U. PORTO

FEP FACULDADE DE ECONOMIA
UNIVERSIDADE DO PORTO

Practical Face Recognition: Building a Complete System Able to Identify Subjects in Real Time

by

Bernardo Maria de Lemos Ferreira Casimiro da Costa

Master's Thesis in Data Analytics

Supervised by
Professor João Manuel Portela da Gama
Dr. Brais Cancela

Biography

Bernardo Maria de Lemos Ferreira Casimiro da Costa was born in Porto, Portugal, on April 3rd 1994. He graduated in Computer Science in 2016 from FCUP. On the same year he started his masters' degree studies in Data Analytics at FEP.

Abstract

Face Recognition has suffered tremendous improvements in the past few years; it has developed a very strong community and is currently a very hot topic. The advances in performance of Face Recognition models have allowed its integration in day-to-day tasks.

This work studies the Face Recognition problem, covering a variety of different systems. These systems are tested under standard benchmarks and on a proposed protocol. The main goal of this research is to show that it is possible to achieve competitive results on standard Face Recognition benchmarks using a mid-range personal machine. The effectiveness and efficiency of the developed systems are presented proving that such goal can be achieved.

Keywords: Face recognition; Deep learning; Convolutional Neural Networks.

Index

Index of Figures

Index of Tables

1 Introduction

The evolution in computer vision is staggering. Deep learning based methods have achieved state-of-the-art performance in a very small time gap. The way image processing is studied has suffered considerable changes over the years (Learned-Miller et al., 2016; Rui & Huang, 1999). Many research fields have developed amazingly, such as image segmentation, object localization and classification and face recognition (He et al., 2017; Redmon & Farhadi, 2016, 2018; Schroff et al., 2015) thanks to the advances in computer hardware and the availability of information.

Face recognition is a challenging and amazing task that consists in classifying face images with a known identity. This problem can be addressed by performing the so-called 'face verification': having two images, the task is to determine whether the two images belong to the same person or not. Recent studies and evolution in computer hardware decreased the gap between computer and human-level performance in face recognition (Taigman et al., 2014). The human-level performance in face verification is approximately ninety-eight per cent (Deng et al., 2017; Taigman et al., 2014). A model presented by a Google research team, FaceNet (Schroff et al., 2015), surpassed human-level performance. A more recent work by the Chinese group, Tencent YouTu Lab (Tencent, 2018), has outperformed FaceNet, yet this work is proprietary and closed. Some very recent state-of-the-art models have been proposed where the models' architecture and the training method are freely and publicly available (Deng et al., 2018; Liu et al., 2017; Wen et al., 2016)

This masters' thesis is organized as follows. In section 2, image processing is addressed, where popular methods and their respective foundations are described. Succeeding, the task of face recognition in contemplated, where the respective challenges and related works are explained. Hereafter, Section 3 describes popular datasets that have been used in face recognition problems. Section 4 presents the technologies that were used in the development of the software. Section 5 exposes the methodology and approaches that were considered in the development of the face recognition system, showing the obtained results and its discussion.

1.1 Motivation

The applications of face recognition technology are immense. Innovative products like *Lighthouse* (2018) allow intelligent house monitoring. The system consists in a camera that is able to recognize a house's most frequent visitors, recognizing children, adults, animals and thieves. A more familiar example is Facebook, that uses face recognition algorithms (Taigman et al., 2014) to automatically tag friends on posted photos. Another developed work, FaceTime (Arsenovic et al., 2017), incorporates this kind of technology to account employees working hours. Tencent (2018), the Chinese group that holds one of the highest performing face recognition algorithms, offers a variety of services: face detection, attribute detection and identification.

Motivated by both the good results and the availability of out-of-the-bag machine learning algorithms, it was established that face recognition would be a tremendous theme to be deeply studied. Works like FaceTime (Arsenovic et al., 2017) use solely pre-trained models, achieving a considerable performance. With all this rich information freely available and a sense of business opportunity, the will to develop such systems grew.

1.2 Problem Definition

Face recognition consists in attributing a known identity to a face image. To that end, a set of known subjects must exist. The process largely depends on the face representation that must capture each face's unique features. The problem also involves the classification/search task that allows the system to separate individuals. Thus, the problem consists in the development of a face representation method and face representation classifier. Several methods (Deng et al., 2018; Schroff et al., 2015; Taigman et al., 2014; Wen et al., 2016) have achieved remarkable performances. They all use neural networks to extract face features, with different architectures, but relying on different classification methods. The work in (Schroff et al., 2015) compares face representations by using a distance threshold, while the works in (Taigman et al., 2014; Wen et al., 2016) use an end-to-end deep neural network model that performs multi-class classification. The proposed work for the masters' thesis is the study of the face recognition problem. Different systems were tested by using a public and well-known face recognition dataset.

2 Face Recognition

This research refers to the problem of face recognition that can be addressed by the paradigms of face verification and identification (Learned-Miller et al., 2016). Verification refers to the task of analyzing two face images and decide whether the two images represent the same person or not (Learned-Miller et al., 2016). Identification problem consists in deciding which images of the *gallery,* set of images representing a group of individuals, are represented by the *probe*, an image or set of images (Learned-Miller et al., 2016). Face recognition is the task of attributing a known identity to a face image from a set of known classes.

To solve the problem of face recognition with unconstrained images one must solve a set of related problems: face localization, alignment, representation and comparison. The best performing methods utilize convolution neural networks (CNNs) to solve this problem (Deng et al., 2018; Schroff et al., 2015; Wen et al., 2016) although other approaches have achieved good results (Devi & Hemachandran, 2016). To resolve the referred sub-problems the characteristics of images must be considered.

2.1 Image Processing

Image processing is the process of manipulating digital images. An image can be processed by using either parametric or non-parametric analytical methods (Ballard, 1981; Chuang et al., 1996), probabilistic distributions (Martin et al., 2012; Stricker & Orengo, 1995), gradients (Dalal & Triggs, 2004), machine learning (He et al., 2017; Kazemi & Sullivan, 2014; Simonyan & Zisserman, 2015; Taigman et al., 2014) and others (Lowe, 2004). The utilized methods depend both on the patterns that one desires to study and on the task.

The complexity of the problem is large since the dimension of digital images is rather high and requires fairly expensive computations.

There are multiple problems in image processing, being the following ones among the most popular: object detection, classification, transformation, pattern recognition, feature extraction and image retrieval. When addressing the face recognition problem, one can't simply compare the raw pixels of two face images. Therefore, the features of each image must be extracted, in order to compare them. This way, the extracted features of each image are compared instead of the raw pixels. Thus, feature extraction and comparison techniques need to be considered

Feature Extraction and Representation

Raw pixels must be represented in a way that allows one to study characteristics among images. Features may be represented through text, like annotations and keywords, and/or by visual features (Rui & Huang, 1999). Features that represent images as text are known as *text-based* features. Bag-of-Visual-Words is a popular text representation of images (Yang et al., 2007), yet it can use a large memory space and it doesn't allow a direct image search like content base image retrieval (CBIR) (Barz & Denzler, 2017). Moreover, images with an unknown class might exist just like complex and cluttered images, making the image search and representation very difficult or even impossible (Barz & Denzler, 2017) .

When extracting features from images one can divide these visual features in two categories. *Low-level* features refer to extracted information from pixels by the computer, like color-histograms, gradient orientation and others. *High-level* features address the features perceived by humans, attributing sense to the image. An example of a *High-level* feature can be contemplated in Figure 6, where the system detects the presence and localization of a face in an image. A *Low-level* feature descriptor HoG example is presented in Figure 1.

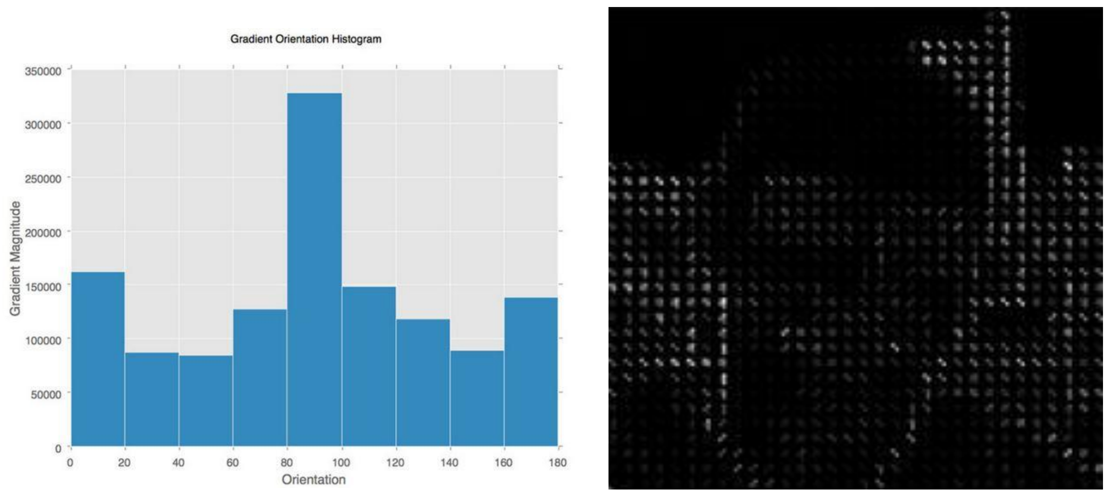


Figure 1 - Low-level face image feature descriptor example using HoG. Left bar graphic represents a pixel's gradient orientation in 20º intervals. Right image yields a visual representation of the extracted HoG.

The *low-level* level features can be further categorized as general or domain-specific features. General features don’t require any specific domain knowledge, being present across all images. Domain-specific features are application dependent; this may include faces, dog breeds and finger-prints, among others.

Visual features include color, shape, texture, color-layout and segmentation; where the first three are global features and the last two are local features.

The characteristics of the features depend on the task, for example, in object detection one requires for the shape representation to be invariant to scale, rotation and translation. Color, texture and shape are widely used in image classification and retrieval tasks (Jain & Vailaya, 1995; Niblack, 1993).

Color is present in all images and it can be described using one or three channels, grey scale and colored respectively. Furthermore, it is independent of image size and orientation. A very popular feature representation method for color is the color histograms (Rui & Huang, 1999). Color histograms simply represent the color distribution in an image. This process focuses on the complete image and not on any of its parts specifically. Another approach consists in using color moments (Stricker & Orengo, 1995). Once again this method extracts the color distribution in the image and is widely used in image retrieval, mainly used for indexing.

Shape is an important visual feature that allows one to accurately identify, detect and describe an object. When localizing or identifying an object in an image it is desirable for the method to be invariant to scale, rotation and translation. Identification consists in indicating the presence of an object, while localization a bounding box surrounding the object must be provided. Popular methods include Histogram of oriented Gradients (HoG) with linear SVM classifier (Dalal & Triggs, 2004), Scale Invariant Feature Transformation (SIFT) (Lowe, 2004) and neural networks (He et al., 2017; Redmon & Farhadi, 2016). HoG combined with SVM as baseline is a popular way of detecting objects (Dalal & Triggs, 2004). HoG detect the changes in pixel intensities, calculating the gradient of an image where it extracts the direction and magnitude of color changes. This is achieved by applying a kernel filter, like the Sobel mask, and Gaussian smoothing. The gradients are later represented as a histogram dived in several blocks, representing angle ranges. A Support Vector Machine (SVM) is a popular machine learning algorithm (Dalal & Triggs, 2004; Devi & Hemachandran, 2016). The goal of a SVM is to find a hyperplane able to separate different labeled data points. Hence the desirable hyperplane represents the largest separations distance between different labeled data points (Smola & Schölkopf, 2004). On the previous method, HoG with SVM classifier, the SVM must be able to draw a hyperplane able to separate a face and a non-face HoG descriptor. SIFT is a powerful method that extracts features and allows image matching in real-time. It is a very involving and complex algorithm that can be divided into the following stages. Scale space construction, LoG approximation, keypoints localization, most representative keypoints extraction, keypoints orientation calculation and generation of SIFT features. Each stage of the process relates to one of the invariance goals: scale, translation and rotation. Finally, the most recent methods in object detection and recognition involve deep neural networks based methods (He et al., 2017; Redmon & Farhadi, 2016, 2018). These methods achieve state-of-the-art accuracy. YOLO9000

(Redmon & Farhadi, 2016) is capable of distinguishing nine thousand different classes of objects in real-time.

Texture refers to the visual patterns that have a homogeneous behavior that doesn't result from the presence of a single color or intensity (Rui & Huang, 1999). The study of texture focuses on the problem of finding relationships between pixels. A common and early method is co-occurrence matrix. The main idea is to extract the distance and orientation between pixels and then compute representative statistics from the matrix to represent texture.

Color-layout resembles to the problem of studying color in sub-regions. Two main approaches were proposed. Subblock-based is when the color distribution of an image is studied by dividing the original image into smaller subimages. Another method is to study only a few subimages. This method is more sophisticated since the image has to be segmented into regions, yet the information retrieved can take smaller spaces of memory and be more representative.

Segmentation is present in many computer vision problems and it is essential when one desires to consider only parts of the image. These methods can be automatic or semi-automatic, meaning that they may use *apriori* knowledge or user feedback (Rui & Huang, 1999). Although many methods have been proposed most recent works that achieve state-of-the-art performance are based on deep neural networks (Bulat & Tzimiropoulos, 2017; He et al., 2017; Redmon & Farhadi, 2016; Ren et al., 2016). Good segmentation is very important in object recognition and the problem is very challenging. Methods based on selective search and neural networks have shown to have good results (Girshick et al., 2012), yet the community walks into complete end-to-end models using only a deep network (He et al., 2017).

In sum, feature extraction allows one to create a feature map of an image. This process allows image comparison, thus when applied to face images one may extract unique face features from different identities, targeting the face recognition problem.

Image Comparison in Large Dimension Spaces

It is necessary to apply efficient and effective multi-dimensional indexing methods in order to obtain scalable systems in large databases (Chuang et al., 1996). The dimensionality of the feature vectors is large and in image retrieval or in clustering problems one may not use Euclidean distances. Thus, when solving, for instance, an image retrieval or clustering problem, a promising approach is to perform dimension reduction and use the appropriate indexing methods (Rui & Huang, 1999). Dimension reduction is common in computer vision and it has been successfully used on face recognition tasks and object detection and recognition (Devi & Hemachandran, 2016; He et al., 2017; Redmon & Farhadi, 2016). When addressing the face recognition problem a face feature vector may have to be compared with others, thus the performance of a system depends largely on the image comparison and indexing techniques.

Similarity and Distance Measures

There are a wide number of similarity and distance measures. These measures may be Euclidean or non-Euclidean.

Although Euclidean distances may not capture the human perception (Rui & Huang, 1999), some Euclidean distances like the *L2* norm have been successfully applied (Barz & Denzler, 2017) just like the squared *L2* distance (Schroff et al., 2015).

Some common non-Euclidean measures are: cosine, correlation, histogram intersection among others (Rui & Huang, 1999). The work in (Jégou & Zisserman, 2014) proved that discarding the magnitude of the features vector and concentrating only on the direction is beneficial for capturing image sense.

Metric Learning

Metric learning is a field of machine learning with multiple application in computer vision, it has been used in face recognition (Schroff et al., 2015) and image retrieval. These methods use *apriori* information, labels, to learn a distance function that maximizes performance. This means that observations that have the same labels should have a small distance, in contrast to observations with different labels.

The Mahalanobis metric learning based techniques reveal a very satisfactory performance. The Mahalonobis metric can be corresponded to the covariance matrix. The goal of this approach is to learn a linear transformation where the most relevant features take a greater importance than irrelevant ones (Martin et al., 2012).
Large Margin Nearest Neighbor (LMNN) (Weinberger & Saul, 2009) metric is among the most popular metric learning techniques (Martin et al., 2012). The main goal is to optimize the K-NN algorithm. It uses a triplet loss-function based on equations 16 and 17. Conceptually the networks presented in (Schroff et al., 2015) are trained based on the LMNN loss-function. The algorithm computes the *k*-nearest neighbors for every observation. Then a perimeter that includes the nearest neighbors that have the same label as the seed, target neighbors, is defined. The samples that have different labels from the seed which invade the perimeter are penalized. This way the correlation between target neighbors strengthens.

2.2 Deep Learning and Computer Vision

Deep Learning (DL) refers to a machine learning field that operates with neural networks that have more than two layers, known as deep networks (Le, 2014). DL based systems have been successfully applied in multiple fields, including natural language processing (Mikolov et al., 2013; Pennington et al., 2014), object localization and recognition (Redmon et al., 2016), time series prediction (Qin et al., 2017) and face recognition (Howard et al., 2017; Schroff et al., 2015).
Ever since AlexNet won the ImageNet Challenge: ILSVRC 2012 (Krizhevsky et al., 2012), Convolutional Neural Networks have become very popular and are the state-of-the-art in several computer vision tasks (Chollet, 2017; Howard et al., 2017; Huang et al., 2017).
To this date, the most promising methods in face recognition are implemented under the deep learning paradigm. They utilize a type of neural networks to extract the features of faces that are able to accurately separate identities.

