Comparing classical and deep approaches for face recognition in a smartgym application

Author: Gonzalo Benito

Abstract

In this paper we approach the problem of face recognition to be applied on the Futbol Club Barcelona' smartgym. The aim is to allow the machines to auto configure themselves according to each user and track athletes performance. This project compares both classical and state-of-the-art deep learning approaches for face recognition on public available data, providing a first baseline to the problem of face recognition in the smartgym scenario. We conclude that for a reduced amount of training data, classical approaches provide a better generalization and less overfitting than deep strategies. In all cases, while recognition performance is reduced for large head pose variations, classical approaches also show a more stable behavior.

Index Terms

Face recognition, deep learning, classical methods, image datasets, eigenfaces, fisherfaces, vgg face

I. Introduction

FACE recognition was one of the first problems addressed by modern computer vision. It is still a challenging matter as human faces belong to a huge data space with many features. Up to the breakthrough of deep learning based on considerably big amounts of data, most of the researchers developed their own datasets along with their algorithms. This has pros and cons because while being true that this allows for a precise control on the properties of the samples, it also holds that generalization power is limited by the restricted information on the sample space. The state-of-the-art in face recognition involves Deep Learning methods with big databases training. In the late years, the creation of the Labeled Faces in the Wild (LFW) dataset [1] allowed to establish a benchmarking database for unconstrained environment face recognition. This turned out be used as a standard measure to compare algorithms on this task. Convolutional neural networks (CNNs) have been used in nearly all of the top performing methods on this dataset. Topping the accuracy ranking, Google FaceNet [2] and DeepID3 [3] are the the current references. However, researchers from Kyoto University tackled face recognition still using non deep learning algorithms in 2014 [4], aiming specifically for small database cases with promising results.

All of the above gains great importance when facing a real world implementation. Considering the FCB's smartgym environment goal poses conditions on the face recognition approach. For instance, the algorithm will work on a restricted environment where the only subjects it has to recognize are the club's players. This makes for small world data and classes to discriminate which implies that classical methods may still present a valid option while deep learning ones find it difficult to perform at their best because of the reduced data. Also, the application scenario requires of robustness against faces position, variation over time and accessories such as beard and hair do. In these cases the classical approaches might be in disadvantage to deep learning ones as the latter can develop face embeddings taking into account variations on the samples as they learn the data features instead of engineering pixel-based features to fit data.

In this work we evaluate three different approaches to the face recognition task for the FCB smartgym project. In the State of the Art section we study popular public databases for face recognition with reduced number of subjects to know how to build a proper database to train the algorithms effectively in a restricted environment. We also introduce some background on the face recognition methods and the current top-performing approaches. The Method section elaborates on the Eigenfaces, Fisherfaces and VGG Face approaches that were chosen as possible implementations, their characteristics, pros and cons. In the Experiments section, we explain the data used, the pre-processing performed and the steps taken to evaluate the algorithms performance in general and also in detail regarding face rotation in both training and testing samples. The Results section contains the data gathered through experimentation over a public face recognition database. We give our conclusions over the results obtained, discuss on the methods viability for implementation and introduce possible future lines of work for the smartgym project. Finally, we include an appendix with a face recognition implementation to be used for the tests in FCBs gymnasium.

Author: Gonzalo Benito, gonzabenito@gmail.com
Advisor 1: Sergio Escalera, Mathematics and Informatics Dept., Universitat de Barcelona and Computer Vision Center
Advisor 2: Josep Llados, Computer Vision Center Direction and Computer Science Department, Universitat Autónoma de Barcelona
Thesis dissertation submitted: September 2017

II. State of the art

In this section we will introduce databases and methods for face recognition. First, we will present three public databases that have been studied for their use both as dummy data for face recognition models early testing, and as guidance for designing Barcelona smartgym's dataset in the project's upcoming stages. Second, the top performance methods in the field considering deep learning and classical ones as well will be analyzed for pros and cons on implementation.

A. Databases

As stated before, the training data is crucial for the final algorithms performance. Many current methods rely on really big databases, most of them being still proprietary. This affects negatively the advances on the field as it can be restrictive for non big companies or universities that find it more difficult to obtain samples for their projects on the field. Therefore, this section studies some of the most used, publicly available databases for face recognition. These provide a platform for the study of possible solutions for our problem. Each has different characteristics, with pros and cons.

1) FERET: This database is the product of a U.S. government program named Face Recognition -hence the name FERET- [5]. Such program began in 1993 and lasted until 1998, being the DoD Counterdrug Technology Development Program Office responsible for its execution. It consisted in three main elements: sponsoring research for face recognition advanced theory application into actual working algorithms, collecting an annotated faces database for the face recognition development, and creating a set of evaluations for algorithms comparison based on FERET database.

The database contains 14126 images of faces of 1199 different individuals. The photographs were collected under relatively unconstrained conditions. To maintain a degree of consistency throughout the database, the same physical setup was used in each photography session. Because the equipment had to be reassembled for each session, there was some minor variation in images collected on different dates. Some individuals had their picture taken in multiple instances of data collection along a two years lapse, allowing for analysis on time influence in recognition process. For the different poses, the subjects were asked to look at marks on the wall, where the marks corresponded to the aspects defined below.

a) Image Details: A set of images of an individual was defined as consisting of a minimum of five views. As the scheme in Fig. 1 shows, two frontal views were taken, labeled fa and fb. One was the first image taken (fa) and the other, fb, usually the last. The subject was asked to change its facial expression for the fb image. Images were also collected at the following head aspects: right and left profile (labeled pr and pl), right and left quarter profile (qr, ql), and right and left half profile (hr, hl). Additionally, five extra locations (ra, rb, rc, rd, and re), irregularly spaced among the basic images, were collected if time permitted. Some subjects also were asked to put on their glasses and/or pull their hair back to add some simple but significant variation in the images. Figure 2 displays a sample of the database.

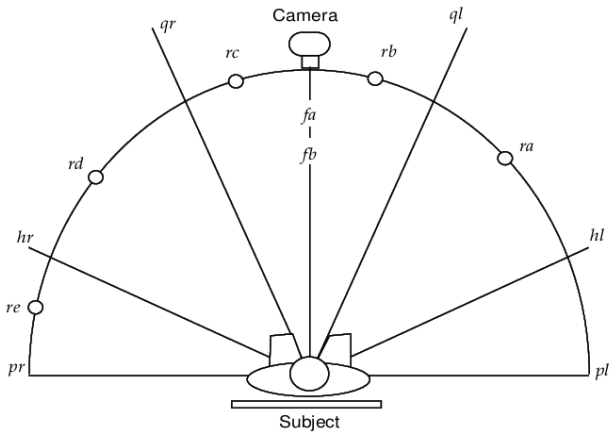

Figure 1: orientations scheme for FERET samples.

b) Training and Testing Protocols: From 1993 to 1994, two phases of image collection had been held in order to prepare for the training and testing of candidate algorithms for the program. By the year 1995, the collection had amounted to 1109 sets of images were in the database, for 8525 total images. There were 884 individuals in the database and 225 duplicate sets of images. The dataset was split in a way that algorithm developers counted with around 300 sets of images as training/validation data set, and the remaining images were held by the government as test set.

To allow the evaluation of the algorithms' robustness with respect to specific variables, the test database was augmented with a set of digitally altered images. Illumination levels of 40 images were changed by using the MATLAB Image Processing

Figure 2: Samples of the FERET database.

Tool Box command "brighten ()" which is a nonlinear function. Collectors created images with the illumination levels reduced by approximately 40 and 60 percent. To test sensitivity to scale changes, they electronically modified 40 images to show 10-, 20-, and 30-percent reductions of scale along each axis. For this, they employed the MATLAB Image Processing Tool Box command "imresize ()" that applies a low-pass filter on the original images to avoid aliasing, and bilinear interpolation to find each pixel density in the reduced image. Finally, using Adobe Photoshop's paint brush tool, the database collectors electronically modified portions of clothing in several of the images to reverse the contrast. The point in this was to see if any algorithms were using cues from clothing for recognition.

2) SCface: Surveillance Camera database was a project carried out by researchers from the University of Zagreb, Croatia [6]. The correspondent publication dates 2011 making it a quite more recent work with respect to FERET II-A1. Its aim was to provide a dataset that allowed development of algorithms oriented to indoor security. For this, the authors made use of an arrange of six commercially available surveillance cameras, a professional digital video surveillance recorder and a computer. Following the principle that for real-world applications, algorithms should operate and be robust on data captured from video cameras on a public place. Cameras image quality may vary, illumination should be uncontrolled with varying direction and strength. Head pose should be as natural as possible. This database was collected mainly having identification rather than verification in mind, pondering the fact that identification results a more demanding recognition problem with its one-to-many comparisons.

