university of groningen

faculty of science and engineering

Analysis of One-Shot Facial Recognition for Real-Time Identity Verification using Deep Learning

Bachelor's Thesis

Primary supervisor: K. Bunte
Secondary supervisor: M. Biehl
Externel supervisor: J.F. van Wezel

Athul Raj Nambiar (s4126351)

August 5, 2022

Contents

Abstract

Facial Recognition is a powerful technology that has the potential to reduce fraudulent activities such as identity theft, help businesses grow by collecting customer data, aid in healthcare and much more. The current level of the technology has room for improvement, particularly when investigating ID related verification tasks.

One of the biggest limitations that affect the accuracy of facial recognition software is the available data set (in this case the number of images). Research within the last two decades has shown that Convolutional Neural Networks (CNNs) can be used to accurately classify the identity of faces, however, it requires hundreds of thousands up to millions of images to achieve this. Siamese Neural Networks (SNNs) on the other hand have the potential to be used in the context of one-shot face and/or image verification tasks, which makes it the ideal model to research, as it can produce very accurate results for face verification tasks with a much smaller scale of images. This research is becoming more important because many companies and organizations require remote identity verification models, however only have very limited training images, and require their software to be able to accurately verify the identity of people unseen by the model. For our research, we investigate two Siamese models, the Siamese FaceNet and Siamese VGG Face, both constructed with state-of-the-art facial recognition CNNs (FaceNet and VGG Face models). We train our models using pre-existing training methods that have obtained high accuracy rates. Furthermore, we investigate the addition of online triplet mining and its potential for improving our training process. Our proposed techniques will be evaluated on a public dataset using 5-fold cross validation, and then we finally apply transfer learning to our models on a demonstration set that emulates our real-world scenario using biometric ID photos and selfies. The proposed models, particularly the Siamese FaceNet, indicate good performance, stability, and are able to generalize well to our out-of-sample demonstration set. We analyze the limitations of the models with respect to image data characteristics such as lighting, facial hair, objects, and others, and outline the basis for future experiments. The goal of this research is to produce an analysis of our SNN models, the training process and the corresponding results in order to provide research that can be contributed to real-time face verification systems.

1 Introduction

In recent years, computer vision is a sector of artificial intelligence that has seen massive growth while playing a crucial role in modern technology and numerous scientific fields. Facial recognition is a topic within computer vision that becomes more important on a daily basis as it has a range of applications, such as for surveillance, commercial applications, and law enforcement [1], [2]. Nowadays, many businesses and government organizations use facial recognition and/or face matching as part of their remote identity verification systems. These systems require a person's identity to be verified by checking the biometric photo on their ID Document and compare it to the holder's face. Research has shown that these systems have their faults, and ID fraud online has increased from 4.1% in 2019 to 5.9% in 2021 [3]. Hence, the importance of accurate remote identity verification technology is only increasing.

"Facial recognition" tends to be an ambiguous term; the definition we will use within our research is the process of identification and verification of faces [4]. Identification is taking an image of a face, and the system task is to identify the person (or the person of the highest probability) from all the people stored in the system. There are many techniques used in facial identification systems, which all use the basic principle of extracting the facial features from an image. Some examples include Local Binary Patterns which is a feature representation method that removes irrelevant information [5] and coupled discriminative feature learning which is a coupled subspace learning method that projects the features of different spectral bands in a common subspace [5], [6]. Verification is different to identification, where verification is defined as taking two images and checking whether the people in the two images have the same identity or not. An example of a face verification technique would be the use of a dimensionality reduction algorithm by Shuicheng et al. [7] that is capable of finding a balanced threshold. Both face identification and verification systems can use similar technology, and both tasks require extracting facial features from images, the difference between the two is how the extracted information is processed and compared.

In the last two decades, the research and development scope of facial recognition technology has changed to focus on deep learning. Deep learning is a sub-field of machine learning, that uses large neural networks to imitate the human brain using data inputs, weights, and biases, that work together to make accurate data-driven decisions [8]. A technique used commonly in deep learning for facial recognition applications is the Convolutional Neural Network (CNN). CNNs are networks that preserve spatial structure by placing weights and biases on objects from input images and learning to generalize those features [9]. CNNs are now one of, if not the most applied approach for almost all recognition [10], [11] and detection tasks [12], [13], and can even reach human performance for certain tasks. This explains their extensive usage within facial recognition software, as research shows Neural Networks tend to produce the highest performance for facial recognition systems [5]. CNNs specifically prove to have very accurate results, however, have one major limitation; they require a lot of data to train the model and obtain accurate results. Furthermore, in the context of facial recogni-

tion, CNNs on their own are used for the classification of faces that were already introduced in the training of the model, meaning face identification tasks. This means that the addition of new images would require the model to be retrained every time an additional face is checked with the system. For a remote face verification task, this method would be both very inefficient and insufficient because millions of new faces introduced to the system would require retraining of the entire model.

One-shot learning techniques tend to be very effective for verification tasks such as image verification [14], and are designed to train models that only contain one image per class [15]. An example of a model used frequently in one-shot learning is the Siamese Neural Network (SNN). A SNN has an architecture built using two identical Artificial Neural Networks (ANNs) [16]. Each twin network requires an input vector, and inputs are then processed in parallel to produce outputs. Finally, these outputs are compared based on the specific task. During the training of a SNN error-back propagation is applied. SNNs show great effectiveness and potential with computer vision applications like face verification [17] and general image verification (examples include images of handwritten texts like alphabets and signatures [14], [18], or even general objects like vehicles [19]). Hence, the study and analysis of an SNN model, based on one-shot learning that overcomes major limitations of CNNs, can provide insightful research to create accurate face verification software.

In this thesis, the focus of our research will be to investigate how to construct an effective SNN model that can be used to accurately verify whether the identity of the person in two input images is the same. The considered application area is ID verification, and hence the task is to verify whether the person in an image or screenshot is likely to be the same person as in an image of an ID document's biometric photo. It is important that during this investigation we focus close attention on our model's capability of minimizing the number of false positives. If our models are able to minimize the percentage of false positives, it also reduces the risk of incorrectly accepting criminals that commit fraud or identity theft. Simultaneously, since there is a lot of research behind CNNs for facial recognition, we explore improvements made around this model, specifically transfer learning to convolutional layers and online triplet mining, and see if these improvements can be adapted to the two ANNs of a one-shot SNN model. We investigate how the described deep learning techniques and their modifications perform in face verification software aimed for ID card verification.

There are three major contributions from this thesis. Our first contribution is that we implement and investigate a face verification system based on a one-shot SNN model. Our second contribution and our primary focus is the research and application of the steps required to accurately verify faces and minimize the number of false positives. Lastly, we investigate the extent to which a Siamese architecture affects the classification accuracy of pre-existing CNN models.

This project is in collaboration with the company Verifai. Verifai specifically would like to know about our implementation and analysis of our face verification model and its suitability for ID verification tasks. Due to the unavailability of data specific to our ID verification task, our models are trained using a public dataset that contains no ID photos. The model is then applied to a smaller demonstration set that contains biometric ID photo images. This contribution is structured as follows. In section 2,

we outline the related work and the basics of face verification. We then introduced the methods and concepts used for our investigation. Specifically, the CNN and SNN architectures we construct are discussed here in detail. In section 3, we discuss our Data Management Plan and the steps required to preprocess the data for our experiments. Next, in section 4 we outline the experiment, outlining the training methodology of our constructed networks and then discussing the metrics we will use to evaluate the model's performance. Once we conduct the experiment and calculate the results, we present and discuss their implications and irregularities in section 5. Finally, we summarize our results and overall contributions of our research, while also suggesting approaches for future research in section 6.

2 Methods

In this section, we first introduce previous work that is related to our research, the basics of face verification, and the theoretical background of the concepts and techniques used in our experiments.

2.1 Related Work

2.1.1 Convolutional Neural Networks for Facial Recognition

CNN based methods can produce higher accuracies than face recognition methods that are based on handcraft features [20], [21]. Up-to-date research is available on large CNNs that are used for image classification based on a person's age, sex, ethnicity, etc. and an example by Parde et al. uses a linear classifier that is trained with cross validation to predict human-assigned trait ratings, where this deep CNN was trained with 494,414 images [22]. Training such large networks could have taken weeks almost a decade ago and nowadays will only take a couple of hours due to advancements in hardware, software and parallel algorithm [23]. In particular, improvements in backpropagation optimization techniques produced from the research of Werbos [24] and LeCunn et al. [25] are one of the major drivers of these advancements. However, this is still an important limitation to consider because high-quality hardware and software requirements are not easily accessible for research purposes. Furthermore, the greatest limitation of CNNs would be the training data required for the network. For example, the EfficientNetV2 [26] consists of CNNs that aims for higher training speeds for small-scale datasets (an example considered is the ImageNet1k with 1.28 million images) [27]. Therefore, the amount of data required to train a CNN is a major disadvantage when considering CNNs for facial recognition software.

Researchers have constructed and investigated many CNN architectures for facial recognition systems because of their performance and ability to produce state-of-the-art results. In fact, CNNs have been used in nearly all the top performing facial recognition problems on the Labelled Faces in the Wild dataset (LFW) [28], the dataset that sets the public benchmark for face verification [29]. Google has designed their own CNN based face recognition algorithm called FaceNet [30]. The smallest FaceNet model is 22 layers deep, however, their other model architectures are much deeper because they connect larger Inception models together, such as NN2 as described in their research. With the FaceNet algorithm, the images are mapped from a face into a position in a multidimensional Euclidean space (in the form of an embedding vector). FaceNet's CNN models are trained with up to 200 million images using triplet loss. Triplet loss calculates a loss metric where the model learns positions for images of the same people are closer in the Euclidean space and images of different people are positioned further apart. FaceNet was able to reach up to 99.63% classification accuracy on the LFW dataset by applying similarity transform alignment to image data. Furthermore, the FaceNet models do not require a CNN bottleneck layer or additional post-processing like other alternatives [31], [32] and perform very well with unseen data. However, the models have high CPU requirements due to

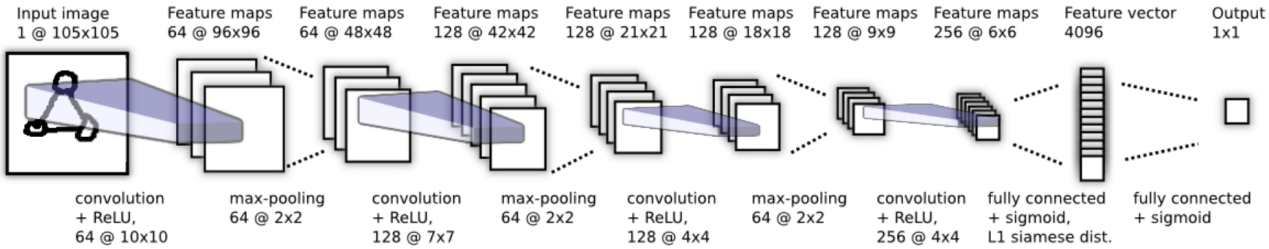


Figure 1: Convolutional Siamese architecture by Koch et al. [14]. The Siamese twin is not shown but connects immediately after the 4096 unit fully connected layer, where the L1 Siamese distance vectors are computed.

their very deep architecture, especially those connected with multiple inception models. The Visual Geometry Group at the University of Oxford also constructed a very high performing CNN that is now referred to as the VGG Face [33]. Their model is 19 layers deep and is trained on 2.6 million images using cross entropy loss. In addition, their research shows that when fine-tuned by training only the fully connected layers using triplet loss, the model is able to achieve 98.95% accuracy on the LFW dataset. Due to the smaller architecture of the VGG Face, it also has a lower CPU requirement compared to FaceNet. Both the FaceNet and VGG Face perform well on facial recognition tasks with different architecture sizes, and we will investigate them alongside one-shot Siamese architectures as part of our research contributions.

2.1.2 Siamese Network for One-Shot Object and Face Verification

One-shot learning is a subject within machine learning that has seen a lot of growth in recent years. The purpose of the entire field of research is to be able to train and produce accurate models with very little data, usually only one image per class. One-shot learning dates back to the 2000s with the work by Li Fei-Fei et al. [34], [35], where the authors present a method for learning object categories using a variational Bayesian framework that uses knowledge from previously learned classes to predict future ones with only a few examples being available from those classes. More recent research has been taking advantage of Siamese Neural Networks for learning object categories and verification tasks [14]. The architecture of their network is seen in Figure 1. They connect two twin CNNs that output 4096 embedding vectors. The CNNs are connected to an L1 distance layer that takes the two embedding vectors and calculates the L1 distance and connects 4096 nodes to a single dense node. To train their model, they use binary cross entropy loss which is useful for binary classification tasks such as image verification (classifying whether the images are the same class or not the same class). They train up to 150,000 images from the Omniglot dataset, a dataset that contains images of 1623 different handwritten characters from 50 different alphabets and was designed for developing more human-like learning algorithms [36]. The Convolutional Siamese Network that they construct can reach up to 93.42% accuracy for Omniglot verification tasks.

