



university of
 groningen

faculty of science
and engineering



Face Verification for Poor Resolution

Images

Bachelor thesis

July 2019

Student: Julius van Dijk (s3212424)

Primary supervisor: dr. André Sobiecki

Secondary supervisor: dr. George Azzopardi

CONTENTS

1	INTRODUCTION	3
2	RELATED WORK	5
2.1	Eigenfaces	6
2.2	Fisherfaces	7
2.3	Local Binary Patterns Histograms	8
2.4	DeepID 1	9
2.5	Deepface	10
2.6	FaceNet	11
2.7	LightCNN	12
2.8	Center Face/Loss	13
2.9	SphereFace	14
2.10	NormFace	15
2.11	Summary	16
3	EXPERIMENTS	17
3.1	Selected Methods	18
3.1.1	Statistical Analysis	18
3.1.2	Convolutional Neural Networks	19
4	RESULTS	20
4.1	Face Verification	20
4.1.1	Eigenface	21
4.1.2	Fisherface	22
4.1.3	Local Binary Patterns Histograms	23
4.1.4	Facenet	24
4.1.5	LightCNN	25
4.1.6	SphereFace	26
4.1.7	Summary	27
4.2	Face Verification after Face Restoration	28
4.3	Face Verification after Downsampling	31
4.4	Robustness of Face Verification	33
5	CONCLUSION	35
5.1	Future work	36

INTRODUCTION

Face detection, recognition (FDR) and verification of face images with poor resolution without labels and ground truths is an important task for forensic investigation. Terrorists and other criminals have posted thousand of videos and images in the internet where the videos are often of low quality. Everyday new illegal videos and images are posted in the internet and many of these videos and images don't have the minimum required quality for methods of face verification. In many cases it is not possible to watch each image and/or video for face verification and/or recognition because the investigators already are overwhelmed with the volume of digital data. Therefore is important that face images with poor digital quality are restored and/or verified and/or recognized.

Methods for FDR are becoming increasingly important in many application areas such as healthcare, the gaming industry, user studies, and homeland defense [30, 33]. Such methods are normally tested, and trained, on good quality datasets and work well on these sets. We define good quality datasets as (mostly) containing face images with standardized (frontal) face positions, good spatial resolution, and controlled lighting conditions. However, images are often acquired in very different conditions, *e.g.*, varying viewpoints, poor lighting, low resolution, and motion blur.

Over the past decades many methods for image restoration have been proposed. The most methods works on any images but there are some methods made specifically for one type of image, face images, for example. The evaluation of these methods are mostly qualitatively and if quantitatively then there is just some PSNR metrics which requires some ground truth image and doesn't address to any application. Most of the of proposed restoration methods are tested with images having artificial noises (blur or salt and paper noises) and these artificial noises are different than the real-world noises. It is unclear if all these restoration methods are useful only for qualitatively purposes or are they also useful for some applications such as face recognition? We believe that the main goal of face restoration methods should be face recognition, artistic puposes or just better visualization. According to our knowledge the methods of face restoration have never been tested in algorithms of face recognition.

The main goal is to know if face restoration add value for face recognition. We perform several experiments to evaluate the methods of face restoration and face recognition. We uses face images with real poor resolution quality and we don't apply any artificial noises.

After the images have been restored we test the methods of face recognition. We compare original low quality face images with restored face images in algorithms of face recognition. The basic principle of our experiments are two tests: firstly we test face recognition using originals and/or poor quality face images and secondly we test face recognition using restored images. We want to know if a restored face image could be easier recognized than an original and low quality resolution face image. In addition to that we benchmark the algorithms against a large dataset where we have introduced artificial noise to receive an indication how the algorithms behave when using good and/or low quality images.

The structure of this thesis is as follows. Section 2 reviews related work and presents the selected methods. Section 3 presents our experiments and how we use these to evaluate the selected methods. Section 4 present the results of the experiments for face restoration, face recognition and face recognition after face restoration. Section 5 concludes the thesis and proposes questions for future work.

RELATED WORK

We are in the possession of a dataset with low quality images of terrorists extracted from online propaganda videos. We want to verify how current methods for face verification perform on these low quality images. The methods discussed in this chapter have been composed by performing a literature study, and searching for open source implementations on Github. The methods are presented in chronological order. The most important criteria for including a method in this chapter is whether it significantly furthered the field of FDR or belongs to a different class of algorithms than others.

The methods are scored on four criteria, listed below. In the next chapter we will provide a summary of our findings and explore the selected methods in better detail.

- **Accuracy** The method presents accuracy figures for one or more public datasets such as Labeled Faces in the Wild (LFW) [9] or YouTube Faces (YTF) [28]. We prefer methods that use the aforementioned datasets as they are industry standards. Old methods, invented prior to the construction of these sets, will not report their accuracy on these sets and hence we will also have to judge the complexity of the datasets used for bench-marking the methods.
- **Availability** Is the method, including all the resources that it needs to operate, publicly available? For methods using a neural network, not only the network structure has to be available but also the pre-trained model weights. It would be unfeasible to retrain all networks of these methods in the scope of this thesis. Some methods have many available implementations, this is preferred over methods having only one public implementation.
- **Complexity** For this criteria we will look at the computational complexity of running the method. For neural networks we will take the training time into account as-well, if available. The computational complexity is important as a complex method might not work on our test computer.
- **Usability** When a method is publicly available we can make an estimation if it will and can work without much fine tuning. Many tunable parameters introduce a new problem. What parameters are the most optimal for our experiments and are they optimal for all experiments.

2.1 Eigenfaces

In 1991 Turk and Pentland presented a paper describing a method for detection and identification of faces in near real time conditions [23]. Before this approach the leading methods were based around the relationship between facial features such as the position of the eyes, nose and mouth. However, research has shown that the direct relationships between these features is not sufficient enough to achieve the same level of face identification as humans [3]. Turk and Pentland took a different approach and were inspired by information theory. The algorithm works by constructing Eigenfaces, which "can be thought of as a set of features which together characterize the variation between face images".

First the algorithm has to be trained, training the algorithm uses known images of identities to construct the corresponding Eigenfaces. When testing the identity of an unknown image it will try to approximate the image based on linear combinations of the Eigenfaces that were generated at the training stage.

Accuracy	The authors have not tested it against a public database, presumably because none were available at that time. However, six years later Belhumeur et al [2] compared their face recognition method against Eigenfaces. They report an accuracy of 75.6% on the Yale face database ¹ .
Availability	Since Eigenfaces are regarded as one of the first successful face recognition methods a lot of implementation are available. The popular image processing library OpenCV includes an implementation ² .
Complexity	In 1991 the authors were able to run the algorithm in real time. The time complexity of the algorithm should be low.
Usability	It is easy to get started using this method, but there are a few constraints. It mostly works on known identities, which are used during the training stage. It is possible to construct new identities based on recognizing patterns in the output. However, if you wish to include these in the model it has to be retrained, it can't simply be added to the current model.

¹ <http://vision.ucsd.edu/content/yale-face-database>

² https://docs.opencv.org/3.4.6/dd/d7c/classcv_1_1face_1_1EigenFaceRecognizer.html

2.2 Fisherfaces

Linear discriminant analysis (LDA) was invented by R. Fisher in 1936 [7]. He used it to solve a taxonomic problem, classifying flowers. Belhumeur et al [2] proposed using LDA for face recognition. According to the authors their extensive testing showed that their method produces a lower error rate than Eigenfaces. They tested both an Eigenface implementation as their Fisherface implementation on a dataset with a lot of variation in the lighting of faces as they assumed that Eigenfaces would not behave well to such changes.

Accuracy	The authors created a public dataset, the Yale face database, which consists of images with a lot of lighting variation ³ . They report an accuracy of 92.3% on this dataset.
Availability	There are many Fisherface implementation such as one in the popular image processing library OpenCV ⁴ .
Complexity	The computational complexity is similar to that of the Eigenface method. Hence, the computational complexity is low.
Usability	It is easy to get started using this method, but just like Eigenfaces there are a few constraints. It mostly works on known identities, which are used during the training stage. It is possible to construct new identities based on recognizing patterns in the output. However, if you wish to include these in the model it has to be retrained, it can't simply be added to the current model.