The database consists of 4,160 images in total, taken in uncontrolled lighting (infrared night vision images taken in dark) with five different quality and resolution surveillance video cameras. Images were taken under uncontrolled illumination conditions and from various distances. Head pose in surveillance images is typical for a commercial surveillance system as the subjects were not looking to a fixed point. Database contains nine different poses images suitable for head pose modeling and/or estimation for each of the 130 subjects such as the samples from Fig. 3.

The procedure for obtaining the images simulated a real-world environment. First they had to walk in front of the surveillance cameras in the dark and then they had to do the same in uncontrolled indoor lighting. During their walk in front of the cameras they had to stop at three previously marked positions. After that, participants were photographed with a digital camera at close range in controlled conditions. Finally, a high quality IR vision surveillance camera was used to take mug shots of the subjects in a dark room. The overall result was of 32 images per individual in the database.

a) Image Details: The whole dataset is divided into smaller sets with specific use each. One of them, the *frontal facial mug shots* set has one image per subject for a total of 130 elements. These images are in high resolution lossless 24 bit color cropped to 1600×1200 pixels. The face occupies approximately 80% of the image and the exact positions of eye centers, tip of the nose and the center of the mouth are manually extracted and stored in a *.txt* file.

The *surveillance cameras images* set was done using 5 cameras attached to the same place. There are three images per subject for each camera, taken at three discrete distances (4.20, 2.60 and 1.00 m). With a total of 15 images per subject in this set (1950 in total), this set was designed to test the face recognition algorithms in real-world surveillance setup. They have been cropped in order to remove as much background as possible. Due to different distances at which the images were taken, new (cropped) images are not all the same size. Cropped images have the following resolutions: 100×75 pixels for distance 1, 144×108 for distance 2 and 224×168 for distance 3.

There is also a 130 samples set for *IR night vision mug shots* which is simply equal to the mug shot set but taken in a dark room using the IR vision mode in grayscale and with a 426×320 pixels resolution for the 130 samples. Aside, there is an IR night vision set made with the surveillance cameras. Its images resolutions are the same as the *surveillance cameras images* but in grayscale. This set has 780 images.

Figure 3: Samples of the SCface database.

Finally there is a set with the same image properties as the *frontal facial mug shots* but including rotated views. There are nine images per subject in this set, which gives 1170 images in total.

3) MIT-CBCL face recognition database: The Center for Biological & Computational Learning (CBCL) at MIT constructed this database with the idea of training component-based face recognition algorithms robust to illumination and pose [7]. A major drawback of the state-of-the-art systems at the time was the need of a large number of training images taken from different viewpoints and under different lighting conditions. These images are often unavailable in real-world applications. To cope with this, the authors decided to employ a synthetically generated images database.

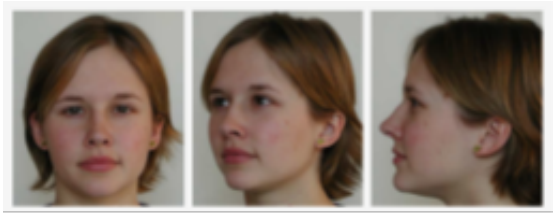

Figure 4: Samples of the triplets used for the 3D models.

a) Image Details: They first generated 3D face models based on three training images of each person. Each triplet consisted of a frontal, a half-profile, and a profile high resolution face image as seen in Fig. 4. According to [7], with a sufficiently large database of 3D face models any arbitrary face can be generated by morphing the ones in the database. An initial database of 3D models was built by recording the faces of 200 subjects with a 3D laser scanner. Then 3D correspondences between the head models by techniques based on optical flow computation. Once the 3D face models of all the subjects in the training database were computed, they generated the synthetic face images under varying pose and illumination to train the component-based recognition system. For a face detection stage, they generated a set morphing the models with a pose range of $\pm 45^\circ$ rotation in depth and $\pm 10^\circ$ rotation in the image plane.

For the recognition phase, the faces were rotated in depth from 0° to 34° in 4° increments and rendered with two illumination models at each pose. The first model consisted of ambient light alone. The second model included ambient light and a directed light source, which was pointed at the center of the face and positioned between -90° and 90° in azimuth and 0° and 75° in elevation. The angular position of directed light was incremented by 15° in both directions. The resulting images are on the lines of the samples shown in Fig. 5.

There are several more faces databases, however most of them share plenty of the characteristics of the aforementioned. The Table I displays the most significant features of some databases. The Database of Faces was included in the comparison, but due to its too small size and a lack for a remarkable difference over the others, it is not analyzed more profoundly.

B. Implementations

Previous image recognition and facial recognition pipelines relied on hand-engineered features such as SIFT, LBP, and Fisher vectors. At the end of 2011 the situation changed with the introduction of large scale feature learning with a sparse autoencoder [8]. This autoencoder was trained using asynchronous Stochastic Gradient Descent (SGD) on 1,000 machines (16,000 cores) at

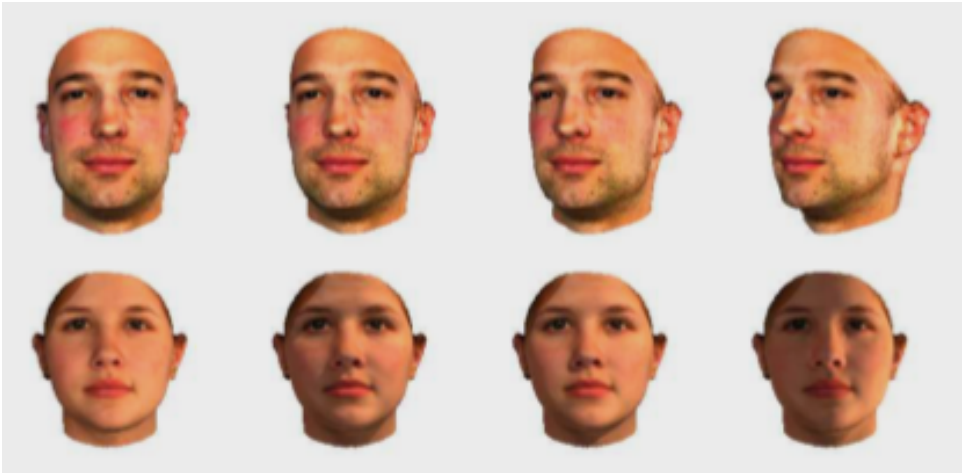

Figure 5: Samples of the face images generated for MIT-CBCL database.

Characteristic	FERET	SCface	MIT-CBCL	Database of Faces
# of images	14126	4160	3240 + 2000*	400
# of individuals	1199	130	10	40
# samples p/individual	12 avg	32	324 + 200*	10
Resolution	256×384	1200×1600	58×58	92×112
Face rotation	180°	180°	0°to 34°	No
Illumination	ambient	ambient + ifr	lfc ctlr	ambient
Artifacts	gl, clcol, ges	mild ges	No	ges, gl

Table I: Databases comparison.

Google using data from YouTube. It was then used to initialize the weights of a DNN which set a new record for performance on ImageNet. Later, in 2012 it was proven that networks with similar performance could be trained with one CPU and two GPUs by Krizhevsky, Sutskever, and Hinton [9]. Taigman, et al. in their 2014 "DeepFace" paper [10] successfully applied this to facial recognition. Since then, Convolutional neural networks (CNNs) have been used in nearly all of the top performing methods on the Labeled Faces in the Wild (LFW) dataset which is currently used as comparison benchmark. Still, there are some implementations which rely on feature-generative methods such as LBP, SVM, PCA and LDA [4] which used to be the state-of-the-art some ten years ago [11].

The main difference between prior methods and current deep learning ones lies in what is being learned by the machine. In LBP facial recognition, weights are applied to specific portions of the face to emphasize or de-emphasize certain regions [12]. Feature learning methods, on the other hand, learn a feature extractor based on the statistics of the training data and have been successfully applied to a variety of different domains and modalities [13].

