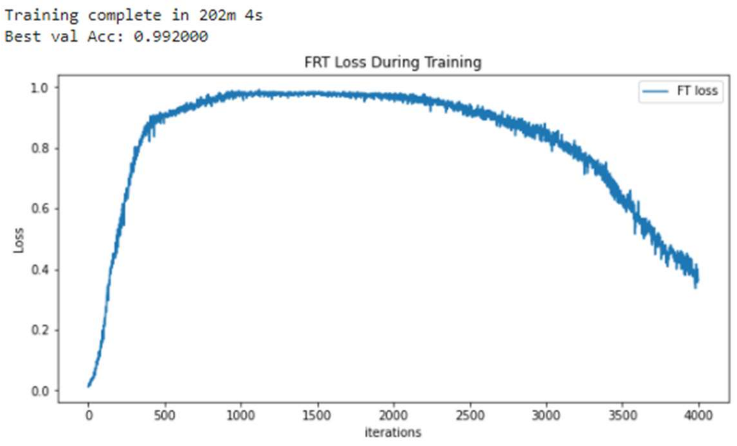


```
Training complete in 18m 47s
Best val Acc: 1.000000
```

Figure 5.3: Accuracy of five-shot 30 classes using 500 iterations (InceptionResnetv1).

Iterations	300	500	1000	2000	4000
Number of classes	50	50	50	50	50
Number of training images	50	50	50	50	50
Test Accuracy	74.4%	87.6%	95.2%	97.6%	99.2%
Training time	~12m	~25m	~51m	~100m	~202m

Table 5.4. One-shot 50 classes (InceptionResnetv1).

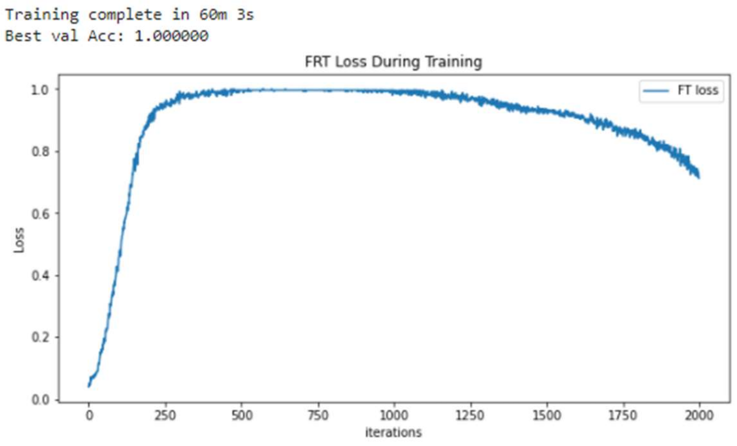


```
Training complete in 202m 4s
Best val Acc: 0.992000
```

Figure 5.4: Accuracy of one-shot 50 classes using 4000 iterations (InceptionResnetv1).

Iterations	300	500	1000	2000	4000
Number of classes	50	50	50	50	50
Number of training images	100	100	100	100	100
Test Accuracy	93.6%	99.2%	99.0%	100%	100%
Training time	~9m	~15m	~30m	~60m	~126m

Table 5.5. Two-shot 50 classes (InceptionResnetv1).

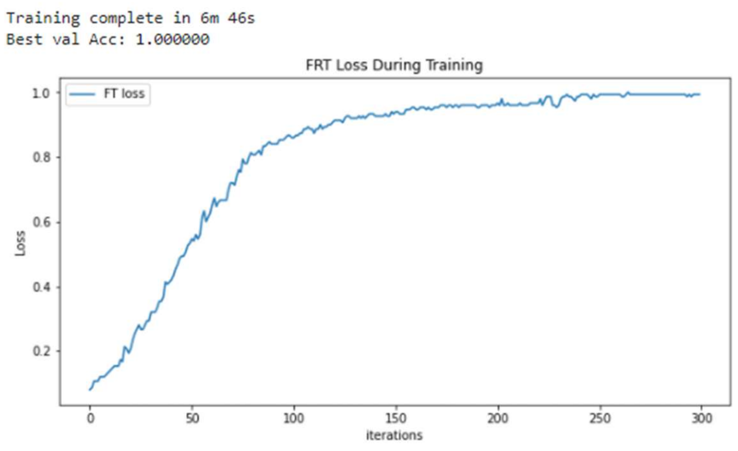


```
Training complete in 60m 3s
Best val Acc: 1.000000
```

Figure 5.5: Accuracy of two-shot 50 classes using 2000 iterations (InceptionResnetv1).

Iterations	300	500	1000	2000	4000
Number of classes	50	50	50	50	50
Number of training images	250	250	250	250	250
Test Accuracy	100%	100%	100%	100%	100%
Training time	~7m	~17m	~35m	~71m	~131m

Table 5.6. Five-shot 50 classes (InceptionResnetv1).



```
Training complete in 6m 46s
Best val Acc: 1.000000
```

Figure 5.6: Accuracy of five-shot 50 classes using 300 iterations (InceptionResnetv1).