Neural Networks

A Neural Network (NN) is a machine learning model inspired by the human brain. These systems are constructed by stacking multiple layers of neurons. In a typical feedforward NN the neurons of contiguous layers are fully connected. This type of layer is referred as fully connected layer, which is the simplest architecture. A NN can be seen as a graph, where the neurons are the vertices and the connections are weighted edges. The input data gets transformed across the layers by applying the edges' weights; this is done through matrix multiplication. Moreover, neuron units apply an activation function, also called squashing function, and a bias term. The application of the activation function allows the introduction of non-linarites, like the sigmoid function (Cybenko, 1989) or the rectifier linear unit (ReLU) (Krizhevsky et al., 2012). The process of applying the connections' weights followed by the bias addition and activation is referred as the feedforward pass. The bias term and the weights are parameters of the network that must be learnt. Training a NN involves computing the feedforward pass followed by the application of the backpropagation algorithm, this process accounts for a full iteration in training which is known as epoch.

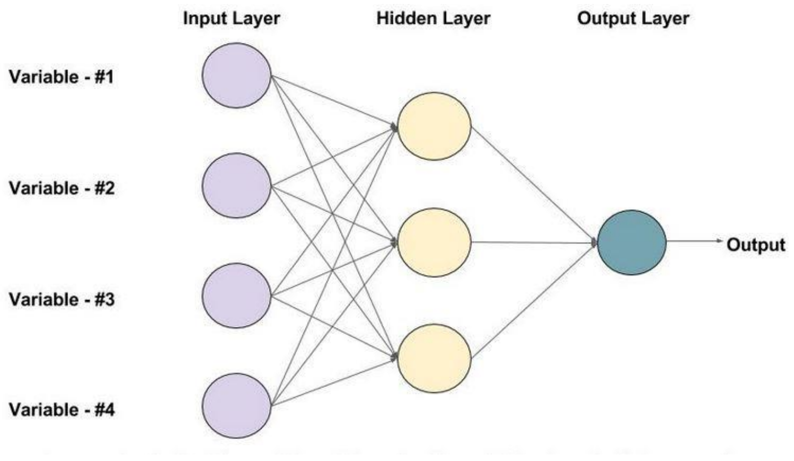


Figure 2 - Neural network with one hidden layer that accepts inputs of spatial dimension four and outputs a single value (Gupta, 2017).

Feedforward Pass

Let X be the network's multivariate input with $x_i \in X$, $w_{ij}^{l}$the weight of the i^{th} connection of the j^{th} neuron of the l^{th} layer and b_j^l the bias term of the j^{th} neuron of the l^{th} layer. Furthermore, the neurons apply an activation function σ, this may be, for example, a sigmoid, tanh or ReLU function. The feedforward pass can be expressed by the following equations:

$$Z_i^0 = x_i,\ \forall x_i \in X, i = 1, \dots |X| \tag{1}$$

$$Z_i^l = Z_i^{l-1} . w_{ij}^l \tag{2}$$

$$a_j^l = \sigma\left(\sum_{i=1}^{n} Z_i^l + b_j^l\right) \tag{3}$$

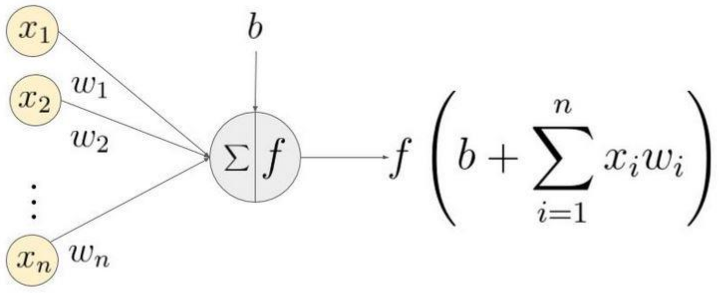


Figure 3 - Feedforward pass of a single neuron, where *b* is the bias term, x_i the i^{th} variable of the input, w_i the i^{th} weight of the neuron and *f* the activation function (Gupta, 2017).

Backpropagation and Optimization

The backpropagation algorithm is used to find the values of the NN's parameters (Le, 2014). This is achieved by computing the gradient of an objective function $\theta(x)$ with respect to the network's parameters. Backpropagation is nothing more than the chain rule applied to neural networks, since the final output is a composed function derived from all the previous layers. In a simple feedforward NN the error is propagated from the output to the input layer.

In order to update the parameters, backpropagation supports optimization algorithms typically based on Gradient Descent (GD) (Le, 2014). The most popular optimization algorithms are: Stochastic Gradient Descent (SGD), Mini-Batch GD, RMS-Prop, Adam and GD with Momentum (Ruder, 2016). SGD is the simplest algorithm and the foundation of the other referred optimization processes. This algorithm uses the gradient value to update the parameters of the network by subtracting or adding it to their current values, depending if it is a minimization or a maximization problem respectively, at each training example. When training a network one must select the learning rate η, which is the step of the gradient.

Using the feedforward pass notation, introducing the learning rate η and a loss function J the SGD algorithm can be expressed by the equations 4 and 5.

$$w_{ij}^{l} = w_{ij}^{l} - \eta \nabla \frac{\partial J}{\partial w_{ij}^{l}} \tag{4}$$

$$b_{j}^{l} = b_{j}^{l} - \eta \nabla \frac{\partial J}{\partial b_{j}^{l}} \tag{5}$$

Convolution Neural Networks

Convolutional Neural Networks are a type of neural networks that have been very successful and ubiquitous (Howard et al., 2017) in computer vision (He et al., 2017; Redmon & Farhadi, 2018; Schroff et al., 2015). They have achieved state-of-the-art performance in image classification (Zoph et al., 2017), face localization (Zhang et al., 2016) and recognition (Chen et al., 2018; Deng et al., 2018; Schroff et al., 2015), among other tasks (He et al., 2017; Qin et al., 2017). A CNN is constituted by one or more convolutional and pooling layers, this way a CNN can be built by stacking multiple convolutional-pooling layers blocks. These blocks are used to transform the image data that will later flow, traditionally, through a fully connected layer for classification.

In face recognition, CNNs, are mostly responsible for extracting the unique features of a face, i.e. extract a face feature map. CNN extracted features have achieved the most notable results on this task (Schroff et al., 2015; Taigman et al., 2014). The community navigates to more complete systems able to solve all of face recognition's sub-problems using a single network, referred as end-to-end networks (Zhong et al., 2017). This type of neural networks is currently the most widely used method in face recognition, since it has been accomplishing constant break troughs.

Convolutional Layer

The convolutional layer contains a set of learnable filters that are responsible for extracting useful information from the input. A simple convolution will produce a two dimensional feature map via point wise multiplication followed by addition across the input channels. When using multiple filters one simply stacks the produced feature maps, hence the number of channels of the output is defined by the number of filters used in the convolutional layer. A typical filter has spatial dimension $3 \times 3 \times n$ or $5 \times 5 \times n$, where n is the number of channels of the input, in a color image it would be equal to 3. A filter must slide across the image at a specified stride. The stride defines the step size that the filter must take vertical and horizontally in order to navigate the image, being 1 a typical value. The stride and convolution combination must be capable of covering the input completely without ever crossing its limits.

There is a broad type of convolutions that have been successfully applied in computer vision besides simple convolutions including: depth-wise separable (Chollet, 2017; Sandler et al., 2018), grouped (Xie et al, 2017) and others (Lin et al., 2014; Zhang et al., 2017).

Padding

Padding involves filling the image borders with a specific value. The most popular version is zero-padding that involves filling the borders with zeros. Thus, padding allows controlling the spatial size of the data across the network. This way with a specified padding of k, with $k \geq 1$, an input with dimension $n \times n$ after applying padding will have dimension $(n+k) \times (n+k)$.

Pooling

Pooling operations are commonly used after successive convolutions. The goal of this operation is to reduce the spatial dimension of the input, hence reduce the number of parameters and computations in the network. Normally, pooling layers apply filters of size 2×2 with stride of 2 that reduces the spatial dimension of the input by half. Although there are several pooling operations, like *average*, *max* pooling is the most common one. *Max* pooling involves selecting the maximum value present in the filter, if one were to use a 2×2 kernel it would involve selecting the maximum value over 4 numbers at every step.

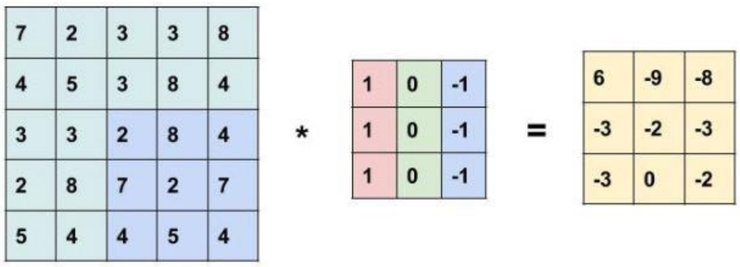


Figure 4 - Application of a 3x3 filter. Right most term of the convolution operation, over an input of dimension 5x5, the leftmost term of the convolution operation, using stride 1 that results in a 3x3 feature map (in yellow). The example is frozen at the last convolution operation, showing the covered surface by the filter on the input (blue region of the input) that results in the bottom right most value of the feature map (Gupta, 2017**).**

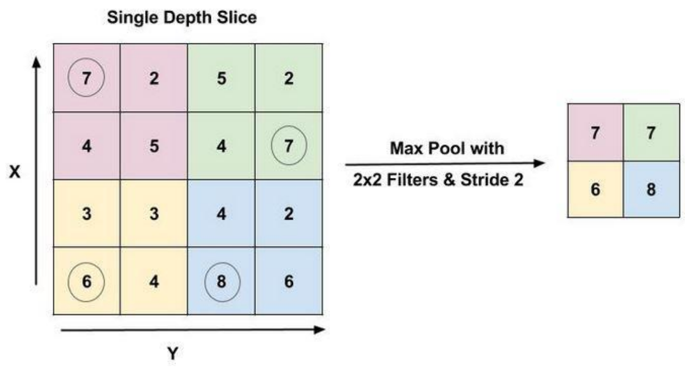


Figure 5 - Max pooling operation over an input of spatial dimension 4x4 using a 2x2 kernel with a stride 2 that results in a 2x2 output volume. The filter is applied four times over the input; the stages are separated by color. At each stage, the maximum value that will be part of the output is inside a circumference (Gupta, 2017).

Although new training approaches and novel loss functions have been suggested (Schroff et al., 2015; Ioffe et al., 2015), the CNNs' architecture has have an obvious impact on the deep learning community. The research and evolution of network architecture has led to improvements on different computer vision tasks (Redmon & Farhadi, 2018; Zoph et al., 2017). The reason for such an impact lays on the fact that new architectures allow deeper and more complex models, reduce convergence and inference time and diminish memory footprint (Chollet, 2017; Iandola et al., 2016). It is clear that the systems that achieved the highest performances in a target task combine state-of-the-art loss functions and CNN architectures (Chen et al., 2018; Schroff et al., 2015). The evolution of CNN architectures has gone from building very deep and computationally heavy models, VGG net (Simonyan & Zisserman, 2015), ResNet (He et al., 2015) and Inception (Szegedy et al., 2015), to shallower, lighter and more sophisticated models that can achieve the same level of performance of much deeper models, but largely reducing the number of parameters, Xception net (Chollet, 2017), MobileNet (Howard et al., 2017) and SqueezeNet (Iandola et al., 2016). A clear example is SqueezeNet that contains 50 times less parameters than AlexNet, but achieves the same level of accuracy. There is a large adoption by the computer vision community of the newest best performing network architectures, being them normally ImageNet Challenge winners. Since 2012, all the winning models of this challenge have been CNNs: AlexNet (Krizhevsky et al., 2012), ZFNet (Zeiler & Fergus, 2014), VGG (Simonyan & Zisserman, 2015), GoogLeNet (Inception) (Szegedy et al., 2015), ResNet (He et al., 2015) and NasNet (Zoph et al., 2017). Since 2015, when the Inception architecture was introduced, the proposals for new network architectures has shifted from building deeper CNNs to the implementation of different types of connections (He et al., 2015) and convolution operations (Chollet, 2017), moving to more innovative ways of building CNNs (Huang et al., 2017; Szegedy et al., 2016) instead of simply stacking simple convolution-pooling layers (Simonyan & Zisserman, 2015). Although ImageNet Challenge winners represent the state-of-the-art in image classification, the winning models take large amounts of memory and computational power (Howard et al., 2017). This way, smaller high performing models have been developed (Arriagaet al., 2017; Howard et al., 2017) that are able to run on mobile devices and to be used in real world applications (Chen et al., 2018; Sandler et al., 2018).

2.3 Face Localization

Face localization consists in outputting the bounding box that surrounds a face. Many approaches have been presented being, among the most popular, the Viola-Jones face detector (Viola & Jones, 2001), HoG and Linear SVM object detector (Dalal & Triggs, 2004) and CNN based methods (Jiang & Learned-Miller, 2017; Zhang et al., 2016).

The work in (Dalal & Triggs, 2004) combines HoG with a linear SVM. The method represents the image with HoG followed by a binary classification using a linear SVM. The binary classification consists in classifying the HoG as person or non-person.

The Viola-Jones object detector was the first method to provide good results in real-time (Viola & Jones, 2001). The method uses a HaaR feature representation, AdaBoost classifiers and a cascade of classifiers. A cascade of classifiers is a type of ensemble learning, but instead of votes the system is divided by stages. Each stage has its own classifier that uses the output information of the previous stage's classifier output, adding information to the input.

Deep learning methods have accomplished extraordinary results in object detection (Redmon & Farhadi, 2018; Zhang et al., 2016). DL based systems use CNNs to localize faces in an image and can address this problem in real time. Some recent and popular methods include Faster R-CNN (Jiang & Learned-Miller, 2017) and MTCNN (Zhang et al., 2016).

Among these methods the Multi-Task CNN (MTCNN) has been successfully used in different face recognition systems (David Sandberg, 2018; Zhong et al., 2017). The model consists of three CNNs: Proposal Net (P-Net), Refine Net (R-Net) and Output Net (O-Net). Each network has a different architecture and only 3x3 and 2x2 filters are used. The networks' architectures can be visualized in figure 21. The PReLu activation function is applied after every convolution and fully connected layers, excluding the output layer. Moreover, non-max suppression (NMS) is performed on every network output. Non-max suppression consists in eliminating bounding boxes that overlay each other's area in more than 50%, hence only the largest bounding box is preserved. Naturally this is performed on proposals of the same image. An image pyramid is fed to the system, meaning that the same image in different sizes is considered. The MTCNN training consists of three tasks: face/non-face classification, bounding box regression, and facial landmark localization. The model is trained using data from WIDER FACE (Yang et al., 2016) and CelebA (Liu et al., 2015). The face classification is a binary problem and the cross-entropy loss is minimized, where p_i is the probability of image x_i being a face and $y_i^{label} \in \{0,1\}$ is the label of image *i*:

$$L_i^{label} = -(y_i^{label}\log(p_i) + \left(1 - y_i^{label}\right)(1 - \log(p_i))) \qquad (6)$$

The bounding boxes are referred as: left, top, height and width. Let $\hat{y}_i^{box}$ the estimated bounding box of image *i* and y_i^{box} the bounding box true values, with $y_i^{landmarks} \in R^4$. For this task the Euclidean loss is employed:

$$L_i^{box} = \left\|\hat{y}_i^{box} - y_i^{box}\right\|_2^2 \qquad (7)$$

The final task, landmarks estimation, also minimizes the Euclidean loss. The landmarks are represented by five points: the center of each eye, nose tip and mouth left and right corners. Let $y_i^{landmarks} \in R^{10}$, using the same notation as equation 7, the loss function can be defined as:

$$L_i^{landmarks} = \left\|\hat{y}_i^{landmarks} - y_i^{landmarks}\right\|_2^2 \qquad (8)$$

Finally, the overall training target can be defined as:

$$min\sum_{i=1}^{N}\left(\alpha_{label}\beta_i^{label}L_i^{label} + \alpha_{box}\beta_i^{box}L_i^{box} + \alpha_{landmarks}\beta_i^{landmarks}L_i^{landmarks}\right) \quad (9)$$

Where N is the number of training samples, α_j is the task importance and $\beta_i^j \in \{0,1\}$ is the sample type indicator. Since different tasks are trained, different images are used for training, i.e. face/non-face image and partially aligned images. Different values of α_j are employed, for P-Net and R-Net the same values are used, concentrating more on the face classification task ($\alpha_{label} = 1, \alpha_{box} = 0.5, \alpha_{landmarks} = 0.5$) while in O-Net the landmarks are more deeply considered ($\alpha_{label} = 0.5$, $\alpha_{box} = 1$, $\alpha_{landmarks} = 0.5$). The MTCNN pipeline can be observed more intuitively in figure 6.

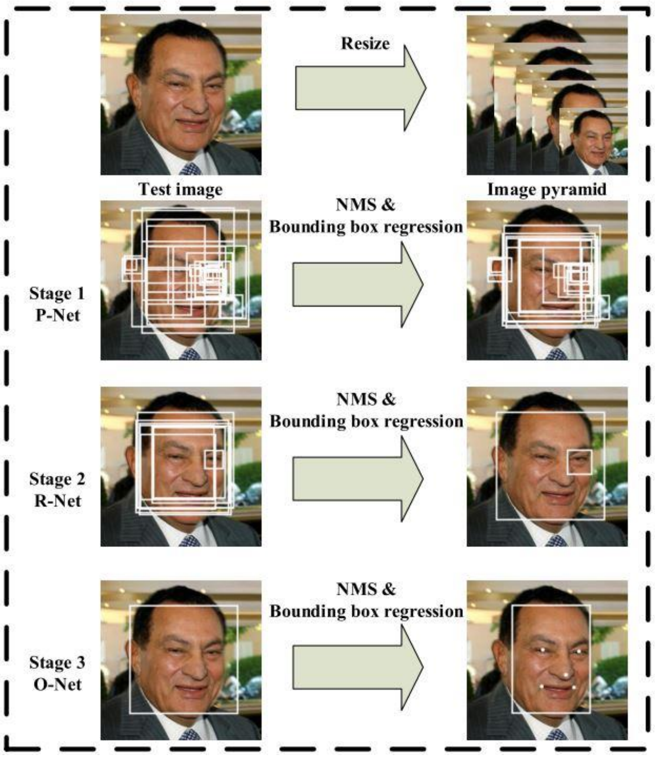


Figure 6 - MTCNN pipeline (Zhang et al., 2016).