Even though most research migrated to deep learning, modern facial recognition pipelines still require re-alignment as domain specific engineering. It is also desirable that data is transformed into a form that is easier to work with. Autoencoders are applied to this task. Unsupervised learning based methods take advantage on the fact that autoencoders can be used to pretrain (initialize the weights of) a DNN which is then fine-tuned with labeled data.

DNNs are not the only deep learning approach as CNNs are also being employed on this field. Nearly every method that performs well on LFW utilizes CNNs. A major advantage of CNNs comes from the nature of the convolution operation: a linear translation in the input data causes a linear translation in the feature map. This provides some degree of translation-invariance which is not found in DNNs and autoencoders. Nowadays top scoring algorithms for face recognition can be seen in Table II.

1) Deep Learning Face Recognition: Among the feature learning oriented neural networks *FaceNet* [2] stands atop. This network was designed to directly learn a mapping from face images to a compact Euclidean space where distances directly correspond to a measure of face similarity. It differs from previous deep learning approaches in the fact that the deep convolutional network is trained to directly optimize the embedding itself, while the others train an intermediate bottleneck layer.

For this direct training approach, they use a scheme as in Fig. 6 with triplets of roughly aligned matching / non-matching face patches generated using an online triplet mining method. The method allowed for state-of-the-art face recognition performance using only 128-bytes per face. The network principle bases on learning an Euclidean embedding per image using a deep convolutional network. The network is trained such that the squared L2 distances in the embedding space directly correspond to face similarity. To perform this training, the authors resort to using a triplet based loss function. The triplets consist of two

Name	Method	Images (millions)	Accuracy	Code
FaceNet [2]	CNN	200	0.9963 ± 0.0009	Unofficial
DeepID3 [3]	CNN	0.29	0.9953 ± 0.0010	No
MFRS [14]	CNN	5	0.9950 ± 0.0036	No
VGG Face EL [15]	CNN	5	$0.9913 \pm -$	Yes
DeepFace [10]	CNN	4.4	0.9735 ± 0.0025	Unofficial
FR+FCN [16]	CNN	0.087	0.9645 ± 0.0025	No
TL Joint Bayesian [17]	Joint Bayesian	0.099	0.9633 ± 0.0108	No
High-dim LBP [18]	LBP	0.099	0.9517 ± 0.0113	Unofficial

Table II: State-of-the-art methods score on LFW.

Figure 6: Structure for FaceNet training model as seen in [2].

matching face thumbnails and a non-matching face thumbnail with tight crops of the face area, no 2D or 3D alignment. The loss aims to separate the positive pair from the negative by a distance margin as in Fig. 7.

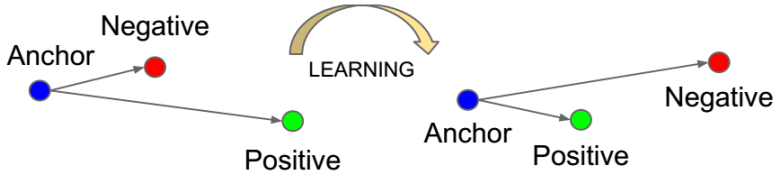

Figure 7: Triplet loss scheme as seen in [2].

The project involved studying two variations of the architecture, called NN1 and NN2 respectively. NN1 is based on regular convolutional layers and has 140 millions of parameters, while NN2 is based on GoogLeNet with inception layers resulting in only 7.5 millions of parameters. table III shows the layers disposition on NN2 network.

Type	output size	depth
conv1	$112 \times 112 \times 64$	1
max pool + norm	$56 \times 56 \times 64$	0
inception (2)	$56 \times 56 \times 192$	2
norm + max pool	$28 \times 28 \times 192$	0
inception (3a)	$28 \times 28 \times 192$	2
inception (3b)	$28 \times 28 \times 256$	2
inception (3c)	$28 \times 28 \times 320$	2
inception (4a)	$14 \times 14 \times 640$	2
inception (4b)	$14 \times 14 \times 640$	2
inception (4c)	$14 \times 14 \times 640$	2
inception (4d)	$14 \times 14 \times 640$	2
inception (4e)	$14 \times 14 \times 640$	2
inception (5a)	$7 \times 7 \times 1024$	2
inception (5b)	$7 \times 7 \times 1024$	2
avg pool	$1 \times 1 \times 1024$	0
fc layer	$1 \times 1 \times 128$	1
L2 norm	$1 \times 1 \times 128$	0

Table III: FaceNet NN2 architecture.

Even though its top holding performance, the fact of its novelty makes for a lack of availability of the network code. There are a few unofficial implementations among the research community developed following the data provided in [2]. Also, this network has been trained using over 200 millions of images posing limitations on third party researchers who should try implementing this architecture. Only big companies such as Facebook and Google's have the capability to gather labeled databases this big so far, and they keep this valuable resource under proprietary rights. Just to have an idea of the massiveness of data in this model we can mention that the training batch size was of 1800 samples.

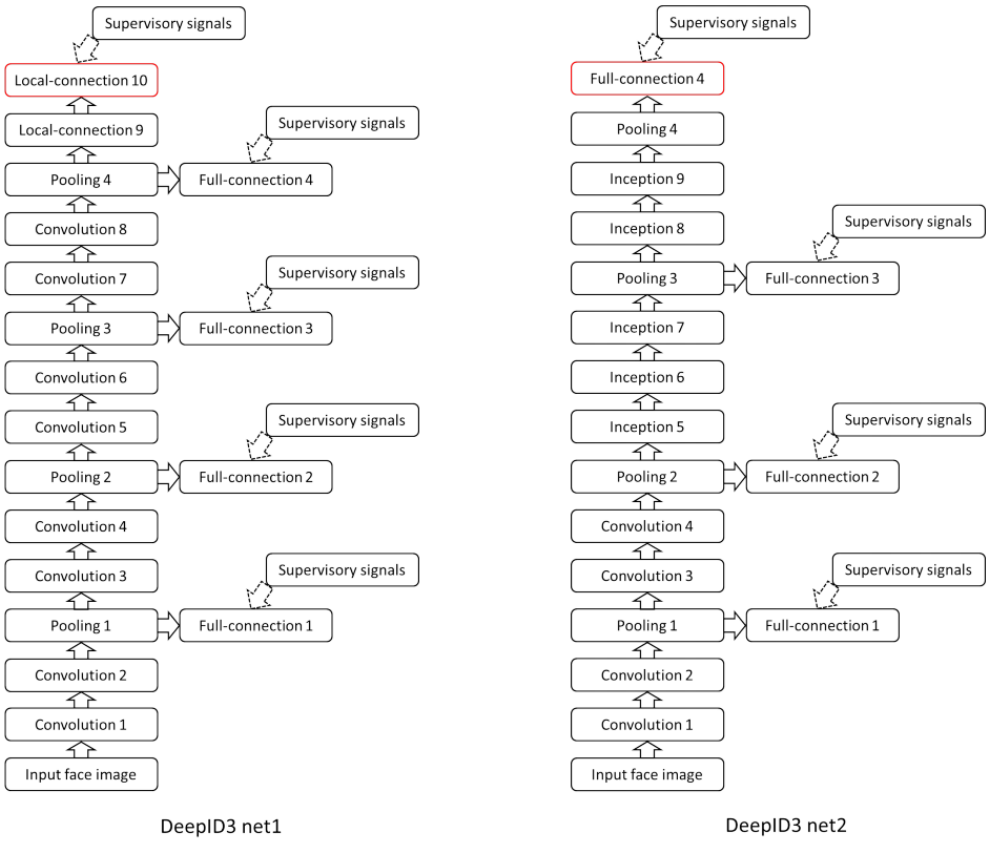

Figure 8: DeepID3 architecture for networks 1 and 2 respectively as depicted in [3].

Another top scorer deep learning method is DeepID3 [3]. The research team in charge of this project aimed for state-of-the-art performance in face recognition through two variations of their network shown in Fig. 8. These two architectures are built from stacked convolution and inception layers as the ones in VGG [19] net and GoogLeNet [20] modified to make them suitable for face recognition. Both consist of fifteen non-linear feature extraction layers plus pooling layers to help to form larger receptive fields and more complex nonlinearities while restricting the number of parameters.

The key difference with other methods are the Joint face identification-verification supervisory signals added to both intermediate and final feature extraction layers during training. This reduced the intra-personal variations in the face representation and alongside with large-scale face identity classification the last hidden layer of the deep neural networks form rich identity-related features.