5.2 One-shot learning results more face recognizers

This section presents the results of six networks, namely Resnet18, Alexnet, VGG11, Squeezenet, Densenet121 and Inception-v3. They are evaluated with one-shot 50 classes dataset. Since the accuracy of these networks was less than InceptionResnetv1, evaluation of 300 iterations was enough. The dataset one-shot 50 classes was increased with the augmentation approach described further in chapter 4.1.

5.2.1 One-shot 50 classes

Algorithm	**Iterations**	**Number of training samples**	**Test Accuracy**	**Training Time**
Resnet18	300	1	49.6%	~9m
AlexNet	300	1	48.5%	~8m
VGG-11	300	1	49.1%	~12m

Squeezenet-v1	300	1	58.1%	~9m
Densenet-121	300	1	49.2%	~14m
Inception-v3	300	1	41.7%	~8m

Table 5.7: Accuracy and training time on several networks.

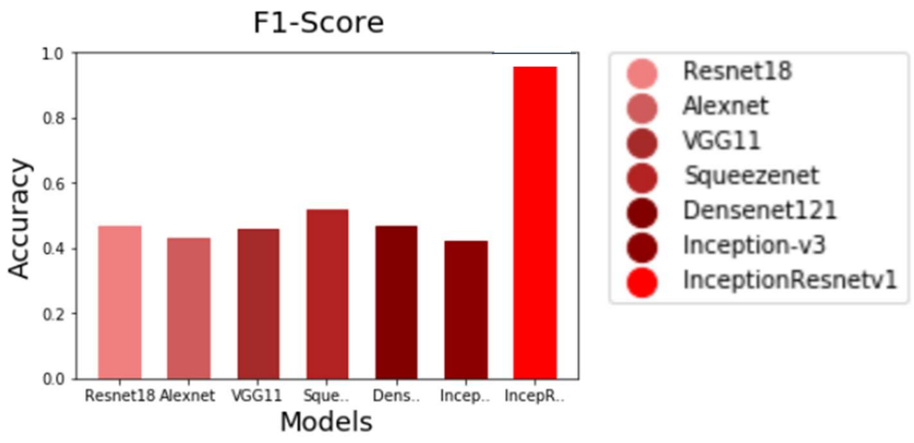


Figure 5.7: Accuracy of *one-shot 50 classes* using several different networks.

5.2.2 50-shot 50 classes with augmentation

Algorithm	Epochs	Number of training samples	Test Accuracy	Training Time
Resnet18	300	50	88.2%	~34m
AlexNet	300	50	87.1%	~38m
VGG-11	300	50	87%	~52m
Squeezenet-v1	300	50	97.2%	~33m
Densenet-121	300	50	92.2%	~85m
Inception-v3	300	50	93.1%	~39m

Table 5.8: Accuracy and training time on several networks with increased dataset.

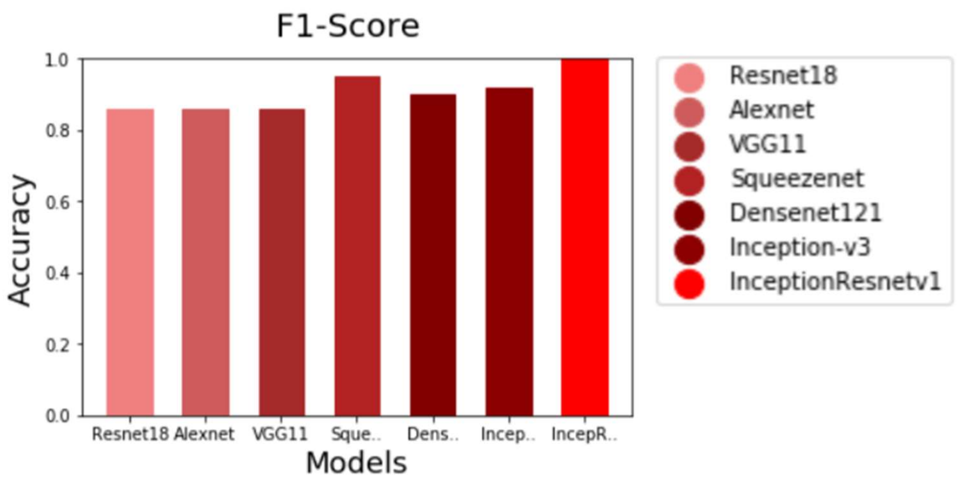


Figure 5.8: Accuracy of *50-shot 50 classes* using increased dataset.

5.3 Face detection results

This section shows the results of the face detection algorithms based on its total number of faces found and their computation times. The computation time is the time it takes for the face detection algorithms to process the video.

5.3.1 Execution time

Figure 5.9 presents the results of the different face detections algorithms. The algorithms were processed on a video that was 1 minute and 12 seconds long. The video contains multiple persons and faces. The time axis represents the time it took for the algorithms to process the video.