Very similar SNNs have also been adapted for face verification tasks. An example is Figure 1 con-

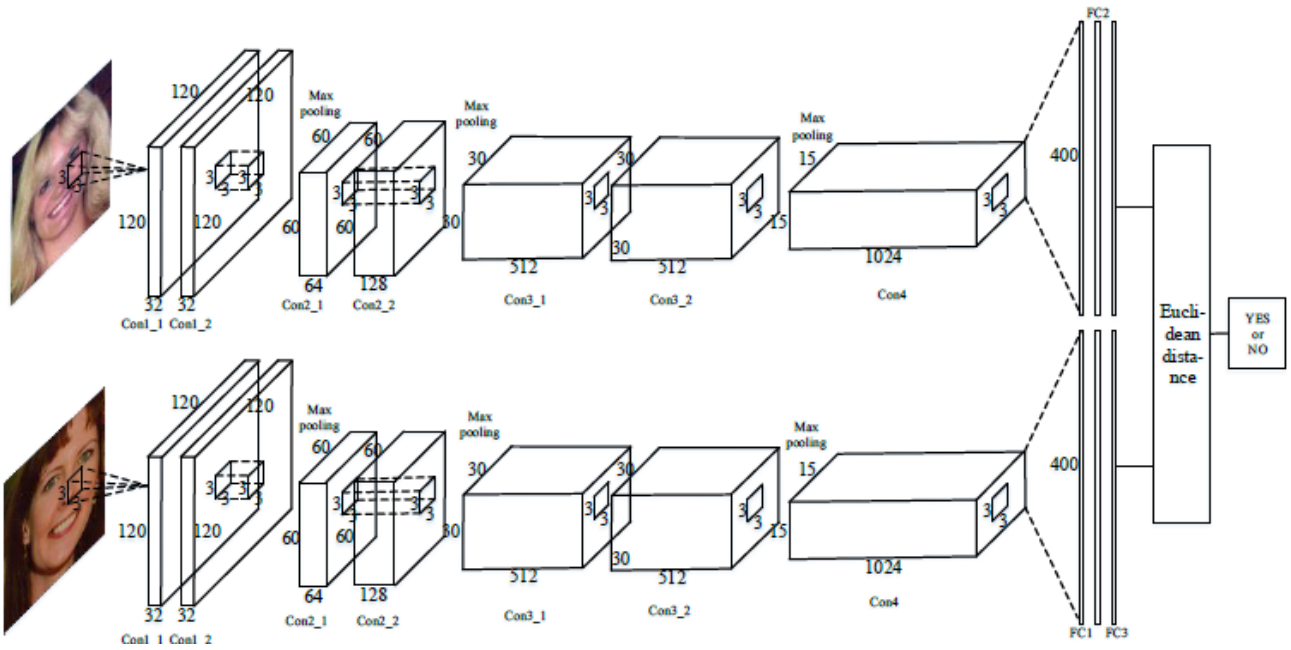


Figure 2: SiameseFace1 network architecture [17]. The network has two twin CNNs. Each twin CNN has three Fully connected layers at the top of the network that output embedding vectors. The Euclidean distance between the two output vectors is calculated and converted to a "yes" or "no" based on a pre-calculated threshold.

structed by Zhang et al. [17]. The similarity between Figures 1 and 2 is the overall structure; two twin CNNs with fully connected top layers, where the output of the twin CNNs connect to a distance layer, which then connects to an output layer to give a single output. For both SNN models, they use output embedding vectors that are output from the CNN layer, after which the distance between the embedding vectors is calculated. Finally, the output value from both SNN models is classified by using a calculated threshold. There are also notable differences between the two SNN models. The CNNs for each model have a different number of fully connected layers, convolutional layers, kernel sizes, and frame sizes. This means that the different CNN structures learn discriminative features at different layers of the CNN. Furthermore, in Figure 1 the output of the distance layer is connected to another single dense node that computes a single value. In Figure 2 on the other hand, the output of the distance layer is directly a single output value because it uses Euclidean distance. The advantage of having the connected node between the distance layer and output is that the value of the output can be mapped to a certain value range based on an activation function. The disadvantage is that it adds an extra training parameter and requires finding a suitable activation function. Another difference is that SiameseFace1 was using the contrastive loss function for the training process. Contrastive loss is similar to triplet loss, where during training the model learns so that embedding vectors output by the CNN are closer for images of the same people and embedding vectors are farther apart for different people. The SiameseFace1 achieved a recognition rate of 98.8% on the AR dataset and 94.80% on the LFW dataset after being trained using contrastive loss with 20,000 images (from both datasets).

These previous works are a good indication of the current level of research conducted on CNNs and SNNs and show the research possibilities of these models in the direction of ID verification tasks.

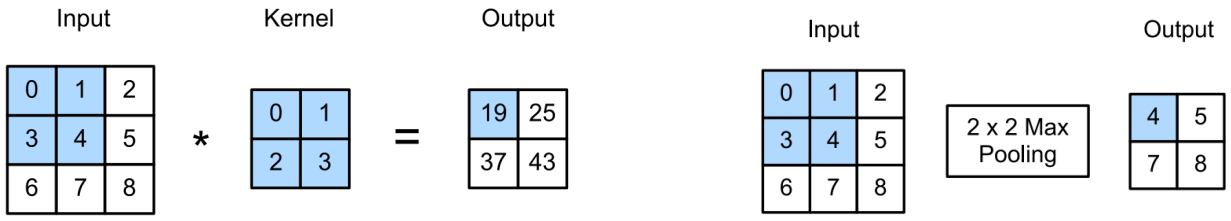
2.2 Face Verification

Face verification is the process of taking two images that contains the faces of people and checking if the identity of the two faces in the two images match. When approaching this problem with deep learning techniques, the general idea is to take the two input images and convert them into a vector by inputting them into an Artificial Neural Network (ANN). Within its layers, the ANN extracts features from the original input. Through a cascade of weighted non-linear transformations, each network outputs a lower dimensional embedding vector. The two lower dimensional output vectors represent a point or position in multidimensional Euclidean space. We calculate the distance between the two points and see if the resulting distance is greater than a certain threshold value. If the distance is greater than the threshold value, then we assume that the two faces do not match and that the faces are of different people, since we estimate dissimilarity. On the other hand, if the distance is less than this threshold, we assume the faces are similar enough to determine a match. In the papers introducing the FaceNet and VGG Face facial recognition CNNs, the authors outline how the CNNs can be used to form position embedding vectors for face verification [30], [33]. While the SNN architecture is built specifically for verification and includes an inbuilt distance layer as seen in Figure 2 [17]. After studying these previous works, we conduct further research that we adapt to our use case of ID verification.

2.3 Convolutional Neural Networks

An ANN also often called multi-layer perceptrons, is a network of hierarchical or multi-layered structure (MLP). It is capable of mathematically learning to represent features at different scales and resolutions and combine them into high-order features [9]. A typical ANN contains an input layer, multiple fully connected (or dense layers) and finally an output layer.

A CNN is an ANN that is able to achieve state-of-the-art results on computer vision and natural language processing tasks [9]. The architecture of CNNs allows them to be able to retain and preserve information on spatial features and generalize those features from input images using weights and biases. The building blocks of these architectures are the convolutional and pool layers, which are usually connected to fully connected layers. For instance, the CNNs in Figures 1 and 2 are built using these building blocks. The layers within a CNN apply mathematical calculations over the course of multiple layers in order for the network to generalize and extract features. In the convolutional layers, the network calculates convolutions. The definition for a single 2-D convolution is defined by this equation, where f is the image input with pixels referred to as x and y, and k is the kernel of size $d \times e$ with i and j representing the position based on the current kernel: $f(x,y) * k = \sum_{i=0}^{d-1} \sum_{j=0}^{e-1} f(x+i, y+j) \cdot k(i,j)$, with $*$ denoting the convolution operation. Figure 3a shows an example of the



(a) Convolution Process

(b) Max Pooling Process

Figure 3: Convolutions are calculated in the Convolutional Layers of a CNN. During the convolution process in (a), the kernel slides over the input image for each location. At each position, the sum of element-wise multiplication of the kernel and the same-sized image pixels is an output [37]. (b) is the max pooling process. This is the simple process that takes place in a max pooling layer of a CNN.

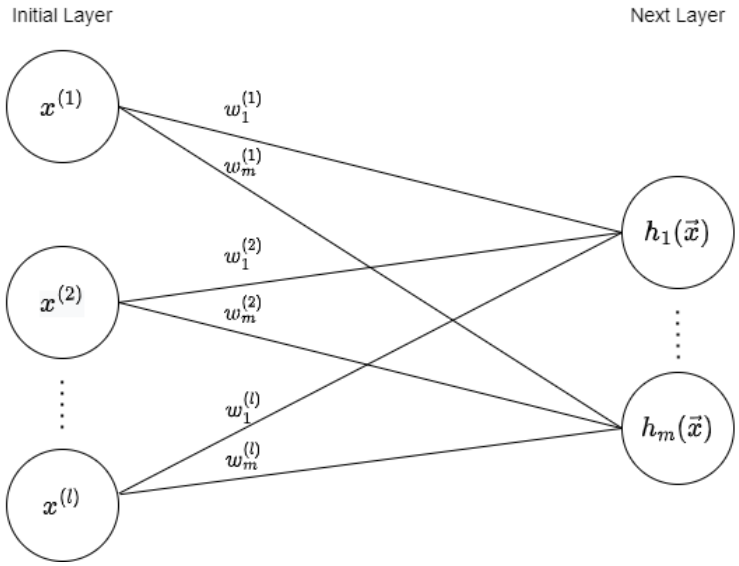


Figure 4: A fully connected layer in an ANN. x presents the values of the node in the initial layer, w represents the weight of the connections, and h is the function that calculates the value at the next layer.

convolutional process that occurs based on the 2-D convolution equation, where the shaded region in the input and the weights of the kernel result in $0\times 0+1\times 1+3\times 2+4\times 3=19$. The convolution layers within a CNN use numerous trainable kernels per layer to extract discriminate features from the input image. In the pooling layer of the CNN, typically the max() operation takes place based on the values we assign for width×height of the receptive field (for example, the shaded region in the input of 3b has a receptive field of 2×2). Other alternative operations that take place are min() or average() in this layer. Pooling layers follow a single or a sequence of convolutional layers and are used to consolidate the learned features[9]. A fully connected layer is displayed in Figure 4. The calculation for the next layer is defined using the function h, where $\vec{x}$ is the input vector defined by the nodes of the initial layer, w are the weights of the connections, and b is an additional bias that is added: $h_j^{(i)}(\vec{x}) = \sum_{i=0}^{l} x^{(i)} \cdot w_j^{(i)} + b$ where $j \in [1,...,m]$. An optional layer that is used in modern CNNs is

the batch normalization layer. This layer is useful for training, and it standardizes the inputs for each mini-batch. Furthermore, it has the effect of stabilizing the learning process and this reduces the number of training epochs required [38]. This equation defines the batch normalization layer, where μ is the mini-batch mean, σ^2 is the mini-batch variance, $x^{(i)}$ is the i^{th} value from the input vector $\vec{x}$, ε is a small value added for numerical stability, and γ and β are parameters learned in the optimization process: $BN_{\gamma,\beta}(x^{(i)}) = \gamma \cdot \left(\frac{x^{(i)}-\mu}{\sqrt{\sigma^2+\varepsilon}}\right) + \beta$. These are the four particular layers used to construct the CNNs that we will investigate.

The CNN architectures we investigate use two activation functions, Rectified Linear Unit (ReLU) and Softmax. Activation functions are used in neural networks to establish and quantify a certain level of importance to a neuron, based on a mathematical function. Research has shown that ReLU activation greatly improved the performance of feedforward networks [39]. ReLU is defined by this equation, where r in the output of a neuron: $\max(0,r)$. Softmax activation, generally used at the end of the CNN architecture, is used to normalize the output nodes based on a probability distribution based on the possible classes that can be predicted. The softmax activation function is applied to networks that are intended for multi-class classification tasks. Softmax is defined by this equation, where s is the softmax function, $\vec{x}$ is the input vector (typically the output of the CNN when softmax is used in the final layer), and M is the size of the input vector $\vec{x}$ but also the number of classes the model is trained to classify: $s(x_i) = \frac{e^{x_i}}{\sum_{j=0}^{M} e^{x_j}}$.

For a CNN, Loss functions are important to evaluate how well data is being modelled and play an important role in the training of a model. During training, the models learn when the weights of the connections between the layers are modified by minimizing the loss through backpropagation. As discussed in section 2.1.1, some typical loss functions used to train CNNs are triplet loss (especially for facial recognition tasks) and cross entropy loss. Triplet loss is explained in section 2.5.1 and the particular CNN models we use for our research were not trained with triplet loss. Cross entropy loss is defined by this equation, where M is the number of classes, y is a binary indicator (0 or 1) if class label c is the correct classification for observation o, and P is the predicted probability of observation o is of class c: $-\sum_{c=1}^{M} y_{o,c} \log(P_{o,c})$. As part of our investigation of particular CNNs, VGG face and FaceNet are detailed in the following section.