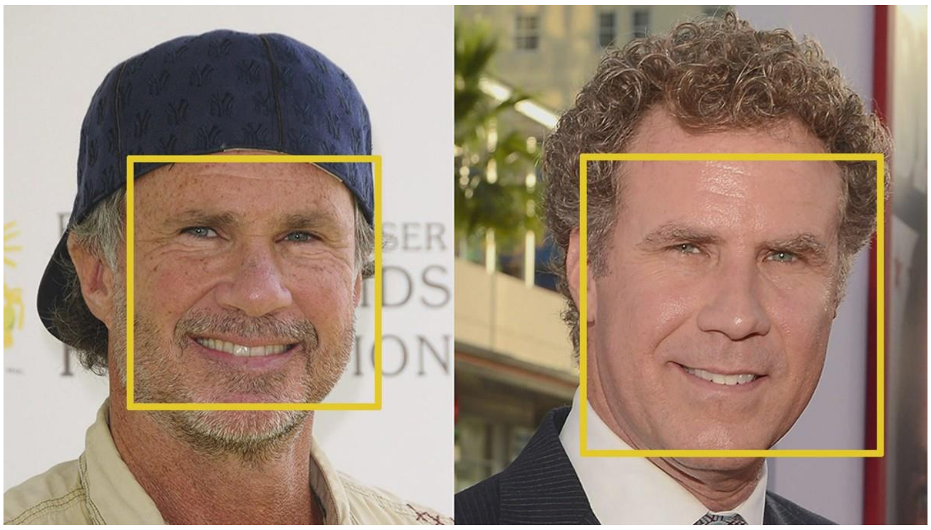


Figure 7 - Face localization with the respective bounding boxes (Geitgey, 2016).

2.4 Face Alignment

Face alignment is the problem of centering a face in an image and perform a transformation that can project the face landmarks in a frontal-view without deforming the image; an example is contemplated in figure 9. Some promising methods and widely used were proposed in literature: 2D face alignment using an ensemble of regressive boost trees (Kazemi & Sullivan, 2014), 2D/3D face alignment using deep neural networks (Bulat & Tzimiropoulos, 2017) and MTCNN (Zhang et al., 2016).

The work in (Kazemi & Sullivan, 2014) uses a cascade of regressors, specifically an ensemble of regressive boost trees to estimate face landmark of 194 points (Figure 8). Beforehand, regressive boost trees is a machine learning algorithm that combines decision trees (Friedman, 1999) for regression with the gradient boosting algorithm. Decision Trees is a machine learning model that builds a path based on the variables' values split in order to obtain a decision, originating a tree structure. The decision tree model (Figure 22) splits variables' values in ranges originating branches. Each node represents a variable and there is only one node per variable, thus the edges (branches) of a node represent a variable's values split. The leaf nodes represent a final decision, in the case of regressive trees a continuous value. Gradient Boosting (Friedman, 1999) is a machine learning technique that uses an ensemble of weak learners to create a strong one. Since the technique uses boosting, it means that miss-classified observations at stage *t* are most likely to be used as input at stage *t+1*. The model ensemble is organized stage-wise, where the model at stage *t* fits the residuals (called *pseudo-residuals*) of model *t-1*. This way it has been shown that this process is equivalent to computing the gradient of the loss function at each step (Friedman, 1999). Thus, gradient boost trees combines and ensemble of regressive trees with gradient boosting. The regressive boost trees landmarks estimator (Kazemi & Sullivan, 2014) trains directly on the pixel intensities, meaning that no feature extraction is needed. The system is capable of detecting the facial landmarks with very high quality in real-time, in fact it only takes 1 millisecond to align and detect the facial features (Kazemi & Sullivan, 2014). The system trains on the popular HELEN dataset (Le et al., 2012) that evaluates 194 points landmarks. The model is trained by minimizing the square error between the predicted landmarks and the ground truth. Let $x_i \in R^2$ define the landmarks' coordinates, $S = (x_i^T, \dots, x_p^T) \in R^{2p}$ represent the landmarks of face I and r_t be the regressor of stage t. This way the predicted landmarks $\hat{S}^{(t)}$ at stage t can be defined as:

$$\hat{S}^{(t+1)} = \hat{S}^{(t)} + r_t\big(I, \hat{S}^{(t)}\big) \tag{10}$$

Function 10 yields a clear special case for $t = 0$. This way r_0 must be initialized. Let n be the number of images, R be the number of initializations per image I_i and $N = nR$, thus for $i = 1, \dots, N$ r_0 can be initialized as follows:

$$\pi_i = \{1, \dots n\} \tag{11}$$

$$\hat{S}^{(0)} \in \{S_1, \dots, S_n\} \backslash S_{\pi_i} \tag{12}$$

$$\Delta S_{\boldsymbol{i}}^{(0)} = S_{\pi_i} - \hat{S}_{\boldsymbol{i}}^{(0)} \tag{13}$$

This way, for $t > 0$ the residuals of regressor r_t are defined as:

$$\Delta S_{\boldsymbol{i}}^{(t+1)} = S_{\pi_i} - \hat{S}_{\boldsymbol{i}}^{(t+1)} \tag{14}$$

Equations 10 to 14 are applied iteratively for the cascade *T* regressors ($r_0,\ldots,r_{T-1}$) using gradient boosting, minimizing the squared error:

$$L\left(I, \Delta\hat{S}_{\boldsymbol{i}}^{(t)}\right) = min \sum_{i=1}^{N} \left\| \Delta S_{\boldsymbol{i}}^{(t)} - \gamma \right\|^2, \ \gamma \in R^{2p} \qquad (15)$$

The trees' splits are chosen, greedily, from a set of randomly generated splits that minimize the sum of square error.

Face Alignment Network (FAN) (Bulat & Tzimiropoulos, 2017) uses deep neural networks to estimate 2D 68 points face landmarks and 3D face landmarks. This method achieved extraordinary results. The work presents three independent networks based on *Hour Glass* (HG) network architecture (Carreira et al., 2016). A network learns 2D face landmarks, another one learns 3D face landmark and finally there's a network that learns how to map 2D to 3D face landmarks. Since the networks are independent it's possible to align a face based on 2D or 3D landmarks.

Multi Task Convolutional Neural Networks (MTCNN) (Zhang et al., 2016) performs face localization and landmarks estimation using a cascade of CNNs. As previously mentioned, the cascade consists of three CNNs: Proposal Net (P-Net), Refine Net (R-Net) and Output Net (O-Net), in this particular order. MTCNN outputs a face's bounding box and five points landmarks – the center of each eye, nose tip and mouth left and right corners. The alignment can be later performed through affine transformation.

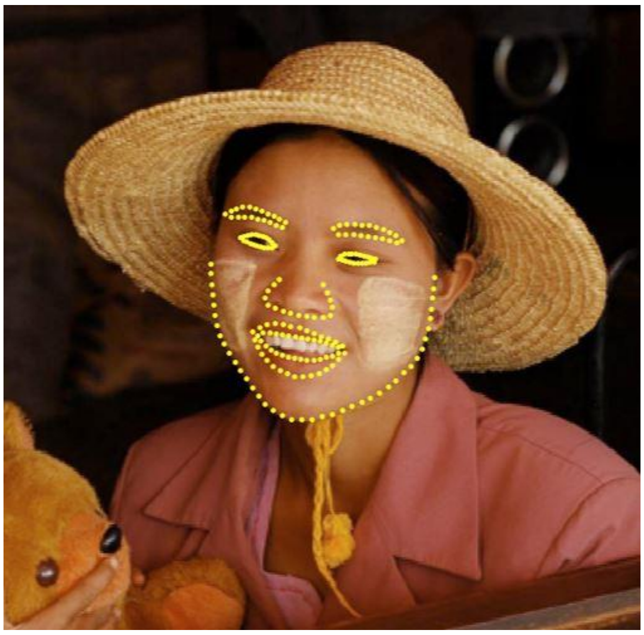


Figure 8 - HELEN face image example with respective 194 face landmarks (Le et al., 2012).

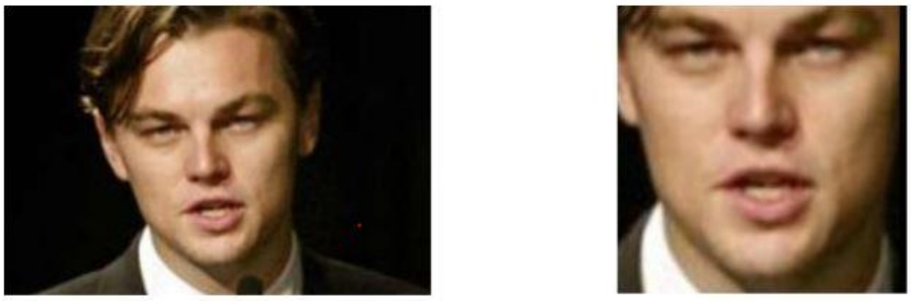


Figure 9 - Face Alignment (Murray, 2017).

2.5 Face Representation

To study a face, the computer must represent the pixels as a feature vector. The representation must capture the faces' main details so that different subjects can be distinguished. The most recent state-of-the-art methods refer to deep learning methods (Deng et al., 2018; Schroff et al., 2015; Wen et al., 2016). These methods perform face embedding, by training the network on a “fake task”. This task works only for the purpose of training the network, since the last layer, the output layer, is removed in order to obtain the embedding of a face, i.e. feature map. Thus the weights of the network act as a look-up table for the embedding map. There have been notable works in this subject: DeepFace (Taigman et al., 2014), DeepID3/2 (Sun et al., 2015; Sun et al., 2014), FaceNet (Schroff et al., 2015), Center Loss with Softmax (Wen et al., 2016) and ArcFace (Deng et al., 2018).

DeepFace (Taigman et al., 2014), developed by Facebook, was the first work to achieve human-level performance, slightly worse, in face recognition. The work proposed a deep learning end-to-end framework that performed 3D alignment followed by embedding. Finally, the face's feature vector is fed into a k-way softmax for classifying the label of the face image. The network was trained on four million Facebook images.

FaceNet (Schroff et al., 2015), developed by Google, proposed a novel loss function and training method for face recognition by training the network with a triplet loss-function based on LMNN (Weinberger & Saul, 2009). The work studies different CNN architectures deeply and concludes that ZF-Net (Zeiler & Fergus, 2014) performs best. The model was trained with 200 million images from a private dataset and achieved 99.63±0.09% accuracy score on LFW. The motivation for the loss function is that faces of the same subject are close and of different subjects are distant. This way the distance between the anchor image and a positive image, image that represents the same subject of the anchor image, is small; and the distance to a negative image is large. Moreover, images must comply with constraint displayed on equation 16 that states that the distance between positive pairs plus a margin, α, is lesser than the distance between negative pairs. The loss-function can be seen in a more intuitive way in equation 17. Here x_i^a is the anchor image, x_i^p the positive image, x_i^n the negative image and Φ the set of images.

$$\left\|f(x_i^a) - f\left(x_i^p\right)\right\|_2^2 + \alpha < \|f(x_i^a) - f(x_i^n)\|_2^2, \; \forall\, x_i^a, x_i^p, x_i^n \in \Phi \quad (16)$$

$$\sum_i^N \left(\left\|f(x_i^a) - f\left(x_i^p\right)\right\|_2^2 - \|f(x_i^a) - f(x_i^n)\|_2^2 + \alpha\right) \quad (17)$$

The triplet-loss function has a very discriminative power and it still represents the state-of-the-art in face recognition (Chen et al., 2018; Deng et al., 2018; Wen et al., 2016). Yet, it is very hard to train a network based on this loss function due to the hard positives and hard negatives search and to its slow convergence (Deng et al., 2018).

More recently, some promising methods that can surpass FaceNet in some challenges with only a fraction of the data have been proposed (Deng et al., 2018; Wen et al., 2016).

The work in (Wen et al., 2016) proposes an end-to-end learning framework with a novel loss function named center loss. This loss function aims to reduce the distance of intra-class feature vectors. A feature vector, with the same dimension as the face feature vector (deep feature), is learnt and updated along the training process for each class center while minimizing the distance of the deep features to the respective class centers. To discriminate the features from different classes the center loss is combined with softmax. The joint supervision of the two function results in an enlargement of the distances of features from different classes while reducing the distances of features from the same class. The proposed model achieved 99.28% accuracy in LFW using only 0.7 million images to train the network, outperforming DeepFace and getting really close to FaceNet on this challenge with only a fraction of the training data. Consider L_C as the center loss, L_S as the softmax loss function and L the joint supervision of the two functions. Omitting the layer superscript from notation let c_{y_i} be the feature vector of the y_i^{th} class center, x_i the i^{th} face feature vector, b the bias term, W_j the weights of the j^{th} neuron and a scalar λ that is used to balance the two loss functions. The formulation is expressed by the equations 18 to 19.

$$L_C = \frac{1}{2}\sum_{i=1}^{m}\left\|x_i - c_{y_i}\right\|_2^2 \qquad (18)$$

$$L_S = -\sum_{i=1}^{m} log\,\frac{e^{W_{y_i}^T x_i + b_{y_i}}}{\sum_{j=1}^{n} e^{W_j^T x_j + b_j}} \qquad (19)$$

$$L = L_S - \frac{\lambda}{2} L_C \tag{20}$$

Very recently DeepInSight deployed a model, ArcFace, that achieved state-of-the-art performance in face recognition obtaining an accuracy of 99.83% on the LFW benchmark (J. Deng et al., 2018). ArcFace separates faces based on arc/angular margin and is largely inspired by the works of (Liu et al., 2017) and (Wang et al., 2018) that use multiplicative angular margin and additive cosine margin respectively. The work proposed by (W. Liu et al., 2017), SphereFace, introduces a parameter *m* that controls the angular margin. The multiplicative angular margin is combined with softmax to discriminate different classes. Softmax supervision is integrated in training to guarantee convergence, turning the model somewhat tricky to train. CosineFace (Wang et al., 2018) is based on additive cosine margin and also separates classes using softmax. This work is easy to implement, has no need for softmax supervision and outperforms SphereFace. Although both works seem very similar the decision boundaries of the models are quite different. The angular margin allows a more clear geometric interpretation (Deng et al., 2018). ArcFace is based on additive angular margin, combining the ideas from both of the previous works and uses a ResNet (He et al., 2015). The authors further enhance the model by introducing the triplet-loss function, evidencing the contribution of (Schroff et al., 2015). Using the notation of the joint supervision function 20, let s be the hypersphere's radius, *m* the angular margin and Θ_i the angle between W_i and x. This way, the ArcFace loss function can be expressed by equations 21 and 22:

$$L_{ArcFace} = \frac{-1}{m} \sum_{i=1}^{m} log \frac{e^{s\left(cos\left(\theta_{y_i}+m\right)\right)}}{e^{s\left(cos\left(\theta_{y_i}+m\right)\right)} + \sum_{j=1, j\neq y_i}^{n} e^{s\left(cos\theta_j\right)}} \tag{21}$$

$$subject\ to: W_j = \frac{W_j}{\|W_j\|}, x_i = \frac{x_i}{\|x_i\|}, cos\theta_j = W_j^T x_i \tag{22}$$

The decision boundaries of the different methods and the respective illustrations are provided in table 1 and figure 10, receptively.

Table 1 - Decision boundaries of different functions on a binary classification example, where Θ_i is the angle between W_i and x, m is the margin and s is the hypersphere radius (Deng et al., 2018).

Loss Functions	**Decision Boundaries**
Softmax	$(W_1 - W_2)x + b_1 - b_2 = 0$
SphereFace (W. Liu et al., 2017)	$\|x\|(cosm\Theta_1 - cos\Theta_2) = 0$
CosineFace (Wang et al., 2018)	$s(cos\Theta_1 - m - cos\Theta_2) = 0$
ArcFace (J. Deng et al., 2018)	$s(cos(\Theta_1 + m) - cos\Theta_2) = 0$

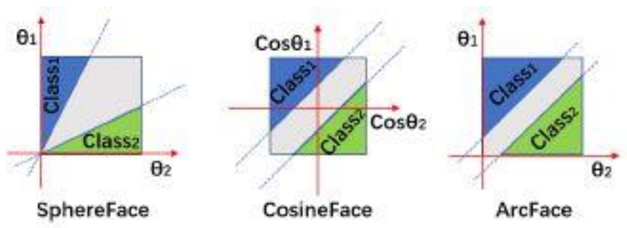


Figure 10 - Decision boundaries for SphereFace, CosineFace and ArcFace (Deng et al., 2018).

Although the accuracy of face recognition models has been evolving, there hasn't been a great effort in enhancing the networks efficiency with respect to speed and size (Howard et al., 2017). Yet, different systems have been proposed using the presented loss functions achieving very good results (Chen et al., 2018; Zhong et al., 2017), combining the shown loss function with more recent network architectures and training methods that require lesser data and achieve the same level of performance as the original proposed models. A very recent work, MobileFaceNets (Chen et al., 2018), combines the ArcFace loss function (Deng et al., 2018) with a network architecture inspired on MobileNetV2 (Sandler et al., 2018). The model trains on the VGG2 dataset (Cao et al., 2018), has less than a million parameters, occupies 4MB of memory space, is two times faster than MobileNetV2, has an inference time of 24 milliseconds and achieves an accuracy score of 99.55% on LFW.