However, after training these networks sole purpose is to extract features from face regions. Moreover, its implementation involves training several networks, each on a different face region. Then an additional Joint Bayesian model [21] is learned on these features for face verification or identification making DeepID3 a mixed model where the classifier lies outside the network. DeepID3 is trained on a dataset of around 290 thousands of images counting over 10000 subjects. This is considerably smaller than FaceNet, but still makes up for a big dataset.

Another interesting example of neural networks face recognition is VGG FACE [15]. Authors here not only studied an algorithm for face recognition, but also tackled the construction of a publicly available dataset for academic research on the given field. Motivation for the second was, as mentioned before, the fact that most state-of-the-art networks were trained on enormous proprietary datasets only achievable by Internet giants.

The implementation itself consists of three versions of the well known VGG [19] architecture properly modified to suit face recognition tasks. The basic design holds five blocks of convolution layers and three fully connected layers. The first stage of training bases on computing the empirical softmax log loss on an N-ways classification problem with **N** being the number of subjects to recognize. Authors removed the classifier layer after learning and made use of the score vectors for face verification through Euclidean distance computation. On top of it, they found it possible to rise the performance by implementing a triplet loss training scheme over the vector scores obtained in the previous training step. This way, the network ends up learning a face embedding of L-dimensions, where L is the output size of the new last fully connected layer. For the face embedding training all layers but last fully connected are frozen, and it takes very few epochs with relatively high learning rate to train.

On the down side of this approach, various stages of data gathering and training require intense supervision and handwork.

The dataset used was built through a 5 stages process and even then, for the triplet loss training another selection had to be done for an effective hard-negative mining. Some data augmentation was performed by using 4 separate crops of the face images and averaging the resulting feature vectors. Multi-scale testing was enabled by rescaling samples, computing the feature vectors and obtaining he average once again. Finally, all these experiments were carried on NVIDIA Titan Black GPUs with 6GB of onboard memory, using four GPUs together due to the computational load.

The cases of [3], [15], [10] and [16] served as reference for the development and comparison of FaceNet. Moreover, VGG Face is one of the three methods chosen to evaluate for performance comparison in the present work.

III. Method

The goal of this work is to assess which algorithms for face recognition are most suitable for the smartgym project. As such, we present three options involving both classical and deep learning methods which are assessed in the following sections. The first two belong to the feature-engineering type, the Eigenfaces and the Fisherfaces methods, while the third method belongs to the feature-learning neural network type using VGG on its face recognition model.

A. Eigenfaces

Out of the three, Eigenfaces [22] is the oldest method. Dating its publication in 1991, this approach treats face recognition as a two-dimensional recognition problem, considering the fact that faces are are normally upright and thus may be described by a small set of 2-D characteristic views. Face images are projected onto a feature space ("face space") that best encodes the variation among known face images. The face space is defined by the "eigen-faces", which are the eigenvectors of the set of faces. Figure 9 shows a typical case of a feature space for faces, where only the first sample of each class is displayed for a clearer visualization.

The approach transforms face images into a small set of characteristic feature images, called "eigenfaces", which are the principal components of the initial training set of face images. Recognition is performed by projecting a new image into the sub-space spanned by the eigenfaces ("face space") and then classifying the face by comparing its euclidean distance with respect to known individuals. Recognition under reasonably varying conditions is achieved by training on a limited number of characteristic views such as a "straight on" view, a 45' view, and a profile view. Back in its introduction date, the approach had advantages over other contemporary face recognition schemes in its speed, simplicity, learning capacity and relative insensitivity to small or gradual changes in the face image.

According to the authors, much of the previous work on automated face recognition considered predefined measures such as eyes nose and mouth position sufficient. That approach ignored the issue of what aspects of the face stimulus are important for identification. They applied a scheme in which the relevant information in a face image was extracted, encoded as efficiently as possible, and then compared one face encoding with a database of models encoded similarly. One of the simplest approaches to extracting the information contained in an image of a face is to somehow capture the variation in a collection of face images, and use this information to encode and compare individual face images.

Mathematically speaking, the goal is to find the principal components of the distribution of faces. These are the eigenvectors of the covariance matrix of the set of face images. These eigenvectors can be thought of as a set of features which together characterize the variation between face images. Each image location contributes more or less to each eigenvector, so what we

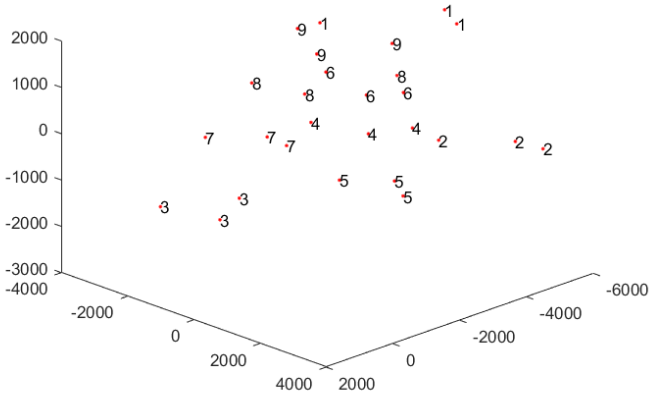

Figure 9: Example of a face space with 10 classes or distinct subjects, showing the relative position of the classes.

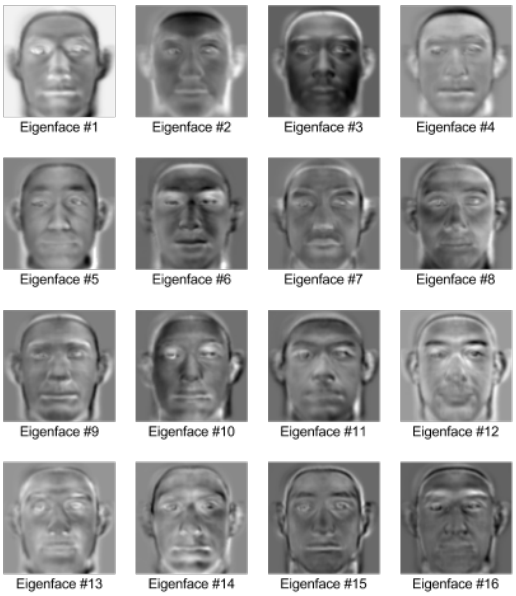

Figure 10: Example of eigenfaces as trained on the MIT-CBCL dataset.

call an eigenface is nothing more than the display of the eigenvector rearranged to the face image dimensions as seen in Fig. 10.

Lets consider a vectorial representation of a gray scale image I of $N \times N$. We can think of it as a vector of dimension N^2, or equivalently a point in an N^2 dimensional space. Therefore a collection of images would map to a point cloud within this space. It is supposed that faces images have a similar overall configuration, so they are not randomly distributed in this huge image space and thus can be described by a relatively low dimensional subspace. By using principal component analysis (or Karhunen-Loeve expansion) there can be found the vectors which best account for the distribution of face images within the entire image space. These vectors define the subspace of face images, which we call "face space". Each vector is of length N^2, describes an $N \times N$ image, and is a linear combination of the original face images. Because these vectors are the eigenvectors of the covariance matrix corresponding to the original face images, and because they are facelike in appearance, the authors refer to them as "eigenfaces".

Let the training set of face images be $x_1, x_2, x_3, ...x_M$. The average face of the set is defined by $\Psi = \frac{1}{M}\sum_{n=1}^{M} x_i$. Each face differs from the average by the vector $\Phi_i = x_i - \Psi$. This set of very large vectors is then subject to principal component analysis which seeks a set of M orthonormal vectors $\mathbf{u}_k$ and their associated eigenvalues λ_k which best describes the distribution of the data. The vectors $\mathbf{u}_k$ and scalars λ_k are the eigenvectors and eigenvalues, respectively, of the covariance matrix

$$\begin{aligned} Cov &= \frac{1}{M}\sum_{n=1}^{M} \Phi_n \Phi_n^T \\ &= AA^T \end{aligned} \tag{1}$$

where $A = [\Phi_1, \Phi_2, ..., \Phi_M]$. Due to matrix Cov being $N^2 \times N^2$, determining the N^2 eigenvectors and eigenvalues is an intractable task for typical image sizes. However, by solving a much smaller $M \times M$ matrix problem. This comes from the condition that the number of data points in the image space is less than the dimension of the space ($M < N^2$), which happens if the number of training samples is relatively small. Given this case, there will be only $M - 1$ meaningful eigenvectors as the rest will have associated eigenvalues equal to zero. Considering the eigenvectors $\mathbf{v}_i$ of $A^T A$ such that