2.3.1 VGG Face

The Visual Geometry Group at the University of Oxford constructed a CNN that is 19 layers deep and connected to a supplementary fully connected layer. Their model is known as the VGG Face. It has 37 Deep Units, and 13 convolutional layers (excluding the fully connected layer), and all convolutional layers are followed by a layer to apply ReLU activation. The model was designed and trained for facial recognition purposes. Their model was trained on 2.6 million images that contained 2622 different people, and it was trained using cross entropy loss. An overview of the CNN is presented in Figure 5. Within the architecture, the fully connected layers at the end are listed as “convolution”

layer	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18
type	input	conv	relu	conv	relu	mpool	conv	relu	conv	relu	mpool	conv	relu	conv	relu	conv	relu	mpool	conv
name	–	conv1_1	relu1_1	conv1_2	relu1_2	pool1	conv2_1	relu2_1	conv2_2	relu2_2	pool2	conv3_1	relu3_1	conv3_2	relu3_2	conv3_3	relu3_3	pool3	conv4_1
support	–	3	1	3	1	2	3	1	3	1	2	3	1	3	1	3	1	2	3
filt dim	–	3	–	64	–	–	64	–	128	–	–	128	–	256	–	256	–	–	256
num filts	–	64	–	64	–	–	128	–	128	–	–	256	–	256	–	256	–	–	512
stride	–	1	1	1	1	2	1	1	1	1	2	1	1	1	1	1	1	2	1
pad	–	1	0	1	0	0	1	0	1	0	0	1	0	1	0	1	0	0	1

layer	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34	35	36	37
type	relu	conv	relu	conv	relu	mpool	conv	relu	conv	relu	conv	relu	mpool	conv	relu	conv	relu	conv	softmx
name	relu4_1	conv4_2	relu4_2	conv4_3	relu4_3	pool4	conv5_1	relu5_1	conv5_2	relu5_2	conv5_3	relu5_3	pool5	fc6	relu6	fc7	relu7	fc8	prob
support	1	3	1	3	1	2	3	1	3	1	3	1	2	7	1	1	1	1	1
filt dim	–	512	–	512	–	–	512	–	512	–	512	–	–	512	–	4096	–	4096	–
num filts	–	512	–	512	–	–	512	–	512	–	512	–	–	4096	–	4096	–	2622	–
stride	1	1	1	1	1	2	1	1	1	1	1	1	2	1	1	1	1	1	1
pad	0	1	0	1	0	0	1	0	1	0	1	0	0	0	0	0	0	0	0

Figure 5: One of the CNN configurations from the Oxford Group. This architecture is what we refer to as VGG Face for our investigation. For each convolutional layer, the filter size, number of filters, stride and padding are presented [33].

but, due to the size of their kernels that match the size of the input data, the convolutions perform the same calculations as a fully connected layer. Furthermore, the softmax activation function is applied at the output layer of the VGG Face. Meaning, this model was designed to classify images to one of the 2622 people the system was trained on.

2.3.2 FaceNet

Google's face recognition algorithm is called FaceNet. FaceNet's CNN models are trained with up to 200 million images. Figure 6 shows the architecture of the Inception Resnet V1 model, which was adapted from FaceNet's NN3 model (one of the multiple models google designed) [30]. For our research purposes, we will use the model in Figure 6 as Google's version of the pre-trained models and weights were not available. The architecture was adapted based on the best performing variations of inception models from Google's research. This version was modified to have the same input size as FaceNet's NN3 model and the overall structure of the Inception Resnet V1 [40]. This variation of FaceNet was trained on the VGGFace2 dataset using softmax loss. Softmax loss is applying the softmax activation on the model's output before calculating cross entropy loss. VGGFace2 contains 3.31 million images of 9131 different identities [41]. For the next sections, we refer to FaceNet as the modified version of the Inception Resnet V1 model on Figure 6.

2.4 Siamese Neural Network

Siamese Networks were initially introduced for signature verification tasks in 1993 [42]. An SNN consists of two twin ANNs that process inputs and then compare the outputs. The twin networks for face and image verification tasks most commonly used are identical CNNs, as they are very useful to extract discriminative features from images. However, as discussed in section 2.1.1, the major limitation with CNNs, if we want to construct a face verification model that is not overfitting and obtains accurate results, is the large amounts of data required to train them. This is where we can

Layer	Size-in	Size-out	Kernel	Stride, Padding	Params	ReLU
Conv BN ReLU	160×160×3	79×79×32	3×3×3	2, 0	32×3×3×3 + 32×3	True
Conv BN ReLU	79×79×32	77×77×32	3×3×32	1, 0	32×3×3×32 + 32×3	True
Conv BN ReLU	77×77×32	77×77×64	3×3×32	1, 1	64×3×3×32 + 64×3	True
MaxPool2D	77×77×64	38×38×64	3×3	2, -	0	True
Conv BN ReLU	38×38×64	38×38×80	1×1×64	2, 0	80×1×1×64 + 80×3	True
Conv BN ReLU	38×38×80	36×36×192	3×3×80	1, 0	192×3×3×80 + 192×3	True
Conv BN ReLU	36×36×192	17×17×256	3×3×192	2, 0	256×3×3×192 + 256×3	True
5× Inception A	17×17×256	17×17×256	Inception A	-	388160	True
Reduction A	17×17×256	8×8×896	Reduction A	-	1711104	True
10× Inception B	8×8×896	8×8×896	Inception B	-	6905600	True
Reduction B	8×8×896	3×3×1792	Reduction B	-	3348096	True
5× Inception C	3×3×1792	3×3×1792	Inception C	-	7957680	True
Inception C	3×3×1792	3×3×1792	Inception C	-	1601536	False
AvgPool2d	3×3×1792	1×1792	3×3	-	0	-
Bottleneck BN	1×1792	1×128	-	-	1792×128 + 128×3	False

Figure 6: We created this Inception Resnet V1 diagram based on the MIT licensed model implemented for our research. The architecture is what is refereed to as FaceNet in this thesis.

apply transfer learning. Transfer learning (or inductive transfer) is the concept of improving the performance on a current task by implementing knowledge or a concept learned on a previous task [43]. We implement transfer learning by constructing a SNN that uses pre-trained CNN models with pre-defined weights for the twin networks. This way we can reach state-of-the-art accuracy and ensure our CNNs do not overfit for our very limited dataset.

For an SNN model, the final output layer can use an activation function that improves performance the overall performance by optimizing the training process. For our investigation, we use the sigmoid activation function. This is the mathematical definition of the sigmoid activation function, where z is just a variable of the Sigmoid function $S(z)$: $S(z) = \frac{1}{1+e^{-z}}$. We use a sigmoid activation function because $0 \leq S(z) \leq 1$ for $y \in \mathbb{R}$, meaning our output would be mapped to a probability between 0 and 1. We chose sigmoid activation because face verification is a binary classification task and sigmoid activation is binary classification in a logistic regression model.

When training our SNN models, we choose a loss function so that the model learns. The ideal loss function for binary classification tasks is cross entropy. Binary cross entropy is a loss function that is fundamental to logistic regression, and consequently for binary classification [44]. The purpose of binary cross entropy is to compare the predicted probabilities to the actual class, which is 0 or 1. Because we use it as a loss function, we calculate a penalty score based on the predicted probability and the actual classification value. The equation for cross entropy can be simplified for a binary classifier. We use (1) to train with binary cross entropy loss:

$$-(y \cdot \log(P) + (1-y) \cdot \log(1-P)) \tag{1}$$

y is the binary indicator (0 or 1) to label the actual class of the prediction (0 if two faces do not match, and 1 if two faces do match). P is the probability of the prediction from the model being 1.

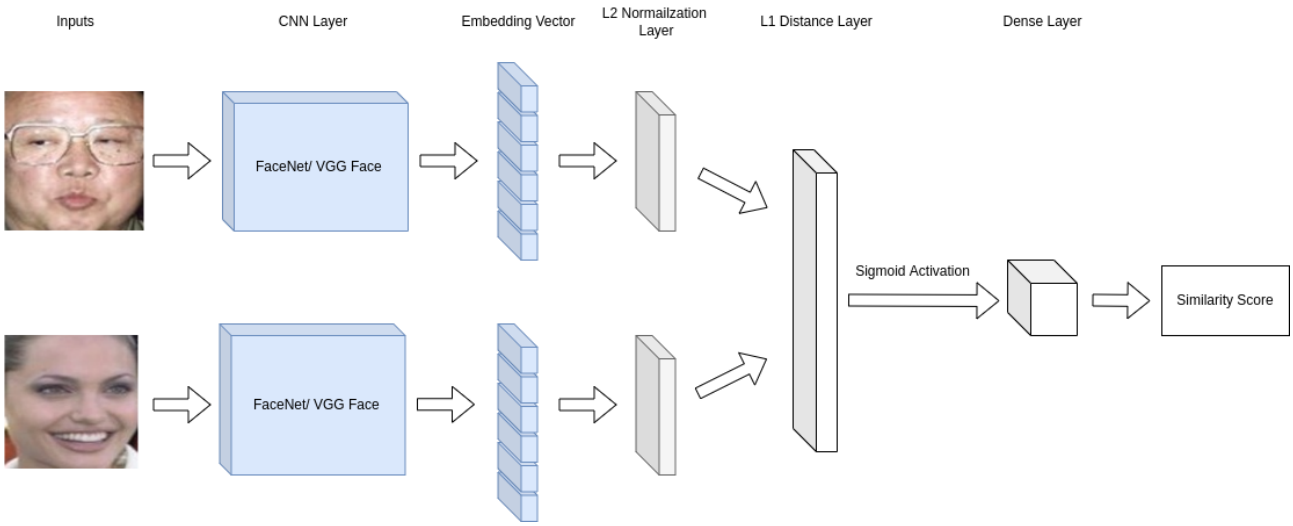


Figure 7: The SNN model for our face verification system.

The research papers in section 2.1 train their facial recognition models using contrastive loss and triplet loss [9], [30], [33]. The research has shown these loss functions improve accuracy for image classification models, especially face classification models, as seen by the results of triplet loss vs cross-entropy [33]. However, in these research papers they train the convolutional networks so that the output embedding vectors of the same class/person are closer together and further apart for different classes/people. Therefore, these two loss functions are powerful tools to train the CNN itself. However, in our model, we apply transfer learning and train our model so that only the fully connected layer is trained. The fully connected layer learns to map the L1 distance of two normalized embedding vectors into a probability, that we can then use to classify whether the faces in the images match or not. This will be discussed further in section 4. We therefore have a binary classification task and do not train the model to change the embedding vectors that are output by the CNN itself. Hence, we use binary cross entropy.

For our research, we will construct a SNN similar to that of Figure 1, which follows a very similar approach to Facebook's DeepFace paper [32]. Figure 7 shows the architecture of our Siamese Network. The following equations are a mathematical representation of the calculations that take place when images are input through the Siamese Network, where t is the number of CNN (1 for the first CNN in the CNN layer and 2 for the other twin CNN), e_t represents the output embedding vector after going through the CNN g_t, D from now on will be referred throughout this thesis as the output size of the embedding vector of the CNN (128 for FaceNet, 2622 for VGG Face), $\vec{X} = (X_1, \ldots, X_D)$ from now on will be referred throughout this thesis as the preprocessed image vector and $\vec{Y} = (Y_1, \ldots, Y_D)$ is the normalized $\vec{X}$:

$$e_t = g_t(\vec{Y}), \text{where } t \in [1,2] \tag{2}$$

$$f_t = \frac{e_t}{\|e_t\|}, \text{ where } ||e_t|| = \sqrt{\sum_{i=1}^{D} e_t^{(i)}},\ t \in [1,2],\ D \in [128, 2622] \tag{3}$$

$$d(f_1, f_2) = |f_1 - f_2| \tag{4}$$

$$SS = S\left(\sum_{i=1}^{D} w^{(i)} d(f_1, f_2)^{(i)}\right), \text{ where } D \in [128, 2622] \tag{5}$$

In Figure 7, first the input image goes through the CNN layer. In the CNN layer, the CNNs are either both FaceNet or both VGG Face (what we refer to throughout this thesis as Siamese FaceNet and Siamese VGG Face respectively), and the layer never has different networks. When we connect the VGG Face from Figure 5 to the Siamese architecture, we remove the softmax activation layer because our facial verification task is a binary classification and not a multi-classification task. FaceNet will be connected to the SNN with the same architecture as shown in Figure 6. We get the embedding vectors by inputting our preprocessed and normalized input images into our CNN in the CNN layer, represented by equation (2). Once we have the embedding vectors, we normalize them in the L2 Normalization Layer using equation (3). With both normalized embedding vectors f_1 and f_2, in the distance layer, we calculate the L1 distance for the two vectors as indicated in equation (4). The L1 distance layer and output layer make up a fully connected layer, where the L1 distance layer has D output nodes fully connected to one dense node in the output layer. The calculation that takes place in this fully connected layer along with sigmoid activation is represented in equation (5). $w^{(i)}$ represents all the weights in the fully connected layer that are learned by the model during training. The similarity score (SS) is the output of the SNN. With the construction of our model, in the next section, we introduce how we investigate sampling methods to help improve the model's performance on difficult test samples.

2.5 Sampling Training Data

When training a model with thousands of samples, it is inefficient and computationally expensive to train on all the permutations of training samples when training our binary classifier. However, only using random samples may not prepare the model for more challenging classification tasks. Tasks such as two images of people that look very similar, but are different people. Therefore, it is important that our model is able to differentiate samples that are more difficult to learn, especially to minimize the number of false positives. In this section, we will discuss triplet loss and how it can be used with triplet mining to sample examples that are most challenging for the model.

2.5.1 Triplet Loss

Triplet loss became a highly regarded loss function for facial recognition tasks after the publication of Google's state-of-the-art model FaceNet [30]. As discussed previously, the embedding vectors output by the CNN are a representation of the input images as a position or a point in the D-dimensional Euclidean space. The goal of triplet loss is to make sure that two images that have the same label (faces that belong to the same person) have positions that are closer together. While on the other hand, two images with different labels (faces belonging to different people) are positioned far apart. However, it is important to point out that the purpose of this loss function is not to train the embedding vectors of each label and group them into clusters. For triplet loss we have triplets, an anchor image (image of a person), a positive image (another image of the SAME person) and a negative image (image of a DIFFERENT person). The requirement of the triplet loss function is that given a set of triplets, meaning the anchor a, positive p and negative n, the negative image should be positioned farther away from the positive by a margin. The following equations indicate how triplet loss is derived:

$$f_u = \frac{e_u}{||e_u||}, \text{ where } ||e_u|| = \sqrt{\sum_{i=1}^{D} e_u^{(i)}}, \ u \in [a,n,p] \tag{6}$$

$$d(f_a, f_v) = \sqrt{\sum_{i=1}^{D} \left(f_a^{(i)} - f_v^{(i)}\right)^2}, \text{ where } v \in [p,n] \tag{7}$$

$$L(f_a, f_p, f_n) = \max(d(f_a, f_p) - d(f_a, f_n) + \alpha, 0) \tag{8}$$

In equations (6)-(8), the subscript a, p and n represent anchor, positive and negative images respectively. In equation (6) we normalize the embedding vector e_u and input that into equation (7). In equation (7), $d(f_a, f_v)$ is how we calculate Euclidean distance between the normalized embedding vectors. In equation (8) we use the Euclidean distances between the anchor and the positive, and the distance between the anchor and the negative to calculate triplet loss. α is the margin we use to make sure the loss function never reaches negative values during training.