³ <http://vision.ucsd.edu/content/yale-face-database>

⁴ https://docs.opencv.org/3.4.6/d2/de9/classcv_1_1face_1_1FisherFaceRecognizer.html

2.3 Local Binary Patterns Histograms

Ahonen et al proposed an adaptation of local binary patterns (LBP) [25], which was used to extract texture features, such that it could be used for face recognition [1].

The local binary pattern operator works by taking a block of a certain size $N \times N$, the kernel, and applying a threshold such that a binary pattern is obtained. This improves upon methods like Eigenfaces and Fisherfaces by extracting features on a smaller scale than the entire face, such that many extracted features are the same for different face representations of the same person.

Accuracy The algorithm was tested against the FERET test sets [15]. This dataset is split up into multiple categories. For each identity 5 pictures were taken. The base image f_a , an image f_b captured shortly after f_a with a different facial expression, f_c a third image taken with a different camera and different lighting conditions, dup I taken within one year of f_a and dup II taken at-least one year after f_a .

Method	f_b	f_c	dup I	dup II	lower	mean	upper
LBP, weighted	0.97	0.79	0.66	0.64	0.76	0.81	0.85
LBP, nonweighted	0.93	0.51	0.61	0.50	0.71	0.76	0.81

Availability Many implementation of LBPH are available such as the following in OpenCV ⁵.

Complexity The local binary patterns algorithm is very fast, and using it for face verification purposes does not influence the execution time significantly.

Usability It is easy to get started using this method, and it has benefits compared to Eigenfaces and Fisherfaces (excluding accuracy). It is not necessary for the model to be retrained for new identities. A local binary pattern histogram can be extracted from an unknown image A and it can be compared with the histogram of an unknown image B. Even if this identity is not in our training set, we can still compare the histograms and check whether they are, likely, the same.

⁵ https://docs.opencv.org/3.4.6/df/d25/classcv_1_1face_1_1LBPHFaceRecognizer.html

2.4 DeepID 1

In 2014 Sun et al introduced the DeepID 1 [19] convolutional neural network. The latest version to date is DeepID 3 [20]. Since the DeepID 3 network is closed source we will discuss DeepID 1. The DeepID network outputs a number of features which can then be used by classifiers to compare them for a use case such as face verification. Hence, DeepID is not a complete solution by itself, a classifier must also be used. This applies to all neural network based methods that we have found. The main difference is that other networks often use a simple distance metric.

The authors use the Joint Bayesian technique for face verification [5] together with their DeepID network. They also train a neural network for verification and compare it to the Joint Bayesian method.

The DeepID model was trained on 80% of the CelebFaces dataset, the other 20% is used to train the verification method (Joint Bayesian/neural network). Afterwards they test the accuracy against the LFW dataset [9].

Accuracy The Joint Bayesian method for verification is approximately 1.8% better than the neural network verification method. The methods were tested on the LFW dataset.

Method	LFW
Joint Bayesian	97.20%

Availability The authors did not release their model themselves. There exists a Github repository⁶ containing a DeepID 1 implementation. This implementation is trained on the CASIA-WebFace dataset. The accuracies reported there are far worse than those in the paper (64%).

Complexity Neither the execution time or the training time of this method have been mentioned by the authors.

Usability Since there isn't a good implementation available the usability is low.

⁶ <https://github.com/joyhuang9473/deepid-implementation>

2.5 Deepface

Taigman et al [22] published DeepFace in 2014, which was one of the first convolutional neural networks that was successful at face verification [26]. It was trained on the largest (private) facial database, that was available at the time, which contained 4 million facial images.

Accuracy The models are tested on the LFW dataset [9]. Since the model in the paper is not public, we will also list the accuracy of the implementation mentioned in the previous section.

Method	LFW
DeepFace (paper)	97.35%
DeepFace (public implementation)	97.30%

Availability The authors did not release their model themselves but there exist multiple implementations in most major languages and frameworks. These implementations often offer a pre-trained model such as this one using Tensorflow⁷.

Complexity The training complexity of this network is large but no exact figures have been provided. Due to the model size we can assume that this method is fairly slow for a CNN.

Usability A separate face detector and feature extractor has to be used, such as VGG-Face [14] or MTCNN [34]. The model also has to be trained using different input data as the authors used a private facial database.

⁷ <https://github.com/ildoonet/deepface>

2.6 FaceNet

Schroff et al [16] introduce two different deep convolutional networks. A Zeiler&Fergus style network [32], and an Inception based network [21]. In the paper they describe a number of networks that can be used on different types of hardware. An algorithm running on a mobile phone has different constraints than one running in a data centre.

The same network can be used for face verification and face clustering. This can be done because the network is trained such that the euclidean length in the embedding space directly corresponds to face similarity. The faces of the same person have small distances and faces of distinct people have large distances.

Accuracy The paper only gives a single accuracy for the LFW dataset and not for each network described in the paper. Since the model is not released publicly we will also state the accuracy of the available network⁸.

Method	LFW Accuracy
FaceNet (paper, no extra alignment)	98.87%
FaceNet (paper, extra alignment)	99.63%
FaceNet (Github, trained on CASIA-WebFace)	99.05%
FaceNet (Github, trained on VGGFace2)	99.65%

Availability As mentioned the model was not released by the authors but open-source implementations exist. These open-source implementations are available in multiple languages and frameworks such as TensorFlow and Keras including pre-trained models.

Complexity The models are trained on a CPU cluster for 1000 to 2000 hours. Their Zeiler&Fergus network contains 140 million parameters and requires 1.6 billion FLOPS per image. One of their GoogLeNet (Inception) based network contains 6.6 to 7.5 million parameters and requires 500 million to 1.6 billion FLOPS per image. Another GoogLeNet based network contains 26 million parameters but only requires 20 million FLOPS per image.

Usability The model is not well suited to train by ourselves (see complexity). Multiple open-source implementation exist, including good documentation and pre-trained models.

⁸ <https://github.com/davidsandberg/facenet>

2.7 LightCNN

Wu et al [29] introduce a framework for light convolutional neural networks for face recognition and face verification. Since it is a framework multiple models are available that make use of this structure.

In the paper they define Light CNN-4, CNN-9 and CNN-29 where the number indicates the number of convolution and max-pooling layers. Light CNN-9 consists of 9 convolution layers and 9 max-pooling layers. The models are trained on the MS-CELEB-1M dataset.

The authors also introduce the Max-Feature-Map (MFM) operation which is an activation function that can be used in the convolutional layers. This operation tries to reduce the amount of bias in the network that occurs due to noisy signals. Since there are many different types of noise in large scale datasets it is crucial that the network can separate the noise from the informative signals.

Other goals of their framework is to be faster than other open-source solutions and to provide a single model that can be used on various face benchmarks, such as face recognition and verification, to get state-of-the-art results.

Accuracy The models are tested on the LFW dataset [9] and the YTF dataset [28].

Method	LFW	YTF
Light CNN-4	97.97%	90.72%
Light CNN-9	98.80%	93.40%
Light CNN-29	99.33%	95.54%

Availability Both the Caffe and Python version are available on Github, complete with the model and weights.

Complexity There is no explicit mention of the training speed or the speed used when verifying a pair of faces. However, the models were invented to provide a faster, and better way than previous models. They also provide 3 different models which differ mainly in complexity which directly translates to execution speed.

Usability There are at-least two open-source implementations. The original one, used by the paper, written in Caffe and an updated version written by the same authors in Python. They advise to use the Python version. The Python version only requires PyTorch. The model can be used by downloading the pre-trained weights but they also offer tools and instructions to (re)train the models using a Python script.

2.8 Center Face/Loss

Center Face or Center Loss was introduced by Wen et al [27]. The authors propose a new supervision signal, called center loss, which is an alternative for the softmax function. Center loss is optimized for face recognition tasks and hence should improve convolutional neural network in accuracy and training speed.