Table 2 shows the size, training data and LFW accuracy score of different face recognition models. It further confirms that state-of-the-art models do not concern the networks' size. Moreover, MobileFaceNet can outperform ArcFace when the models are trained using the same dataset, VGG2 (Cao et al., 2018), evidencing the impact of the network architecture on the face recognition task.

Table 2 - Models and respective training data size and LFW accuracy score. *Number of Models* refers to the number of models used in the ensemble. [1] Private dataset. [2] Web-Face (Yi et al., 2014). [3] VGG2 (Cao et al., 2018). [4] MS-CELEB-1M (Guo et al., 2016).

Methods	**Model Size**	**#Models**	**Dataset Size**	**LFW Acc.**
DeepFace (Taigman et al., 2014)	-	3	4M[1]	97.35%
FaceNet (Schroff et al., 2015)	30MB	1	200M[1]	99.63%
Softmax + CenterLoss (Wen et al., 2016)	250MB	1	>0.4M[2]	99.28%
SphereFace (W. Liu et al., 2017)	-	1	>0.4M[2]	99.42%
ArcFace (J. Deng et al., 2018)	112MB	1	>13M[3,4]	99.83%
ArcFace (J. Deng et al., 2018)	112MB	1	3.8M[4]	99.50%
MobileFaceNet (S. Chen et al., 2018)	4.0MB	1	3.8M[4]	99.55%

Face Comparison

In face recognition face comparison can be divided in distance based and classification based methods (Deng et al., 2018; Schroff et al., 2015; Taigman et al., 2014; Wen et al., 2016). The image comparison stage must consider the feature representation, since they may not be compatible (Barz & Denzler, 2017).

In (Schroff et al., 2015) a nearest neighbor search approach is followed by learning a distance threshold that can maximize the face verification task.

In (Devi & Hemachandran, 2016) the authors use a hybrid method, combining CBIR with a SVM binary classifier. The system collects the closest images to the input face and then performs a binary classification on a one-vs-all approach, comparing each one of the faces returned by the CBIR system with the input image, outputting if two faces represent or not the same subject.

In (Deng et al., 2018; Taigman et al., 2014; Wen et al., 2016) a classification based approach is taken by training the networks with a softmax loss function or by feeding the feature representation into a k-fold softmax function.

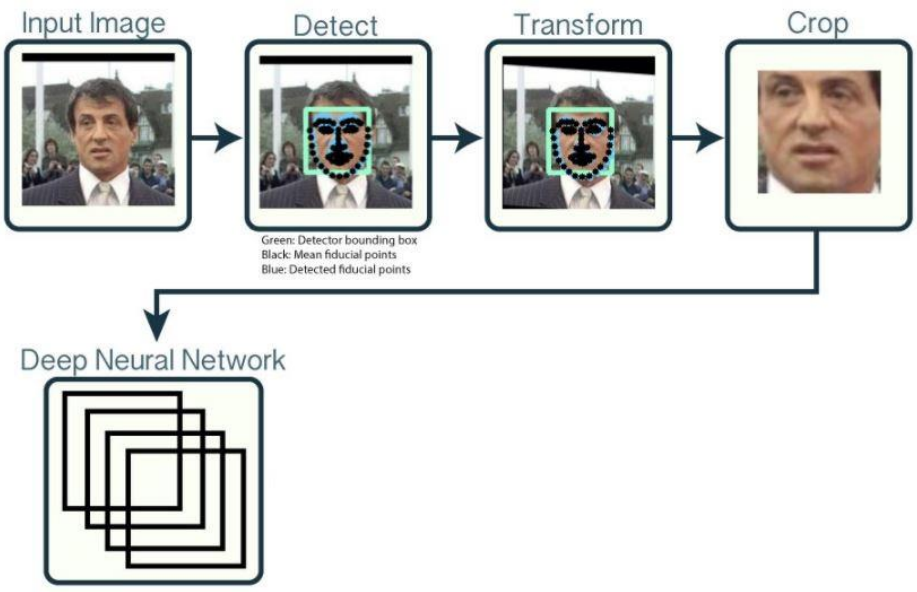

Input Image
Detect
Transform
Crop
Green: Detector bounding box
Black: Mean fiducial points
Blue: Detected fiducial points
Deep Neural Network

Figure 11 - Face embedding workflow (Amos, 2016).

3 Datasets

When developing a face recognition model one can choose from a variety of databases in which to train and test the model. Multiple datasets from several different sources exist. There are unconstrained images, colored or grey scaled, funneled, aligned faces and more. Although there is a vast number of datasets, this work will focus on those that have been the standard in face recognition works and are publicly available.

3.1 VGGFace 1/2

The VGGFace dataset (Parkhi et al., 2015), from VGG (Visual Geometry Group), is a large scale dataset built of 2.6M images of 2.6K celebrities, 375 images per person. The data was collected combining automation and human in a loop. The motivation for the construction of this dataset is the importance of large quantities of training data in face recognition models (Parkhi et al., 2015). Moreover, the group is moved by the access that large companies have to enormous quantities of private data, such as Google; allowing the company to train FaceNet on 200M images of 8M identities. This way, VGG contributed with a large scale, high quality face image dataset that further allowed the group to train a state-of-the-art face recognition model (Parkhi et al., 2015).

More recently, VGG released a larger dataset, VGGFace2 (Cao et al., 2018). VGGFace2 contains 3.31M images of about 9K subjects. The images were downloaded from Google Image and have large variation in pose, ethnicity, illumination and profession. VGGFace2 delivers a dataset with a large number of identities and images per identity (average 362.6 images per person), large range of illumination, pose and ethnicity and low label noise. The work in (Cao et al., 2018) evidences the quality of the VGGFace2 by showing the enhancement of the studied models when trained on VGGFace2. Although it is a very recent dataset is has gained some notorious popularity, being used in a variety of projects (Grewal, 2018).

3.2 CASIA-WebFace

CASIA-WebFace (Yi et al., 2014) is a public dataset, commonly referred as WebFace, built of almost 500k images of about 10k subjects. Although it is a small dataset when compared to MS-Celeb-1M (Guo et al., 2016) or MF1/2 (Kemelmacher-shlizerman, Seitz, Miller, & Brossard, 2016; Nech & Kemelmacher-shlizerman, 2016) it is possible to train state-of-the-art models using its data (W. Liu et al., 2017; Wen et al., 2016; Yi et al., 2014).

3.3 Labeled Faces in the Wild

The Labeled Faces in the Wild (LFW) database (Huang et al., 2007) is the standard dataset for unconstrained face verification (Deng et al., 2017). It contains face photographs designed for studying the problem of unconstrained face recognition. The dataset contains 13233 images of faces collected from the web. There are 5749 different identities. Each face has been labeled with the name of the person pictured. A total of 1068 of the people pictured have two or more distinct photos in the dataset. The only constraint on these faces is that they were detected by the Viola-Jones face detector. The dataset is highly unbalanced with respect to sex, since it contains more than 10000 images of males and less than 3000 images of females.
There are four different sets of LFW images including the original and three different types of aligned images. The aligned images include funneled images (ICCV 2007), LFW-a, which uses an unpublished method of alignment, and deep funneled images (NIPS 2012). Among these, LFW-a and the deep funneled images produce superior results for most face verification algorithms over the original images and over the funneled images (ICCV 2007). Moreover, the LFW database already contains pairs of matched and mismatched images for train and test, see figure 23.

3.4 Fine-grained Labeled Faces in the Wild

Fine-grained LFW (FGLFW) database is a renovation of LFW (Deng et al., 2017). The FGLFW database was developed by that fact that almost all the negative face pairs of LFW are quite easy to distinguish (Deng et al., 2017). The negative pairs are randomly selected from different individuals, and it is common that two random individuals have large differences in appearance. Many face pairs even have different genders. These differences in appearance can be seen easily in figure 23. Thus, verification is, by its nature a problem in which many examples are very easy with large inter-class variance. Since FGLFW only modifies the mismatched face samples from LFW, the images from the last are directly used. FGLFW offers a text file in the form of a 10-fold cross validation using splits founded by LFW. There are ten sets, every of which consists of three hundred matched pairs and three hundred mismatched pairs. The list is formatted as follows: ten sets are arranged in order. The differences in LFW and FGLFW mismatched pairs can be contemplated in figure 24. The accuracy of this dataset hasn't been verified neither its reliability by LFW's team.

LFW vs. FGLFW

Although incredible results have been achieved with state-of-the-art models for the standard dataset LFW, their performance drop about ten to twenty percent when performing face verification on the FGLFW database. These methods include state-of-the-art deep learning, metric-learning and face descriptors methods. It is contemplated in table 3 that the methods that deteriorate the most from LFW to FGLFW are deep learning based methods.

Table 3 - Verification accuracy (%) on datasets LFW and FGLFW under image unrestricted setting using labeled outside data (Zhong & Deng, 2017).

Method	**Training images**	**LFW**	**FGLFW**
DeepFace (Taigman et al., 2014)	0.5M	92.87%	78.78%
DeepID2 (Sun et al., 2014)	0.2M	95.00%	78.25%
VGG-Face (Simonyan & Zisserman, 2015)	2.6M	96.70%	85.78%
DCMN (W. Deng et al., 2017)	0.5M	98.03%	91.00%
Noisy Softmax (Chen et al., 2017)	0.5M	99.18%	94.50%
Human (W. Deng et al., 2017)	n/a	99.85%	92%

4 Technology Stack

Different technologies are used to solve the proposed face recognition problem.
The developed software uses the *Python* programming language (2018) version 3.6.2. This language is highly portable, powerful and fast. It is largely used by the machine learning and data mining community and has a wide number of libraries that permit efficient data manipulation and out-of-the-bag models.
Several projects have used this language and its libraries to address the face recognition problem effectively (Amos, 2016; Murray, 2017; Sandberg, 2018), showing that it is possible to achieve state-of-the-art results under the proposed conditions.

4.1 General Purpose Libraries

This work uses the following libraries. *NumPy* (2018) is a fundamental package for scientific computing with Python. It offers a powerful N-dimensional array object manipulation, linear algebra functions, among other things. *OpenCV* (2018) is a computer vision and machine learning library that contains thousands of optimized algorithms. It is available in multiple programming languages. *Dlib-ml* is a machine learning toolkit developed in C++ aimed for those who need to develop complex software and solve real world problems (King, 2009). Although it is targeted for C++ developers it is available in other programming languages. This library offers a great number of machine learning and optimization algorithms and optimized linear algebra operations. *Scikit-learn* is a python library that disposes out-of-the-box machine learning models, like SVM, and a set of utility functions, including pre-processing methods, evaluation metrics, etc (Pedregosa et al., 2010).

4.2 Deep Learning Libraries

TensorFlow is a library for numerical computation using data flow graphs (Abadi et al., 2016). It has a flexible architecture that allows one to deploy computations to the CPU or GPU. It was developed by the Google Brain Team for the purpose of conducting machine learning and deep neural networks research. *Keras* is a high level *python* API that uses *TensorFlow*, *CNTK* or *Theano* as a backend. It offers an intuitive interface for development and training of deep neural networks while providing a set of data modeling tools. It was developed with a focus on enabling fast experimentations.

Table 4 - Used libraries and respective versions.

Libraries	**Versions**
Dlib	1.19
TensoFlow GPU	1.5.0
Numpy	1.14
OpenCV	3.4.0.12
CUDA	9.0
CuDNN	7
Keras	2.1.3
Scikit-learn	0.19.1

5 Experimental Results

The work uses the technologies presented in Section 4, to address the face recognition problem. The methods used during the experiments have been explained deeply in previous Sections.

The goal of the masters' thesis is to achieve high subject verification performance. But as it was presented throughout this thesis state-of-the-art methods use enormous amounts of data and require immense computational power. It has been shown that such methods take a long time to train, from several days to even months, making it infeasible for one to train a model from scratch in the context of a master thesis that must be developed in approximately six months. This way, the proposed work will use transfer learning, like pre-trained models and pre-computed weights.

Although the proposed goal is very challenging, recent works suggest that face recognition using transfer learning and open source libraries to achieve very accurate results, with accuracy larger than ninety percent, is possible in personal powerful machines.

Thereby, the objective of the master's thesis is to develop computer software capable of distinguishing different subjects with a high accuracy rate.

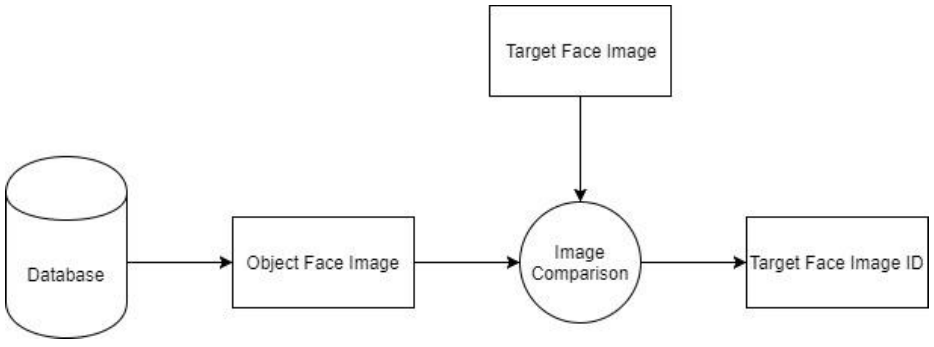


Figure 12 - General application diagram.

5.1 Method

The developed work focuses on face recognition. The main goal is to accurately identify the identity of a face image. The problem is challenging and requires one to solve a set of related problems. Yet, the work will mainly study the face representation and comparison problems. The problems can be divided into four sub-problems.

First, one must detect faces in the image. In this work an image with a single face will be fed into the system as input. It is intended to use the images of the LFW and FGLFW databases. Moreover, this task will also involve giving to the system input images with no faces. The complexity of the problem is reduced, since the problem of detecting multiple subjects in an image won't be addressed. The HoG with SVM face detector (Dalal & Triggs, 2004) and MTCNN (Zhang et al., 2016) were considered in this stage.

Second, the face must be aligned. The system must be invariant to pose and rotations of faces, in order to efficiently distinguish different subjects. Although LFW provides the necessary adjustments to produce thumbnails from the face images, this work intends to build a complete system that does not require anything more than a face image as input and a face images database with the respective labels. The system will rely on two models to localize a face's landmarks: ensemble of regressive boost trees (Kazemi & Sullivan, 2014) and MTCNN (Zhang et al., 2016).

Third, the system must learn the features of a face. For this purpose, a face encoding will be performed, more specifically a face embedding into a Euclidean space. Since it is infeasible to train a model from scratch, pre-trained models for the purpose of embedding the faces will be used. As previously discussed, FaceNet achieves state-of-the-art accuracy by embedding a face into a 128 dimension vector. The work also provides a way to verify faces by simply using nearest neighbour search with a distance threshold, thus the face representations with distances greater than the computed threshold are classified as not being the same subject. Thereby, the models FaceNet (Schroff et al., 2015) and dlib's CNN face encoder (King, 2017) are used for the purpose of face embedding.

Finally, the face representations must be compared, so that the subjects can be correctly identified. As contemplated on the previous step, one can use the approach adopted by FaceNet that has shown extraordinary results. Other techniques will also be considered like the one based on CBIR and SVM.
In the context Convolutional Neural Networks transfer learning may consist in the three major scenarios. CNN feature extraction consists in extracting the last fully connected layer from the network, then consider the rest of network as a fixed feature extractor. Another scenario would be fine-tuning of a network; this involves updating the weights of the networking by providing additional data. Finally, one can use the whole model, since training a model from scratch takes a large amount of time. It is intuitive to see that these scenarios can translate to any other model, where one can reuse, fine-tune or extract weights/coefficients. The guide provided by Stanford University for transfer learning will be taken into account (Karpathy, 2018).

Data

During the development and evaluation phases the LFW and FGLFW were considered as the main data sources. The LFW database was considered, since it is the *de facto* benchmark for face recognition and verification (Learned-Miller et al., 2016). Since it was shown that many algorithms have better performance on LFW's test set than on the FGLFW, the last one is also considered for performance comparison. The goal is to study the impact of the pairs on the systems.
The models are evaluated using 10-fold cross-validation provided by LFW. Each fold contains 300 positive pairs and 300 negative pairs, hence each folds contains 600 pairs, i.e. 1200 images. The pairs were obtained randomly.
FGLFW also provides a pre-computed 10-fold for cross-validation. Yet the pairs are carefully selected, so that each pair can be very similar, turning the verification task harder. As LFW's 10-fold, FGLFW's contains 300 positive and negative pairs per fold. The FGLFW set is more balanced than LFW considering subjects' sex. On average

LFW's cross-validation set has 3.9 times more male subjects than female subjects. Meanwhile on FGLFW the number of female subjects per fold is, on average, 2.4 times smaller than the number male subjects. Moreover, FGLFW considers 1.5 more identities per fold the LFW. This way FGLFW is a more balanced and diversified set than LFW.

Table 5 - LFW 10-fold cross-validation statistics.

LFW/Fold	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**Mean**
#females	286	332	325	298	292	330	316	292	312	274	305.7
#males	914	868	875	902	908	870	884	908	888	926	894.3
#identities	435	409	397	423	422	422	443	443	439	451	428.4

Table 6 - FGLFW 10-fold cross-validation statistics.