2.5.2 Online Triplet Mining

With the triplet loss function, we can group triplets into three categories:

- Easy triplets: $d(a,p) + \alpha < d(a,n)$
- Hard triplets: $d(a,n) < d(a,p)$
- Semi-hard triplets: $d(a,p) < d(a,n) < d(a,p) + \alpha$

For the definition of these categories, we need to analyze the position of the negative, relative to the anchor and the positive. Therefore, we can represent these categories with respect to the negatives as seen by Figure 8.

The concept of online triplet mining, introduced in the FaceNet paper [30], is to compute the most useful triplets for the model to learn during the training process itself. During training each training epoch or certain training epochs, we choose the specific types of triplets we want from the batch of data. This process is useful because as the model learns, we use the updated model from the previous epochs to calculate the triplets for the current epochs.

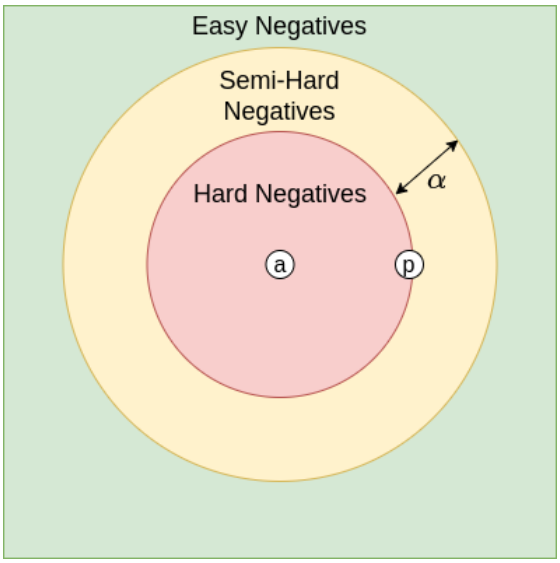


Figure 8: Hard, semi-hard and easy negative regions within the Euclidean space. Points a and p are the anchor and positive.

In an ideal world, we would use online triplet mining with hard triplets in order for our model to learn the most difficult classification tasks (for instance images of people who have very similar faces but are different people; examples would be siblings or doppelgängers). However, because our data set is limited and training batches with only hard triplets is unrealistic with the available samples. Therefore, in each batch, for each anchor, we sample the hardest positives and the hardest negatives among the batch. The hardest positives would be defined as $\max(d(a,p))$, and the hardest negatives would be $\min(d(a,n))$ from the batch. These triplets would be the hardest triplets among the batch, but not necessarily always hard triplets. They can be easy, semi-hard or hard triplets. We sample the hardest triplets from the batch using online triplet mining. The alternative sampling method was calculating the average triplet losses of only hard and semi-hard triplet batches from a batch. However, research has shown between sampling the hardest triplets and this alternative sampling approach, the hardest sampling yields the better performance [45].

3 Data

In this section, we discuss all the details of the data we used for our research. We outline the Data Management Plan (DMP) to summarize our testing, training and demonstration data, and summarize details regarding how our data is handled and used. Then we will summarize the steps taken to preprocess the data for our experiments.

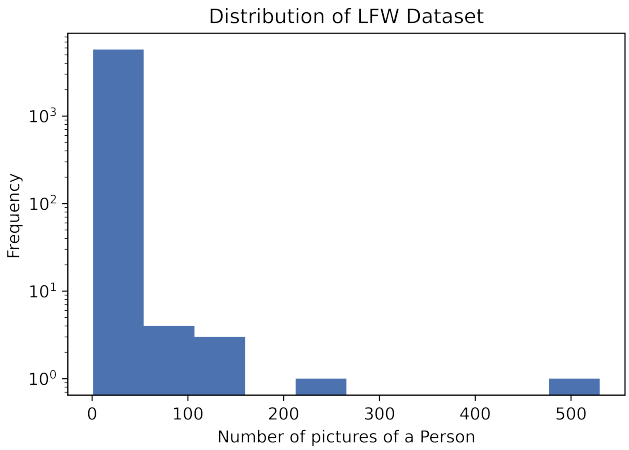


Figure 9: LFW distribution.

3.1 Data Management Plan

For our research, we use the DMP template from Horizon 2020 FAIR [46]. The purpose of the DMP and by following this structure is to present our data as findable, accessible, interoperable and reusable (FAIR).

1. **Data Summary:** For our research, we do not have access to a data set that has images of ID Photos and a corresponding image (i.e., a selfie) of that same person. Therefore, we chose to use the Labelled Faces in the Wild (LFW) dataset which is a public benchmark for face verification [29]. The dataset contains 13233 images of 5749 people, where 1680 of those people had two or more distinct images in the data set. The distribution of the data set can be seen in Figure 9. The number of distinct images per person is heavily skewed, for example with over 4000 people having only a single image. For research purposes, we will use images from the LFW dataset for training, validation and testing.
   As an addition to this data set, we have created a small demonstration dataset. This data set contains 96 images, from 16 different people. Each person has a single ID photo and 5 selfies facing different angles (straight, right, left, top, bottom). This demonstration set is to apply transfer learning with the models trained on the LFW dataset and evaluate its performance on the demonstration set.
2. **FAIR Data:** We have a main CSV file that contains the file path and the class label of each image in the entire LFW dataset. The data was organized so that online triplet mining can effectively be applied and images of people are more evenly distributed to avoid overfitting of certain people with distinct features. We ordered the data so that in a batch of 100 it can only have up to a maximum of 30 images of a single person and that each batch has at least 10 positives overall so that we can sample at least 10 triplets. The batch size of 100 was chosen as it provided a fair distribution of images the throughout our data, and the optimized training batch was still not calculated at the time. After ordering the data, the data is split into 5 different sets of training, validation and testing sets that are stored in different CSV's in the same format. We split the data this way because we applied 5-fold cross validation (discussed in section 4.2).

For our Siamese architecture we created training, testing and validation data for each cross validation iteration. The Siamese data is saved in CSV file, where each row has the file path for the two input images, class label of the two images and a binary category (0 or 1) to indicate whether the two input images belong to the same class or not. It is important to know that all Siamese data files are distributed so that the number of positives (two image inputs from the same class/person) are equal to the number of negatives (two input images are from different classes/people). We achieve this by placing each image in the dataset with a positive and a negative. To avoid an imbalance in the Siamese data, we remove images from the dataset that have only one image of that person, because they have zero positive images. However, because there is a skew in the distribution of data as discussed in the **data summary**, we created a subset of training, validation and testing data from the entire LFW data that was organized. We have CSV files for people who only have 1 image available, and another file for people who have exactly 5 images available in the entire LFW dataset. The purpose of this is discussed in section 4.3.1. Finally, we create CSV documents for the demonstration set with the column structure as the Siamese data, however, the demonstration set is only for testing and therefore was not split up into different sets for cross validation.

The LFW is openly accessible to the public [29], however demonstration set and all the CSV files containing organized data and classification labels are only accessible to the employees at Verifai. The data along with descriptions will be made available in the shared network of the company.

3. **Allocation of Resources:** There is very minimal cost in making the data FAIR in our project. The storage space required to store the data in the network storage space is very little (no more than 2 GB). It will be our responsibility to organize and describe the data for future use/ implementation. While Verifai will be responsible for providing the storage space and making decisions on how long the data will be stored in the company network.
4. **Data Security:** During the research project, multiple copies of the data have been stored on multiple hard drives and a backup cloud server. The images from the LFW dataset are public and therefore there is no privacy issue. The demonstration set was provided by volunteers and allowed their data to be used for testing. Only employees of Verifai will have access to these images in their secure network.
5. **Ethical Aspects:** The images of the demonstration set were provided by Verifai employees and other volunteers. The data was provided under the assumption they will only be used for testing the model and therefore must not be used to train any current or future model. Furthermore, the data shall not be published without additional permission requested from the original employee/volunteer. This is because there was no agreement of a sort to use the images for this purpose, and therefore would be a privacy issue to present/publish the images anywhere without permission.
6. **Other:** Not applicable.

3.2 Data Preprocessing

When working with neural networks, we need to format our input data so that they fit the requirements of our ANN. Furthermore, we preprocess the data to make it easier for our model to interpret and use the input data to achieve more accurate results.

Here are the steps that we take to prepare our data for training, validation and testing:

1. **Detect and Crop Faces using MTCNN**
   The first step in preprocessing our data is to crop the faces from the images themselves. This is because using the LFW dataset there are many images with different backgrounds and if we were to input the entire image into our SNN we would compare irrelevant features in the network like background, hair, faces of other people, and other objects that are insignificant and a hindrance for face verification. Therefore, we need to first detect the face within an input image and crop out the face from the entire image. We do this by inputting our images into a Multi-task Cascaded Convolutional Neural Network (MTCNN) [47]. The MTCNN can recognize faces and return landmark locations on the face such as the eye, nose and mouth.

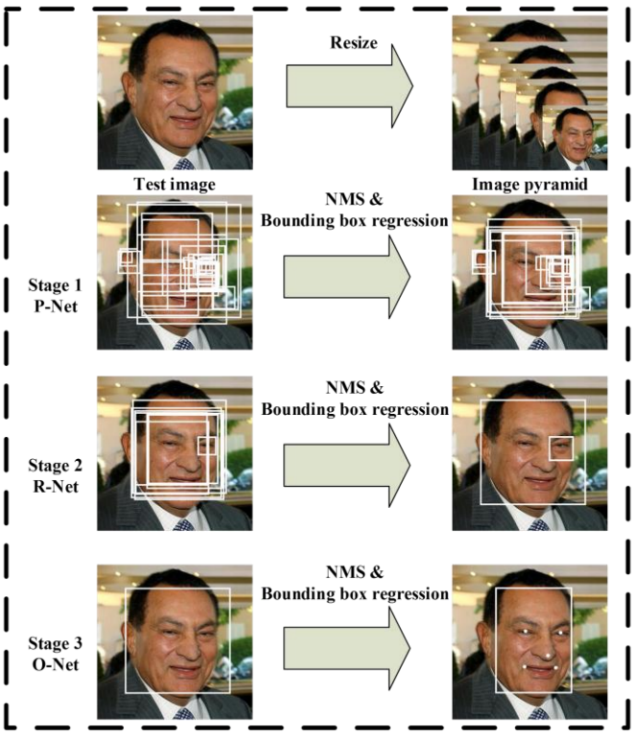


Figure 10: The figure describes the steps that occur when an image is input into an MTCNN [47].

Figure 10 shows the pipeline when an image of a person is input into the MTCNN. When an image is input, it initially starts by resizing the image into different scales and creates an image pyramid. Then follows the three stages [47]:

Stage 1 is using a P-Net, which is a CNN to retrieve the candidate windows of the face and their Bounding Box Regression (BBR) vectors. BBR vectors are a prediction of the localization boxes of features of the face. With this information, the candidates are calibrated based on the regression vector estimations. After which the Non-maximum Suppression (NMS) is applied and merges the overlapped candidates to remove redundant information. The final output at this stage would be all candidate windows after refinement.

Stage 2 places all the candidate windows into the R-Net, which removes numerous false candidates. Calibration with BBR and NMS is then applied. At this stage, if the image is classified as a face, the outputs are the bounding box of the face and the localizations of the facial landmarks.

Stage 3 is similar to the previous stage, but the O-Net is used to describe the face in more detail

and identify face regions with more supervision. The network specifically outputs the five facial landmark positions.

Once our image goes through these stages, we used the bounding box output by the MTCNN to select the region of the image that has the face and crop it out, then save the cropped image into a new JPEG.

2. **Resize images**

   Once we have obtained the cropped face images, it needs to be resized so that they fit the input size of our Siamese networks. The Siamese FaceNet has an input size of 160×160 and the Siamese VGG Face has an input of 224×224. We resize the images to the appropriate input size when preparing the data using TensorFlow (technology discussed in section 4.1).

3. **VGG16 preprocessing**

   Once the images have been resized to the correct input size, we apply VGG16 preprocessing to the images. The VGG16 is a CNN created by the Visual Geometry Group from Oxford, and it won the ILSVRC(ImageNet) competition in 2014 [48]. We use the same preprocessing method in order to make it easier for our models to be able to extract features from the inputs and perform better. What occurs during VGG16 preprocessing is that the images are first converted from RGB to BGR, and then each color channel is zero-centered with respect to the ImageNet data set.

4. **Normalize values**

   The final step of preprocessing our data is to normalize the data between the range of -1 and 1. By applying equation (9) to the tensors produced from step 3, we are able to achieve this:

$$Y^{(i)} = 2 \cdot \left(\frac{X^{(i)} - \min(\vec{X})}{\max(\vec{X}) - \min(\vec{X})} \right) - 1 \tag{9}$$

   where Y_i is the normalized data at the i^{th} vector position.

4 Experimental Setup

In previous sections, we discussed the problem we are trying to research, and understand the concepts we need to understand in order to conduct our research and the data that needs to be prepared for the research. In this section, we will use all that prior knowledge and discuss the experiments we ran to first familiarize ourselves with the technology and concepts, outline the training methodology we designed for our research, and then finally discuss the evaluation metrics for our experiment.