When the model was released it achieved a record accuracy on the MegaFace challenge dataset [10] under the evaluation protocol of small training set.

The network is trained on CASIA-WebFace, CACD2000 and Celebrity+. The total set contains 0.7 million images of 17,189 unique identities.

In the paper they use three models, a model using only softmax loss (**model A**), a model using softmax loss and contrastive loss (**model B**) and a model using softmax loss and center loss (**model C**).

Accuracy The models are tested on the LFW dataset [9], the YTF dataset [28] and the MegaFace challenge.

Method	LFW	YTF	MegaFace
model A	97.37%	91.1%	41.297%
model B	99.10%	93.8%	69.987%
model C (proposed)	99.28%	94.9%	76.516%

Availability The authors released the source code, written in Caffe, including the pre-trained models on Github⁹. There are other implementations of the center loss function but not of the same model as specified in the paper.

Complexity The models were trained on two TitanX GPUs. The training time ranged from 14 hours to 22 hours depending on the model.

Usability The documentation is not complete. It only contains instructions for training the model but not for using the pre-trained models.

⁹ <https://github.com/ydwen/caffe-face>

2.9 SphereFace

Liu et al [13] address a specific face recognition problem. They argue that, at the time of writing, there were few suitable loss functions in models to specifically compare facial features. They introduce a variation on the softmax loss function, a-softmax. This loss function is able to make a better comparison between extracted facial features than regular softmax.

The model was trained on the CASIA-WebFace dataset.

Accuracy The models are tested on the LFW dataset [9] and the YTF dataset [28].

Method	LFW	YTF
SphereFace	99.42%	95.0%

At the time this paper was released it performed better on the LFW dataset than most methods such as DeepFace, Deep FT, DeepID and Center Face. FaceNet achieved a better accuracy of 99.65%.

Availability There are at-least 13 open-source implementations of SphereFace written in different languages, using different frameworks such as PyTorch, Tensorflow, Caffe or Keras. The original implementation is written in Caffe. The model weights, and the training data is publicly available.

Complexity There is no mention of the time it takes to validate a single pair of faces or the amount of time it took to train the networks.

Usability A separate face detection algorithm has to be used, such as MTCNN [34]. The SphereFace model uses these features as input for verification.

2.10 NormFace

Wang et al [24] propose a way to improve the performance of convolutional neural network models. They propose two strategy changes that can be applied during the training of the network.

The first being a modification to the softmax loss function such that it is optimized for comparing facial features. This is related to the changes proposed by SphereFace. The second strategy improves on the complexity of metric learning. Metric learning, in this context, usually compares either pairs or triplets as opposed to categorical learning. For a certain amount of training samples N , the number of suitable pairs or triplets is $O(N^2)$ and $O(N^3)$. This amount is often so large that you can't train every sample and hence a selection has to be made. The authors propose two new loss functions, C-contrastive and C-triplet loss, which are designed to replace contrastive (pair) loss, and triplet loss.

It is important to note that these improvements can be used on many different convolutional network models. In the paper they test their improvements using Light CNN [29] and Center Face [27].

Accuracy The models are tested on the LFW dataset [9].

Baseline model	Original accuracy	NormFace
LightCNN	98.13%	98.78%
Center Face	99.03%	99.21%

Availability A complete implementation of these improvements on both the Light CNN and Center Face model is available on Github. This includes both the model as the weights for the model.

Complexity There is no mention of the time it takes to validate a single pair of faces or the amount of time it took to train the networks.

Usability There are some requirements before this model can be used. A separate face and facial landmark detector has to be used, such as MTCNN [34]. In addition to that only an implementation using Caffe exists, which according to the author should work on Windows and Linux. Only a GPU with CUDA support, is supported.

2.11 Summary

Algorithm	Accuracy	Availability	Complexity	Usability
Eigenfaces	-	+	++	+/-
Fisherfaces	+/-	+	++	+/-
LBPH	+	+	++	+/-
DeepID 1	+	-	+/-	+
Deepface	+	-	+/-	+/-
FaceNet	++	+	+	++
LightCNN	++	++	++	++
Center Face/Loss	++	+	+/-	+/-
SphereFace	++	++	+	++
NormFace	+	++	+	+/-

Table 1: Comparison table of the 4 criteria per method.

We have scored the methods discussed in this chapter based on four criteria: Accuracy, availability, complexity and usability. We have summarized this in table 1. The scale used has five possible values:

- -: The lowest score possible.
- -: The method does not have optimal performance on the criteria.
- +/-: The method performs at-best averagely on the criteria.
- +: The method performs well on the criteria.
- ++: The method excels on the criteria.

EXPERIMENTS

For executing the different experiments we adopted the implementation of the methods such that they have a uniform application programming interface (API). Prior to face verification, detection and alignment of an image has to be performed for most of the methods. We have created a single pipeline, depicted in figure 1, which can use any method adhering to our API.

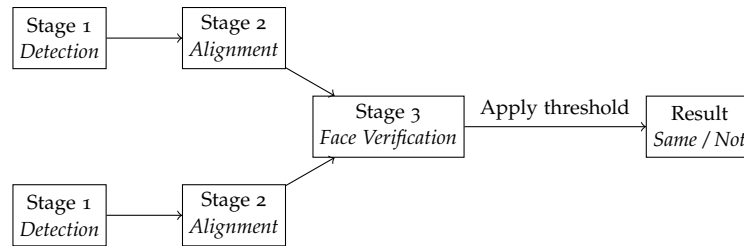


Figure 1: Pipeline used for all experiments

After finishing stage 3 of the pipeline a threshold is applied to give an answer whether the two images are of the same person. For a single test this results in four cases:

1. True positive (TP): The images are of the same identity and the algorithm agrees.
2. False positive (FP): The images are not of the same identity but the algorithm determines that it is the same person.
3. True negative (TN): The images are not of the same identity and the algorithm agrees.
4. False negative (FN): The images are of the same identity but the algorithm determines that they are not.

To show the performance of a method, independent of a certain threshold, we use the receiver operating characteristic curve (ROC curve). It plots the true positive rate (TPR) against the false positive rate (FPR). All other statistics can be derived from these two.

- $TNR = 1 - FPR$
- $FNR = 1 - TPR$

Based on this we can determine the equal error rate (EER). The equal error rate is defined as having an equal probability of miss identifying a true or false sample, or having an equal FPR and TNR.

We use two different methods for determining what threshold should be used for determining the accuracy. The first being the threshold at the EER. The second being the best possible accuracy. For the best possible accuracy many thresholds are tested and the threshold resulting in the highest overall accuracy will be kept. The latter is not always viable as not all of our tests have an equal amount of match cases and no match cases.

We have used three different datasets for the four experiments discussed in the following sections.

- Terrorists: 20 face images of 10 individuals. For each person there are two images: (A) a very low quality image (approximately 45x45 pixels) and (B) a better quality face image (approximately 200x200 pixels). All these images are resized to 128x128 such that they have a normalized size. Bi-cubic interpolation was used for the resizing process. These face images come from videos of potential terrorist. The images have been acquired by running facial detection algorithms proposed by [34] on videos of potential members of the IS terrorist group that were posted on YouTube. Most images suffer from low resolution, blur, and/or noise. In addition, face details are sometimes obscured by cap covers and/or facial hair. Based on expertise from a video surveillance company, we selected these images to be typical of those that (a) are typical in surveillance tasks, but (b) surveillance software have difficulty in analyzing and recognizing.
- Terrorists restoration results: 80 images. We use the output of eight restoration methods of a previous bachelor project [8]. Dataset (A) was used as the input for these algorithms.
- LFW database: Is one of the most used databases in other papers. We created a low quality version of the dataset, as we will discuss in section 4.3

3.1 Selected Methods

Based on the summary of the previous chapter we have chosen six methods for further evaluation: Eigenfaces, Fisherfaces, LBPH, Facenet, LightCNN, SphereFace.

These methods can be divided into two classes that captures the major differences between the methods.

3.1.1 Statistical Analysis

Eigenfaces, Fisherfaces and LBPH fit into this class.