FGLFW/Fold	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**	**Mean**
#females	276	399	423	334	335	328	346	345	407	286	347.9
#males	924	801	777	866	865	872	854	855	793	914	852.1
#identities	293	275	265	279	270	272	311	270	285	290	281

In addition, different systems are evaluated on the test set. The test set consists of 500 positive and negative pairs, which is rather small. Although a training set is also provided, this was only used to train a SVM binary classifier. The training set contains 1100 positive and negative pairs, i.e. 2200 pairs. Hence the LFW dataset, when compared to other databases, like VGG2 or CASIA-WebFace, is indeed a very small. Furthermore, although LFW contains 13233 images, only a small fraction is considered for test and training. The training set contains little more than 1500 identities. Both training and test sets are very unbalanced with respect to the subjects' sex. On both sets the number of male face images is about three times greater than the number of female images, which is on the same level of disparity of the cross-validation set. The training set has 3.9 images per identity while test has 3.

Table 7 - LFW's training and test set statistics.

Dataset	#females	#males	#identities
Training	1150	3250	1519
Test	568	1549	696

Models

As previously explained in the beginning of this Section, the system will have four different stages, being stage three and four deeply considered. The data flow between each stage should be independent of the models used on each one, this way different approaches can be studied easily, simplifying the development and comparison of different methods. Most of the tested methods were previously trained by the referred authors, taking advantage of transfer-learning.

Face Detection

The first stage involves detecting faces in the input image. As anteriorly stated, the system will only consider a single face per image, if multiple faces are detected the largest one is considered, in order to diminish the problem complexity. Negative samples were also fed to the system, i.e. images without a face. The methods must be capable of stopping the process when a face isn't found; otherwise a cropped image, thumbnail, will be given as input to the next stage.

Several promising methods have been proposed. Since it is desirable to verify a subject identity in real-time, this work will consider the following methods. HoG with SVM face detector (Dalal & Triggs, 2004) and MTCNN (Zhang et al., 2016). There are multiple sources containing these models pre-trained freely on the web and in several packages. This phase won't be deeply considered since, surprisingly, it has been shown that variations in face localization bounding boxes doesn't affect the face recognition task significantly (Zhong et al., 2017). Moreover, the MTCNN model has shown to be a good model for localizing faces on the studied task (Zhong et al., 2017). Thus the HoG based face detector will solely be used as a baseline comparison.

The HoG with linear SVM was considered as a baseline detector. A pre-trained model was collected from Dlib. The training process is as follows. The model computes the HoG descriptor of 3000 LFW images. Each descriptor is later fed to a linear SVM that classifies it as person or non-person.

Also, a pre-trained MTCNN model is used. It is implemented in Tensorflow and is freely available (Sandberg, 2018), yet the training process isn't disclosed. There are better performing MTCNN models, but they are implemented in a different deep neural networks framework, Matlab/Caffe. Since the work in (Zhong et al., 2017) shows that the variation of the bounding boxes does not have a significant impact on the face recognition task, having the fittest bounding boxes isn't a main concern. Furthermore, the face encoder is also implemented in Tensorflow, which facilitates the workflow of the system.

Face Alignment

The second stage assures that the system is invariant to face pose, rotation and translation. Here the system gets a face thumbnail as input, derived from stage one. The system will transform each picture so that the face landmarks are always in the same place in the image. To do so a model for estimating the localization of the face landmarks and to center and project the face must be taken into account.

The face landmarks can be found using the method proposed in (Kazemi & Sullivan, 2014) that uses an ensemble of regression trees to estimate the face landmarks. The system is capable of estimating the face landmarks in one millisecond. Finally, the face can be centered. Another method was considered, MTCNN (Zhang et al., 2016), that outputs a face's bounding box and five points landmarks. Only 2D face alignment will be considered since 3D face alignment has revealed to not increase face recognition performance significantly (Banerjee et al., 2018).

Dlib offers a pre-trained facial features estimator, referred as Dlib_68, and aligner. The facial features are estimated using an ensemble of regressive boost trees. The model is trained similarly to the one presented on the original work (Kazemi & Sullivan, 2014), only differing on the training set; Dlib's model trains on the iBUG 300-W dataset (Sagonas et al., 2015). This way, the model outputs 68 points face landmarks as shown in figure 13 opposing to the 194 of the original model.

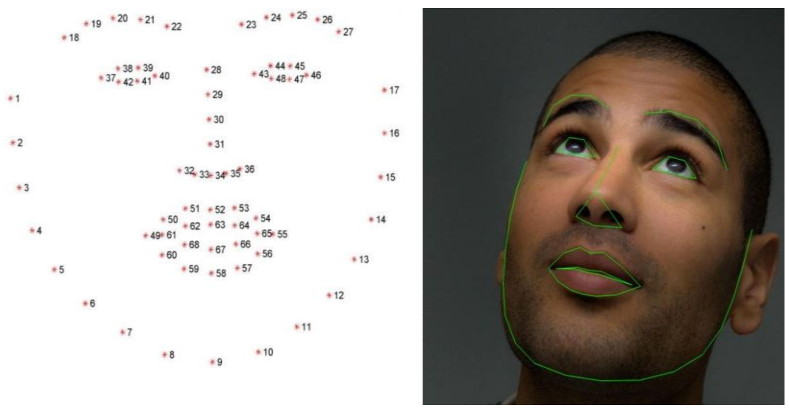


Figure 13 - 2D sixty-eight points face landmarks (Rosebrock, 2017).

As previously referred, the MTCNN model used doesn't disclose training details. This model has the advantage of outputting both the bounding boxes and landmarks of a face, excluding the need for a face landmarks estimator model, thus simplifying the process.

Face Representation

In this stage the system has already transformed the image completely. The input is now an aligned face that will serve as input to the model that will learn a new representation for the face images. The learned representation must be able to capture unique face features that will permit to distinguish the different subjects.

Incredible methods, close to human capabilities (Taigman et al., 2014), some surpassing them (Deng et al., 2018; Schroff et al., 2015; Wen et al., 2016), have been developed. The scores obtained on the dataset Unconstrained LFW are the standard for face verification. As presented before, these methods take a lot of time to train making it impossible for one to train a model from scratch on a personal machine. For this reason, transfer learning will be used. Several pre-trained networks trained using different loss function are freely available on the web (Amos et al., 2016; Sandberg, 2018).

Since FaceNet still represents state-of-the-art in face recognition (Deng et al., 2018; Wen et al., 2016) the pre-computed weights of this model are used. The work of (Sandberg, 2018), offers a set of pre-trained networks trained using the triplet-loss. These models are trained on the VGGFace2 and CASIA-WebFace datasets using Tensorflow, and will be referred as CNN_VGG2 and CNN_CASIA respectively. This way, the impact of the training data can be evaluated at test time. Both of the models use the Inception-ResNet-V1 architecture (Szegedy et al., 2016) and use the same training procedure, thus they only differ on the data used for training. Both of the models were trained on an Nvidia Pascal Titan X GPU, Tensorflow r1.7, CUDA 8.0 and CuDNN 6.0. They use a learning rate of 0.05. To avoid overfitting, Dropout (Srivastava et al., 2014) is used, which consist in deactivating, with a certain probability, each neuron of the network; a Dropout of 0.4 is used. Moreover a data augmentation technique referred as random flip is included. This technique flips an image horizontally with a certain probability, commonly 0.5. Hence a random flip with 50% probability is applied to the training data. Finally, for optimization the Adam algorithm was selected (Kingma & Ba, 2014). Using this training conditions, CNN_VGG took 30 hours to train, CNN_CASIA training times isn't disclosed.

Dlib's face encoder (King, 2017) is also considered for testing, referred as CNN_Dlib. This model is a ResNet network (He et al., 2015) with twenty nine convolutional layers. The network was trained on the VGGFace dataset (Parkhi et al., 2015), FaceScrub dataset (Ng & Winkler, 2014) and scraped images by the author. The model is implemented using Dlib's neural network framework. The loss function tries to project the identities into non-overlapping spheres of radius 0.6 and includes hard-negative and hard-positive mining; unfortunately the loss function isn't explicitly mentioned neither other training specifications.

Table 8 - The used embedding models and respective information.

Model	Network Architecture	Size (MB)	Loading Time	Training Data	Loss function	Embedding Size
CNN_Dlib	ResNet	21.4	1.256	VGG, FaceScrub, Scrapped data	-	128
CNN_CASIA	Incepition-ResNet-V1	93.5	19	CASIA-WebFace	Triplet-loss	512
CNN_VGG2	Incepition-ResNet-V1	91.3	21	VGG2	Triplet-loss	512

Subject Identification

Finally, after obtaining the face embedding representation the system must now correctly identify the subject face. Each face image in the database will be labeled with the respective identity. Thus the number of labels is equal to number of subjects.

There are numerous approaches for face recognition, including nearest neighbor search (Schroff et al., 2015), CBIR with SVM (Devi & Hemachandran, 2016) and neural networks (Deng et al., 2018; Taigman et al., 2014; Wen et al., 2016). FaceNet uses a nearest neighbor approach by introducing a distance threshold of 1.242, which is obtained by detecting the optimal split between subjects. Different thresholds can be computed, since it depends on the data. The problem can also be addressed as a multi-class classification problem by using a SVM for subject classification. The work in (Devi & Hemachandran, 2016) uses a hybrid approach, based on CBIR and SVM. On a first phase the system collects a set of face images that are close to the input face image with a nearest neighbor approach, thus using CBIR. Finally, the system uses a SVM that makes a binary classification, outputting if two images represent the same subject or not. This way the system can reduce the computational cost of comparing the input image to the whole database, since it only compares the input image to the images returned by the CBIR system. This work will consider the threshold, SVM and CBIR with SVM approaches.

Evaluation Metrics

To evaluate the performance of the models different metrics must be taken into account. To estimate the models' efficiency and effectiveness the computational time used during the test phase and multiple classification metrics must be contemplated, using different model complexities and data sizes. It is intended to identify which of the stages is the system's bottleneck. For this reason, each stage must be evaluated individually.

Several metrics that are commonly present in object and face recognition literature will be considered to evaluate the models. These metrics are: Precision, Recall, ROC curves and accuracy (He et al., 2017; Redmon & Farhadi, 2016; Simonyan & Zisserman, 2015; Viola & Jones, 2001).

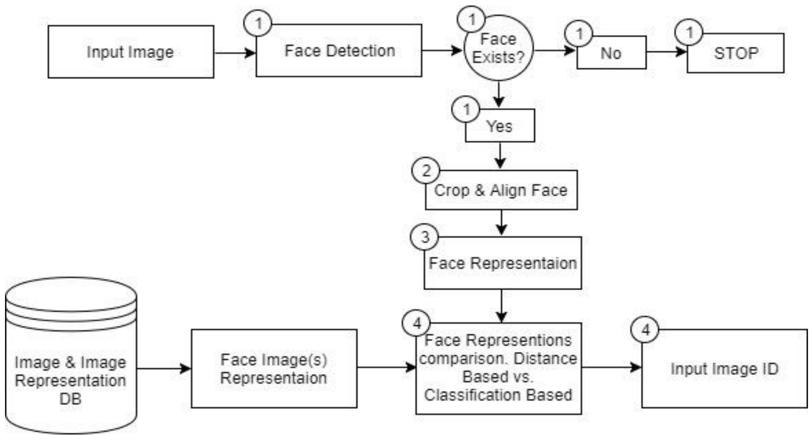

Input Image
1
Face Detection
1
Face Exists?
1
No
1
STOP
1
Yes
2
Crop & Align Face
3
Face Representaion
Image & Image Representation DB
Face Image(s) Representaion
4
Face Representions comparison. Distance Based vs. Classification Based
4
Input Image ID

Figure 14 - Detailed application diagram.

5.2 Experimental Results

All the obtained results were computed using the author’s personal machine, an ASUS ROG Strix GL553VW, running the Ubuntu 16.04 LTS operating system. The specifications of the machine can be contemplated in table 9.

Table 9 - Machine specifications.

Part	**Model**
CPU	Intel® Core™ i7-6700HQ @ 2.60GHz
GPU	NVIDIA® GeForce® GTX 960M, VRAM GDDR5 4GB 2500 MHz
Memory	DDR4 16GB (1066 MHz)
Disk	1TB HDD

The machine has a fairly nice computational power and can run the evaluated models in real time with no hassle. The computations, face localization, landmarks detection and face embedding, are computed by the GPU that has a computing capability of 5.0, according to NVIDIA’s evaluation benchmark.
The different stages of the face recognition process were evaluated separately; naturally the performance of a single stage depends on the models used in previous ones, excluding face localization.
Evaluating the computational cost of each model is straight forward and the Python library *time* was used for this purpose. Common input image sizes include: 96x96 (Schroff et al., 2015), about 128x128 (J. Deng et al., 2018; Schroff et al., 2015; Zhong et al., 2017) and 224x224 (Krizhevsky et al., 2012; Schroff et al., 2015). The system was tested on images of size 250x250, the original size of LFW’s images. Note that this is the size that is fed to the face localization models. The images are cropped, using the obtained bounding boxes, in order to produce a face thumbnail; hence the size will be smaller than the original one. The thumbnails are resized using bilinear interpolation to 180x180. Furthermore, since the performance of the complete system depends on its parts, the combination of multiple models will be studied as a whole and stage-wise.

Implementation Details

While developing the face recognition system flexibility was a primary concern, so that different model combinations could be tested easily. This way, an API was developed to abstract and facilitate testing. The implementation consists of multiple functions, one for each stage and common computations such as reading images from a database. The models used in each stage can be easily changed by simply modifying a function's argument, as other parameters such as the image size used. Hence the system is flexible and allows one to rapidly test different system combinations.

When testing the different models it was deeply desirable that repeated computations didn't occur, this includes: face localization, landmarks estimation and face alignment and face representation. As seen all the stages' computations, except the fourth, can be recycled. This way, the face thumbnails for both the HoG with SVM and MTCNN models were saved. Furthermore, the aligned faces produced by the regressive trees and MTCNN were also saved. Finally, considering MTCNN and regressive trees face aligners over the MTCNN thumbnails were embedded using the three proposed methods and later serialized using the *pickle* library. While classifying each pair the desirable embedding for each face image can be easily obtained by accessing the serialized object. Each embedding object is a *Python* dictionary, which is a Hash Table implementation. This way, an embedding of an image is indexed by its name that turns the retrieval of the face representation fast and easy. Hence, most of the computations are recycled.

Experimental Results Evaluation

Only two databases for the face recognition task are considered for this work - LFW and FGLFW. The system is evaluated as a whole. Nevertheless, the contribution of a stage's model for the face recognition task can be evaluated. This is done by changing the model used on a single stage. Hence the impact of a model for the face recognition task is addressed.

The time of each model is evaluated excluding intermediate computations between stages, such as image resizing. This way, the time of each model can be evaluated by simply introducing a timestamp at the beginning and ending of a model's computation, considering a function call time irrelevant.

Face Localization

In this stage two models were tested on the whole LFW dataset. Not surprisingly the HoG with SVM model turned out to be a bad choice to integrate the system. When localizing faces, this model failed to detect 58 faces, taking, on average, 0.0393s of inference time per image. In contrast MTCNN was able to detect all the faces and still be faster than the previous model, taking 0.0309s per image, on average. MTCNN considers an image pyramid and it consists of a cascade of three CNNs, so it is surprising that it can be faster than the other considered model. A possible reason for this behaviour relies on the fact that MTCNN runs on the GPU while HoG with SVM runs on the CPU. This way, MTCNN reveals to be a very appealing method. Even more, it has the ability of also outputting a face's features, removing the need for applying a model for face landmarks estimations, thus reducing computational time.

Table 10 – Face localization inference time per face and number failed faces on the LFW dataset.

Models	**Average Inference Time (s)**	**#Failed Faces**
HoG + SVM	0.0393	58
MTCNN	0.0309	0

Since MTCNN didn't fail any example and is faster than the other method, all the next stages solely use faces localized by the MTCNN model.

Face Landmarks Detection and Alignment

Dlib's regressive boost trees implementation for face landmarks estimation (Dlib_68) was applied to the face thumbnails produced by MTCNN. Just like MTCNN, Dlib_68 is able to estimate every face pose. Moreover, it reveals to be extremely fast, taking on average 0.0028s to estimate the 68 face landmarks per image.

Table 11 – Inference time per face and failed facial features on the LFW dataset.

Models	**Average Inference Time (s)**	**#Failed Faces**	**Localization + Landmarks Estimation Time (s)**
Dlib_68	0.0028	0	0.0338
MTCNN	-	0	0.0309

After detecting the face landmarks using Dlib_68, Dlib's face aligner was used to align the thumbnails. The aligner is very fast, since it only takes 0.0016s to align a thumbnail. Naturally, Dlib's face aligner cannot be used with MTCNN 5 points face landmarks, since it expects to receive facial features built of 68 points. Thus, the aligner was only tested using Dlib_68's estimations.

Table 12 - Inference time per face, number of failed face alignments and Stage 1 and 2 combined times on the LFW dataset.

Models	**Average Inference Time (s)**	**#Failed Faces**	**Stage1 + Stage2 Time (s)**
Dlib's Aligner	0.0016	0	0.0354
MTCNN	0.0309	0	0.0309

Face Representation

All three models were applied on MTCNN and Dlib aligned images, resulting in 6 different embedding sets. All three models use the GPU during inference time, yet

CNN_Dlib is not implemented using the same library as CNN_VGG and CNN_CASIA. They have yet another difference, CNN_Dlib outputs 128 feature vectors while CNN_VGG and CNN_CASIA output 512 feature vectors.