$$A^T A \mathbf{v}_i = \mu_i \mathbf{v}_i \tag{2}$$

Pre multiplying both sides by A we have

$$AA^T A \mathbf{v}_i = A\mu_i \mathbf{v}_i \tag{3}$$

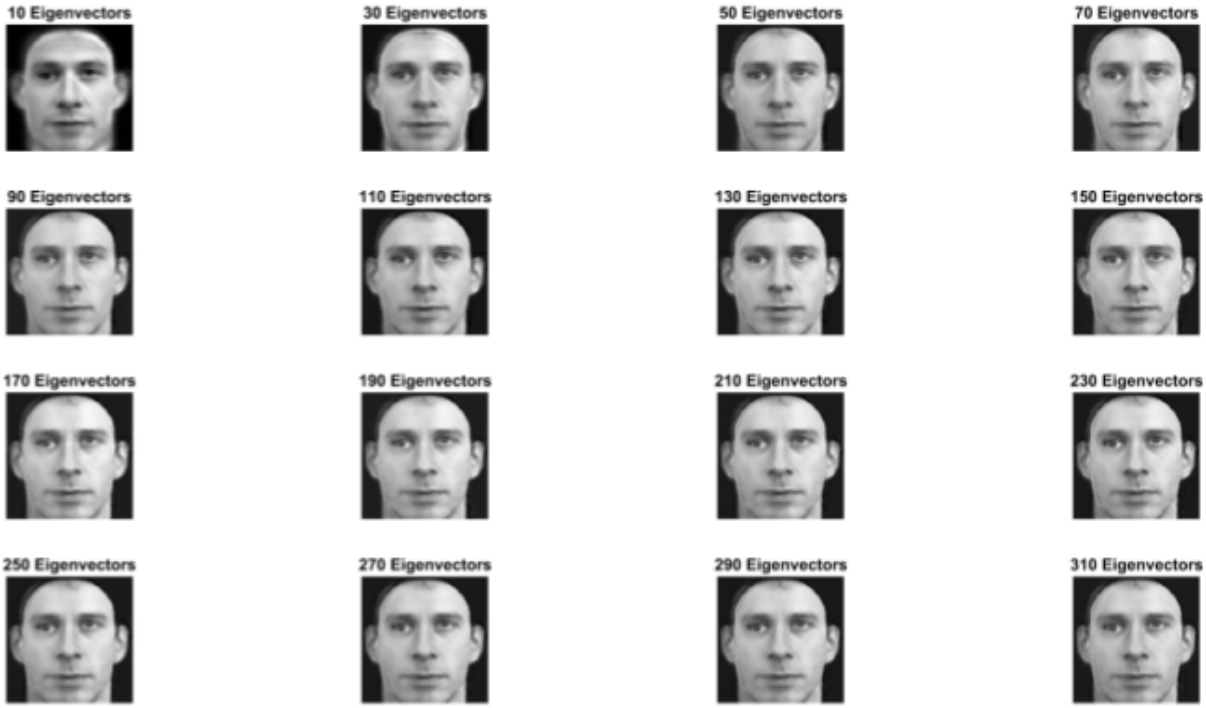

Figure 11: Example of how eigenfaces are used to reconstruct a given image. Notice that the more eigenfaces are used, the more precise the reconstruction.

And so we see that $A\mathbf{v}_i$ are eigenvectors of $Cov = AA^T$. By following this analysis we construct $L = A^T A$ matrix of $M \times M$, where $L_{mn} = \Phi_m^T \Phi_n$ and find the M eigenvectors $\mathbf{v}_l$ of L. These vectors define the linear combination of the M training face images to form the eigenfaces $\mathbf{u}_l$ according to Equation 4.

$$\mathbf{u}_l = \sum_{k=1}^{M} \mathbf{v}_{lk} \Phi_k \qquad l = 1, 2, ..., M \tag{4}$$

In practice, $M \ll N^2$ so the problem's solution es feasible to compute.

Once the eigenfaces are created identification becomes a pattern recognition task. The eigenfaces span an M'-dimensional subspace of the original N^2 image space. The M' significant eigenvectors of the L matrix are chosen as those with the largest associated eigenvalues. The number of eigenfaces to be used is chosen heuristically based on the, eigenvalues.

A new face image (x) is projected into the 'face space' -which means that it is transformed into its eigenface components- by a simple operation,

$$\omega_k = \mathbf{u}_k^T (x - \Psi) \qquad k = 1, ..., M' \tag{5}$$

This describes a set of point-by-point image multiplications and summations. Figure 11 shows how once the eigenfaces' weights have been determined, it is possible to reconstruct the original face image x. The more eigenfaces used, the more accurate the reconstruction, although the computational cost rises too.

The weights form a vector $\Omega^T = [\omega_1, \omega_2, ..., \omega_{M'}]$ that describes the contributions of each eigenface in representing the input face image. The vector is used to find which of a number of pre-defined face classes, if any, best describes the face. The simplest method for determining which face class provides the best description of an input face image is to find the face class *k* that minimizes the Euclidean distance $\epsilon_k = ||(\Omega - \Omega_k)||$, where Ω_k is a vector describing the *k*th face class. Then by simple thresholding over the ϵ_k values, one can decide on the class a given face image belongs to. Also, this technique allows to distinguish up to a certain level which images even contain an image, due to the fact that face images should not lie too far from the 'face space' given by the eigenfaces.

This method is simple to implement, adapts well to image sets with reduced size of classes and training time is proportional to the number of classes. It does have drawbacks when there are too many classes as effective separation between them becomes harder to achieve.

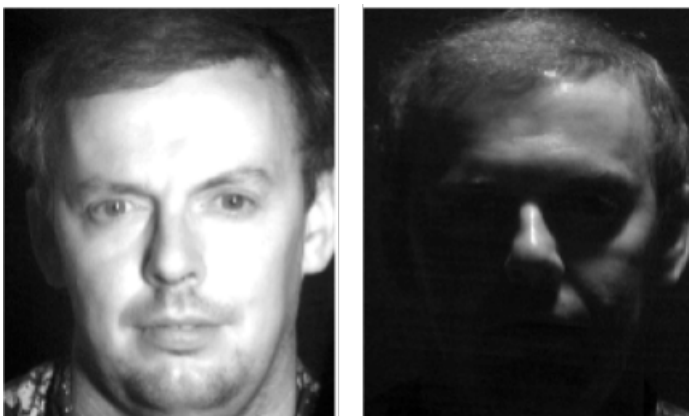

Figure 12: Difference in appearance of the same face with greatly different lighting conditions for better understanding, as shown in [23].

B. Fisherfaces

This method was developed under the same theoretical principles of linearly projecting the image space to a low dimensional subspace as Eigenfaces. However, Fisherfaces was demonstrated to have error lower error rates [23]. Based on Fisher's Linear Discriminant, authors proved that this method produces well separated classes in a low-dimensional subspace, even under severe variation in lighting and facial expressions such as the case of Figure 12.

The main exploits for Fisherfaces approach are the following two principles of photometry and 3D geometry:

1) All of the images of a Lambertian surface, taken from a fixed viewpoint, but under varying illumination, lie in a 3D linear subspace of the high-dimensional image space.
2) Because of regions of shadowing, specularities, and facial expressions, the above observation does not exactly hold. In practice, certain regions of the face may have variability from image to image that often deviates significantly from the linear subspace, and, consequently, are less reliable for recognition.

Authors made use of these observations to find a linear projection of the faces from the high-dimensional image space to a significantly lower dimensional feature space which is insensitive both to variation in lighting direction and facial expression. Then comes the key difference to Eigenfaces, because while using PCA to yield projection directions that maximize the total scatter across all classes -meaning all images of all faces-, this one retains unwanted variation product of intra-class illumination and expressions diversity. Unlike the aforementioned, Fisherfaces seeks to projection directions that are nearly orthogonal to the within-class scatter, projecting away variations in lighting and facial expression while maintaining discriminability.