4.1 Tools and Technologies

For the preparation of data, we used many python libraries. We used an MTCNN library along with image manipulation packages from python to crop out the faces from the dataset and store them as a new JPEG. Python libraries were used to resize the image. To apply VGG16 preprocessing, we use functions that are available with the Keras application for the VVG16 model. The preprocessed images are in the format of NumPy arrays, to which we apply the normalization equation (9) to.
In order to create our models, we used TensorFlow which allowed us to import the Keras library. With Keras, we were able to construct both FaceNet and VGG Face and upload their pre-trained weights. Then we used those models and constructed our Siamese architecture and add the L1 distance and output layer to the twin CNNs. Using TensorFlow, we made the weight of only the fully connected layer trainable, while all other layers were frozen.
For training, we required GPUs that initially trained experimental models to see if the design was correctly implemented. Furthermore, we implemented 5-fold cross validation for 2 models with 2 training variations; Siamese FaceNet, Siamese VGG Face and both variation with hard sampling (discussed in section 4.2). Therefore, our training required powerful GPUs to speed up the process. For the training of our model, we used the Peregrine Cluster from the University of Groningen (with Nvidia Tesla v100 and Nvidia Tesla K40 GPUs). We also trained our models using two Nvidia GeForce 1080 Ti GPUs provided by Verifai.

4.2 Training

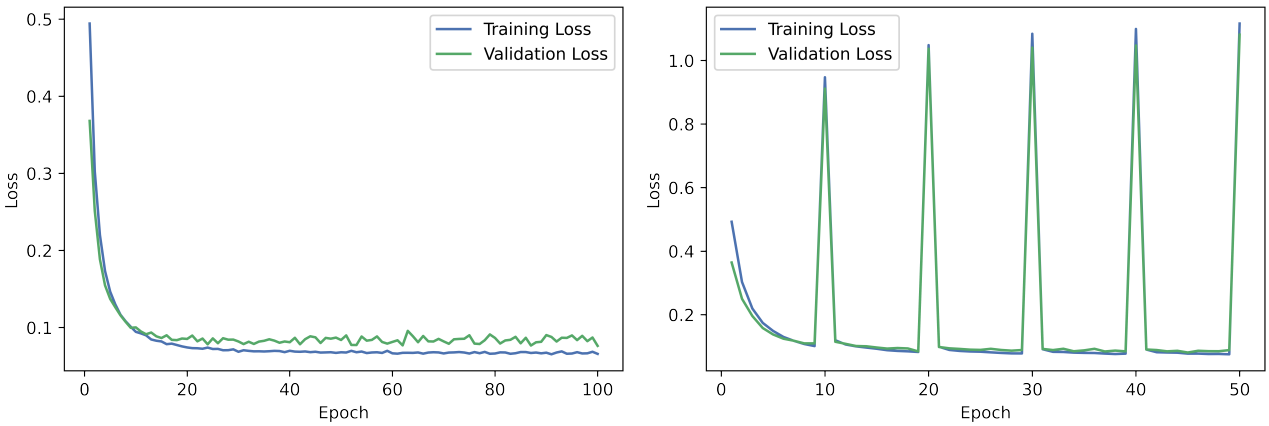
To test the performance of our models with our limited dataset, we apply 5-fold cross validation. Additionally, cross validation is also to avoid overfitting our models to our dataset. We split our data into 5 splits, so 80% training and 20% testing data. The validation data is retrieved from the last 20% of the training data from each cross validation iteration. Once the data is split appropriately, we apply data preprocessing as explained in section 3.2 before starting training for our models.
We train two models, our Siamese FaceNet and Siamese VGG Face (described in section 2.4). During the training process of both models, we make sure that only the weights of the fully connected layer between the L1 distance layer and output layer are trainable (meaning 128 trainable weights for Siamese FaceNet and 2622 trainable weights for Siamese VGG Face), while all other trainable

weights of both models are frozen. To identify the hyperparameters that best suit the training of our model, we take the training set and create a subset for the validation set. We use these sets to find the right hyperparameters by training them models for 20 epochs and monitoring both accuracies and losses for both training and validation.
To train our models, we set an initial learning rate of 0.01 and train for 100 epochs. During training, we created a learning rate scheduler that applied exponential decay every 10,000 steps with a base of 0.9. We use the Adam optimizer, which is robust and well-suited to a wide range of non-convex optimization problems, and obtained very good results for online convex optimization frameworks [49]. Furthermore, the empirical results from the research show that Empirical results demonstrate that Adam works well in practice and compares favorably to other stochastic optimization methods. We train our model using binary cross entropy loss and monitor the accuracies and losses for training and validation to ensure that the model is learning and making sure no overfitting occurs.

4.2.1 Training with Online Triplet Mining

For our model to correctly differentiate challenging test cases (specifically images of different people who visually look very similar), we applied online triplet mining (OTM). We train both Siamese FaceNet and Siamese VGG Face using the same 5-fold cross validation split. For our implementation, we mine triplets of the hardest positives and hardest negatives from each batch, as discussed in section 2.5.2. When training the Siamese FaceNet and Siamese VGG Face with OTM, what occurs is that we start with the initial SNN that has the pre-trained CNNs, but the fully connected weights between the L1 distance layer and the output layer are untrained. The models are then trained for 50 epochs, where every 10th epoch applies triplet mining of the hardest samples. We apply the same hyperparameters as described in the training for the previous section (4.2), also using exponential decay for the learning rate and binary cross entropy loss. Figure 11b shows the learning curves between regular training (a) and training with OTM (b). For the regular training process, we see an expected learning curve with no signs of overfitting. During every 10th epoch, with OTM applied to the training process, the loss is much higher because the hardest positives and hardest negatives are sampled, resulting in the increased binary cross entropy loss. However, OTM training also has no indication of overfitting.

How the triplets are calculated: In our implementation of online triplet mining, at every 10th epoch during training we calculate the hardest positive and negatives of a batch. This is done by first calculating all the embedding vectors of all the images in our batch. With the list of embedding vectors, we calculate the pairwise Euclidean distance between all the vectors and place it into a 2-D matrix. Each row and column of the matrix corresponds to an image. For each row, the embedding (which corresponds to an image) is treated as an anchor image. We check the row of that anchor to find all the images that have the same label as our anchor and get a list of positives. From the positives, we retrieve the image with the largest Euclidean distance to the anchor (the hardest positive). Similarly, for the negatives, we check for all the other images in the row that have a different label and get a list



(a) Siamese FaceNet training

(b) Siamese FaceNet with OTM applied to training

Figure 11: The learning curves for Siamese FaceNet regular training (a) and training with online triplet mining applied (b). The curves outline the training process of iteration 1 of cross fold validation.

of negatives. We then retrieve the image that has the smallest Euclidean distance to the anchor (the hardest negative). This is how the hardest triplets are mined for this anchor, and then this process is repeated for the rest of the rows. We make sure to skip anchors that have no positives in the batch since they cannot form a triplet (however as discussed in section 3.1, we try to minimize this by making sure images that have positives have up to 30 positives for a single person in a batch of 100 and that each batch has at least 10 positives overall).

4.3 Evaluation

In this section, we investigate the metrics best suited to evaluate our results for both the LFW dataset and our demonstration set and the method used to calculate the optimal thresholds for our models.

4.3.1 LFW Dataset

For the LFW data, we used two evaluation metrics to display our results, the F1-score and the False Acceptance Rate. The F1-score is used in binary classification as a means to measure a model's accuracy, and it does this by calculating the harmonic mean between precision and recall. Precision is the proportion of positive predictions that were actually correct, and recall is the proportion of actual positives that were correctly predicted. Maximizing the F1-Score leads to higher performance of our model. Note throughout this thesis we refer to TP as true positives, FP as false positives, TN as true negatives, and FN as false negatives. The following equations explain how we calculate the F1 score,

where *Pr* and *Re* are precision and recall, respectively:

$$Pr = \frac{TP}{TP+FP} \tag{10}$$

$$Re = \frac{TP}{TP+FN} \tag{11}$$

$$F_1\ Score = 2 \cdot \frac{Pr \cdot Re}{Pr+Re} \tag{12}$$

$$\tag{13}$$

We have split our training data into an equal number of actually correct positives and negatives and discussed in the DMP in section 3.1. This was done so that there is no imbalance between the correct positives and the correct negatives, which can result in a misleading F1-Score because of how precision is calculated. This is explained in more detail in section 4.3.2, where we can't use the F1 score due to the imbalance in the correct positives and negatives.

The second metric we use to evaluate our trained models is the False Acceptance Rate (FAR). In testing a biometric system, it is very important to look a FAR metrics [50]. FAR is the percentage of false positives out of all the correct negatives. Therefore, it is a way to measure all the faces that are able to bypass our system; the percentage of all the faces that should not match, but our system incorrectly indicates that they do match. This is a very crucial metric to avoid fraud and identity theft, hence the FAR results will be of the utmost importance when presenting our facial verification model. Minimizing the FAR leads to higher performance of our models. It is calculated using the formula in equation (14):

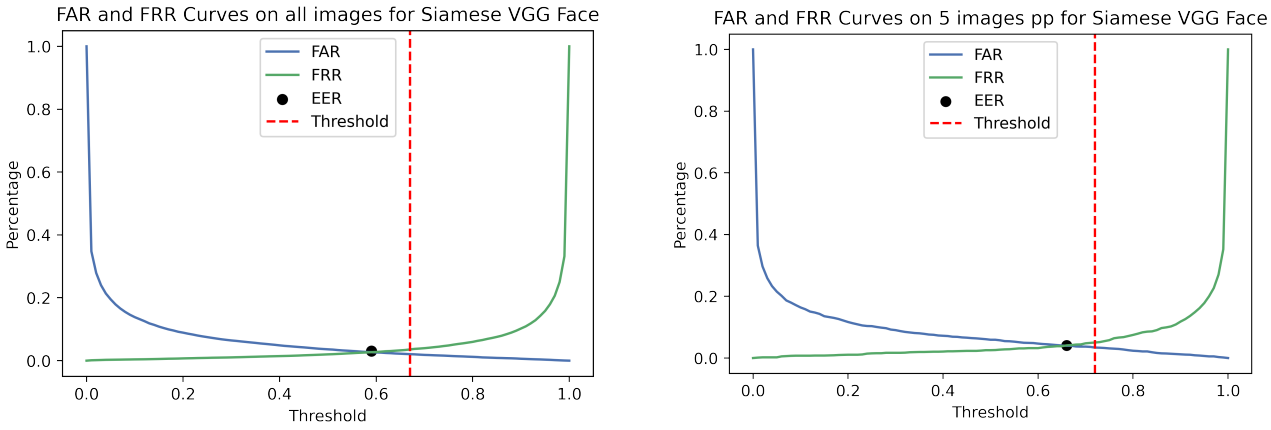
$$FAR = \frac{FP}{FP+TN} \tag{14}$$

Before running calculations for these two metrics, a threshold needs to be set for our models. Within biometric systems, calculating the Equal Error Rate (EER) is common practice to find the threshold of a model [50]. The EER is the point at which the FAR and the False Rejection Rate (FRR) are equal. Figure 12 shows the intersection between FAR and FRR, which marks the EER. FRR and EER are defined in equations (15-16):

$$FRR = \frac{FN}{FN+TP} \tag{15}$$

$$EER : FRR = FAR \tag{16}$$

Calculating the EER is a suitable method for assigning a threshold for most biometric systems. However, due to the distribution of the LFW dataset, and maintaining a balance between the correct positives and correct negatives, our model is trained with a slight bias. It is biased because our threshold becomes skewed depending on the type of training set we use to identify the EER. Looking at Figure 12, we can see that the threshold is shifted higher for the subset of the LFW dataset with 5 images per



(a) (b)

Figure 12: Example FAR and FRR curves for the Siamese VGG Face. (a) are the curves calculated from the training set of all images in the LFW dataset, and (b) are the curves calculated from a subset of the training set of the LFW dataset, where the condition of the subset is that all the people have exactly 5 images in the dataset.

person vs the entire LFW dataset. Furthermore, looking at Figure 13 and the corresponding FAR AUC values from Table 3, we see that the AUC is the lowest for all data in the LFW dataset (0.0614 for Siamese VGG Face) and then increases for the subset of data with 5 images PP (0.0805 for Siamese VGG Face) and 1 image PP (0.0805 for Siamese VGG Face). This indicates that our model is better at reducing the FAR for a dataset with a distribution like our LFW distribution, which has numerous images for the same faces, and therefore the EER is present at a lower threshold. Furthermore, when a set with a smaller number of images per person is available, the EER is at a higher threshold. In a real-world application of our system, it is much more realistic for there to be very few images of the same person being input into the system. Therefore, the EER would not be the optimal indication of the threshold for our system. After discussion with Verifai and analysis of the behavior of the thresholds between Siamese VGG Face and FaceNet, we decided to define the threshold by the point at which $FRR = 2 \times FAR$ instead of the EER. Figure 12 contains red dash lines for both sub-figures that represent this threshold for the Siamese VGG Face model.