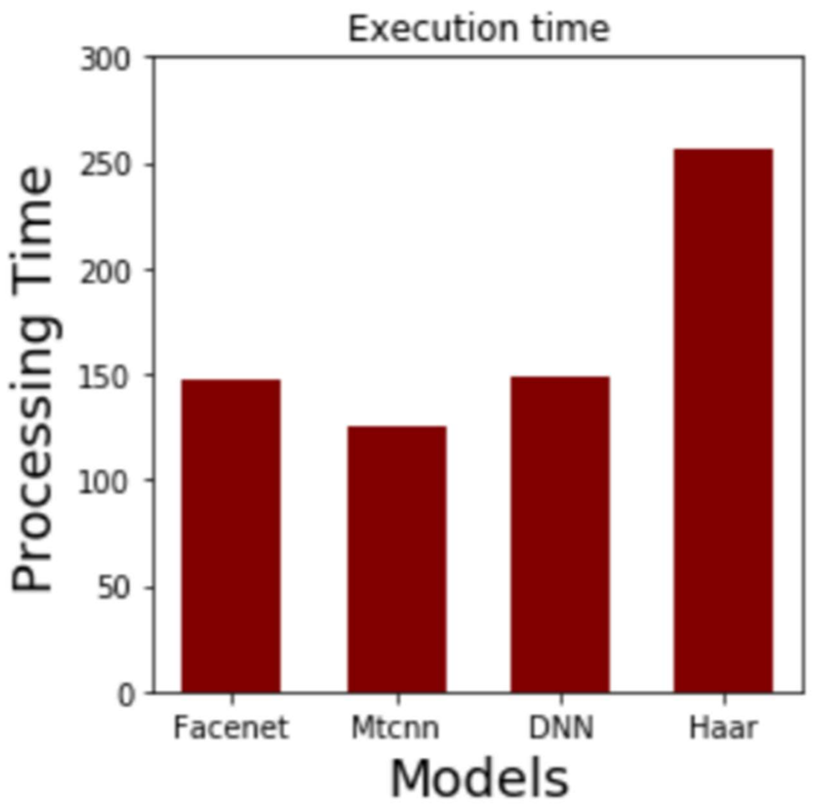


Figure 5.9: The time for each algorithm to process the video, in seconds.

5.3.2 Total number of faces found

Table 5.9 presents the number of faces the algorithms find. The number of faces is evaluated on how many faces they found in a video. The length of the video is 1 minute and 12 seconds and contains several persons and faces. The purpose of this table is to compare the four face detection algorithms in terms of how many faces they found in the video. See table 5.9.

FaceNet	Mtcnn	OpenCV_DNN	OpenCV_Haar
1742	1736	747	638

Table 5.9: Number of faces found in the video.

5.4 FPS results

This section presents the results in FPS while the system was only running with the face detection algorithm. Figure 5.10 shows that the FPS is 16.54 on a resolution of 640 x 320 pixels. See figure 5.10.

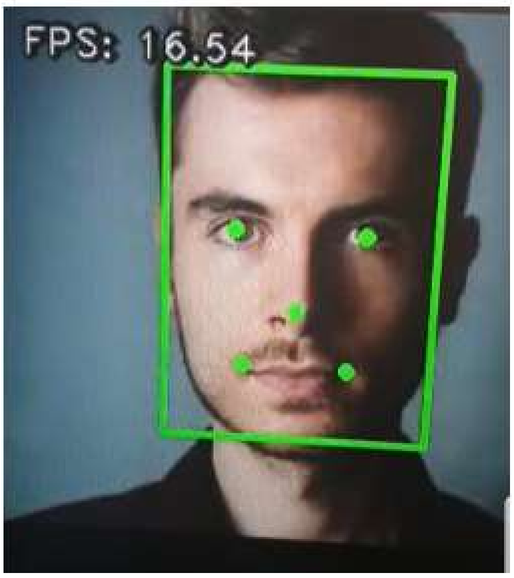
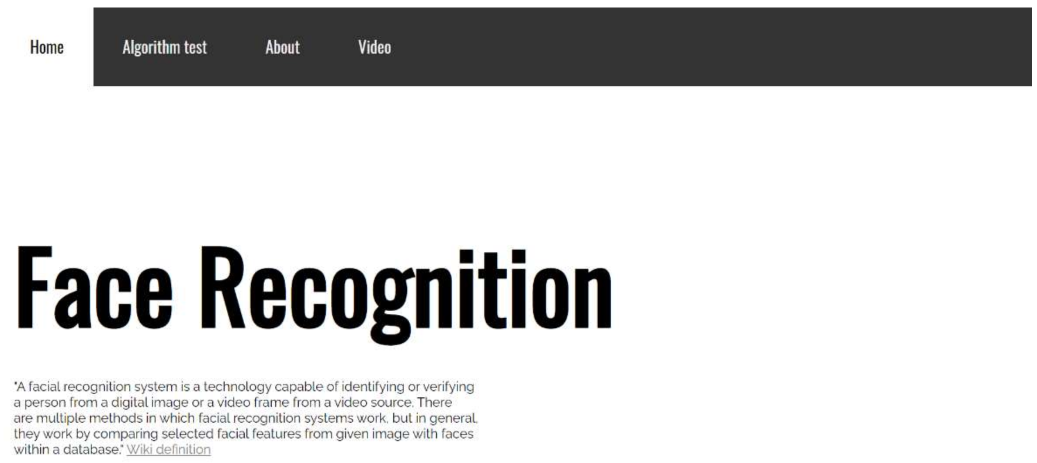


Figure 5.10: FPS for face detection algorithm.