These methods are the oldest included in our summary. The algorithms of Eigenfaces and Fisherfaces are extremely similar. The

difference being the statistical analysis method used for reducing the dimensionality of the image, Principal Component Analysis (PCA) for Eigenfaces and Linear Discriminant Analysis (LDA) for Fisherfaces.

Before we can use these methods for our experiments they have to be trained. All three methods are best used in closed set face verification environments and perform best when having multiple images per identity during the training process. Our real world dataset and our variation on the LFW dataset [9] does not satisfy the requirement and hence our results will be sub-optimal.

Each method returns the distance from the verification image to the best matching representation of a certain identity, 0 being the best possible score. We negate the distance of each output such that a higher score means represent a higher confidence in the match.

Face alignment is not explicitly required by these methods but we will still provide aligned images.

We use the OpenCV library ¹ implementation of the methods for our experiments.

3.1.2 Convolutional Neural Networks

The remaining three methods, Facenet, LightCNN and SphereFace are included in this class.

Most networks require an input of a fixed size. These methods can differ from their preferred size but this lowers the accuracy. Thus we will provide each method with an aligned preprocessed image in the preferred size of the network.

The methods return a high dimensional (eg 1024D) feature vector representing the analysed image. For each method, and all experiments we use the cosine similarity metric to determine the similarity (figure 2).

$$\text{similarity} = \cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \|\mathbf{B}\|}$$

Figure 2: Definition of cosine similarity. **A** denotes the extracted feature vector of the first image, **B** denotes the extracted feature vector of the second image used for verification.

After applying a certain threshold to these scores we can determine whether the method accepts two images as a match, or rejects it as a match. There is not a uniform threshold for the best results, generally a higher threshold yields less false positives but more false negatives while a lower threshold yields more false positives but more true negatives.

¹ <https://opencv.org/>

RESULTS

To answer the questions proposed in the introduction we have structured our results into four sections. In the first section, Face Verification, we test the six selected methods using an experiment that uses our low resolution dataset. The second section, Face Verification after Face Restoration, repeats this test but on the datasets that have been constructed by applying image restoration techniques to the low resolution dataset. The section Face Verification after Downsampling, tests the influence of poor resolution images on face verification. The last section, Robustness of Face Verification, tests the robustness of face verification methods using multiple photos of the same individual in different facial positions.

4.1 Face Verification

For this experiment we extracted a low resolution sample, combined with a higher resolution sample of the same individual from videos.

The low resolution samples have all been up-scaled to 128x128 pixels using bi-cubic interpolation. For each identity we compare a high resolution image against a low resolution image of all 10 identities.

The output scores for each method are shown in a color coded table (figure 3). Different cell colors represent whether the score will be accepted as a match under a certain threshold. We have chosen to use the threshold associated with the lowest equal error rate (EER) as opposed to choosing the threshold associated with the best accuracy as that would result in a minimal accuracy of 90% since the test contains only 10 true samples and 90 false samples.

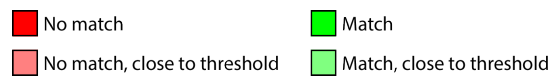
























Figure 3: Color coding used in the results.





















4.1.1 *Eigenface*

											
		-2,894.17	-5,709.82	-6,815.54	-7,097.12	-5,695.68	-7,022.54	-5,252.42	-4,174.39	-4,535.53	-4,072.50
											
											
											
											
											
											
											
											
											
											

The threshold used is -5219.595 which results in an EER of 30%. Under this threshold the total accuracy is 73%.

An EER of 30% is better than just randomly guessing but it still yields in a low accuracy of 73%. This is not unexpected as the method performs better when it has multiple images per identity.





















4.1.2 *Fisherface*

										
	-2,791.68	-5,646.47	-6,764.85	-7,091.15	-5,607.21	-6,968.47	-5,230.28	-4,161.65	-4,475.24	-4,072.09
	-5,857.94	-5,305.90	-4,280.63	-6,315.06	-5,721.38	-7,007.11	-7,641.31	-5,917.45	-6,553.78	-6,796.76
	-4,956.78	-4,392.09	-4,522.62	-5,310.61	-4,253.37	-6,206.33	-6,881.39	-4,566.62	-5,725.96	-6,383.67
	-5,869.34	-3,339.39	-2,379.07	-3,452.42	-3,718.96	-4,719.49	-6,232.56	-5,159.74	-4,885.80	-7,067.96
	-4,784.15	-4,018.70	-5,707.41	-5,307.53	-4,785.13	-5,613.87	-5,371.29	-3,718.92	-5,183.06	-5,003.91
	-5,948.51	-3,547.11	-5,230.55	-2,271.11	-4,129.14	-1,857.89	-5,073.41	-4,966.24	-5,189.31	-6,276.92
	-6,529.99	-7,040.46	-8,021.47	-6,321.31	-7,670.40	-7,366.03	-4,416.19	-5,881.38	-5,860.96	-3,656.22
	-5,720.85	-6,263.90	-5,274.15	-7,694.98	-6,464.41	-8,059.87	-7,929.81	-4,873.10	-6,702.28	-6,526.12
	-4,823.49	-6,876.86	-6,728.01	-7,846.36	-7,022.65	-7,930.26	-5,784.01	-4,730.17	-5,157.67	-5,158.83
	-6,117.96	-5,883.78	-6,858.05	-6,187.76	-6,100.36	-6,567.73	-6,806.40	-5,679.21	-5,922.26	-5,357.05

The threshold used is -5073.409 which results in an EER of 28%. Under this threshold the total accuracy is 73%.






















The EER is slightly lower than the EER of Eigenface but it still yields the same accuracy. The experiment does not contain much lighting variation, in which Fisherface outperforms Eigenface according to its authors. Hence, a performance similar or equal to Eigenfaces was expected.

4.1.3 Local Binary Patterns Histograms

										
	-118.84	-165.68	-175.13	-175.41	-153.05	-185.06	-186.33	-193.50	-201.04	-180.05
	-162.84	-134.13	-189.59	-167.91	-172.99	-177.55	-191.41	-184.84	-188.77	-184.84
	-160.31	-164.45	-157.90	-181.57	-145.94	-192.75	-209.67	-205.63	-208.20	-185.66
	-164.78	-156.80	-162.89	-164.11	-149.90	-181.13	-171.72	-183.92	-173.38	-187.25
	-157.96	-155.07	-180.15	-176.49	-145.24	-169.69	-182.59	-186.36	-184.56	-177.58
	-176.76	-169.25	-196.21	-163.11	-169.96	-146.62	-182.87	-185.99	-188.18	-191.18
	-162.03	-156.21	-187.35	-142.00	-163.05	-172.00	-171.93	-165.31	-193.86	-179.75
	-183.19	-180.48	-193.21	-169.98	-180.91	-193.69	-197.58	-172.64	-189.73	-179.97
	-167.30	-173.61	-172.56	-170.78	-167.38	-190.49	-187.95	-187.69	-193.62	-186.90
	-176.40	-174.69	-167.61	-180.82	-164.78	-180.09	-180.91	-189.96	-176.91	-177.98

Based on the ROC curve we determined the equal error rate (EER), with the accompanying threshold. The threshold used is -171.932 which results in an EER of 32%. Under this threshold the total accuracy is 68%.






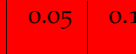
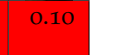
4.1.4 *Facenet*

											
	0.93	0.36	0.30	0.56	0.55	0.38	0.39	0.29	0.26	0.14	
	0.30	0.92	0.49	0.41	0.34	0.41	0.23	0.37	0.35	0.17	
	0.39	0.47	0.38	0.42	0.40	0.33	0.22	0.23	0.29	0.21	
	0.59	0.57	0.46	0.81	0.48	0.24	0.23	0.27	0.22	0.11	
	0.55	0.50	0.48	0.55	0.73	0.61	0.35	0.30	0.48	0.23	
	0.27	0.38	0.39	0.29	0.61	0.75	0.47	0.53	0.54	0.52	
	0.40	0.39	0.58	0.26	0.53	0.60	0.79	0.44	0.62	0.23	
	0.21	0.47	0.48	0.32	0.35	0.60	0.30	0.75	0.35	0.27	
	0.44	0.62	0.52	0.30	0.44	0.63	0.47	0.59	0.64	0.33	
	0.20	0.18	0.14	0.11	0.34	0.49	0.31	0.21	0.39	0.93	

The threshold used is 0.586 which results in an EER of 10%. Under this threshold the total accuracy is 90%.