This method selects ω in 5 to form a vector Ω in such a way that the ratio of the between-class scatter and the within-class scatter is maximized. Let C be number of classes in the training set, Ψ be the average image of all the samples and ψ_i the average image of each class. Let the between-class scatter matrix be defined as

$$S_B = \sum_{i=1}^{C} M_i(\psi_i - \Psi)(\psi_i - \Psi)^T \tag{6}$$

and the within-class scatter matrix be defined as

$$S_W = \sum_{i=1}^{C} \sum_{x_k \epsilon X_i} (x_k - \psi_i)(x_k - \psi_i)^T \tag{7}$$

where ψ_i is the mean image of class X_i and M_i the number of samples in X_i. If S_W is nonsingular, the optimal projection Ω_{opt} is chosen as the matrix with orthonormal columns which maximizes the ratio of the determinant of the between-class scatter matrix of the projected samples to the determinant of the within-class scatter matrix of the projected samples, ie.,

$$\begin{aligned} \Omega_{opt} &= \arg\max_{\Omega} \frac{|\Omega^T S_B \Omega|}{|\Omega^T S_W \Omega|} \\ &= [\omega_1, \omega_2, ..., \omega_{M'}] \end{aligned} \tag{8}$$

where $\{\omega_i \mid i = 1, 2, ..., M'\}$ is the set of generalized eigenvectors of S_B and S_W corresponding to the M' largest generalized eigenvalues $\{\lambda_i \mid i = 1, 2, ..., M'\}$, ie.,

$$S_B\omega_i = \lambda_i S_W \omega_i, \qquad i = 1, 2, ..., M'. \tag{9}$$

Note that there are at most $C-1$ nonzero generalized eigenvalues, and so an upper bound on M' is $C-1$, where C is the number of classes.

In the face recognition problem, one is confronted with the difficulty that the within-class scatter matrix $S_W \in \mathbb{R}^{N\times N}$ is always singular. This stems from the fact that the rank of S_W is at most $M-C$, and, in general, the number of images in the learning set M is much smaller than the number of pixels in each image N^2. This means that it is possible to choose the matrix Ω such that the within-class scatter of the projected samples can be made exactly zero.

In order to overcome the complication of a singular S_W, Fisherfaces avoids this problem by projecting the image set to a lower dimensional space so that the resulting within-class scatter matrix S_W is nonsingular. This is achieved by using PCA to reduce the dimension of the feature space to $M-C$, and then applying the standard Fisher Linear Discriminant [24] to reduce the dimension to $C-1$. Formally,

$$\Omega_{opt}^T = \Omega_{fld}^T \Omega_{pca}^T \tag{10}$$

where

$$\begin{aligned} \Omega_{pca} &= \arg\max_{\Omega} |\Omega^T S_T \Omega| \\ \Omega_{fld} &= \arg\max_{\Omega} \frac{|\Omega^T \Omega_{pca}^T S_B \Omega_{pca} \Omega|}{|\Omega^T \Omega_{pca}^T S_W \Omega_{pca} \Omega|}. \end{aligned} \tag{11}$$

Here the optimization for Ω_{pca} is performed over $N^2 \times (M-C)$ matrices with orthonormal columns, while the optimization for Ω_{fld} is performed over $(M-C)\times M'$ matrices with orthonormal columns. In computing Ω_{pca}, we have thrown away only the smallest $C-1$ principal components.

To resume, Fisherfaces approaches the face recognition problem in a similar way to Eigenfaces considering that both aim to find a lower dimensional space projection that allows for linear discrimination over classes. The main difference lies in the scatter maximization performed. Fisherfaces allows to tackle illumination and expression variations in a more effective way, reaching lower error rates without a much higher cost in terms of training time for reduced training sets.

C. VGG Face

The VGG neural network was originally developed to prove that replacing a low number of layers of large filters in CNNs by a bigger series of layers of smaller filters, preserved the effective receptive area, reduced number of parameters to compute and provided better performance in terms more complex features learned and more precision in classification tasks.

As mentioned in Section II, this approach was developed alongside a public dataset with the intention of showing both a protocol to create useful databases for deep learning, and a face recognitionn method with state-of-the-art performance [15]. Feature learning, also known as deep learning methods share with classical feature engineering approaches the concept that within all set of data there is an underlying structure. The difference in these approaches is that while the latter aims to design the feature vectors to classify this underlying structure effectively, the first aims to learn the intrinsic features of this structure that allow classification. Note that following this concepts, comparison between methods becomes more than just noting positive recognition rates.

The inherent nature of deep learning states that a limited amount of training samples will not suffice for obtaining a powerful classifier with good generalization power. On the other hand, with enough samples per subject and a proper training, a neural network can achieve great performance even for a large number of classes where Eigenfaces or even Fisherfaces might find it difficult to achieve an effective linear discrimination.

Initially in this face recognition implementation, the deep architectures ϕ are bootstrapped by considering the problem of recognizing $C = 2622$ unique individuals, setup as a C-ways classification problem. The CNN associates to each training image $x_i, i = 1, 2, ..., M$ a score vector $\mathbf{y}_i = W\phi(x_i) + b \in \mathbb{R}^C$ by means of a final fully-connected layer containing C linear predictors $W \in \mathbb{R}^{C\times D}, b \in \mathbb{R}^C$, one per identity. These scores are compared to the ground-truth class identity $c_i \in \{1, 2, ..., C\}$ by computing the empirical softmax log-loss

$$E(\phi) = -\sum_i \log\left(\frac{e^{\langle \mathbf{e}_{c_i}, \mathbf{y}_i\rangle}}{\sum_{q=1,2,...,C} e^{\langle \mathbf{e}_q, \mathbf{y}_i\rangle}}\right) \tag{12}$$

where $\mathbf{y}_i = \phi(x_i) \in \mathbb{R}^D$, and $\mathbf{e}_c$ denotes the one-hot vector of class $\mathbf{c}$.

After learning, the classifier layer (W, b) can be removed and the score vectors $\phi(x_i)$ can be used for face identity verification using the Euclidean distance to compare them. However, the scores can be significantly improved by tuning them for verification in Euclidean space using a "triplet loss" training scheme. The latter is essential to obtain a good overall performance according to the authors.

With the triplet-loss training, authors aimed at learning score vectors that performed well in the final application, i.e. identity verification by comparing face descriptors in Euclidean space. This is similar in spirit to "metric learning", and, like many metric learning approaches, is used to learn a projection that is at the same time distinctive and compact, achieving dimensionality reduction at the same time.

The output $\phi(x_i) \in \mathbb{R}^D$ of the CNN, pre-trained following the previous paragraph's explanation, is L2-normalized and projected to a $L \ll D$ dimensional space using an affine projection

$$\mathbf{y}_i = W' \frac{\phi(x_i)}{||\phi(x_i)||_2} \qquad , W' \in \mathbb{R}^D. \tag{13}$$

While this formula is similar to the linear predictor learned above, there are two key differences. The first one is that $L \neq D$ is not equal to the number of class identities, but it is the (arbitrary) size of the descriptor embedding (in [15] the authors set $L = 1024$). The second one is that the projection W' is trained to minimize the empirical triplet loss

$$E(W') = \sum_{(a,p,n)\in T} \max\left\{0 \quad , \quad \alpha - ||\mathbf{y}_a - \mathbf{y}_n||_2^2 + ||\mathbf{y}_a - \mathbf{y}_p||_2^2\right\}, \qquad \mathbf{y}_i = W' \frac{\phi(x_i)}{||\phi(x_i)||_2}. \tag{14}$$

Differently from the previous learning stage, there is no bias being learned here as the differences in 14 would cancel it. Here $\alpha \geq 0$ is a fixed scalar representing a learning margin and T is a collection of training triplets. A triplet (a, p, n) contains an anchor face image a as well as a positive $p \neq a$ and negative n examples of the anchor's identity. The projection W' is learned on target datasets such as LFW and YTF following their guidelines.

Being based on the VGG-16 classifier network from [19], VGG Face relies in the power from a deep Convolution Neural Network architecture -see Fig. 13- to learn face classification features. It comprises 11 blocks, each containing a linear operator followed by one or more non-linearities such as ReLU and max pooling. The first eight such blocks are of convolutional type as the linear operator is a bank of linear filters (linear convolution) each operating over a restricted area of the image. The last three blocks are instead Fully Connected (FC); here the size of the linear operator filters matches the size of the input data, such that each filter "senses" data from the entire image. All the convolution layers are followed by a rectification layer (ReLU) as in [9]; however, differently from [9] and similarly to [19], they do not include the Local Response Normalization operator.

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 13: VGG Face net layers scheme, as seen in [15].