At inference time, all the models were all faster when computing the feature maps on the Dlib aligned thumbnails, 0.001s faster on average. This is a consistent behavior that might be caused by the way that the Dlib library generates its thumbnails. Yet, the impact isn't concerning. Furthermore, CNN_Dlib is considerably faster than the other models, about 2.5 times faster. CNN_CASIA is slightly faster than CNN_VGG, which surprising since the training conditions and architectures are exactly the same, excluding the training set. While CNN_Dlib uses a small ResNet arquitecture, CNN_CASIA and CNN_VGG use the original Inception-ResNet-V1 architecture which is indeed very deep. Thus it is natural for CNN_Dlib to be faster than the other models.

Table 13 - Embeddings' inference time and Stage 1, 2 and 3 combined times.

Models	**Stage 2 Models**	**Inference Time (s)**	**Stage 1 + Stage 2 + Stage 3 Time (s)**
CNN_Dlib	Dlib_68 + Dlib Aligner	0.0089	0.0443
CNN_Dlib	MTCNN	0.0091	0.0400
CNN_CASIA	Dlib_68 + Dlib Aligner	0.0227	0.0581
CNN_CASIA	MTCNN	0.0243	0.0552
CNN_VGG	Dlib_68 + Dlib Aligner	0.0235	0.0589
CNN_VGG	MTCNN	0.0247	0.0556

Subject Identification

For this stage different approaches were studied. This includes two different 10-fold cross-validation sets, LFW and FGLFW. The face recognition system is tested using distance thresholds and SVM classification. The last is only tested on LFW's test set due to training conditions. Still, the threshold approach is also conducted on the test set.

Firstly, subject identification using a threshold is addressed. Let $I_i \in M$, where M is the set of face images, and ϑ an embedding function such that $\vartheta(I_i) = v_i$, where $v_i \in R^d$ is the feature vector representation of I_i. Two face images represent the same identity if their distance e lesser or equal than a defined threshold. For this purpose, the Euclidean distance was employed. Hence, considering a threshold $th \geq 0$, the face verification task decision can be defined as follows:

$$d(v_i, v_j) = \left\| v_i - v_j \right\|_2^2, \ i,j = \ 1, \ldots, |M| \tag{23}$$

$$\begin{cases} d(v_i, v_j) \leq th, \ v_i \ and \ v_j \ represent \ the \ same \ identity \\ othewise, \ v_i \ and \ v_j \ don't represent \ the \ same \ identity \end{cases} \tag{24}$$

For CNN_VGG and CNN_CASIA the upper bound of the considered thresholds was based on FaceNet’s proposed thresholds: 1.1 and 1.242. Thus, the tested thresholds for these two models range from 0.7 to 2.0 with a 0.001 step, hence each test considered 1300 thresholds. Meanwhile, for CNN_Dlib, the thresholds’ values range should be upper bounded by 0.6, since the same identities lye in a ball of 0.6 radius. Yet, a 0.7 upper bound was considered and a lower bound of 0. Also, a 0.001 step was considered, thus 700 thresholds were considered per test. The best performing threshold, i.e. the threshold that has the largest mean accuracy over the 10-fold cross-validation set, is reported for each model.

Generally the best performing face encoder is CNN_VGG, followed by CNN_CASIA and finally CNN_Dlib on the LFW cross-validation set. CNN_Dlib performs very badly under Dlib’s face aligner. Dlib’s face alignment method outputs far worse results in face verification, CNN_Dlib’s accuracy increases 36 percent points, while the other methods increase 10 percent points when switching from Dlib’s face aligner to MTCNN’s. The standard deviation of the accuracy score suffers a great decrease from the Dlib to the MTCNN method, showing that the results of the last are more consistent. Since Dlib’s face aligner is clearly worse than MTCNN’s, the ROC analysis is performed using MTCNN face alignment only. All the models output large areas under the curves

(AUC), at least 0.97. Although the regressive boost trees face landmarks estimator and aligner seemed very promising the MTCNN face aligner achieved far better results, yielding a very large accuracy gap between the two. It is also interesting to notice that CNN_Dlib best performing threshold on LFW's cross-validation set under MTCNN's face aligner is greater than 0.6. This means that the model isn't able to separate well different identities, specifically the same identity may be in different hyperspheres, meaning that different identities are not separated by the system.

Table 14 - CNN face embeddings top performing threshold using Dlib_68 and Dlib's face aligner on LFW's 10-fold cross validation data.[1]Threshold.

Models	**Th.**[1]	**Acc. (Mean)**	**Acc. (Std)**	**Recall (Mean)**	**Recall (Std)**	**Precision (Mean)**	**Precision (Std)**
CNN_Dlib	0.589	0.604	0.015	0.527	0.018	0.624	0.022
CNN_CASIA	1.155	0.876	0.008	0.862	0.024	0.888	0.013
CNN_VGG	1.148	0.884	0.013	0.860	0.023	0.904	0.011

Table 15 - CNN face embeddings top performing threshold using MTCNN face alignment on LFW's 10-fold cross validation data. [1]Threshold.

Models	**Th.**[1]	**Acc. (Mean)**	**Acc. (Std)**	**Recall (Mean)**	**Recall (Std)**	**Precision (Mean)**	**Precision (Std)**
CNN_Dlib	0.634	0.966	0.009	0.960	0.015	0.970	0.007
CNN_CASIA	1.079	0.974	0.007	0.967	0.012	0.980	0.007
CNN_VGG	1.115	0.985	0.004	0.984	0.008	0.987	0.006

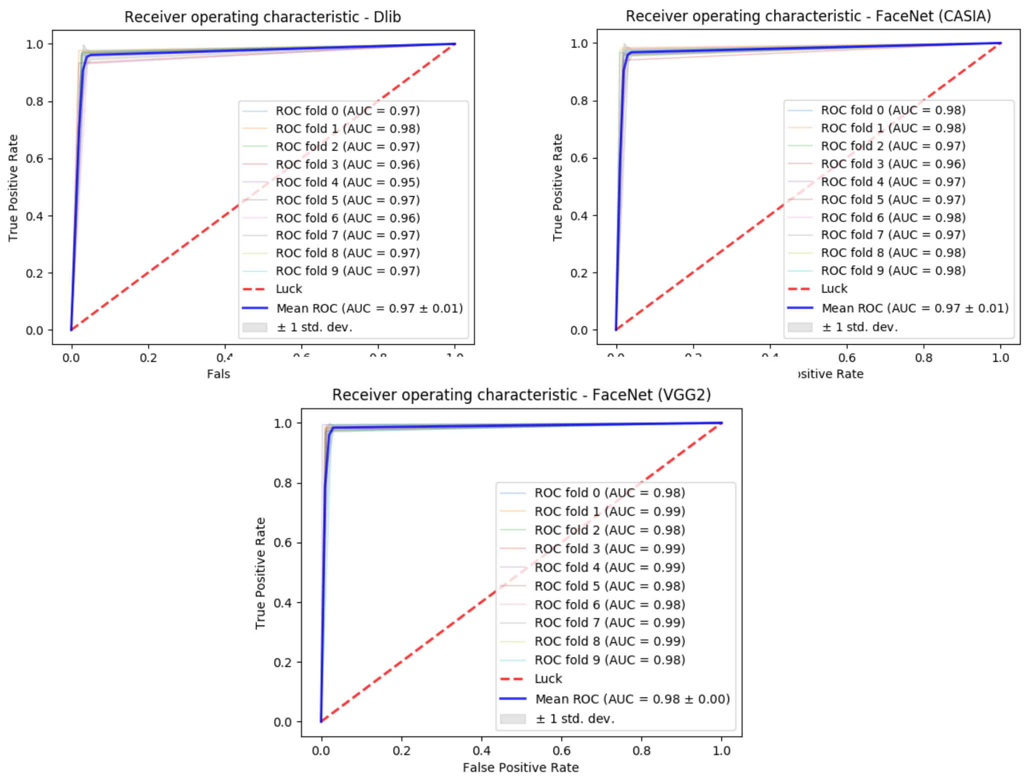


Figure 15 - LFW's 10-fold cross validation curves. Top left: CNN_Dlib ROC curves. Top right: CNN_CASIA ROC curves. Bottom center: CNN_VGG ROC Curves.

All the models' thresholds decrease from the LFW to the FGLFW dataset. This is expected since FGLFW's pairs are more similar than LFW's. Thus, the distance between FGLFW's pairs is smaller, therefore the threshold distance that is able to better separate a pair of identities decreases. Even though the CNN_Dlib has very low performance using Dlib_68 on LFW, the model is able to achieve a very similar performance to CNN_CASIA. CNN_Dlib is able to outperform all tested methods on FGLFW using Dlib_68 by 10 percent. This behavior is very surprising, since CNN_Dlib is the worst performing model in every test. It's also astonishing that CNN_Dlib performs better using Dlib_68 on FGLFW than LFW, except for this special case all the other methods perform worse on FGLFW. Moreover, the models show larger standard deviation accuracy on FGLFW than on LFW, indicating that the performance is more inconsistent. ROC analysis also outputs smaller AUCs.

Table 16 - CNN face embeddings top performing threshold using Dlib_68 and Dlib's face aligner on FGLFW's 10-fold cross validation data. [1]Threshold.

Models	**Th.**[1]	**Acc. (Mean)**	**Acc. (Std)**	**Recall (Mean)**	**Recall (Std)**	**Precision (Mean)**	**Precision (Std)**
CNN_Dlib	0.557	0.878	0.013	0.859	0.031	0.893	0.022
CNN_CASIA	1.065	0.762	0.018	0.725	0.022	0.783	0.025
CNN_VGG	0.997	0.755	0.018	0.662	0.022	0.813	0.024

Table 17 - CNN face embeddings top performing threshold using MTCNN face alignment on FGLFW's 10-fold cross validation data. [1]Threshold.

Models	**Th.**[1]	**Acc. (Mean)**	**Acc. (Std)**	**Recall (Mean)**	**Recall (Std)**	**Precision (Mean)**	**Precision (Std)**
CNN_Dlib	0.561	0.878	0.016	0.867	0.016	0.887	0.026
CNN_CASIA	0.988	0.897	0.012	0.895	0.011	0.898	0.021
CNN_VGG	0.970	0.912	0.012	0.913	0.013	0.911	0.019

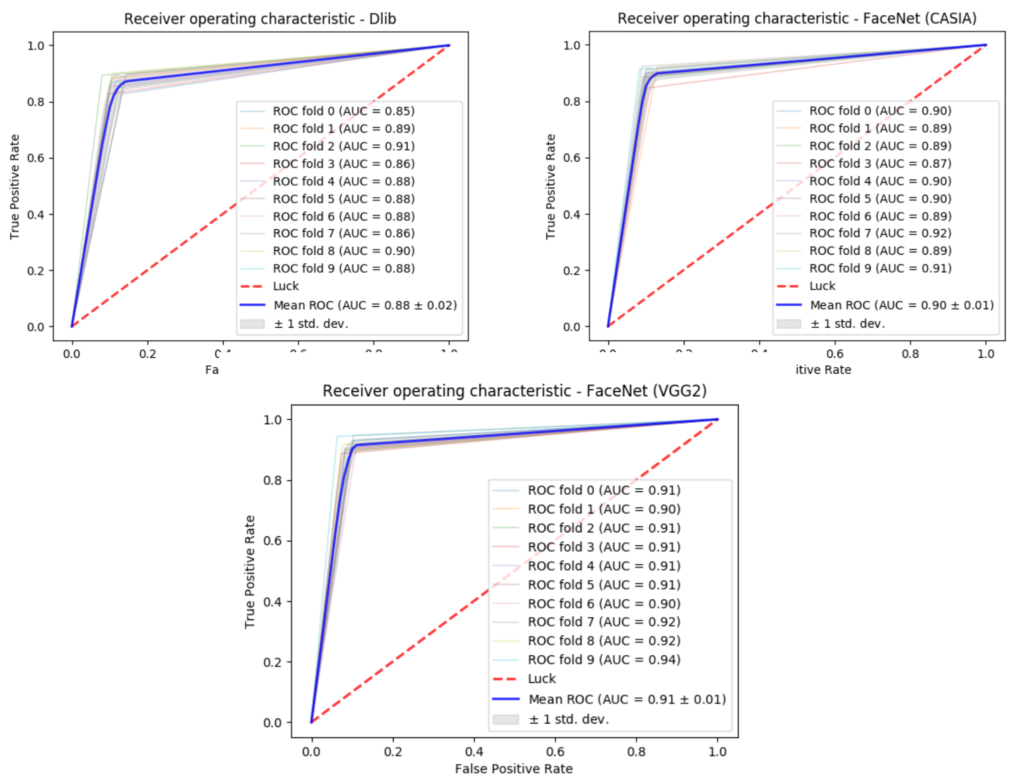


Figure 16 - FGLFW's 10-fold cross validation curves. Top left: CNN_Dlib ROC curves. Top right: CNN_CASIA ROC curves. Bottom center: CNN_VGG ROC Curves.

The best performing models on both LFW and FGLFW cross-validation sets were selected for testing. Thus a total of 6 models were considered, 3 from each set. All the tested models use MTCNN face aligner. Although CNN_Dlib has the same mean accuracy on FGLFW when using Dlib_68 or MTCNN, when using Dlib_68 it has lower standard deviation, -0.001. This difference is insignificant, yet for comparison purposes, CNN_Dlib with MTCNN face aligner is considered. The test set is provided by LFW and contains 1000 face image pairs. The FGLFW thresholds perform worse than LFW thresholds. Since this is a LFW test this is expected, since LFW pairs are more distant than the ones in FGLW. Hence, as contemplated the number of false negatives increases and the number of false positives decreases when using FGLFW's thresholds. This is a natural result triggered by the smaller thresholds.

Contemplating the results, it can be stated that CNN_VGG is the best performing model. CNN_VGG is able to outperform CNN_CASIA that is trained under the same conditions, showing the impact of training data at test time. Yet the difference in performance is very small, thus the following questions remain: Does triplet-loss has enough discriminative power? Is the quality of CASIA-WebFace superior to VGGFace2?
Although CNN_Dlib performs worse than the other models, it revealed very competitive results, while being two times faster.

Table 18 – LFW's best performing threshold score under the face recognition protocol.

Models	**Accuracy**	**#TP**	**#FP**	**#TN**	**#FN**
CNN_Dlib	0.953	479	26	474	21
CNN_CASIA	0.971	482	11	489	18
CNN_VGG	0.975	491	16	484	9

Table 19 – FGLFW's best performing threshold score under the face recognition protocol.

Models	**Accuracy**	**#TP**	**#FP**	**#TN**	**#FN**
CNN_Dlib	0.945	447	2	498	53
CNN_CASIA	0.944	444	0	500	56
CNN_VGG	0.963	463	0	500	56

Finally, regarding the test set pairs provided by LFW, a linear SVM binary classifier was trained. The SVM is implemented under *Scikit-learn*, turning the model easy to test and train. The SVMs are trained on LFW's training set. The SVM must train on the

pairs face representations, this way the SVM looks at two faces at once. The training and test sets were obtained by concatenating the face pairs' embeddings into one. The concatenated vectors are labelled as 0 and 1, different and same identity. Since the feature maps obtained from the MTCNN aligner by CNN_Dlib, CNN_CASIA and CNN_VGG were the best performing ones during cross-validation and test, they're included on the SVM classification task. Hence, three SVMs were trained. A stratified 10-fold cross-validation was generated with the training data, leaving 20% for validation.
The learning curves of the three SVM all have the same behaviour (Figure 17). Both validation and cross-validation scores converge to a very low value, approximately 50%, with the increasing size of the training set. Thus, there wouldn't be great benefits with more training data. This way, the learning curves suggest that the linear SVM isn't a good model for this task. Further parameterization of the model or even a different estimator could leave to a lower bias.

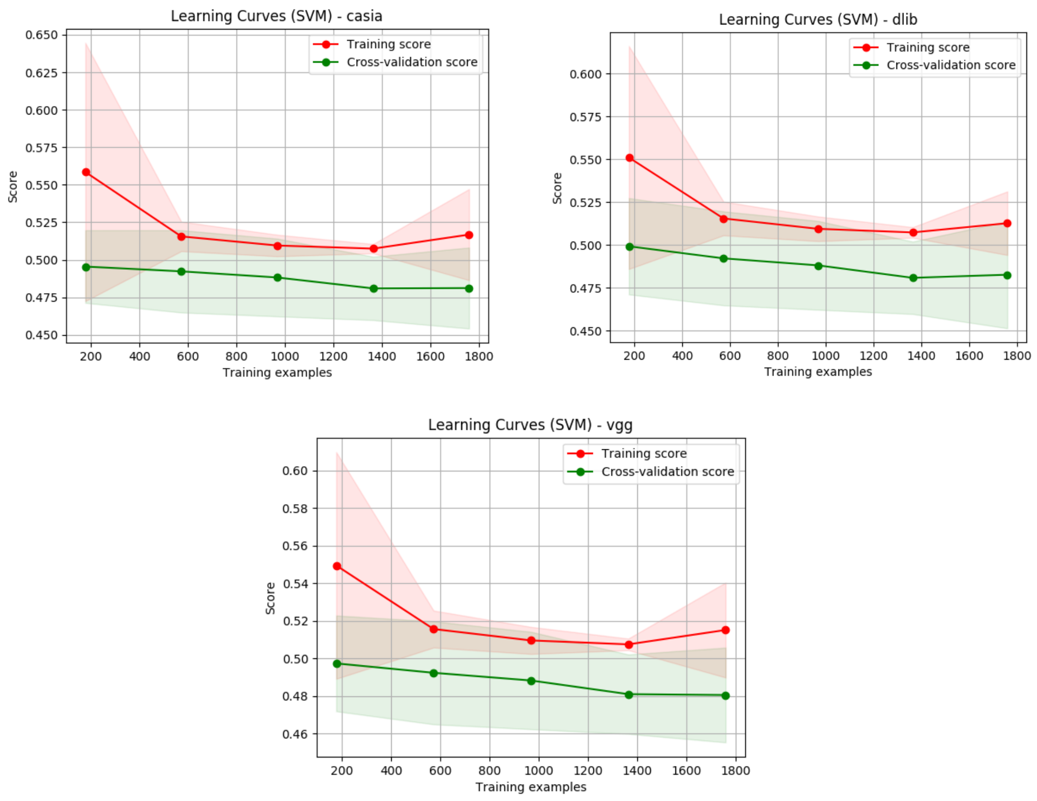


Figure 17 - SVMs' learning curve. Top left uses CNN_CASIA embeddings. Top right uses CNN_Dlib embeddings. Bottom center uses CNN_VGG embeddings. The lines in red represent the training score, while the green lines represent the cross-validation score.