It is important to note that this is not the best accuracy possible. The low similarity score of the third image results in a $1/10 = 10\%$ false positive rate. Hence, a threshold is chosen such that the EER becomes 10%, even though a threshold can be chosen which results in no false positives, 0.64.


4.1.5 *LightCNN*

										
	0.88	0.13	0.08	0.38	0.24	0.19	0.03	0.05	0.14	0.10
	0.10	0.86	0.16	0.19	0.18	0.24	0.04	0.31	0.18	0.10
	0.22	0.21	0.55	0.34	0.37	0.19	0.01	0.02	0.19	0.11
	0.39	0.26	0.18	0.78	0.23	0.18	-0.05	0.15	0.23	0.06
	0.39	0.13	0.23	0.29	0.86	0.27	0.21	0.01	0.27	0.19
	0.07	0.28	0.16	0.16	0.15	0.74	0.25	0.30	0.27	0.14
	0.13	0.11	0.34	0.05	0.15	0.49	0.68	0.21	0.30	0.09
	0.11	0.28	0.25	0.13	0.16	0.50	0.22	0.69	0.38	0.23
	0.17	0.08	0.23	0.20	0.21	0.40	0.28	0.45	0.65	0.26
	0.17	0.11	0.23	0.01	0.19	0.14	0.09	0.27	0.31	0.99

Based on the ROC curve we determined the equal error rate (EER), with the accompanying threshold. The threshold used is 0.554 which results in an EER of 0%. Under this threshold the total accuracy is 100%.

As the equal error rate is 0%, the accuracy of LightCNN on this experiment is 100%. In addition to that, there is only one case where the returned value is close to the threshold. This result, which is the same as the threshold, is picture 3. Even if we would choose a non optimal threshold of 0.50, it would only add 1 false accept case which still results in a high accuracy.

4.1.6 *SphereFace*

										
	0.84	0.17	0.09	0.39	0.05	0.12	0.02	-0.08	-0.03	0.02
	0.07	0.79	0.02	0.21	0.05	0.06	0.00	0.10	0.13	0.10
	0.06	0.12	0.32	0.15	0.27	0.17	0.38	0.34	0.41	0.21
	0.29	0.30	-0.02	0.85	0.16	0.12	-0.01	-0.04	0.11	-0.05
	0.12	0.11	0.25	0.16	0.74	0.24	0.31	0.16	0.33	0.23
	0.03	0.18	0.22	0.00	0.09	0.57	0.22	0.16	0.17	0.23
	0.23	0.23	0.27	-0.02	0.06	0.42	0.32	0.12	0.30	0.17
	-0.03	0.06	0.15	0.02	-0.04	0.35	0.27	0.47	0.28	0.22
	0.04	0.29	0.19	0.13	0.32	0.32	0.22	0.21	0.50	0.10
	0.09	0.17	0.33	-0.06	0.29	0.20	0.28	0.46	0.44	0.99

Based on the ROC curve we determined the equal error rate (EER), with the accompanying threshold. The threshold used is 0.293 which results in an EER of 10%. Under this threshold the total accuracy is 92%.

A possible explanation for the higher error rate than LightCNN could be the crop size. The input images of dataset B (orig 128x128) are square, while this network takes rectangular images (96x112) as input.

4.1.7 Summary

The area under the curve (AUC) of an ROC graph is a good measure of the performance of the model. An AUC of 1.0 means perfect accuracy, it has a false positive rate and false negative rate of 0%.

When the AUC is 0.5 it means that the method cannot distinguish between the positive and negative class. The false positive rate and false negative rate are 50%, hence you don't have any certainty whether something is true positive or false positive.

This statistic is irrelevant of a threshold as the true negative rate is inversely proportional to the true positive rate.

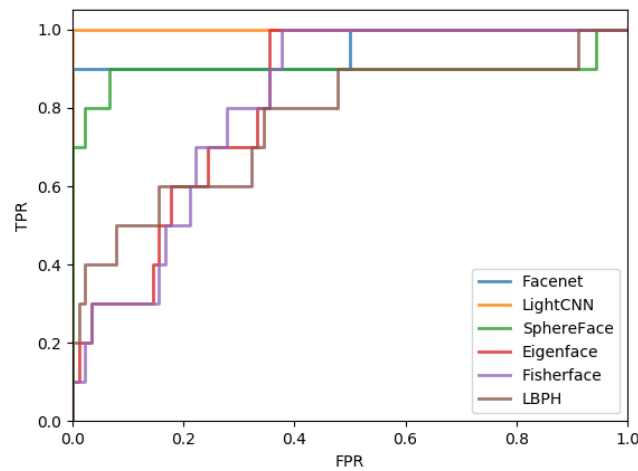


Figure 4: ROC Curves

The ROC curves (figure 4) clearly denote that the newer, CNN based, methods perform better on this experiment.

When comparing the matrices of the different methods we observe that Eigenface and Fisherface provide similar results. This is not surprising as the methods work in a similar way. SphereFace and FaceNet look a lot worse than LightCNN. This is mostly because of choosing a threshold that results in an equal error rate. When choosing a slightly higher threshold, the number of false positives will drop dramatically increasing the overall accuracy.

4.2 Face Verification after Face Restoration

In this experiment we repeat the previous experiment but instead of comparing the high resolution images with the low resolution images we use the result of 6 super resolution methods.

We used the following methods for increasing the resolution of the images:

1. DeblurGAN [11]
2. FSRNet [6]
3. NeuralEnhance [4]
4. Mixing: as of yet this method has not been published yet. It consists of mixing the image generated by FSRNet with the original image.
5. SmartDeblur [31]
6. Sobiecki10 [17]
7. Sobiecki11 [18]



Figure 5: (a) original, (b) bicubic interpolation, (c) Smart Deblur, (d) Neural Enhance, (e) DeblurGAN, (f) FSRNet, (g) Sobiecki10, (h) Sobiecki11 and (i) Mixing

For each method we recorded the output in the same manner as the previous experiment. In the graph, we only record the value for the

true, or matching, cases. This value should be as high as possible as that indicates that the algorithm has the most confidence of a match between these two images.

We will also plot the accuracy of each method, again at the equal error rate, as that takes all test cases into account.

The Area under curve (AUC) statistic is explained in section 4.1.7.

For all methods except LightCNN a discrepancy between the highest scoring methods based on AUC and Accuracy is present. As explained in the previous section the accuracy at the equal error rate is not necessarily the best possible accuracy while the AUC is a performance metric independent of a chosen threshold. Thus we can assume that in the case of Facenet, not only DeblurGAN and FSRNet have a improved accuracy but also the mixing method. The largest discrepancy is present for SphereFace, at the equal error rate only the accuracy improves with DeblurGAN but the AUC scores better for Sobiecki10, NeuralEnhance, DeblurGAN and Sobiecki11.

For Facenet the difference in accuracy between DeblurGAN and the resized images is 3%.