The first two FC layer outputs are 4,096 dimensional and the last FC layer has either C = 2,622 or L = 1,024 dimensions, depending upon the loss functions used for optimization, either C-way class prediction or L-dimensional metric embedding. In the first case, the resulting vector is passed to a softmax layer to compute the class posterior probabilities. The architecture in Fig. 13 is not the only one the authors of [15] tested, as they evaluated performance of two more variations which had more convolutional layers. The input to these VGG Face networks is a face image of size 224×224 with the average face image (computed from the training set) subtracted – this is critical for the stability of the optimization algorithm.

Even though the authors of [15] used triplet loss training to enhance the performance, VGG Face can already be used for face verification by implementation of the softmax loss training stage. This will be proven useful in the following Experiments Section as limited time for this work completion restricted the tests over the evaluated methods that could be finished.

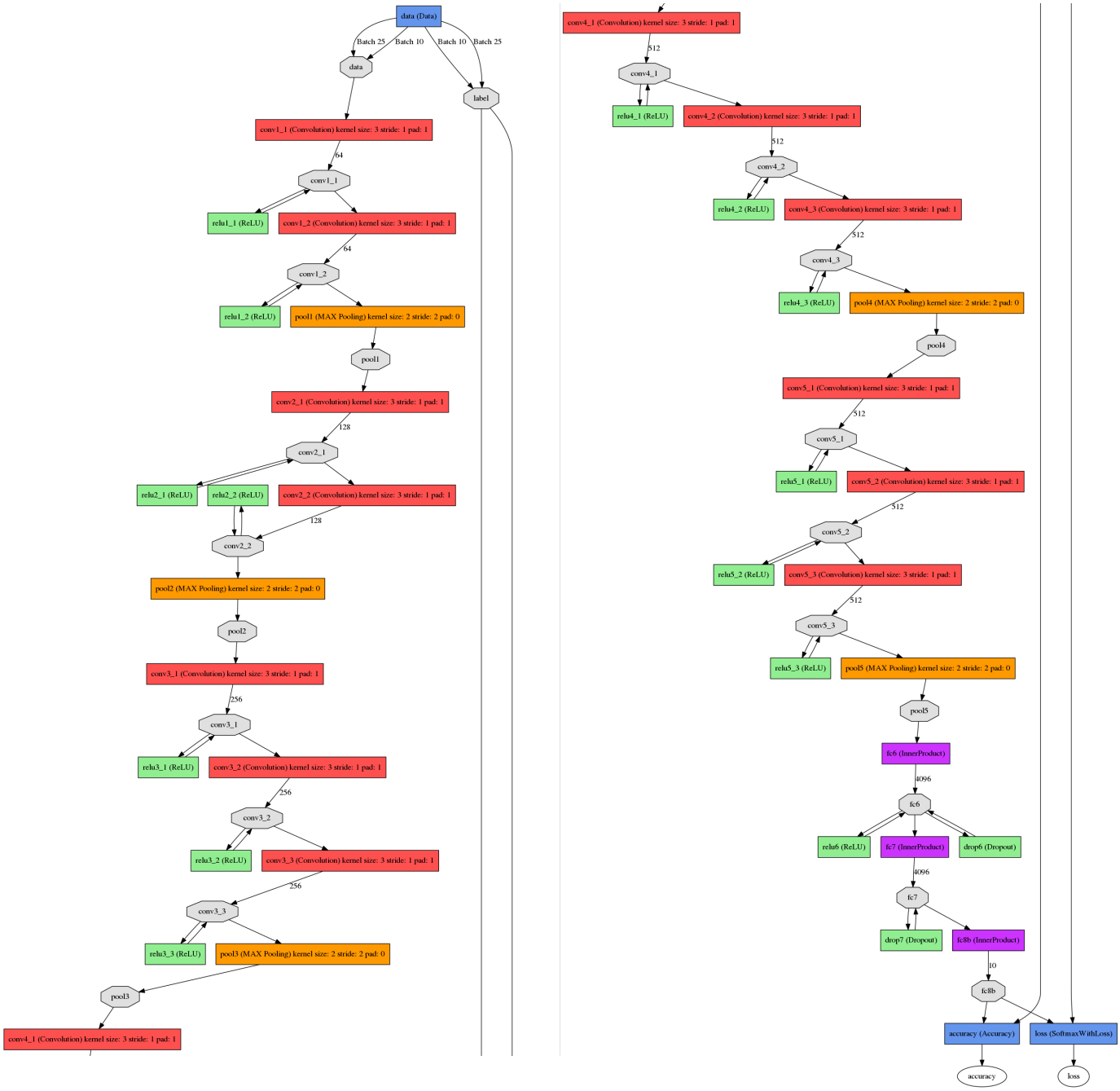

Figure 14: VGG Face net as implemented in this work.

The architecture implemented in the present work is shown in Fig. 14. It follows the same structure as the one from Fig. 13 with exception of the number of outputs in the last FC layer, which have been adapted to the number of individuals of the experimental database. Due to our dataset being several times smaller than that one of [15], the use of Dropout layers and the limitation to just one FC layer with an output of 4096 becomes important in a sense of reducing the network capacity, seeking to avoid overfitting this way.

IV. Experiments

In the previous section we exposed the methods chosen as possible implementations for the smartgym project. In this section we will explain the experiments performed to compare each method's pros and cons. First, some aspects about the public database employed for prototyping will be discussed. Then we will explain the pre-processing carried out on the data used which involves face detection and frontalization of the samples. Following this we describe the final experimental dataset. Finally the experiments for overall performance as well as evaluation on face rotation and sample size will be detailed.

A. The Data

The original goal was to test performance over a public dataset first to tune-up code, and then move to a dataset made of samples taken in FCB's junior basketball gymnasium, which is one of the places where the final system should be implemented. Given the magnitude of the project, the present work was carried parallel to the setup of the cameras and server system in the facilities of FCB for field tests. The present performance tests were carried out on the public MIT-CBCL [7] dataset, replicating conditions to be expected from the FCB's data allowing an exhaustive assessment of the methods on a public database.

MIT-CBCL dataset was not the only one used for early stages, SCFace [6] was also studied. The main reason for choosing MIT's set was the fact that the synthetically generated face masks allowed for a great samples per subject rate. Furthermore, the samples included controlled illumination changes (36 lighting conditions overall) and precise face rotation (from $0°$ to $34°$ in steps of $4°$). Also, the lack of background in the samples helps to avoid training interference caused by artifacts at the back, which influence in performance is not to be tested until further on in the project.

The low number of subjects remains a main drawback, but priority at this early stage of the project was given to cleanliness of the data for an easier debugging of the methods implementations. It was this aspect that conditioned the decision of not using SCFace, which had thirteen times the amount of subjects of MIT-CBCL, but the rotated views amounted to only 7 per subject and exhibited an undesired imprecision regarding angles. Some rotated views in the right direction did not match in angular intensity with their left counterpart. This could have been tackled by a process similar to the one used in [7] to generate the 3D synthetic masks, but it exceeded the scope of this work.

B. Images pre-processing

For the implementation, it has been proposed a scheme most face recognition algorithms use, where the faces in the incoming images are detected and frontalized before using them as input to the evaluated methods. This sets some leverage for this early stage testing, as Fisherfaces and VGG Face are inherently more robust to position changes of faces than Eigenfaces. While many frontalization algorithms attempt to approximate 3D facial shapes for each query image, we follow the lines of [25] which uses a single, unmodified, 3D surface as an approximation to the shape of all input faces. On the one hand, the authors of this method have demonstrated through tests over LFW dataset that any impact this simplification has on facial appearances is typically negligible. On the other hand, this leads to a straightforward, efficient and easy to implement method for frontalization as seen in Fig. 15. In fact, faces remain easily recognizable despite the approximation and, more importantly, the frontalized faces are aggressively aligned thereby improving performances over previous alignment methods.

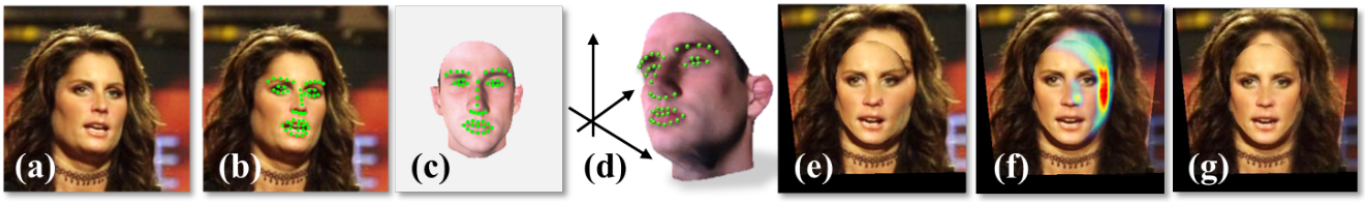

Figure 15: Frontalization process overview as explained in [25]. (a) Query photo; (b) facial feature detections; (c) detection placed over 3D model; (d) textured 3D computer graphics model rendered; (e) back-projection of the query intensities to the reference coordinate system; (f) estimated visibility due to non-frontal poses, overlaid on the frontalized result. Warmer colors reflect less visible pixels; (g) the final frontalized result.