4.3.2 Demonstration Set

One of the sub-goals of our research is to apply transfer learning with our trained Siamese models, and apply them to our demonstration set (explained in section 3.1). The demonstration set is a small-scale example of the actual tasks our verification system needs to run. The actual task is that our verification system needs to be able to verify if the biometric photo from an ID document matches the face of a person from a photo for any pair of images unseen by the model. The way we organize our demonstration set is by testing our models on 2 subsets of the entire demonstration set. The first subset is pairs of ID photos and Straight Face selfies, as shown in Figure 14. The second subset is

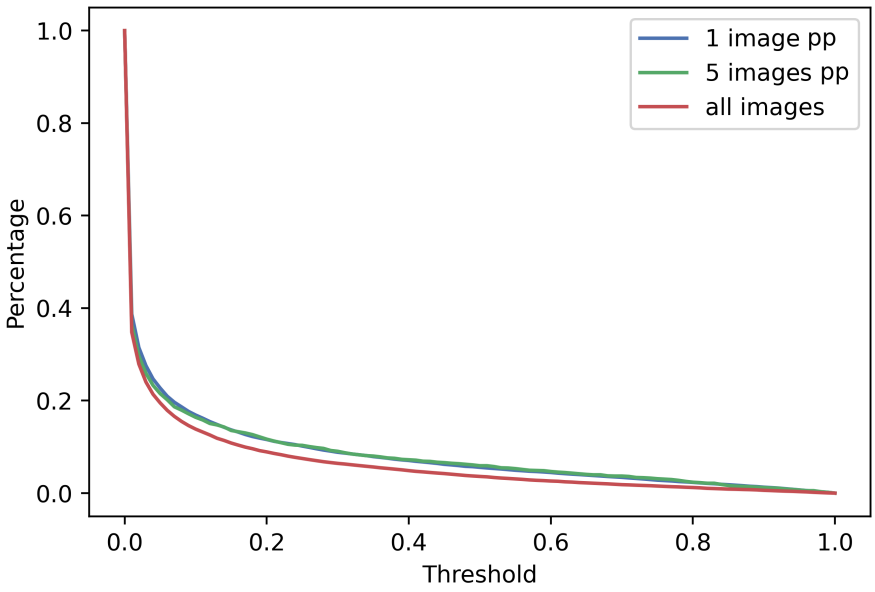


Figure 13: The FAR curves for the Siamese VGG Face for different training sets (set with exactly 1 image per person, exactly 5 images per person and the entire LFW dataset).

pairs of ID photos and other angle selfies (top, bottom, left and right angles of people's faces). For both these subsets, the ratio of positives to negatives of image pairs is 1:15, meaning that there is a big imbalance in data. Due to this imbalance in data, we noticed that the F1-Score was not a fair performance metric with our data. This can be explained using the confusion matrix in Table 1. The confusion matrix is calculated using the ID photo and straight face subset on the Siamese FaceNet model. The F1-Score finds the harmonic mean between the precision and recall, however, due to the imbalance of positives to negatives, precision is a misleading metric. Equation 10 shows the equation for precision, where it considers the number of false positives. However, due to the imbalance in data, the number of false positives (53) is greater than half of the true positives (80) and leads to a precision of 63.88%. This seems like our model obtains much poorer results, but in actuality, it performs well. Because when looking at only the actual values individually, we obtain a very small percentage of false positives (FAR) and a very small percentage of false negatives (FRR). A FAR of 4.4% and a FRR of 0% is obtained. The issue with the F1-Score is that when calculating precision, small increases in the false positives will greatly reduce precision due to the very small number of actual positives in our demonstration set. Therefore, due to the data imbalance, it would be more appropriate to analyze the results of the positives and negatives individually using FRR and FAR. The metrics are also a standard when analyzing biometric models in machine learning [50]. We want to minimize both FRR and FAR to achieve greater performance of our models for the demonstration set.

We do not have access to a training set for our demonstration set as it is too small. Therefore, we apply transfer learning to calculate the appropriate threshold. The thresholds we set for all models when testing our demonstration set are the same thresholds calculated in section 4.3.1. We specifically use

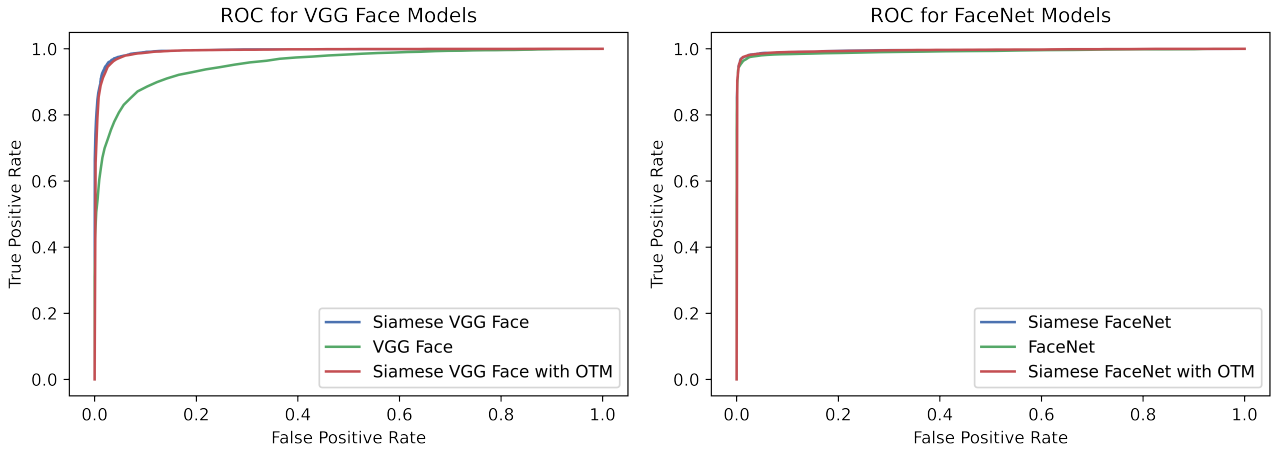
(a) ID photo (b) Straight Face

Figure 14: An example of what a positive pair from the demonstration set would look for, ID and Straight face [51]. Examples from our original demonstration set are not included in this report for privacy reasons.

Table 1: Confusion matrix for ID and Straight Face for the Siamese FaceNet.

		Actual Values	
		Positive	Negative
Predicted Values	Positive	80	53
	Negative	0	1147

the thresholds calculated using the subset of the LFW training data containing people with exactly 5 images. This is because in our demonstration we have a similar distribution, where there are exactly 5 selfies of each person.



(a) (b)

Figure 15: ROC graphs constructed using the validation set. OTM stands for online triplet mining.

5 Results and Discussion

This section describes the results we obtained using the experiment introduced in section 4. Finally, we analyze and evaluate our results and what they indicate in our research.

5.1 LFW Results

Figure 15 displays the ROC curves of all variations of our models. These curves and the corresponding AUC values, listed in Table 2, were calculated using the validation set. A ROC is a visual representation of our model's ability for binary classification tasks. The interpretation of the ROC curve is that the closer our curve is to the top-left corner, the better the specific model performs for its classification task. The AUC values listed in Table 2 is a numeric representation of this same performance, where an AUC value closer to 1 means higher accuracy. Looking at the AUC results, we see that the original FaceNet model and our versions of Siamese FaceNet and Siamese FaceNet trained with online triplet mining (OTM) are almost indistinguishable in terms of performance. Although the AUC values do show that the Siamese FaceNet variations show a slight improvement from the original FaceNet model. With the VGG Face, we see a much greater with the Siamese VGG Face models, which is much more distinguishable.

Table 3 depicts all the results for our Siamese models and the variations trained with OTM. Furthermore, it details the F1-Scores and FARs for each sets of testing data (all data, subset of people with exactly 5 images, subset of people with exactly 1 image). For all sets, we see that the F1-Scores of the Siamese FaceNet and Siamese FaceNet with OTM have less than a 0.2% difference between each other, which is just standard error and therefore indistinguishable difference between the two models. Similarly, there is no notable difference between the FAR values, where the greatest difference is less than 0.36%. The difference between Siamese VGG Face and Siamese VGG Face with OTM on the

Table 2: AUC values for the ROC curves, as visualized in Figure 15. An AUC value of 1 indicates the highest accuracy possible. OTM stands for online triplet mining.

	AUC
Siamese VGG Face	0.9939
Siamese VGG Face (OTM)	0.9923
VGG Face	0.9547
Siamese FaceNet	**0.9951**
Siamese FaceNet (OTM)	0.995
FaceNet	0.9919

other hand is much more notable. Implementation of OTM for the Siamese VGG Face decreases the F1-Score by 1% up to 1.3% for the different sets. While the FAR score increases from 0.8% up to 3% for the different sets.

Looking at the results from the AUC of our ROC curves, there is a clear improvement as expected when creating a Siamese architecture of our two original CNNs (VGG Face and FaceNet). Furthermore, the F1-Score of the Siamese VGG Face reaches up to 94.74% and the Siamese FaceNet reaches up to 97.53%. While FAR percentages were as low as 3.34% and 0.89% for the Siamese VGG Face and Siamese FaceNet respectively. Both models obtained high-performance results for our face verification tasks, when tested on LFW testing data.

The goal of OTM was to yield higher performance than the regular SNN training. However, looking at the results we can see that for the Siamese VGG Face, the model performs worse after training with the hardest triplets. We see that the F1-Score decreases and the FAR increases, both indicating a reduction in performance. After thorough analysis, the most probable reason for this occurrence is due to the structure of our network. One layer is not enough for our model to be able to learn how to differentiate the most difficult samples. The OTM method was designed to train CNNs to output positions with greater distances for harder negatives and closer distances for harder positives. During our training, we took a different approach where our models learn to take the distance of two embedding vectors and learn to classify them as 1 or 0 (matching or not matching). From our understanding, the one fully connected layer of all our models does not have enough weights that can properly learn to differentiate between regular examples and hardest examples. This resulted in the weights being slightly reset/randomized without proper learning. Another indication of this is from the difference between the Training FAR AUC values from Table 3, where the values are all at least 5x greater when OTM is applied to the models. This shows that at each threshold, in most cases the FAR will be higher, indicating that the single fully connected layer was not enough for the model to learn to accurately verify faces for the hardest samples. For the Siamese FaceNet, we saw that there was a very slight increase in performance when OTM was applied, but it was too small to be a notable improvement and is likely a cause of the standard error. Therefore, OTM is better suited when we train the CNNs directly and not as useful when training our model to classify the distance of two

Table 3: Results for each set of data (The entire LFW dataset, a subset of people with exactly 5 images, a subset of people with exactly 1 image). The F1-Score and FAR are calculated using LFW test data. The Training FAR AUC is the FAR curve's AUC values calculated using the training data.

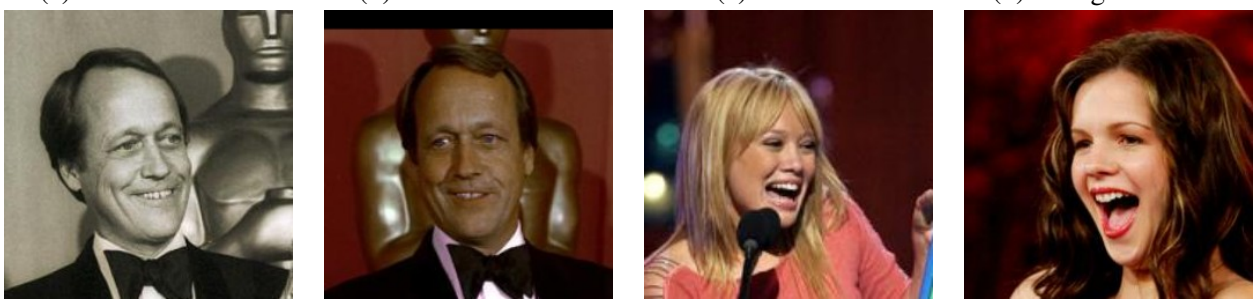
		F1-Score (%)	FAR (%)	Training FAR AUC
All data	Siamese VGG Face	94.7374	3.8092	0.0614
	Siamese VGG Face (OTM)	93.7192	4.6278	0.3757
	Siamese FaceNet	**97.5307**	**1.5807**	0.0342
	Siamese FaceNet (OTM)	97.6099	1.4626	0.5035
5 PP	Siamese VGG Face	94.107	3.3628	0.0805
	Siamese VGG Face (OTM)	92.7743	6.3717	0.425
	Siamese FaceNet	**97.3897**	**0.885**	0.032
	Siamese FaceNet (OTM)	97.5654	0.531	0.4899
1 PP	Siamese VGG Face	-	3.3407	0.0805
	Siamese VGG Face (OTM)	-	4.8882	0.4236
	Siamese FaceNet	-	**1.0563**	0.0343
	Siamese FaceNet (OTM)	-	1.0071	0.4937

embedding vectors.

After calculating our results, we took a deeper look into the irregularities and classification errors made by our models. Our observations were as follows: We first analyzed both the Siamese VGG Face and Siamese FaceNet for image pairs of the same people that were incorrectly classified as 0 (false negatives). There were common characteristics in the images causing them to be incorrectly classified. These characteristics are differences in make-up, different angles of the face, facial hair, lighting, glasses on one image, and objects covering the face such as trophies awards and hands. The characteristics make image pairs of the same people look quite different, and therefore the Siamese models are not able to classify them as the same person. Another observation is that during the data preprocessing, the MTCNN incorrectly crops faces or crops the wrong person's face from the image. Hence, it makes complete sense why this small percentage of incorrectly cropped pairs would be classified as different people because the faces in the images are of different people. (a) and (b) from Figure 16 show images of Nicole Kidman with different make-up for a movie role, where she looks quite different; this pair was incorrectly classified as 0 by both models. For both the Siamese VGG Face and Siamese FaceNet, there were image pairs of the different people that were incorrectly classified as 1 (false positives). The common characteristics that we saw between these image pairs were their similar facial structures and expressions. We noticed that siblings and other family members were also incorrectly classified as the same person. (c) and (d) from Figure 16 are an example of a pair who are also brothers, that were incorrectly classified as the same person by both models.