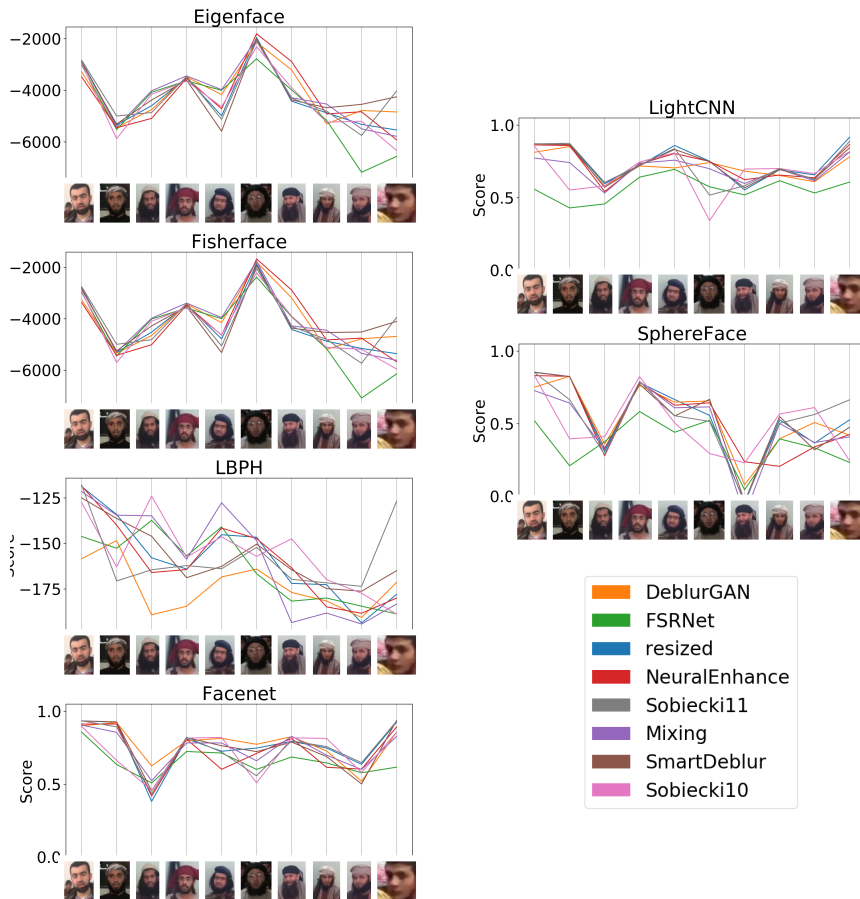
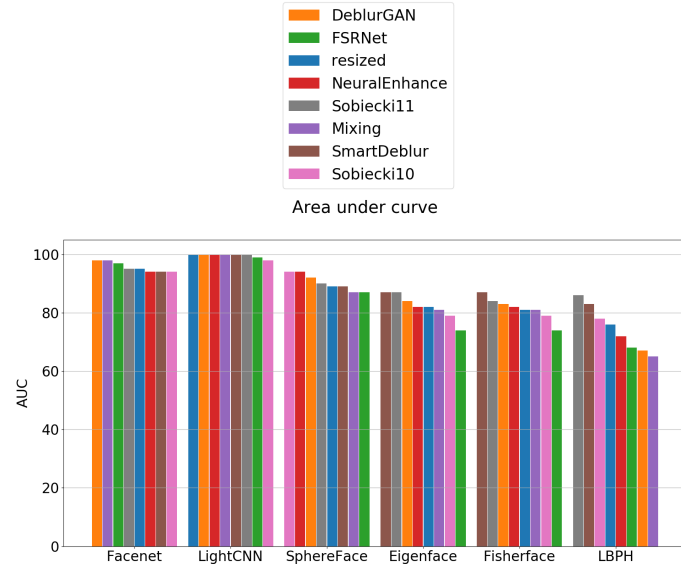
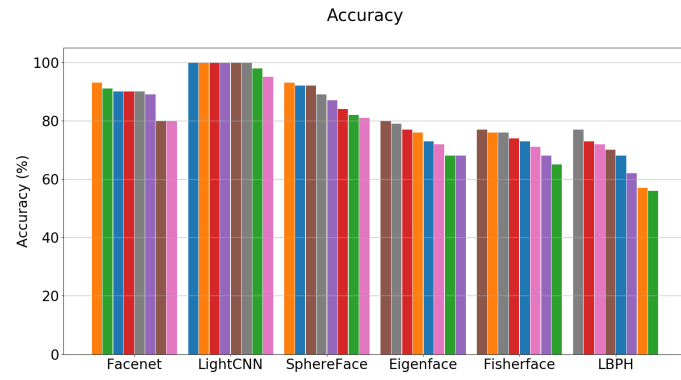


Figure 6: Output score for each of the 10 match cases, reported per method.



(a)



(b)

Figure 7: (a) denotes the AUC for each method, (b) reports the accuracy.

4.3 Face Verification after Downsampling

In this experiment we have lowered the resolution, using bi-cubic interpolation, of each image in the LFW dataset [9] to 50x50 pixels, afterwards we increased the resolution back to 128x128 pixels such that we could use it as input for each method. We will refer to this set as the low quality dataset.



Figure 8: Image a is the original image taken from LFW, image b is our bad quality variation of the image.

It is important to note that this will also have detrimental effects to the face detection algorithm, which is used for aligning the images before running face verification methods. We have opted to use the original face detection and alignment landmarks such that we can run the test in the same manner as the authors of the dataset recommend.

The LFW dataset contains guidelines for testing the accuracy of methods. They include 10 test sets, which each contain 300 matches and 300 non match cases. We will now refer to the 10 test sets as a single test. This test has been run three times per method. Once testing the original LFW dataset against the original dataset. Once using the original LFW dataset against the low quality variation. Once using the low quality dataset against the low quality set.

The OpenCV implementation of Fisherface failed for this experiment and hence no results are included.

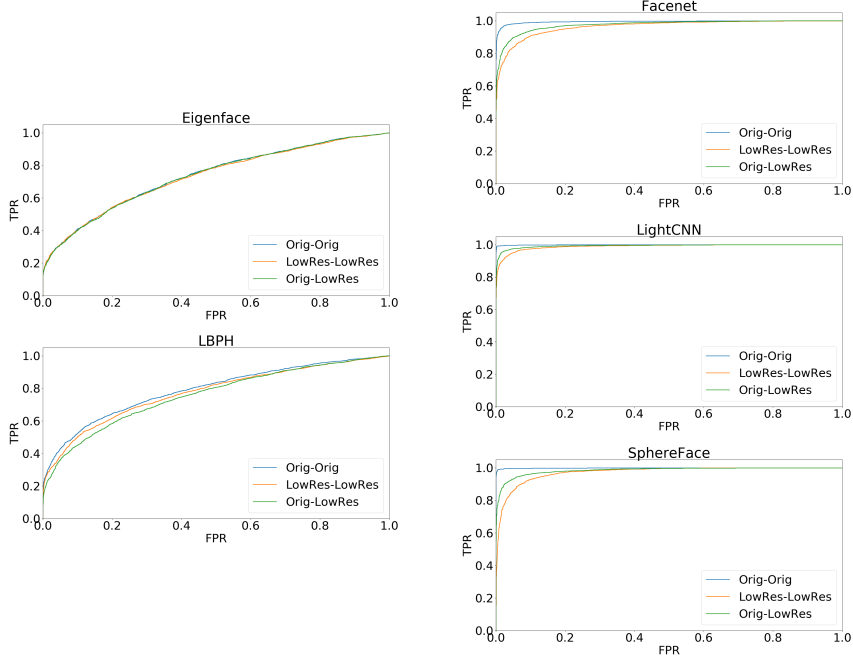


Figure 9: ROC curves per method. Each graph contains the result of the 3 tests that were executed.

Algorithm	O-O Acc @ EER ($\pm\sigma$)	O-L Acc @ EER ($\pm\sigma$)	L-L Acc @ EER ($\pm\sigma$)
Eigenface	68.1% ($\pm 2.3\%$)	68.0% ($\pm 2.4\%$)	67.9% ($\pm 1.8\%$)
Fisherface	-	-	-
LBPH	73.3% ($\pm 1.4\%$)	70.4% ($\pm 1.6\%$)	71.9% ($\pm 2.0\%$)
Facenet	97.5% ($\pm 0.8\%$)	92.8% ($\pm 0.9\%$)	90.7% ($\pm 0.9\%$)
LightCNN	99.4% ($\pm 0.5\%$)	97.15% ($\pm 0.5\%$)	95.5% ($\pm 0.4\%$)
SphereFace	99.2% ($\pm 0.5\%$)	94.5% ($\pm 0.5\%$)	92.0% ($\pm 1.0\%$)

Table 2: Accuracies at the EER for each method and test executed. *O-O* denotes the original dataset compared with the original dataset. *O-L* denotes the original dataset compared with the low quality dataset. *L-L* denotes the low quality dataset compared with the low quality dataset.