The FPS is shifting a lot. For instance, if only the face detection algorithm is running it reaches an FPS of 16-17. However, if both the face detection and the face recognizer are running simultaneously the FPS decreases to 6-11.

5.5 Web server results

This section presents the results from the web server. Figure 5.11 illustrates the front page meaning the first page that the users get into when visiting the web server. Figure 5.11 illustrates which options the user has to navigate to different pages or entry points.

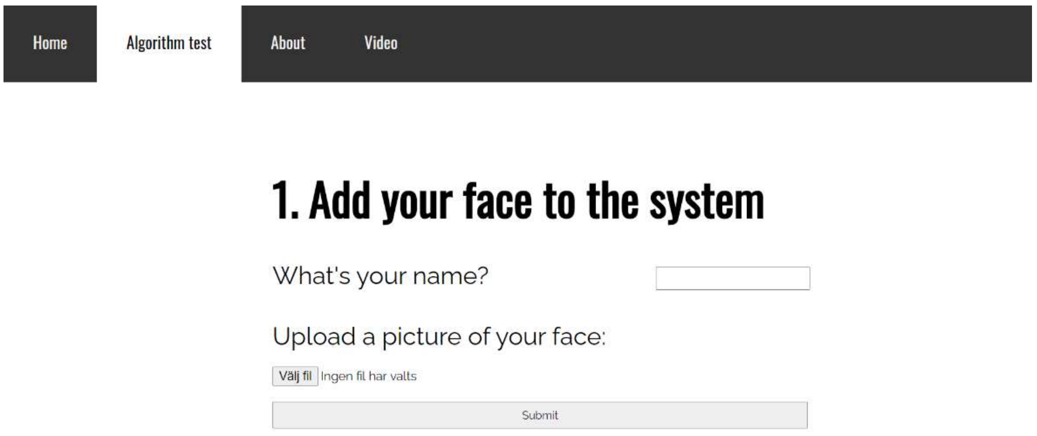


Face Recognition

"A facial recognition system is a technology capable of identifying or verifying a person from a digital image or a video frame from a video source. There are multiple methods in which facial recognition systems work, but in general, they work by comparing selected facial features from given image with faces within a database." Wiki definition

Figure 5.11: Home page in the web server.

Figure 5.12 illustrates the contents that is shown in the next webpage. The user can add a face to the system by either choosing an image from its directory or by snapping an image from a web camera or mobile phone. When the user has added a face to the system the next step is to test and predict using the network. Thus, it is possible to upload a new image on the same person in order to predict which is illustrated in figure 5.13.

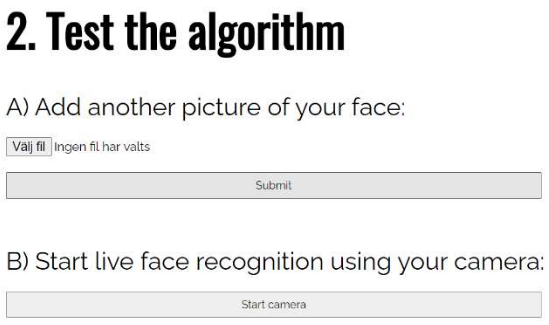


1. Add your face to the system

What's your name?

Upload a picture of your face:

Figure 5.12: User can add its face to the system.



2. Test the algorithm

A) Add another picture of your face:

B) Start live face recognition using your camera:

Figure 5.13: Predict a new image.

As figure 5.13 shows there is also another option (referred to as B above). When pressing this button, a real-time face recognition will start. During this phase, several algorithms are running simultaneously. They all predict in real time in the video stream. Face detection, face expression, face landmarks, age and gender are all algorithms that are running simultaneously. See figure 5.14.

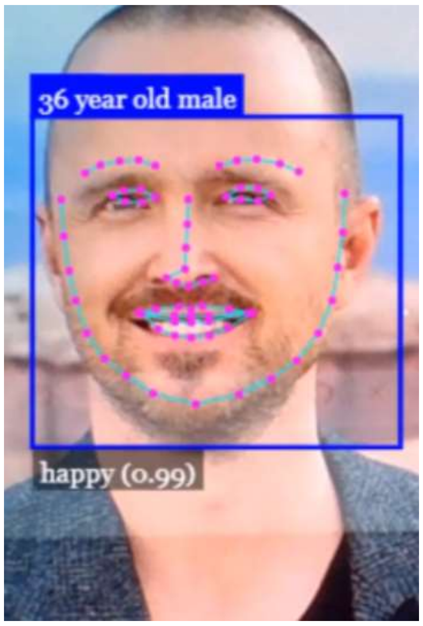


Figure 5.14: Real-time face recognition.