The differences between face verification and recognitions were addressed. It is very intriguing how face recognition models are tested under face verification constraints (Deng et al., 2018; Schroff et al., 2015; Wen et al., 2016; Zhong et al., 2017). From the presented approaches only one performs real face recognition: the CBIR with SVM approach (Devi & Hemachandran, 2016). The work presents a face recognition pipeline based on CBIR. Given a face image to the system; it will retrieve a set of images based on a similarity measure (CBIR). Using a SVM the system performs face verification using the set of returned face images and the input image.

For the face recognition task none of the presented works provided a protocol to test a model under such conditions. For that reason, a simple test protocol was created. To test the face recognition system a complete database is considered, in this case the LFW dataset. The system will iterate over every image of the database once. For each image the system will try to search its identity in the database by comparing the input's embedding with all the other images' embedding, excluding the input image itself. Consider a database *D*, where image $I_i \in D$ and $v_i \in V$ be the embedding of face image I_i, where V is the set of embeddings of the face images in *D*. Consider a system S such that:

$$S(I_i) = argmin\left\|v_i - v_j\right\|_2^2, \ \forall I_i \in D, v_i \in V, \forall v_j \in V \backslash \{v_i\} \qquad (25)$$

Following the proposed protocol this work tests two different methods, threshold based and CBIR with SVM based. For this task only the face representations collected from CNN_VGG under MTCNN's face aligner were considered. For each system the number of false and true positive and negative predictions are presented. The LFW database has 4069 identities that have only one image sample; this way for these images' embeddings no identity should be returned. This means that there should be 4049 true negatives.

For the threshold based only, the CNN_VGG face encoder with the MTCNN face aligner system was tested, using the best performing thresholds for this system on LFW and FGLFW, 1.115 and 0.970. The system searches every embedding in the database in a naïve way, i.e. sequentially. After verifying every distance the system returns the vector that is closest to the input, like in equation 25. Finally, if the computed distance is lesser or equal than the defined threshold the input image is classified with the same identity as the returned vector.

Table 20, shows that the largest threshold is far too lose, since it has no false or true negative. This means that the system is always able to find a representation that is at a distance to the target face vector that is lesser or equal than the threshold. Hence 1.115 is a very large threshold for the face recognition task. The smallest threshold outputs better results and is able to detect negative pairs, yet the difference in accuracy is very small. The results show that using a shortest threshold may reduce the number of false positives. As expected searching times are very poor, once the systems searches the complete database, taking a fifth of second to return an identity.

Table 20 - Face recognition benchmarks using different thresholds. [1]Average search time per input image.

Threshold	**#TP**	**#FP**	**#TN**	**#FN**	**Accuracy**	**Search time[1] (s)**
1.115	8491	4742	0	0	0.642	0.209
0.970	8487	4645	97	4	0.649	0.209

Regarding the CBIR with SVM system, the goal of the SVM is to reduce the number of false positive and negatives. For that purpose the previously presented SVM trained on LFW under CNN_VGG embedding is used. No distance threshold was used, since the use of SVM is solely to study its impact as a performance enhancer. Thus the SVM classifies the vector that is at a minimum distance to the input vector. From previous results (Table 20) it is known that for every image embedding there's always another embedding of a different image that is at a maximum distance of 1.115. The results in table 21 show that using a SVM classifier results in a drop of accuracy of more than 10%, making it a poor choice for this task. Yet, this system performed very similarly on the face verification task in contrast to the threshold based system.

Table 21 - Face recognition benchmarks using CBIR with SVM. [1]Average search time per input image.

Model	**#TP**	**#FP**	**#TN**	**#FN**	**Accuracy**	**Search time[1] (s)**
CBIR + SVM	3926	1444	3298	4565	0.546	0.197

When comparing the results of the proposed face recognition protocol and LFW's face verification task a huge drop in accuracy is contemplated. The models' accuracy scores decrease about 30% using the same thresholds. The results show that face recognition needs a greater discriminative power than face verification.

6 Conclusion

This work addressed the problem of face recognition. The challenges of this task were exposed by analyzing each of its stages independently. Different methods were presented and the respective state-of-the-art models were explained. Multiple face recognition systems were tested, varying the face localization, landmarks detection, encoding and comparison algorithms. The tests reveal that it is possible to develop a very high performing system on standard face recognition benchmarks on a personal machine through transfer learning. Furthermore, the work shows the great difference in performance between face verification and recognition. For this purpose a test protocol was proposed. This way, the goal of the masters' thesis was achieved, since high levels of accuracy were achieved on face recognition benchmarks in real time.

References

Abadi, M., Agarwal, A., Barham, P., Brevdo, E., Chen, Z., Citro, C., … Zheng, X. (2016). TensorFlow: Large-Scale Machine Learning on Heterogeneous Distributed Systems. https://doi.org/10.1038/nn.3331

Adam Geitgey. (2016). Machine Learning is Fun! Part 4: Modern Face Recognition with Deep Learning. Retrieved June 15, 2018, from https://medium.com/@ageitgey/machine-learning-is-fun-part-4-modern-face-recognition-with-deep-learning-c3cffc121d78

Adrian Rosebrock. (2017). Facial landmarks with dlib, OpenCV, and Python - PyImageSearch. Retrieved June 15, 2018, from https://www.pyimagesearch.com/2017/04/03/facial-landmarks-dlib-opencv-python/

Amos, B., Ludwiczuk, B., & Satyanarayanan, M. (2016). OpenFace: A general-purpose face recognition library with mobile applications. *Technical Report CMU-CS, CMU School of Computer Science*, *16*(118). https://doi.org/10.5281/zenodo.32148

Andrej Karpathy. (2018). CS231n Convolutional Neural Networks for Visual Recognition. Retrieved June 15, 2018, from http://cs231n.github.io/transfer-learning/

Arriaga, O., Valdenegro-Toro, M., & Plöger, P. (2017). Real-time Convolutional Neural Networks for Emotion and Gender Classification. Retrieved from http://arxiv.org/abs/1710.07557

Arsenovic, M., Sladojevic, S., Anderla, A., & Stefanovic, D. (2017). FaceTime – Deep Learning Based Face Recognition Attendance System, 53–58.

Ballard, D. H. (1981). GENERALIZING THE HOUGH TRANSFORM TO DETECT ARBITRARY SHAPES *, *11*(2).

Banerjee, S., Brogan, J., Krizaj, J., Bharati, A., RichardWebster, B., Struc, V., … Scheirer, W. (2018). To Frontalize or Not To Frontalize: Do We Really Need

Elaborate Pre-processing To Improve Face Recognition? Retrieved from http://arxiv.org/abs/1610.04823

Barz, B., & Denzler, J. (2017). Automatic Query Image Disambiguation for Content-Based Image Retrieval.

Brandon Amos. (2016). OpenFace. Retrieved June 15, 2018, from http://cmusatyalab.github.io/openface/

Bulat, A., & Tzimiropoulos, G. (2017). How far are we from solving the 2D &amp; 3D Face Alignment problem? https://doi.org/10.1109/ICCV.2017.116

Cao, Q., Shen, L., Xie, W., Parkhi, O. M., & Zisserman, A. (2018). VGGFace2 : A dataset for recognising faces across pose and age.

Carreira, J., Agrawal, P., Fragkiadaki, K., & Malik, J. (2016). Stacked Hourglass Networks for Human Pose Estimation. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, *2016–Decem*, 4733–4742. https://doi.org/10.1109/CVPR.2016.512

Chen, B., Deng, W., & Du, J. (2017). Noisy softmax: Improving the generalization ability of DCNN via postponing the early softmax saturation. *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, *2017–Janua*, 4021–4030. https://doi.org/10.1109/CVPR.2017.428

Chen, S., Liu, Y., Gao, X., & Han, Z. (2018). MobileFaceNets : Efficient CNNs for Accurate Real-time Face Verification on Mobile Devices. https://doi.org/arXiv:1804.07573v2

Chollet, F. (2017). Xception: Deep learning with depthwise separable convolutions. *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, *2017–Janua*, 1800–1807. https://doi.org/10.1109/CVPR.2017.195

Chuang, G. C., Kuo, C. J., & Member, S. (1996). Wavelet Descriptor of Planar Curves Theory and Application, *5*(1).

Cole Murray. (2017). Building a Facial Recognition Pipeline with Deep Learning in

Tensorflow. Retrieved June 15, 2018, from https://hackernoon.com/building-a-facial-recognition-pipeline-with-deep-learning-in-tensorflow-66e7645015b8

Cosma, S., & Shalizi, C. R. (2006). Lecture 10 : Regression Trees. *October*, 1–7. Retrieved from papers2://publication/uuid/2182D360-C21A-406A-B060-E4C868399517%5Cnhttp://www.stat.cmu.edu/~cshalizi/350-2006/lecture-10.pdf

Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function. *Mathematics of Control, Signals, and Systems*, (2), 203–314. https://doi.org/10.1007/BF02836480

Dalal, N., & Triggs, W. (2004). Histograms of Oriented Gradients for Human Detection. *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition CVPR05*, *1*(3), 886–893. https://doi.org/10.1109/CVPR.2005.177

David Sandberg. (2018). GitHub - davidsandberg/facenet: Face recognition using Tensorflow. Retrieved June 15, 2018, from https://github.com/davidsandberg/facenet/

Deng, J., Guo, J., & Zafeiriou, S. (2018). ArcFace: Additive Angular Margin Loss for Deep Face Recognition. Retrieved from http://arxiv.org/abs/1801.07698

Deng, W., Hu, J., Zhang, N., Chen, B., & Guo, J. (2017). Fine-grained face verification: FGLFW database, baselines, and human-DCMN partnership. *Pattern Recognition*, *66*(July 2016), 63–73. https://doi.org/10.1016/j.patcog.2016.11.023

Devi, N. S., & Hemachandran, K. (2016). Retrieval and Recognition of faces using Content- Based Image Retrieval (CBIR) and Feature Combination method.

Friedman, J. H. (1999). Greedy Function Approximation: A Gradient Boosting Machine. https://doi.org/10.1214/aos/1013203451

Girshick, R., Donahue, J., Darrell, T., Malik, J., & Berkeley, U. C. (2012). Rich feature hierarchies for accurate object detection and semantic segmentation.

Guo, Y., Zhang, L., Hu, Y., He, X., & Gao, J. (2016). MS-Celeb-1M : A Dataset and Benchmark for Large-Scale Face Recognition.

Gupta, V. (2017). Image Classification using CNNs in Keras | Learn OpenCV. Retrieved June 17, 2018, from https://www.learnopencv.com/image-classification-using-convolutional-neural-networks-in-keras/

He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2017). Mask R-CNN.

He, K., Zhang, X., Ren, S., & Sun, J. (2015). Deep Residual Learning for Image Recognition. https://doi.org/10.1007/s11042-017-4440-4

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., … Adam, H. (2017). MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. https://doi.org/arXiv:1704.04861

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. *University of Massachusetts Amherst Technical Report*, *1*, 07-49. https://doi.org/10.1.1.122.8268

Huang, G., Liu, Z., Van Der Maaten, L., & Weinberger, K. Q. (2017). Densely connected convolutional networks. *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, *2017–Janua*, 2261–2269. https://doi.org/10.1109/CVPR.2017.243

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 <0.5MB model size, 1–13. https://doi.org/10.1007/978-3-319-24553-9

Jain, A. K., & Vailaya, A. (1995). Image Retrieval using Color and Shape 1 Introduction.

Jégou, H., & Zisserman, A. (2014). Triangulation embedding and democratic kernels for image search. *Cvpr*.

Jiang, H., & Learned-Miller, E. (2017). Face Detection with the Faster R-CNN. *Proceedings - 12th IEEE International Conference on Automatic Face and Gesture Recognition, FG 2017 - 1st International Workshop on Adaptive Shot Learning for Gesture Understanding and Production, ASL4GUP 2017, Biometrics* *in the Wild, Bwild 2017, Heteroge*, 650–657. https://doi.org/10.1109/FG.2017.82

Kazemi, V., & Sullivan, J. (2014). One Millisecond Face Alignment with an Assemble of Regression Trees. *27th IEEE Conference on Computer Vision and Pattern Recognition*, 1867–1874. https://doi.org/10.13140/2.1.1212.2243

Kemelmacher-shlizerman, I., Seitz, S. M., Miller, D., & Brossard, E. (2016). The MegaFace Benchmark : 1 Million Faces for Recognition at Scale.

King, D. (2017). dlib C++ Library: High Quality Face Recognition with Deep Metric Learning. Retrieved June 17, 2018, from http://blog.dlib.net/2017/02/high-quality-face-recognition-with-deep.html

King, D. E. (2009). Dlib-ml: A Machine Learning Toolkit. *Journal of Machine Learning Research*, *10*, 1755–1758. https://doi.org/10.1145/1577069.1755843

Kingma, D. P., & Ba, J. (2014). Adam: A Method for Stochastic Optimization, 1–15. https://doi.org/http://doi.acm.org.ezproxy.lib.ucf.edu/10.1145/1830483.1830503

Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. *Advances In Neural Information Processing Systems*, 1–9. https://doi.org/http://dx.doi.org/10.1016/j.protcy.2014.09.007

Le, V., Brandt, J., Lin, Z., Bourdev, L., & Huang, T. S. (2012). Interactive facial feature localization. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *7574 LNCS*(PART 3), 679–692. https://doi.org/10.1007/978-3-642-33712-3_49

Le, Q. V. (2014). A Tutorial on Deep Learning Part 1: Nonlinear Classifiers and The Backpropagation Algorithm. *Tutorial*, 1–18. https://doi.org/10.1007/3-540-49430-8

Learned-Miller, E., Huang, G. B., RoyChowdhury, A., Li, H., & Hua, G. (2016). Labeled faces in the wild: A survey. *Advances in Face Detection and Facial Image Analysis*, 189–248. https://doi.org/10.1007/978-3-319-25958-1_8

LFW. (n.d.). Retrieved June 15, 2018, from http://vis-www.cs.umass.edu/lfw/devTrain.html

LIGHTHOUSE. (2018). Lighthouse AI: The only camera that gets your busy life at home. Retrieved June 15, 2018, from https://www.light.house/

Lin, M., Chen, Q., & Yan, S. (2014). Network In Network.

Liu, W., Wen, Y., Yu, Z., Li, M., Raj, B., & Song, L. (2017). SphereFace: Deep hypersphere embedding for face recognition. *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, *2017–Janua*, 6738–6746. https://doi.org/10.1109/CVPR.2017.713

Liu, Z., Luo, P., Wang, X., & Tang, X. (2015). Deep learning face attributes in the wild. *Proceedings of the IEEE International Conference on Computer Vision*, *2015 Inter*, 3730–3738. https://doi.org/10.1109/ICCV.2015.425

Lowe, D. G. (2004). Distinctive Image Features from Scale-Invariant Keypoints, *60*(2), 91–110.

Martin, K., Hirzer, M., Wohlhart, P., Roth, P. M., Bischof, H., Fn, F. P., & Fn, F. P. (2012). Large Scale Metric Learning from Equivalence Constraints ∗, (Ldml).

Mikolov, T., Sutskever, I., Chen, K., Corrado, G., & Dean, J. (2013). Distributed Representations ofWords and Phrases and their Compositionality TomasMikolov. https://doi.org/10.1162/jmlr.2003.3.4-5.951

Nech, A., & Kemelmacher-shlizerman, I. (2016). Level Playing Field for Million Scale Face Recognition.

Ng, H. W., & Winkler, S. (2014). A data-driven approach to cleaning large face datasets. *2014 IEEE International Conference on Image Processing, ICIP 2014*, 343–347. https://doi.org/10.1109/ICIP.2014.7025068

Niblack, C. W. (1993). QBIC project: querying images by content, using color, texture, and shape. *Proceedings of SPIE*, *1908*(1), 173–187. https://doi.org/10.1117/12.143648

NumPy — NumPy. (2018). Retrieved June 15, 2018, from http://www.numpy.org/

OpenCV library. (2018). Retrieved June 15, 2018, from https://www.opencv.org/

Parkhi, O. M., Vedaldi, A., & Zisserman, A. (2015). Deep Face Recognition, (Section

3).