4.4 Robustness of Face Verification

In the first and second test we only tested two images per identity. In this test we aim to determine how the methods perform when encountering a variation of face expressions and lighting conditions. We plot the output score of each image, as verified with the reference image, such that we can compare how the variations influence the score.

Figure 10 depicts the reference image used for this experiment, figure 11 shows the ten images of the same individual used in the experiment and figure 12 shows the ten images that should not match the reference image.



Figure 10: The reference image used in the experiment.



Figure 11: 10 images that are of the same identity as our reference image.



Figure 12: 10 images, all of different identities that are not the same as our reference image.

The threshold that results in the best accuracy has been plotted using a black line. All the results above this line are accepted as a match, while the results below are not. Surprisingly each method achieved a 100% accuracy on this experiment.

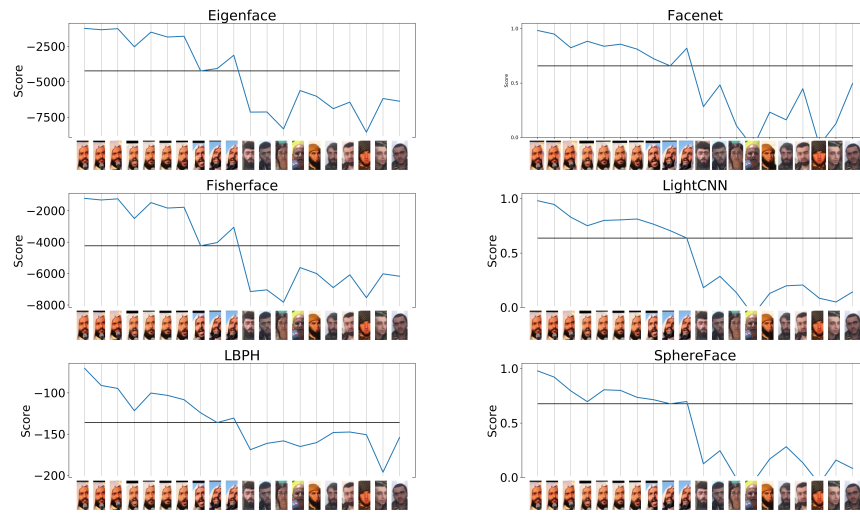


Figure 13: Output scores per method.

CONCLUSION

Our main goal is to know if face restoration adds value for face recognition. We tested the performance of six methods on a real world dataset containing photos of terrorists, taken out of propaganda videos, and on a larger artificial dataset to analyze the impact of low quality images on the verification process.

The experiments showed that restoration methods do influence the performance of verification methods, and they can improve the accuracy. However, the influence of a restoration method is not the same for all verification methods. One explanation is due to the fact how CNNs operate. The networks try to reduce the dimensionality of the input by extracting important features. Restoration methods can either enhance or distort these features, making the method more accurate or less accurate.

Qualitative analysis of the restoration methods showed that FSRNet the best restoration method. Hence, we expected the verification results to reflect that. The accuracy of Facenet was only slightly raised by FSRNet, while it even lowered the accuracy of LightCNN and SphereFace dramatically. The best restoration method for LightCNN can not be determined as it achieved a perfect accuracy on the non-improved images. The best restoration method for Facenet and SphereFace is DeblurGAN, which is surprising as the qualitative analysis classified this as one of the worst restoration methods. Again, a possible explanation lies in the way how CNNs operate. The networks extract certain features from the input image and does this in a different way than humans. What looks best to the human eye does not necessarily look the best for a CNN.

The test on the Labeled Faces in the Wild (LFW) [9] has shown us the importance of good quality input images. On average the accuracy between the original images compared with the original image was 3.9% higher than the low quality images compared with the low quality images. Eigenfaces are less susceptible to the difference in quality. The CNNs are influenced in a large way by the lower quality images. The largest difference was reached by SphereFace: 7.2%.

Our real world tests show promising results for using face restoration to enhance face verification. The dataset that we used consists mainly of unlabeled data, this makes running large tests impossible. Our current test only contained 10 true/match cases, and 90 false/no match cases. An argument can be made why this should be avoided. However, testing an equal amount of match as no match cases is also not representative of real world use cases [12]. An improvement

can be made by running the test on more true matches, for instance by including more images per identity or by labeling more unique identities.

The experiments on the LFW dataset show a discrepancy between the accuracy reported by the authors and our findings for several methods. A possible explanation is the variation in the alignment algorithm used. For each method we used the same detection and alignment algorithm, while the original implementations of the tested methods did not use the same detection and alignment method or implementation. Facenet, LightCNN and SphereFace all use the same method for detection, but different implementations which result in close (within 2-3px), but not matching output of the detection algorithm.

5.1 Future work

When generating the low quality version of the LFW dataset the face detection algorithm failed on more than 20% of the cases. Without a working detection method, verification is impossible. We suspect that face restoration methods might have a dramatic improvement on the detection of faces in low quality photos.

BIBLIOGRAPHY

- [1] T. Ahonen, A. Hadid, and M. Pietikainen. "Face Description with Local Binary Patterns: Application to Face Recognition". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 28.12 (Dec. 2006), pp. 2037–2041. ISSN: 0162-8828, 2160-9292. DOI: [10.1109/TPAMI.2006.244](https://doi.org/10.1109/TPAMI.2006.244). URL: <http://ieeexplore.ieee.org/document/1717463/>.
- [2] Peter N Belhumeur, Joao P Hespanha, and David J Kriegman. "Eigenfaces vs. Fisherfaces: Recognition Using Class Specific Linear Projection". In: *IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE* 19.7 (1997), p. 10.
- [3] S Carey and R Diamond. *From Piecemeal to Configurational Representation of Faces*. - PubMed - NCBI. Jan. 21, 1977. URL: <https://www.ncbi.nlm.nih.gov/pubmed/831281>.
- [4] A. J. Champandard. *Neural Enhance: Super Resolution for images using deep learning*. <https://github.com/alexjc/neural-enhance>. 2016.
- [5] Dong Chen et al. "Bayesian Face Revisited: A Joint Formulation". In: *Computer Vision – ECCV 2012*. Ed. by Andrew Fitzgibbon et al. Red. by David Hutchison et al. Vol. 7574. Berlin, Heidelberg: Springer Berlin Heidelberg, 2012, pp. 566–579. ISBN: 978-3-642-33711-6 978-3-642-33712-3. DOI: [10.1007/978-3-642-33712-3_41](https://doi.org/10.1007/978-3-642-33712-3_41). URL: http://link.springer.com/10.1007/978-3-642-33712-3_41.
- [6] Y. Chen et al. "FSRNet: End-to-End Learning Face Super-Resolution with Facial Priors". In: *Proc. IEEE CVPR*. 2018, pp. 2492–2501.
- [7] R. A. Fisher. "THE USE OF MULTIPLE MEASUREMENTS IN TAXONOMIC PROBLEMS". In: *Annals of Eugenics* 7.2 (Sept. 1936), pp. 179–188. ISSN: 20501420. DOI: [10.1111/j.1469-1809.1936.tb02137.x](https://doi.org/10.1111/j.1469-1809.1936.tb02137.x). URL: <http://doi.wiley.com/10.1111/j.1469-1809.1936.tb02137.x>.
- [8] Hidde Folkertsma. "Super-Pixel Resolution for Face Images". In: (2018).
- [9] Gary B Huang et al. "Labeled Faces in the Wild: A Database for Studying Face Recognition in Unconstrained Environments". In: (), p. 11.
- [10] Ira Kemelmacher-Shlizerman et al. "The MegaFace Benchmark: 1 Million Faces for Recognition at Scale". In: *IEEE*, June 2016, pp. 4873–4882. ISBN: 978-1-4673-8851-1. DOI: [10.1109/CVPR.2016.527](https://doi.org/10.1109/CVPR.2016.527). URL: <http://ieeexplore.ieee.org/document/7780896/>.