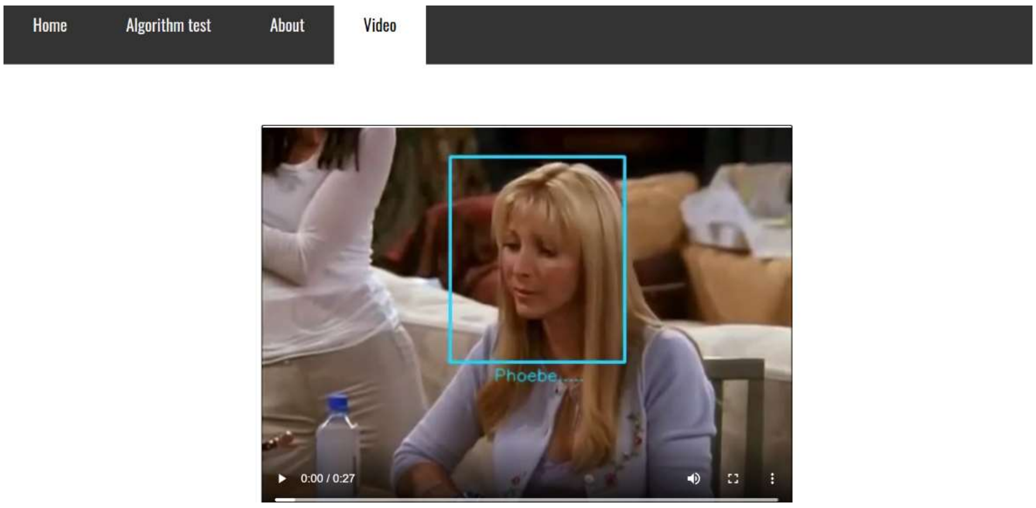


Figure 5.15: (face detection) Mtcnn and (face recognition) InceptionResnetv1. The video is from the TV series Friends.

Figure 5.15 illustrates the content presented in the last entry point. InceptionResnetv1 and Mtcnn are the algorithms that are running simultaneously on the video stream.

6 Discussion

This chapter discuss the results which is presented in chapter 5.

6.1 Discussion one-shot learning InceptionResnetv1

The final system has InceptionResnetv1 as the face recognizer. Several evaluations are done with this network for instance, one-shot 30 classes, two-shot 30 classes, five-shot 30 classes, one-shot 50 classes, two-shot 50 classes and five-shot 50 classes are all evaluated with InceptionResnetv1. The idea of the evaluation is to compare the results with the different datasets.

Table 5.1. One-shot 30 classes: As Table 5.1 presents it is clear that from 300 iterations compared to 4000 iterations there are differences between the accuracies. With 300 iterations for one-shot 30 classes, the network reaches an accuracy of 88.7% compared to 99.3% with 4000 iterations.

The training time for one-shot 30 classes is depending on iterations, see table 5.1. For instance, it takes around 4 minutes for InceptionResnetv1 to train during 300 iterations compared to 86 minutes with 4000 iterations.

Table 5.2. Two-shot 30 classes: However, table 5.2 presents the results from two-shot 30 classes. The dataset contains two training instance per class compared from table 5.1. With 300 iterations, the network reaches an accuracy of 100%. Table 5.2 illustrates that the upcoming evaluations reach an accuracy of 100% as well.

The training time for two-shot 30 classes took longer time compared to table 5.1 (one-shot 30 classes). For example, in table 5.1 it takes around 4 minutes to train the network, compared to 6 minutes which table 5.2 shows.

Table 5.3. Five-shot 30 classes: The results show that five-shot 30 classes reach an accuracy of 100%. The training time is not much longer compared to the other evaluations.

Table 5.4. One-shot 50 classes: This table presents the results from One-shot 50 classes meaning that the classes increase from 30 to 50. Table 5.4 presents that the network reaches an accuracy of 74.4% with 300 iterations. With 4000 iterations, it reaches an accuracy of 99.2%, see table 5.4.

The training time for one-shot 50 classes takes almost 3 times longer than one-shot 30 classes with 300 iterations.

Table 5.5. Two-shot 50 classes: As the table illustrates, the network reaches an accuracy of 93.6% which has improved a lot compared to one-shot 50 classes with 300 iterations which table 5.4 shows. It also presents that it reaches an accuracy of 99 % with 500 iterations.

The training time for two-shot 50 classes takes around 12 minutes with 300 iterations compared to 202 minutes with 4000 iterations.

Table 5.5. Five-shot 50 classes: As the table illustrates, the network reaches an accuracy of 100%. It also presents that it reaches an accuracy of 99 % with 500 iterations.

The training time for 50-shot 50 classes does not take much longer compared to the other evaluations.