Pedregosa, F., Varoquaux, G., Gramfort, A., & Michel, V. (2010). scikit-learn: machine learning in Python — scikit-learn 0.19.1 documentation. Retrieved July 3, 2018, from http://scikit-learn.org/stable/index.html

Pennington, J., Socher, R., & Manning, C. (2014). Glove: Global Vectors for Word Representation. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 1532–1543. https://doi.org/10.3115/v1/D14-1162

Qin, Y., Song, D., Chen, H., Cheng, W., Jiang, G., & Cottrell, G. (2017). A Dual-Stage Attention-Based Recurrent Neural Network for Time Series Prediction. Retrieved from http://arxiv.org/abs/1704.02971

Redmon, J., & Farhadi, A. (2016). YOLO9000: Better, Faster, Stronger Joseph. https://doi.org/10.1109/CVPR.2017.690

Redmon, J., & Farhadi, A. (2018). YOLOv3: An Incremental Improvement. https://doi.org/10.1109/CVPR.2017.690

Ren, S., He, K., Girshick, R., & Sun, J. (2016). Faster R-CNN : Towards Real-Time Object Detection with Region Proposal Networks, 1–14.

Ruder, S. (2016). An overview of gradient descent optimization algorithms. Retrieved from http://arxiv.org/abs/1609.04747

Rui, Y., & Huang, T. S. (1999). Image Retrieval : Current Techniques , Promising Directions , and Open Issues, *62*, 39–62.

Sagonas, C., Antonakos, E., Tzimiropoulos, G., Zafeiriou, S., & Pantic, M. (2015). 300 Faces In-The-Wild Challenge: database and results. *Image and Vision Computing*, *47*, 3–18. https://doi.org/10.1016/j.imavis.2016.01.002

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., & Chen, L.-C. (2018). MobileNetV2: Inverted Residuals and Linear Bottlenecks. Retrieved from http://arxiv.org/abs/1801.04381

Schroff, F., Kalenichenko, D., & Philbin, J. (2015). FaceNet: A unified embedding for face recognition and clustering. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, *07–12–June*, 815–823. https://doi.org/10.1109/CVPR.2015.7298682

Shah Nawaz Grewal. (2018). GitHub - Shahnawazgrewal/Enhanced-Center-Loss: Code for enhanced center loss function. Retrieved June 15, 2018, from https://github.com/Shahnawazgrewal/Enhanced-Center-Loss

Simonyan, K., & Zisserman, A. (2015). Very Deep Convolutional Networks for Large-Scale Image Recognition. https://doi.org/10.1016/j.infsof.2008.09.005

Smola, A. J., & Schölkopf, B. (2004). A tutorial on support vector regression. *Statistics and Computing*, *14*(3), 199–222. https://doi.org/10.1023/B:STCO.0000035301.49549.88

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research*, *15*, 1929–1958. https://doi.org/10.1214/12-AOS1000

Stricker, M. A., & Orengo, M. (1995). Similarity of color images, 381. https://doi.org/10.1117/12.205308

Sun, Y., Liang, D., Wang, X., & Tang, X. (2015). DeepID3: Face Recognition with Very Deep Neural Networks, 2–6. Retrieved from http://arxiv.org/abs/1502.00873

Sun, Y., Wang, X., & Tang, X. (2014). Deep Learning Face Representation by Joint Identification-Verification, 1–9. https://doi.org/10.1109/CVPR.2014.244

Szegedy, C., Ioffe, S., Vanhoucke, V., & Alemi, A. (2016). Inception-v4, Inception-ResNet and the Impact of Residual Connections on Learning. https://doi.org/10.1016/j.patrec.2014.01.008

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., … Rabinovich, A. (2015). Going deeper with convolutions. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, *07–12–June*, 1–9. https://doi.org/10.1109/CVPR.2015.7298594

Taigman, Y., Yang, M., Ranzato, M., & Wolf, L. (2014). DeepFace: Closing the gap to human-level performance in face verification. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 1701–1708. https://doi.org/10.1109/CVPR.2014.220

Tencent. (2018). Face Analysis|Tencent Youtu. Retrieved June 15, 2018, from http://bestimage.qq.com/faceanalyze.html

Viola, P., & Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*, *1*, I-511-I-518. https://doi.org/10.1109/CVPR.2001.990517

Wang, F., Liu, W., Liu, H., & Cheng, J. (2018). Additive Margin Softmax for Face Verification. https://doi.org/10.1109/LSP.2018.2822810

Weinberger, K. Q., & Saul, L. K. (2009). Distance Metric Learning for Large Margin Nearest Neighbor Classification. *The Journal of Machine Learning Research*, *10*, 207–244. https://doi.org/10.1126/science.277.5323.215

Welcome to Python.org. (2018). Retrieved June 15, 2018, from https://www.python.org/

Wen, Y., Zhang, K., Li, Z., & Qiao, Y. (2016). A Discriminative Feature Learning Approach for Deep Face Recognition Yandong. https://doi.org/10.1007/978-3-319-46478-7

Xie, S., Girshick, R., & Doll, P. (2017). Aggregated Residual Transformations for Deep Neural Networks.

Yang, J., Jiang, Y.-G., Hauptmann, A., & Ngo, C.-W. (2007). Evaluating Bag-of-Visual-Words Representations in Scene Classification, 197–206.

Yang, S., Luo, P., Loy, C. C., & Tang, X. (2016). WIDER FACE: A face detection benchmark. *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, *2016–Decem*, 5525–5533. https://doi.org/10.1109/CVPR.2016.596

Yaoyao Zhong, & Weihong Deng. (2017). Fine-grained LFW (FGLFW) Database.

Retrieved June 15, 2018, from http://www.whdeng.cn/FGLFW/FGLFW.html

Yi, D., Lei, Z., Liao, S., & Li, S. Z. (2014). Learning Face Representation from Scratch.

Zeiler, M. D., & Fergus, R. (2014). Visualizing and understanding convolutional networks. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *8689 LNCS*(PART 1), 818–833. https://doi.org/10.1007/978-3-319-10590-1_53

Zhang, K., Zhang, Z., Li, Z., Member, S., Qiao, Y., & Member, S. (2016). Joint Face Detection and Alignment using Multi - task Cascaded Convolutional Networks. *Spl*, (1), 1–5. https://doi.org/10.1109/LSP.2016.2603342

Zhang, X., Zhou, X., Lin, M., & Sun, J. (2017). ShuffleNet: An Extremely Efficient Convolutional Neural Network for Mobile Devices.

Zhong, Y., Chen, J., & Huang, B. (2017). Toward End-to-End Face Recognition Through Alignment Learning. *IEEE Signal Processing Letters*, *24*(8), 1213–1217. https://doi.org/10.1109/LSP.2017.2715076

Zoph, B., Vasudevan, V., Shlens, J., & Le, Q. V. (2017). Learning Transferable Architectures for Scalable Image Recognition. Retrieved from http://arxiv.org/abs/1707.07012

Appendix

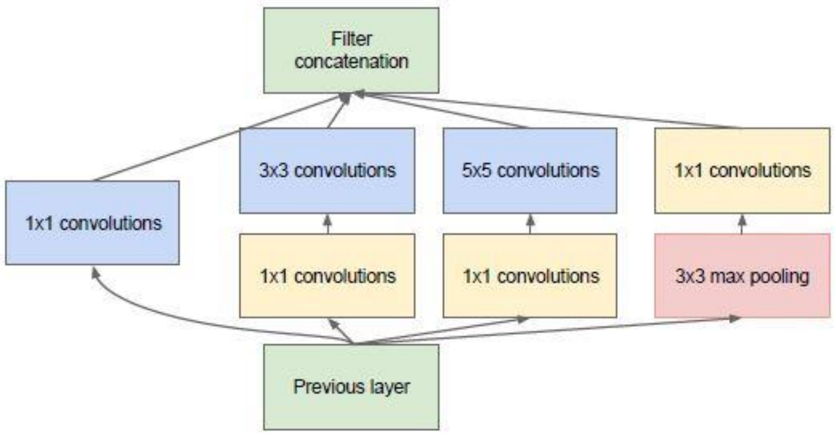


Figure 18 - Inception Network (GoogLeNet) building block (Szegedy et al., 2015).

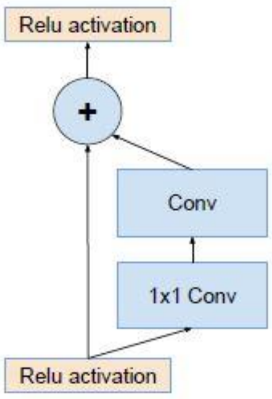


Figure 19 - ResNet (He et al., 2015) building block (Szegedy et al., 2016).

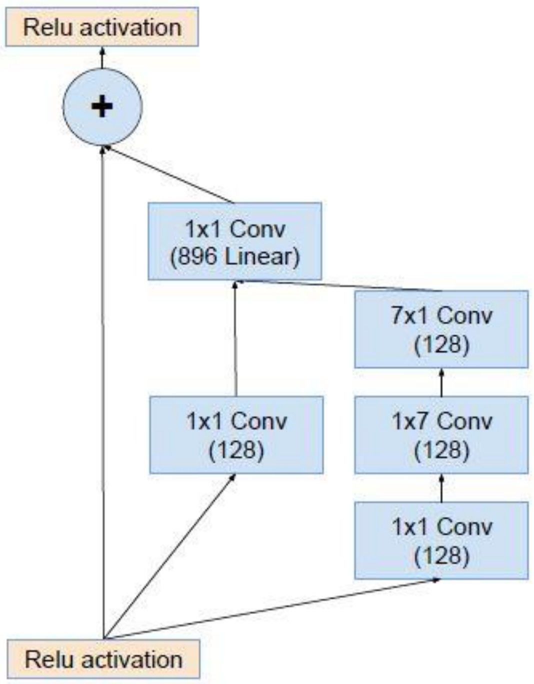


Figure 20 - Inception-ResNet-V1 block example (Szegedy et al., 2016).

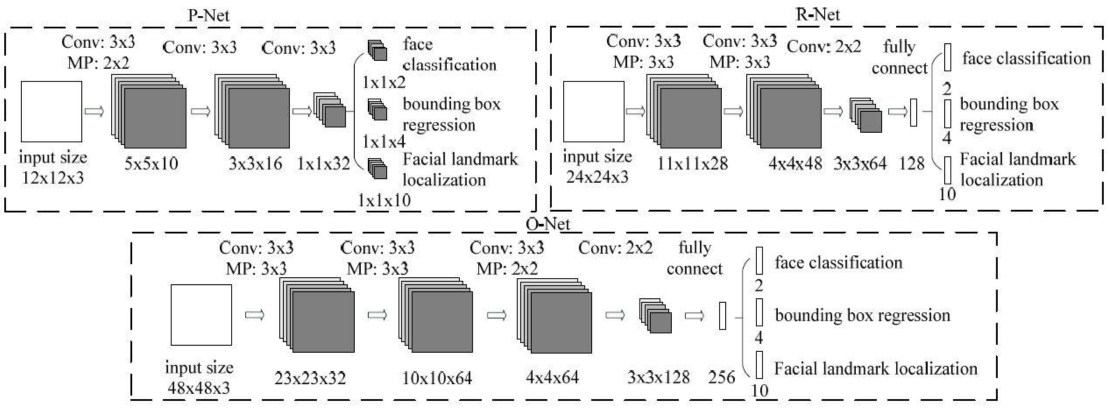
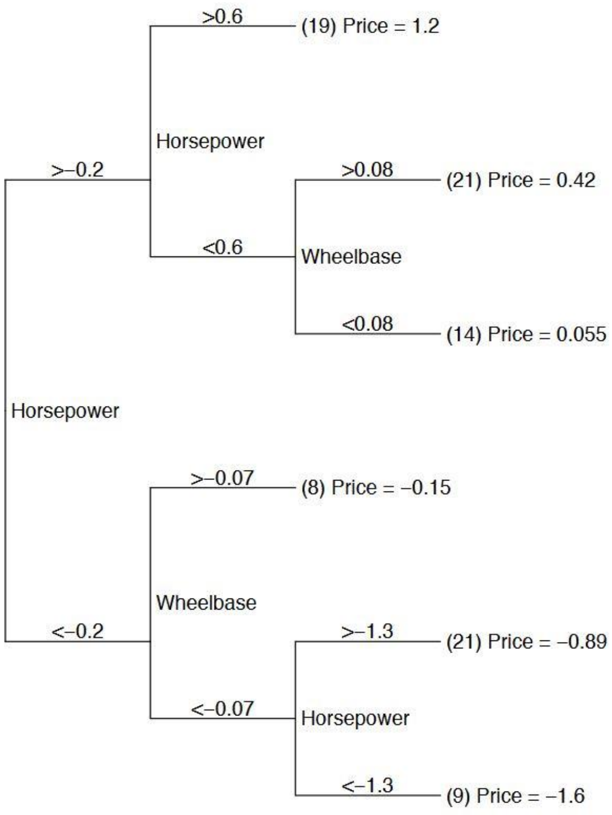


Figure 21 - MTCNN's P-Net, R-Net and O-Net architectures (Zhang et al., 2016).



1.

Figure 22 - Regressive Tree car price prediction example (Cosma & Shalizi, 2006).

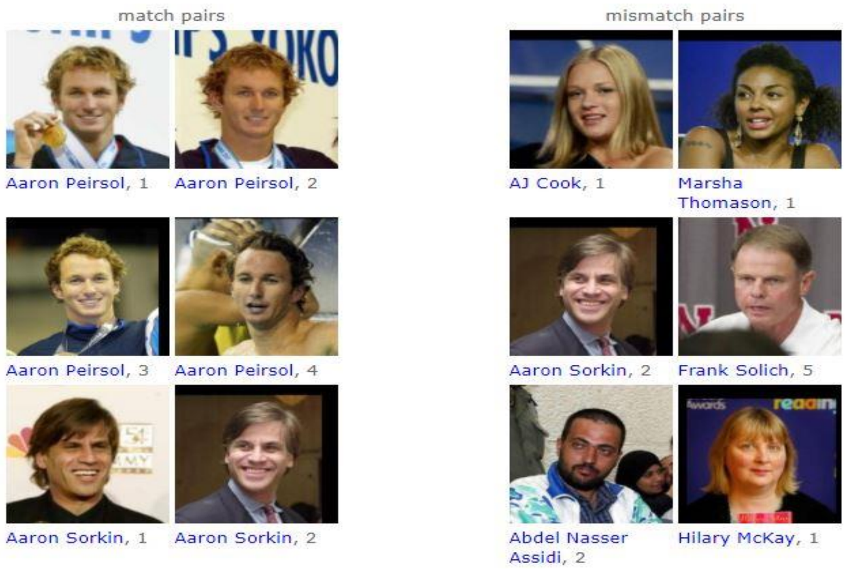


Figure 23 - LFW's training set sample, matched and mismatched pairs ("LFW," n.d.), non- aligned faces.

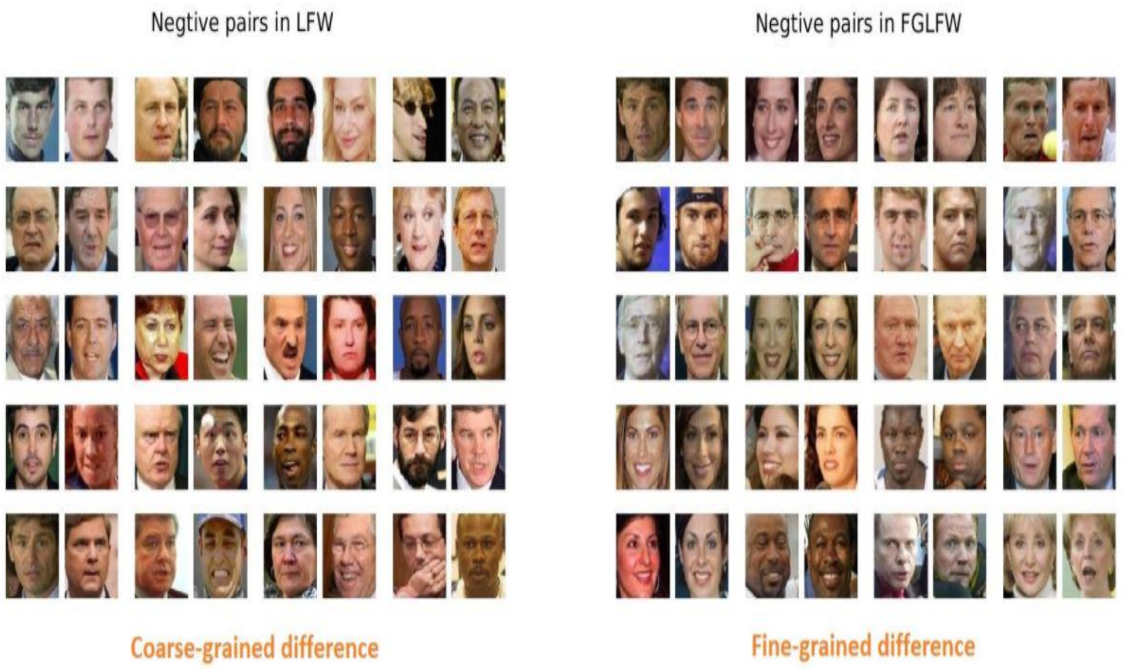


Figure 24 - On the left: mismatched LFW's pairs; on the right: mismatched FGLFW's pairs (Zhong & Deng, 2017).