- [11] O. Kupyn et al. "DeblurGAN: Blind Motion Deblurring Using Conditional Adversarial Networks". In: *Proc. IEEE CVPR*. 2018.
- [12] Shengcai Liao et al. "A Benchmark Study of Large-Scale Unconstrained Face Recognition". In: IEEE, Sept. 2014, pp. 1–8. ISBN: 978-1-4799-3584-0. DOI: [10.1109/BTAS.2014.6996301](https://doi.org/10.1109/BTAS.2014.6996301). URL: <http://ieeexplore.ieee.org/document/6996301/>.
- [13] Weiyang Liu et al. "SphereFace: Deep Hypersphere Embedding for Face Recognition". In: IEEE, July 2017, pp. 6738–6746. ISBN: 978-1-5386-0457-1. DOI: [10.1109/CVPR.2017.713](https://doi.org/10.1109/CVPR.2017.713). URL: <http://ieeexplore.ieee.org/document/8100196/>.
- [14] Omkar M. Parkhi, Andrea Vedaldi, and Andrew Zisserman. "Deep Face Recognition". In: British Machine Vision Association, 2015, pp. 41.1–41.12. ISBN: 978-1-901725-53-7. DOI: [10.5244/C.29.41](https://doi.org/10.5244/C.29.41). URL: <http://www.bmva.org/bmvc/2015/papers/paper041/index.html>.
- [15] Syed A Rizvi, P Jonathon Phillips, and Hyeonjoon Moon. "The FERET Verification Testing Protocol for Face Recognition Algorithms". In: (Oct. 1998), p. 16.
- [16] Florian Schroff, Dmitry Kalenichenko, and James Philbin. "FaceNet: A Unified Embedding for Face Recognition and Clustering". In: *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (June 2015), pp. 815–823. DOI: [10.1109/CVPR.2015.7298682](https://doi.org/10.1109/CVPR.2015.7298682). arXiv: [1503.03832](https://arxiv.org/abs/1503.03832). URL: <http://arxiv.org/abs/1503.03832>.
- [17] A. Sobiecki, L.A.P. Neves, and C.E. Thomaz. "To a Better Digitalization and Visualization of Frontal Face Photographs". In: *IV European Conference on Computational Mechanics* Paris, France (2010).
- [18] A. Sobiecki et al. "Segmentacao e Restauracao Digital para Eliminacao de Artefatos em Imagens Frontais de Face". In: *VII Workshop de Visão Computacional* Curitiba, Brazil (2011).
- [19] Yi Sun, Xiaogang Wang, and Xiaoou Tang. "Deep Learning Face Representation from Predicting 10,000 Classes". In: IEEE, June 2014, pp. 1891–1898. ISBN: 978-1-4799-5118-5. DOI: [10.1109/CVPR.2014.244](https://doi.org/10.1109/CVPR.2014.244). URL: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6909640>.
- [20] Yi Sun et al. "DeepID3: Face Recognition with Very Deep Neural Networks". In: (Feb. 3, 2015). arXiv: [1502.00873](https://arxiv.org/abs/1502.00873) [cs]. URL: <http://arxiv.org/abs/1502.00873>.
- [21] Christian Szegedy et al. "Going Deeper with Convolutions". In: IEEE, June 2015, pp. 1–9. ISBN: 978-1-4673-6964-0. DOI: [10.1109/CVPR.2015.7298594](https://doi.org/10.1109/CVPR.2015.7298594). URL: <http://ieeexplore.ieee.org/document/7298594/>.

- [22] Yaniv Taigman et al. "DeepFace: Closing the Gap to Human-Level Performance in Face Verification". In: IEEE, June 2014, pp. 1701–1708. ISBN: 978-1-4799-5118-5. DOI: [10.1109/CVPR.2014.220](https://doi.org/10.1109/CVPR.2014.220). URL: <http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6909616>.
- [23] M. Turk and A. Pentland. "Eigenfaces for Recognition". In: *Journal of Cognitive Neuroscience* 3.1 (1991), pp. 71–86. ISSN: 0898-929X. DOI: [10.1162/jocn.1991.3.1.71](https://doi.org/10.1162/jocn.1991.3.1.71). PMID: 23964806.
- [24] Feng Wang et al. "NormFace: L_2 Hypersphere Embedding for Face Verification". In: ACM Press, 2017, pp. 1041–1049. ISBN: 978-1-4503-4906-2. DOI: [10.1145/3123266.3123359](https://doi.org/10.1145/3123266.3123359). URL: <http://dl.acm.org/citation.cfm?doid=3123266.3123359>.
- [25] Li Wang and Dong-Chen He. "Texture Classification Using Texture Spectrum". In: *Pattern Recognition* 23.8 (Jan. 1990), pp. 905–910. ISSN: 00313203. DOI: [10.1016/0031-3203\(90\)90135-8](https://doi.org/10.1016/0031-3203(90)90135-8). URL: <https://linkinghub.elsevier.com/retrieve/pii/0031320390901358>.
- [26] Mei Wang and Weihong Deng. "Deep Face Recognition: A Survey". In: (Apr. 18, 2018). arXiv: [1804.06655](https://arxiv.org/abs/1804.06655) [cs]. URL: <http://arxiv.org/abs/1804.06655>.
- [27] Yandong Wen et al. "A Discriminative Feature Learning Approach for Deep Face Recognition". In: *Computer Vision – ECCV 2016*. Ed. by Bastian Leibe et al. Vol. 9911. Cham: Springer International Publishing, 2016, pp. 499–515. ISBN: 978-3-319-46477-0 978-3-319-46478-7. DOI: [10.1007/978-3-319-46478-7_31](https://doi.org/10.1007/978-3-319-46478-7_31). URL: http://link.springer.com/10.1007/978-3-319-46478-7_31.
- [28] Lior Wolf, Tal Hassner, and Itay Maoz. "Face Recognition in Unconstrained Videos with Matched Background Similarity". In: IEEE, June 2011, pp. 529–534. ISBN: 978-1-4577-0394-2. DOI: [10.1109/CVPR.2011.5995566](https://doi.org/10.1109/CVPR.2011.5995566). URL: <http://ieeexplore.ieee.org/document/5995566/>.
- [29] Xiang Wu et al. "A Light CNN for Deep Face Representation with Noisy Labels". In: (Nov. 9, 2015). arXiv: [1511.02683](https://arxiv.org/abs/1511.02683) [cs]. URL: <http://arxiv.org/abs/1511.02683>.
- [30] Ming-Hsuan Yang, D.J. Kriegman, and N. Ahuja. "Detecting faces in images: a survey". In: *IEEE TPAMI* 24.1 (2002), pp. 34–58.
- [31] V. Yuzhikov. *SmartDeblur deconvolution software*. 2015. URL: <http://smartdeblur.net>.
- [32] Matthew D. Zeiler and Rob Fergus. "Visualizing and Understanding Convolutional Networks". In: *Computer Vision – ECCV 2014*. Ed. by David Fleet et al. Vol. 8689. Cham: Springer International Publishing, 2014, pp. 818–833. ISBN: 978-3-319-10589-5 978-3-319-10590-1. DOI: [10.1007/978-3-319-10590-1_53](https://doi.org/10.1007/978-3-319-10590-1_53). URL: http://link.springer.com/10.1007/978-3-319-10590-1_53.

- [33] C. Zhang and Z. Y. Zhang. "A Survey of Recent Advances in Face Detection". In: *Technical Report MSR-TR-2010-66*. Microsoft Research, 2010.
- [34] Kaipeng Zhang et al. "Joint Face Detection and Alignment Using Multitask Cascaded Convolutional Networks". In: *IEEE Signal Processing Letters* 23.10 (Oct. 2016), pp. 1499–1503. ISSN: 1070-9908, 1558-2361. DOI: [10.1109/LSP.2016.2603342](https://doi.org/10.1109/LSP.2016.2603342). URL: <http://ieeexplore.ieee.org/document/7553523/>.