To summarize, several face recognizer networks are evaluated during this project, namely InceptionResnetv1, Resnet18, Alexnet, Squeezenet, VGG11, Densenet121 and Inceptionv3. It is clear that InceptionRensetv1 performs better than the rest. However, it can be noted that the other algorithms perform much better when the amount of data is increased. For instance, figure 5.8 shows that Squeezenet reaches an accuracy near 98% with the 50-shot 50 classes dataset. InceptionResnetv1 performs well during all evaluations and does not require much data to get an accuracy of 99% and above.

Since one of the goals was to reduce the data and implement a network that still performs well with low amounts of data, InceptionResnetv1 is used in the final project.

6.2 Discussion results face detection

This section presents the results of the face detection algorithms which are responsible to find a face in a single image or in a video stream. The section starts by presenting the results according to their total number of faces found, followed by the execution time.

6.2.1 Total number of faces found

As table 5.9 illustrates, there are four different algorithms that are evaluated. The total number of faces found is measured on a video stream. In other words, the four algorithms predict simultaneously when they spot a face in the video stream. As table 5.9 presents, Mtcnn and Facenet perform best on this video followed by Opencv_Dnn and Openv_Haar.

Given by the total number faces found, there are no large differences in performance between Facenet and Mtcnn. But the final system uses Mtcnn as the face detection algorithms due to the performance in speed which is described below.

6.2.2 Execution time

The execution times between the four face detection algorithms are varying. For instance, Mtcnn performs best and can process the video faster than the rest of face detection algorithms. As figure 5.9 shows, the slowest algorithm is Opencv_Haar which almost is 2x slower than Mtcnn which is implemented in the final system.

However, I will not investigate which face detection algorithms are the fastest existing in the world in this study, as I only compared four face detections algorithms with each other. But according to several papers, Mtcnn is a state-of art face detection algorithm and the fastest algorithm that exists today.

6.3 Discussion one-shot learning results

In this section the result from chapter 5.2 is discussed.

6.3.1 Training time

Table 5.7 illustrates the results from the different algorithms that are trained with one-shot learning. Several face recognizer networks are evaluated using one-shot 50 classes and 50-shots 50 classes, generated using augmentation. As Table 5.7 illustrates, two algorithms have the same training time, namely Alexnet and Inception-v3. The slowest of them all is Densenet121 which has a training time of 14 minutes. VGG11 had almost 2x the time in order to train compared to Alexnet and Inceptionv3 One reasoning why some of the algorithms perform slower than others is that they have different amounts of parameters which increase the size of the networks. This make the size of the network larger.

6.3.2 Accuracy

Figure 5.7 and 5.8 presents the accuracy of the algorithms. As it shows it is clear that these algorithms needs more data in order to perform well. In figure 5.7 where the algorithms are evaluated with one-shot 50 classes they have an accuracy around 50% whereas when they are evaluated with 50-shot 50 classes they are improving in accuracy. For instance, Squeezenet has an accuracy near 97%.

6.4 Discussion of augmented dataset

This section will present the execution time and the accuracy for the same algorithms that were presented in section 6.3. However, the difference is that the dataset has been enhanced and has been increased from one sample to 50 samples per class.

6.4.1 Execution time

This section presents the results of the algorithms with an increased dataset, namely 50 instances for each class, using the augmentation method. A generating function is implemented in order to increase the dataset. However, Table 5.8 shows that Densenet121 took 85 minutes to train. Densenet121 is the slowest network and the fastest of them is Resnet18 which had 34 minutes of training. Alexnet is not far behind as it perform the training in 38 minutes.

6.4.2 Accuracy

The algorithms are evaluated with F1-Score. Figure 5.7 and 5.8 present the results. It is an interesting result since the models that are evaluated were trained with 50 instances per class, meaning that the accuracy has been improved by the augmentation. However, InceptionResnetv1 still performs best, with an accuracy of 100%, as illustrated in figure 5.8. However, it is interesting to see that all other algorithms namely Resnet18, Alexnet, Squeezenet, VGG-11, Densenet-121 and Inception-v3 all improves with augmentation. Table 5.8 presents that Resnet18 has an accuracy of 86% which is an increase compared to when it was trained with a single instance, namely one-shot 50 classes.

6.5 Discussion FPS

Section 5.4 presents the result of the FPS on the face detection algorithm. It is evaluated when Mtcnn is running and Figure 5.10 illustrates the result. It shows an FPS of 16.54 and the resolution of the windows is set to 640 x 340 pixels. When the resolution of the windows is increased, the FPS goes down. However, 640 x 340 pixels is a good resolution in this project.

6.6 Discussion web server

Section 5.5 presents the web server and the server that was the final system. It contains all of the algorithms that is presented in Chapter 4. Mtcnn is used because the evaluation of the face detection illustrates that it has the fastest performance while being accurate. The face recognition algorithm that is used in the web server is Inceptionresnetv1 since it performs best during the accuracy test and it is also fast relative to the training.