Since the Siamese VGG Face model performed worse than the Siamese FaceNet, we wanted to under-

(a) Nicole Kidman (b) Nicole Kidman (c) Jeb Bush (d) George W. Bush



(e) George Roy Hill (f) George Roy Hill (g) Hilary Duff (h) Amber Tamblyn

Figure 16: Examples of errors from our models on the LFW test set. Both Siamese FaceNet and Siamese VGG Face incorrectly classify (a) and (b) as different people, and incorrectly classify (c) and (d) as the same person. Only Siamese VGG Face incorrectly classifies (e) and (f) as different people and incorrectly classifies (g) and (h) as the same person. (a)-(h) are the original images before they go through data preprocessing.

stand why. When observing the results on the Siamese VGG Face for classifying like pairs (images of the same person), we noticed that lighting and skin color in the images made a difference. Lighting affects the skin color, shadows in the face, color perception of facial features and therefore the Siamese VGG Face was not able to correctly classify images of the same people that had differences of these characteristics in the image pairs. This is likely because the original pre-trained VGG Face model must have not trained to overcome these potential differences between like image pairs. The embedding vectors greatly affect the classification abilities, and lighting was not specifically addressed when introducing the VGG Face's training images. (e) and (f) from Figure 16 are an example of an image pair of the same person, that could not be classified as the same person by the Siamese VGG Face. It is clear that (e) is a greyscale image with different lighting and a different angle of the face compared to (f), and this was registered by the model to be very different people. In the case of false positives, it was the same issue with the lighting and skin color. The Siamese VGG Face had more false positives compared to the Siamese FaceNet, where the images of different people had very similar lighting, skin color and facial expressions. (g) and (h) from Figure 16 is an example of this, where the image pairs for different people that were classified to be the same person by the Siamese VGG Face. To understand how well our model performs on an ID verification task, we apply our model to our demonstration set and discuss the results in the next section.

Table 4: Results for the demonstration set. The demonstration set consists of two sets tested. The first set where each image of the biometric ID photo

		FRR (%)	FAR (%)
ID and Straight Face	Siamese VGG Face	5	18.75
	Siamese FaceNet	**0**	**4.4167**
ID and Other Angles	Siamese VGG Face	6.25	19.2708
	Siamese FaceNet	**2.1875**	**3.6875**

5.2 Demonstration Results

Table 4 shows the calculated results for the demonstration data. The results were only calculated for the models without OTM during the training process, due to the results in section 5.1 showing no improvement with this training variation. The Siamese FaceNet outperforms the Siamese VGG Face in both metrics for both demonstration subsets. This is to be expected, looking at our results for the LFW dataset in section 5.1. Both models are much better at minimizing the percentages of false negatives (FRR), than false positives (FAR). The Siamese VGG Face has especially high percentages of FAR, which can result in a major security risk in a verification system. The Siamese FaceNet performs much better. Overall, considering that we are applying transfer learning where our models are trained with no ID photos, the Siamese FaceNet performs very well. The FAR when testing Siamese FaceNet on ID photos with other angles instead of straight faces is lower. This is likely because the images of other angles are more different than straight face images, and therefore more different than the ID photo. Hence, it is less likely to register those image pairs as a false positive. Although the opposite is seen looking at the Siamese VGG Face FAR results. This could simply be due to standard error, or another possibility is that our Siamese VGG Face model learns features that could easily be tricked using different angles (this is a major consideration to make, especially when trying to avoid fraud). Due to the sample size of our demonstration data, we cannot draw any major conclusions about these discussion points. What we can strongly conclude is that the Siamese FaceNet outperforms the Siamese VGG Face and at the threshold we learned from the LFW training set, the FRRs are lower than the FARs.

6 Conclusions and Future Work

This project was inspired by the research conducted on facial recognition models and the ever-growing need for remote ID verification systems. With the recent increase in online identity-related fraud and crime, the demand for highly accurate ID verification models is only growing. Through our investigation, we aimed to create a face verification model, that can be applied to any biometric ID photos and corresponding selfie images. We started our investigation with CNNs and their ability to retain facial features from images. Then we explore SNNs as a one-shot learning technique that has powerful applications for face verification tasks. We construct our Siamese FaceNet and Siamese VGG Face models by applying transfer learning and utilizing high-performance CNNs. We also consider the implications of data sampling and investigate the implementation of online triplet mining. Our models required training, validation and testing data, therefore we created a Data Management Plan. We then formulated an experiment to train our Siamese models using 5-fold cross validation. Lastly, to analyze our research we evaluated the performance of all model variations on both the LFW test set and our custom demonstration set.

Through our research, we set out to make three major contributions. Our first contribution was creating face verification software that uses a one-shot learning model. We were successfully able to implement two types of SNN models (Siamese FaceNet and Siamese VGG Face) that used different CNNs to create embedding vectors of face images and classify their similarity of the inputs using the distance of the embedding vectors. Our models use the same concepts of one-shot learning, allowing us to train our networks with a smaller dataset. The secondary contribution of our research and our primary focus is to outline what steps can be taken for our constructed models to accurately verify faces and minimize the number of false positives. During our investigation, we studied the architecture of Siamese Networks and how they used a CNN to calculate embedding vectors as positions for images. Our research lead us to the understanding that facial recognition CNNs reach over 95% accuracy, only when trained with millions of images. We did not have access to such a large dataset for the ID verification task, nor the resources required to conduct such training, and therefore we focused our efforts on transfer learning. We first used transfer learning and constructed our SNNs using highly accurate pre-trained CNNs (FaceNet and VGG Face) and trained only the fully connected layer of our SNN so that our models have been adapted to our data. With this process on the LFW dataset, we were able to obtain F1-Scores of up to 97.53% and 94.74% for the Siamese FaceNet and Siamese VGG Face models respectively. Furthermore, reduce the percentage of false positives (FAR) up to 0.885% and 3.36% for the two models. Finally, we applied transfer learning using the models trained on the LFW dataset and tested our demonstration set. This also lead to strong results, where the Siamese FaceNet achieved a FRR of 0% and a FAR of 4.41%. While the Siamese VGG Face did not perform as well and achieved a FRR of 5% and a FAR of 18.75%. However, with our results on the demonstration set, we cannot claim that we will be able to produce similar results in a real-time face verification system. This is because our demonstration set is very small and limited in terms of

the types of photos that can be put into it (factors such as facial hair, make-up, and age have not been tested due to the limited volunteers for the demonstration set). What conclusions we can draw from our results are that the Siamese FaceNet outperforms our Siamese VGG Face, our training process is able to achieve great results, and we expect to further reduce the FAR by applying the same experiment with a training dataset that contains biometric ID photos and the corresponding person's selfie. Additionally, while investigating OTM, our results indicated that the method was not suitable for our training process, where we froze all layers of our CNN and therefore performed worse than the regular training. Our final contribution was to investigate to what extent the construction of our SNNs using pre-trained facial recognition CNNs would improve the classification accuracy. Our results proved that FaceNet was already very capable when it came to verifying faces accurately, therefore constructing and training the Siamese FaceNet, the classification capabilities only increased by 0.32% based on the ROC AUC values. FaceNet already provided outstanding classification accuracies, and therefore our implementation had caused only a minor improvement that could also potentially be interpreted as standard error. However, we saw great improvement from constructing and training the Siamese VGG Face from VGG Face, where the classification accuracy improved from 0.9547 to 0.9939 for the ROC AUC; an increase of 4.1%. These results only further support our conclusion that our SNN construction and training process will be able to achieve great results for facial verification tasks and can make up for the limitated data.

6.1 Future Work

To fully understand our model and improve its limitations, future work could consist of augmenting our training data to overcome all the observed weaknesses of our current model. We could especially focus on augmenting and filtering the training data so that our model can overcome these difficult verification tasks:

- **Lighting, color and make-up:** The images could be augmented so that the amount of lighting and saturation is varied for images, particularly for the images of the same people. Furthermore, images could also be augmented so that the color and greyscale are changed because many ID photos (such as those from the Netherlands) have greyscale images of the person's face. Images of the same people with different make-up could also be included, but augmenting the colors would have the same effect.
- **Angles:** We can modify the data by flipping and rotating the images in order to change the angles of the faces in our images.
- **Facial hair:** We could include images of the same person with different amounts of facial hair, and look at how the model performs. This is a test that we could not run in detail.
- **Objects:** Filtering out images where the face is partially covered could also be beneficial. For example, images with glasses, hands, trophies, etc. and because in the context of a real-life application, these objects need to be detected and then from the photo in the first place.

- **Incorrect cropping:** After examination of the testing data, we noticed that many faces were incorrectly cropped. To avoid this in the future, the easiest fix could be to remove images where multiple faces are detected, as the wrong faces are cropped during data preprocessing using the MTCNN. The other possible method could be to use another face detection network and compare the two detected faces and filter out the ones that have completely different (therefore detect different faces).

To investigate improvements in the future to our current data preprocessing steps, we would like to evaluate the use of 2D face alignment. This concept uses a CNN to detect key point positions (for example, the eyes on a face) and apply a rotation transformation to the image. Research has shown that aligning the face images before running them through the facial recognition network improves verification and classification accuracy, even with networks that already obtain over 95% accuracy [33], [52].

As part of supplementary research, it is also important to compare our current models with other techniques. This is important because our current model requires large training times for multiple models, especially when applying 5-fold cross validation. After researching alternative methods, we came across Large Margin Nearest Neighbor (LMNN) as a distance learning metric [53]. The general idea of this distance learning metric imitates the same behavior as triplet loss because triplet loss is based on LMNN. The concept of LMNN is to pull target neighbors closer together and push differently labelled samples further apart. Using the embedding vectors of the CNNs, for future research we could apply this or other similar distance learning metrics and verify if we are able to produce similar or even better results with potentially much lower training times.

6.1.1 Future Experiments

After collecting our results from our experiment, we designed an additional modification that could not be implemented due to the time required to train all variations of our models with different sampling. Therefore, in this subsection we will describe how this modification was designed and how it has the potential to improve our Siamese models and therefore be used for future research. As discussed in section 4.2, we only train the weights in the fully connected layer of our Siamese architecture. However, our new implementation would require us to first train the CNNs individually, and then connect them to construct an SNN just as in Figure 7, and finally train the SNN.

The goal of our new implementation is to first apply transfer learning and fine-tune the FaceNet and VGG Face CNNs individually before connecting them to our SNN. This means that we would train only the top layers of both CNNs. Looking at Figure 5, to fine-tune VGG Face, we removed the softmax activation layer and then make only layers fc6 – fc8 trainable. To fine-tune the FaceNet we only make the Bottleneck BN layer shown in Figure 6 trainable. The purpose of only making the top layers trainable is because we use the weights of the pre-trained layers that can effectively extract discriminative features. The pre-trained model's weights are very valuable as they are learned from

millions of images. By only training the top layers, we fine-tune the model and create a model that is able to make good CNN embedding vectors for our specific dataset.
The CNNs would be trained using the triplet loss function and online triplet mining (finding the hardest samples). Section 2.5.1 discusses triplet loss and how it is highly regarded for facial recognition tasks. With triplet loss, the CNN will learn to output embedding vectors where images of the same person will be positioned much closer together and images of different people will be positioned further apart. Online triplet mining would be used along with triplet loss when training the top layers of the CNNs because then the model will be able to differentiate the more challenging samples. This is discussed in further detail in section 2.5.2. Once the CNNs have been trained, we connect them to our Siamese architecture, as in Figure 7. Then the final step would be to train the SNN as described in section 4.2; by freezing the CNN layer weights, using binary cross entropy and learning rate decay.
Using triplet loss and online triplet mining will theoretically improve the positioning of images input into the CNNs and consequently improve the accuracy of the face verification capability from our Siamese architecture. This is because if the CNN can position images according to the concept of triplet loss, then the L1 distance layer will have very small distances between the anchor and positive images, and large distances between the anchor and negative images; making it very easy to identify if a face matches an ID photo depending on the optimized threshold we choose (optimization of threshold is discussed in section 4).

6.2 Suggestions for Verifai

Currently, this first investigation of face verification models was successful in outlining a research experiment that can be translated into an ID verification task. Through our investigation, we were able to produce a model that obtained great performance. Despite the great results, it is still a bit early to present the system to clients or integrate it with the Verifai Expert Back-Office (VEBO). The VEBO is the layer between the customer's and Verifai's clients, in which experts from Verifai can organize and verify the scanned identity documents. The reason our models are not ready for production is that the percentage of false positives (FAR) by our best model is not particularly high for the demonstration set, but the demonstration set cannot provide conclusive percentages considering its small sample size. Therefore, it requires more testing in an ID verification context. Our suggestion first suggestion is for Verifai to create a dataset that includes both faces and biometric ID photos. However, we understand that is not a simple task, so our next suggestion would be to take the current models and apply the experimental method outlined in section 6.1.1. The written code will be provided to Verifai. There is a strong indication that this method will help reduce the FAR. On top of applying this training method, we suggest implementing the data augmentation and filtering discussed in section 6.1. Finally, I would suggest implementing face alignment as part of the data preprocessing step; there are python libraries available to easily implement this task.

Acknowledgments

I would like to express my appreciation for all the help provided by my supervisor, Professor Kerstin Bunte. From our meetings, I am thankful for the advice, knowledge, and perspectives Professor Bunte provided when discussing the challenges and progress of the project. Our discussions especially helped me keep focus and scope the project.
I would also very much like to thank my colleagues at Verifai. The opportunity to be a full member of the team and be a part of the day-to-day helped me keep focus, stay motivated and understand the overall mission of Verifai and my project. I would especially like to thank my supervisor at Verifai, Jelle van Wezel who gave me great advice on how to tackle challenges and was always available to assist me if I came to him with questions. I would like to thank Jeewon, Carolien and Leander for the discussions we had about machine learning topics and research ideas I could explore. I would also like to thank Tim and Stan for their interest, tips and general advice, helping me transition smoothly into a working environment and making me feel a part of the Verifai team. Finally, I would like to thank all my Verifai colleagues for the very social, welcoming and inclusive work environment that made the past few months a great experience.