7 Conclusions

There are several goals which are presented in section 1.3 and the first goal is *research and find different approaches in order to solve the one-shot learning problem*. The goal is achieved and the face recognizer algorithm which is called InceptionResnetv1 presents accuracy up to 100% with one-shot learning. Another approach to solve the one-shot learning problem is to use augmentation meaning that a function generates more images from one image in order to increase the dataset. All algorithms that are evaluated during this project presents better results when they got an enhanced dataset with augmentation. However, as mentioned in the beginning of this section InceptionResnetv1 shows extremely good results without augmentation. Therefore, this network is implemented in the final system.

The next goal of this study is to *implement a face recognition system with only one sample per class*. This goal is achieved with two different approaches. The final system uses a face recognizer network that calls InceptionResnetv1 and it shows good results. Thus, this network is implemented in the final system. However, another approach that solves the one-shot learning problem is with augmentation. The remaining face recognition algorithms that are not used in the final system shows results up to 85-97% with augmentation.

The third goal that section 1.3 presents is *implement a suitable face detection algorithm*. This goal has been solved with a face detection algorithm namely, Mtcnn. The final system uses this algorithm since it presented best results due to the execution time and it have good accuracy as well. From several paper it is also known that Mtcnn is a state-of-art algorithm.

The fourth goal is *implement algorithms that can classify the age, gender, face landmarks and facial expressions of a person.* This goal has also been achieved with several algorithms. These algorithms are activated and running on the client side and are therefore implemented with Javascript. This enables ability to several users from different networks to use the system.

The fifth goal is *implement a webserver with all algorithms included.* This goal is achieved. A webserver is implemented and the entire face recognition system is implemented within it. Namely InceptionResnetv1, Mtcnn, Face landmarks, Face expression, age and gender. REST APIs are implemented in order to communicate between the server and the clients. The server is called Flask and it is implemented in Python.

The sixth goal is *give the possibility to add a new user to the system and re-train the network.* This goal was achieved as the webserver gives the possibility to add a new user to the system.

The last goal is to *implement the entire system on a Jetson Nano and optimize it.* This goal is solved the entire system is implemented on the Jetson Nano and it is optimized as well with GPU programming.

7.1 Ethical aspects

Photography ethics are the principles that guide how we take and share images. Photography ethics are subjective, contextual and fluid, meaning that every person's ethics will be different since ethics are based on a person's life and experience and values. This project suggests a solution on a face recognition system with one-shot learning. Thus, it is important to care about the ethics since there is a camera included in the system. When the algorithms in the system spot a face of a person it will pre-process the image and save it in a database. This can be infringement of an individual's privacy. For instance, if the system is implemented on a public place where people are not aware about the system and its potential power there will be a problem. Thus, it is important to create awareness about the system to the users with warning signs or something similar that inform the users. Photography ethics matter since no one want to be photographed without knowing it.

To summarize, this paper suggests a solution on how to implement an advanced and optimized face recognition system that can be used in several fields and applications. Its strengths and its usefulness can help people and companies on a daily basis. For instance, as a security system to give entrance rights. However, beyond all benefits it can bring, it is crucial to consider the ethical aspects if it should be deployed in a live setting.

7.2 Future work

For future work, it could be interesting to investigate in more suitable algorithms that can perform well on small amounts of data. The results of this study indicate that InceptionResnetv1 performs well on one-shot learning problems. However, the other algorithms that were tested can still be improved. It would also be interesting to see how this project's solution would work when implemented as part of a full scale solution, such as in a car. Another idea for future work could be to further develop the applications for various devices. Algorithms could further be adapted to the requirements of the different operating systems on the market, for instance. This solution contains limitations in the use of real-time camera for at least one of the operating systems, iOS. Furthermore, additional efforts to improve user experience and adapt it to broader user groups, for instance vision disabled users, could also be conducted.

References

[1] Tan, X., Chen, S., Zhou, Z. H., & Zhang, F. (2006). Face recognition from a single image per person: A survey. *Pattern recognition*, *39*(9), 1725-1745.

[2] Zhao, W. , Chellappa, R., Phillips, P. J. and Rosenfeld, A., Face Recognition: A Literature Survey, ACM Computing Survey, December Issue (2003) 399-458.

[3] Chellappa R, Wilson C L, Sirohey S. Human and machine recognition of faces: a survey. Proceedings of the IEEE, 83(5) (1995) 705-740.

[4] Daugman J. Face and gesture recognition: Overview, IEEE Trans. Pattern Analysis and Machine Intelligence, 19(7)(1997) 675-676.

[5] Guo, Y., & Zhang, L. (2017). One-shot face recognition by promoting underrepresented classes. *arXiv preprint arXiv:1707.05574*.

[6] Shao, L., Zhu, F., & Li, X. (2014). Transfer learning for visual categorization: A survey. *IEEE transactions on neural networks and learning systems*, *26*(5), 1019-1034.

[7] Pan, S. J., & Yang, Q. (2009). A survey on transfer learning. *IEEE Transactions on knowledge and data engineering*, *22*(10), 1345-1359.

[8] Sharma, R., Kumar, D., Puranik, V., & Gautham, K. (2019, April). Performance Analysis of Human Face Recognition Techniques. In *2019 4th International Conference on Internet of Things: Smart Innovation and Usages (IoT-SIU)* (pp. 1-4). IEEE.