References

[1] F. P. Mahdi, M. M. Habib, M. A. R. Ahad, S. Mckeever, A. S. M. Moslehuddin, *et al.*, "Face recognition-based real-time system for surveillance," *Intelligent Decision Technologies-netherlands*, vol. 11, no. 1, pp. 79–92, Apr. 2017, ISSN: 1872-4981. DOI: 10.3233/idt-160279.

[2] J. R. Barr, K. W. Bowyer, P. J. Flynn, and S. Biswas, "Face recognition from video: A review," *International Journal Of Pattern Recognition And Artificial Intelligence*, vol. 26, no. 5, Si, Aug. 2012, ISSN: 0218-0014. DOI: 10.1142/s0218001412660024.

[3] I. F. R. 2022, *Identity fraud report 2022*, Dec. 2021. [Online]. Available: https://onfido.com/resources/insights/identity-fraud-report-2022 (visited on 06/28/2022).

[4] W. Chao, "Face recognition," *GICE, National Taiwan University*, 2007.

[5] L. L. Chambino, J. S. Silva, and A. Bernardino, "Multispectral facial recognition: A review," *IEEE Access*, vol. 8, pp. 207 871–207 883, 2020, ISSN: 2169-3536. DOI: 10.1109/access.2020.3037451.

[6] Y. Jin, J. Lu, and Q. Ruan, "Coupled discriminative feature learning for heterogeneous face recognition," *IEEE Transactions on Information Forensics and Security*, vol. 10, no. 3, pp. 640–652, 2015. DOI: 10.1109/tifs.2015.2390414.

[7] S. Yan, D. Xu, and X. Tang, "Face verification with balanced thresholds," *IEEE Transactions On Image Processing*, vol. 16, no. 1, pp. 262–268, Jan. 2007, ISSN: 1057-7149. DOI: 10.1109/tip.2006.884939.

[8] J. D. Kelleher, "Introduction to deep learning," in *Deep learning*. MIT Press, 2019, pp. 1–8.

[9] B. Zohuri and S. Zadeh, "Deep learning," in *Artificial intelligence driven by machine learning and Deep Learning*. Nova Science Publishers, 2020, pp. 96–150.

[10] K. Simonyan and A. Zisserman, *Very deep convolutional networks for large-scale image recognition*, 2014. DOI: 10.48550/arxiv.1409.1556. [Online]. Available: https://arxiv.org/abs/1409.1556.

[11] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, "Deepface: Closing the gap to human-level performance in face verification," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, 2014, pp. 1701–1708. DOI: 10.1109/cvpr.2014.220.

[12] P. Sermanet, D. Eigen, X. Zhang, M. Mathieu, R. Fergus, *et al.*, *Overfeat: Integrated recognition, localization and detection using convolutional networks*, 2013. DOI: 10.48550/arxiv.1312.6229. [Online]. Available: https://arxiv.org/abs/1312.6229.

[13] J. Tompson, R. Goroshin, A. Jain, and C. LeCun Y.and Bregler, *Efficient object localization using convolutional networks*, 2014. DOI: 10.48550/arxiv.1411.4280. [Online]. Available: https://arxiv.org/abs/1411.4280.

[14] G. Koch, R. Zemel, and R. Salakhutdinov, "Siamese neural networks for one-shot image recognition," in *ICML deep learning workshop*, Lille, vol. 2, 2015, pp. 1–30.

[15] L. Fei-fei, "Knowledge transfer in learning to recognize visual object classes," in *In: International Conference on Development and Learning (ICDL)*, 2006.

[16] D. Chicco, "Siamese neural networks: An overview," in *Artificial Neural Networks*, H. Cartwright, Ed. New York, NY: Springer US, 2021, pp. 73–94, ISBN: 978-1-0716-0826-5. DOI: 10.1007/978-1-0716-0826-5_3. [Online]. Available: https://doi.org/10.1007/978-1-0716-0826-5%5C_3.

[17] J. Zhang, X. Jin, Y. Liu, A. K. Sangaiah, and J. Wang, "Small sample face recognition algorithm based on novel siamese network," *Journal Of Information Processing Systems*, vol. 14, no. 6, pp. 1464–1479, Dec. 2018, ISSN: 1976-913x. DOI: 10.3745/jips.02.0101.

[18] D. Das Chakladar, P. Kumar, P. P. Roy, D. P. Dogra, E. Scheme, *et al.*, "A multimodal-siamese neural network (msnn) for person verification using signatures and eeg," *Information Fusion*, vol. 71, pp. 17–27, Jul. 2021, ISSN: 1566-2535. DOI: 10.1016/j.inffus.2021.01.004.

[19] J. Zhu, H. Zeng, Y. Du, Z. Lei, L. Zheng, *et al.*, "Joint feature and similarity deep learning for vehicle re-identification," *IEEE Access*, vol. 6, pp. 43 724–43 731, 2018, ISSN: 2169-3536. DOI: 10.1109/access.2018.2862382.

[20] D. Chen, X. Cao, F. Wen, and J. Sun, "Blessing of dimensionality: High-dimensional feature and its efficient compression for face verification," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, Jun. 2013.

[21] K. Simonyan, O. M. Parkhi, A. Vedaldi, and A. Zisserman, "Fisher vector faces in the wild.," in *Bmvc*, vol. 2, 2013, p. 4.

[22] C. J. Parde, C. Hu Y.and Castillo, S. Sankaranarayanan, and A. J. O'Toole, "Social trait information in deep convolutional neural networks trained for face identification," *Cognitive Science*, vol. 43, no. 6, Jun. 2019, ISSN: 0364-0213. DOI: 10.1111/cogs.12729.

[23] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, pp. 436–444, May 2015. DOI: 10.1038/nature14539.

[24] P. J. Werbos, *The roots of backpropagation: From ordered derivatives to neural networks and political forecasting*. Wiley, 1994.

[25] Y. Le Cun, "Learning process in an asymmetric threshold network," in *Disordered Systems and Biological Organization*, E. Bienenstock, F. F. Soulié, and G. Weisbuch, Eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 1986, pp. 233–240, ISBN: 978-3-642-82657-3.

[26] M. Tan and Q. V. Le, "Efficientnetv2: Smaller models and faster training," *CoRR*, vol. abs/2104.00298, 2021. arXiv: 2104.00298. [Online]. Available: https://arxiv.org/abs/2104.00298.

[27] M. Tan and Z. Dai, *Toward fast and accurate neural networks for image recognition*, Sep. 2021. [Online]. Available: https://ai.googleblog.com/2021/09/toward-fast-and-accurate-neural.html (visited on 03/28/2022).

[28] S. Balaban, "Deep learning and face recognition: The state of the art," *CoRR*, vol. abs/1902.03524, 2019. arXiv: 1902.03524. [Online]. Available: http://arxiv.org/abs/1902.03524.

[29] G. B. Huang, M. Mattar, T. Berg, and E. Learned-Miller, "Labeled faces in the wild: A database forstudying face recognition in unconstrained environments," in *Workshop on Faces in 'Real-Life' Images: Detection, Alignment, and Recognition*, Erik Learned-Miller and Andras Ferencz and Frédéric Jurie, Marseille, France, Oct. 2008. [Online]. Available: https://hal.inria.fr/inria-00321923.

[30] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering," in *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2015, pp. 815–823. DOI: 10.1109/cvpr.2015.7298682.

[31] Y. Sun, X. Wang, and X. Tang, "Deeply learned face representations are sparse, selective, and robust," *CoRR*, vol. abs/1412.1265, 2014. arXiv: 1412.1265. [Online]. Available: http://arxiv.org/abs/1412.1265.

[32] Y. Taigman, M. Yang, M. Ranzato, and L. Wolf, "Deepface: Closing the gap to human-level performance in face verification," in *2014 IEEE Conference on Computer Vision and Pattern Recognition*, Jun. 2014, pp. 1701–1708. DOI: 10.1109/cvpr.2014.220.

[33] O. Parkhi, A. Vedaldi, and A. Zisserman, "Deep face recognition," British Machine Vision Association, 2019, pp. 1–12.

[34] L. Fe-Fei, Fergus, and Perona, "A bayesian approach to unsupervised one-shot learning of object categories," in *Proceedings Ninth IEEE International Conference on Computer Vision*, 2003, 1134–1141 vol.2. DOI: 10.1109/iccv.2003.1238476.

[35] L. Fei-Fei, R. Fergus, and P. Perona, "One-shot learning of object categories," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 28, no. 4, pp. 594–611, 2006. DOI: 10.1109/tpami.2006.79.

[36] B. M. Lake, R. Salakhutdinov, and J. B. Tenenbaum, "Human-level concept learning through probabilistic program induction," *Science*, vol. 350, no. 6266, pp. 1332–1338, 2015. DOI: 10.1126/science.aab3050. [Online]. Available: https://www.science.org/doi/abs/10.1126/science.aab3050.

[37] A. Zhang, Z. C. Lipton, M. Li, and A. J. Smola, "Dive into deep learning," *CoRR*, vol. abs/2106.11342, Jun. 2021. arXiv: 2106.11342. [Online]. Available: https://arxiv.org/abs/2106.11342.

[38] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," *CoRR*, vol. abs/1502.03167, 2015. arXiv: 1502.03167. [Online]. Available: http://arxiv.org/abs/1502.03167.

[39] I. Goodfellow, Y. Bengio, and A. Courville, "Deep feedforward networks," in *Deep learning*. MIT Press, 2016, pp. 226–227.

[40] C. Szegedy, S. Ioffe, and V. Vanhoucke, "Inception-v4, inception-resnet and the impact of residual connections on learning," *CoRR*, vol. abs/1602.07261, 2016. arXiv: 1602.07261. [Online]. Available: http://arxiv.org/abs/1602.07261.

[41] Q. Cao, L. Shen, W. Xie, O. M. Parkhi, and A. Zisserman, *Vggface2: A dataset for recognising faces across pose and age*, Oct. 2017. DOI: 10.48550/arxiv.1710.08092. [Online]. Available: https://arxiv.org/abs/1710.08092.

[42] J. Bromley, J. Bentz, L. Bottou, I. Guyon, Y. Lecun, *et al.*, "Signature verification using a "siamese" time delay neural network," *International Journal of Pattern Recognition and Artificial Intelligence*, vol. 7, p. 25, Aug. 1993. DOI: 10.1142/s0218001493000339.

[43] R. Vilalta, C. Giraud-Carrier, P. Brazdil, and C. Soares, "Inductive transfer," in Jan. 2011, pp. 545–548. DOI: 10.1007/978-0-387-30164-8_401.

[44] C. C. Aggarwal, "The basic architecture of neural networks," in *Neural networks and deep learning: A textbook*. Springer International Publishing AG, 2018, pp. 14–16.

[45] A. Hermans, L. Beyer, and B. Leibe, "In defense of the triplet loss for person re-identification," *CoRR*, vol. abs/1703.07737, Mar. 2017. arXiv: 1703.07737. [Online]. Available: http://arxiv.org/abs/1703.07737.

[46] *Research & innovation - funding & tenders portal h2020 online manual*, 2020. [Online]. Available: https://ec.europa.eu/research/participants/docs/h2020-funding-guide/cross-cutting-issues/open-access-data-management/data-management_en.htm.

[47] K. Zhang, Z. Zhang, Z. Li, and Y. Qiao, "Joint face detection and alignment using multitask cascaded convolutional networks," *IEEE Signal Processing Letters*, vol. 23, no. 10, pp. 1499–1503, 2016. DOI: 10.1109/lsp.2016.2603342.

[48] S. Liu and W. Deng, "Very deep convolutional neural network based image classification using small training sample size," in *2015 3rd IAPR Asian Conference on Pattern Recognition (ACPR)*, Nov. 2015, pp. 730–734. DOI: 10.1109/acpr.2015.7486599.

[49] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *ICLR (Poster)*, Jun. 2015. [Online]. Available: http://arxiv.org/abs/1412.6980.

[50] L. Taylor and M. Shepherd, "Chapter 12 - performing the security tests and evaluation," in *FISMA Certification and Accreditation Handbook*, L. Taylor and M. Shepherd, Eds., Burlington: Syngress, 2007, pp. 187–209, ISBN: 978-1-59749-116-7. DOI: https://doi.org/10.1016/B978-159749116-7/50017-2. [Online]. Available: https://www.sciencedirect.com/science/article/pii/B9781597491167500172.

[51] V. Arak, *9 tips for breezing through identity verification*, Aug. 2019. [Online]. Available: https://www.veriff.com/blog/9-tips-for-successful-verification (visited on 07/22/2022).

[52] T. Lu, Q. Zhou, W. Fang, and Y. Zhang, "Discriminative metric learning for face verification using enhanced siamese neural network," *Multimedia Tools And Applications*, vol. 80, no. 6, pp. 8563–8580, Mar. 2021, ISSN: 1380-7501. DOI: 10.1007/s11042-020-09784-8.

[53] K. Q. Weinberger, J. Blitzer, and L. Saul, "Distance metric learning for large margin nearest neighbor classification," in *Advances in Neural Information Processing Systems*, Y. Weiss, B. Schölkopf, and J. Platt, Eds., vol. 18, MIT Press, 2005. [Online]. Available: https://proceedings.neurips.cc/paper/2005/file/a7f592cef8b130a6967a90617db5681b-Paper.pdf.