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

[10] LeCun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278-2324.

[11] Gu, J., Wang, Z., Kuen, J., Ma, L., Shahroudy, A., Shuai, B., ... & Chen, T. (2018). Recent advances in convolutional neural networks. *Pattern Recognition*, *77*, 354-377.

[12] Szegedy, C., Ioffe, S., Vanhoucke, V., & Alemi, A. (2016). Inception-v4, inception-resnet and the impact of residual connections on learning. *arXiv preprint arXiv:1602.07261*.

[13] Szegedy, C., Vanhoucke, V., Ioffe, S., Shlens, J., & Wojna, Z. (2016). Rethinking the inception architecture for computer vision. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 2818-2826).

[14] He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 770-778).

[15] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems* (pp. 1097-1105).

[16] Iandola, F. N., Han, S., Moskewicz, M. W., Ashraf, K., Dally, W. J., & Keutzer, K. (2016). SqueezeNet: AlexNet-level accuracy with 50x fewer parameters and< 0.5 MB model size. *arXiv preprint arXiv:1602.07360*.

[17] Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. (2017). Densely connected convolutional networks. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 4700-4708).

[18] Schroff, F., Kalenichenko, D., & Philbin, J. (2015). Facenet: A unified embedding for face recognition and clustering. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 815-823).

[19] Chopra, S., Hadsell, R., & LeCun, Y. (2005, June). Learning a similaritymetric discriminatively, with application to face verification. In 2005IEEE Computer Society Conference on Computer Vision and PatternRecognition (CVPR'05) (Vol. 1, pp. 539-546). IEEE.

[20] Face Detection using Haar Cascades
https://docs.opencv.org/3.4/d7/d8b/tutorial_py_face_detection.htm
l
Accessed: 10 June 2020.

[21] Support Vector Machines
https://scikit-learn.org/stable/modules/svm.html
Accessed: 10 June 2020.

[22] Gaussian Naïve Bayes
https://scikit-learn.org/stable/modules/naive_bayes.html
Accessed: 13 June 2020.

[23] Nearest Neighbors
https://scikit-learn.org/stable/modules/neighbors.html
Accessed: 20 June 2020.

[24] Face-api.js
https://justadudewhohacks.github.io/face-api.js/docs/index.html
Accessed: 25 June 2020.

[25] Guo, Y., & Zhang, L. (2017). One-shot face recognition by promoting underrepresented classes. *arXiv preprint arXiv:1707.05574*.

[26] Zhao, W., Krishnaswamy, A., Chellappa, R., Swets, D. L., & Weng, J. (1998). Discriminant analysis of principal components for face recognition. In *Face Recognition* (pp. 73-85). Springer, Berlin, Heidelberg.

[27] Edy, W., Imam Husni, A. A., Herny, F., Wiwien, H., Muchamad Taufiq, A., & Prajanto Wahyu, A. ATTENDANCE SYSTEM BASED ON FACE RECOGNITION SYSTEM USING CNN-PCA METHOD AND REAL-TIME CAMERA. *IEEE*.

[28] Sameem, M. S. I., Qasim, T., & Bakhat, K. (2016, December). Real time recognition of human faces. In *2016 International Conference on Open Source Systems & Technologies (ICOSST)* (pp. 62-65). IEEE.

[29] Goodfellow, I., Bengio, Y., Courville, A., & Bengio, Y. (2016). *Deep learning* (Vol. 1). Cambridge: MIT press.

[30] Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., ... & Kudlur, M. (2016). Tensorflow: A system for large-scale machine learning. In *12th {USENIX} symposium on operating systems design and implementation ({OSDI} 16)* (pp. 265-283).

[31] NumPy
https://numpy.org/doc/stable/about.html
Accessed: 2 July 2020.

[32] Matplotlib
https://matplotlib.org/
Accessed: 10 July 2020.

[33] Paszke, A. Gross, S. , Chintala, S. , Chanan, G. , Yang, E. , DeVito, Z., … & Lerer, A. (2017). Automatic differentiation in pytorch.

[34] OpenCV
https://docs.opencv.org/3.4/d7/d8b/tutorial_py_face_detection.html
Accessed: 10 July 2020.

[35] Jupyter
https://jupyter.org/
Accessed: 10 July 2020.

[36] Scikit-learn "scikit-learn",
https://scikit-learn.org/stable/
Accessed: 20 July 2020.

[37] Flask server
https://flask.palletsprojects.com/en/master/quickstart/
Accessed: 10 June 2020.

[38] Glob
https://docs.python.org/2/library/glob.html
Accessed: 17 July: 2020.

[39] Albumentations
https://albumentations.readthedocs.io/en/latest/
Accessed: 18 July 2020.

[40] Pytube3
https://python-pytube.readthedocs.io/en/latest/
Accessed: 2 August.

[41] Split-folders
https://pypi.org/project/split-folders/
Accessed: 10 August.