The process following the sequence in Fig. 15 comes down to entering the query photo and running the facial feature detections. Then the same detector is used to localize the same facial features in a reference face photo, produced by rendering a textured 3D computer graphics model. From the 2D coordinates on the query and their corresponding 3D coordinates on the model a projection matrix is estimated to back-project query intensities to the reference coordinate system. To obtain the final frontalized result, the facial appearance in regions where pixels are less visible is produced by borrowing colors from corresponding symmetric parts of the face.

This system has some limitations though, as one cannot expect to find a matrix that can effectively back-project when the hidden regions of the face are too big or too many, as in the case of high angular rotations. The first stage of the experiments was indeed to assess up to which degree of face rotation, the frontalization could offer resulting images that were usable for the training of the face recognition algorithms. Figure 16 shows an example of a good frontalization.

C. The Experimental Dataset

The experimental database used for evaluation of the face recognition algorithms was obtained by processing MIT-CBCL entire synthetic images (a total of 3240) with the frontalization algorithm from [25]. Once frontalized, the images were inspected in order to check that they were semantically correct. The inspection demonstrated that face rotations over $24°$ produced poor results in this aspect, and so those samples of higher angle were discarded. Figure 17 is an example of a semantically incorrect

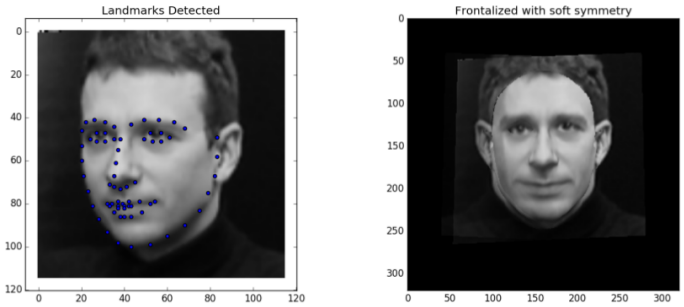

Figure 16: Frontalization example using one of MIT-CBCL test set images.

frontalization due to over rotation of the face. Once bad frontalization cases are removed, the database contains 2520 samples, 250 per subject with rotation ranging from $0°$ to $24°$. With this number, we can evaluate progressively the influence of samples per class in the final performance of each method.

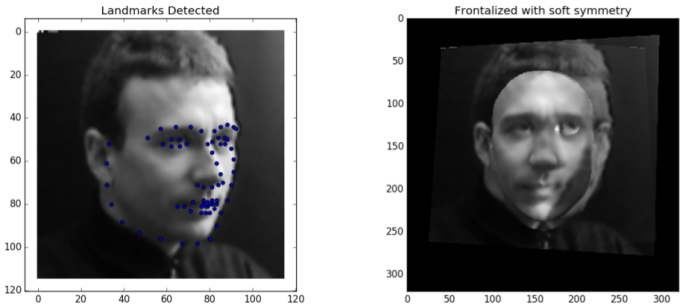

Figure 17: High rotation angles do not allow for a good face reconstruction as in this case.

After frontalization, an of-the-shelf object detector based on Viola-Jones [26] is passed through the entire set so as to eliminate as much background as possible and make the region of interest occupy as much image space as possible. The processed images are resized to 227×227 (input size for the VGG Face net), and turned to gray scale as for the moment we are not interested in the information color might provide. Figure 18 displays the output of this post-frontalization processing.

Figure 18: Face image pre-processing overview: (left) input synthetic image; (center) frontalization output; (right) face detection and image resize.

To maximize the fidelity of the results even though the limited amount of data available, we resort to a $\boldsymbol{k}$-fold cross validation scheme. We generate 10 datasets made of the 2500 samples, each one with its own train and validation set. For this, we split the images in a 90% for training and 10% for test, with no image is shared by each test set. Therefore, we produce 10 sets of 2020 training images with 250 test samples. Each individual has equal number of samples as the others, both in training and testing in order to avoid unwanted biases.

D. Training

When the training for Eigenfaces and Fisherfaces consists in the resolution of a system of equations, CNN's training scheme differs. VGG Face's training consists in the forward passing through the net of a batch of data images followed by a back-propagation back to the input in order to compute the gradients to modify the filter weights. This requires the use of a solver structure, which was configured to set different learning rates progressively.

As we use the original weights from [15], we need to perform a fine tuning rather than a training from scratch. Being this case we start with a low learning rate of 0.0001, and it is lowered by one degree of magnitude by epoch $\#80$, then again on $\#120$ and further on by epochs $\#350$, $\#425$, $\#450$ and $\#500$. Batch size is set to 20 for hardware restrictions and the weight decay was set to 0.0005 and we used 'RMSProp' as solver.

E. Performance Assessment

The goal of the experiments was to obtain overall performance metrics on the three methods chosen. For this, and considering the current limited database it was implemented a 10-fold cross validation scheme with the following protocol:

1) Train the algorithm on the first training set.
2) Evaluate the True Positive Rate when running the trained model over the corresponding test set.
3) Repeat 2) ten times and average the TPR obtained to eliminate aleatory biases.
4) Select the following training set and repeat process from 1) with its corresponding test set.
5) Once the algorithm has run over the ten sets, compute the final TPR measure as the mean among the averaged TPRs obtained to eliminate sample dependent biases.

This amounts to 100 runs of the face recognition method over the data.

The evaluation protocol was applied to a series of experiments, each focusing on extracting different data about the methods. First we conducted a global performance measure by using the entire dataset on the 10-fold cross validation protocol to have a general idea of what to expect on the behavior of each algorithm on a 10 persons database with plenty of images per subject to train from. In the case of Eigenfaces and Fisherfaces, training time was also computed and averaged.

Next experiment consisted in restricting the number of samples per subject within each training set to evaluate performance dependency on the sample per class size. As the dataset accounted for 7 angle views: $[0°, 4°, 8°, 12°, 16°, 20°, 24°]$, the floor set for samples per subject was 7 samples, one of each rotation angle. From then on, the other samples per subject rates were set to 35 (5 per angle), 70 (10 per angle), 140 (20 per angle) and 196 (28 per angle).

Again, the evaluation protocol was conducted, one run per each sample-per-subject size for a total of 700 runs for each method. The number of times each algorithm is run on the data pursues the elimination of stochastic biases that may appear during tests, as well as data dependent biases because of the specific samples each training and test set have. It is important to mention that having equal number of samples per angle and for each subject is meant for adding information on the influence of face rotation in the performance of the three methods in comparable way. For this, not only global TPRs per sample size are computed, but also the TPR for each angle of face rotation is calculated and saved for results analysis.

Notice that the last amount of samples per subject was set to 196 instead of 210. This is due to further restrictions on the dataset that are required for VGG Face training which needs not only a train and test set, but also a validation test for the training process. Therefore, in the case of VGG Face, the sets were split in a ratio of 80% to train, 10% to validate and 10% to test, leaving a train set of 202 images per subject at most and the others with 25 images per subject. Moreover, in the case of VGG Face the training is performed only once per sample size as it is further time consuming than the other two and we deem this aspect as non defining for the results.

To resume the experiments, the goal was to test how well each of the three algorithms did when having full range data, how their performance varied according to the degree of face rotation in the images, and how was the positive rate affected by the size of the sample. The experiments were conducted on an Intel® Core i7 6500U CPU processor with 2.6 GHz clock, 12 GB DDR3L 1600 MHz SDRAM and one NVIDIA® GeForce® GT 940M 2GB DDR3 platform. The machine ported both Windows 10 and Ubuntu 16.04LTS operative systems. Eigenfaces and Fisherfaces were implemented in MATLAB® R2016b on Windows, while VGG Face was implemented in PyCaffe on Ubuntu due to limited support for Caffe on Windows. The following section presents the results obtained for these experiments.

V. RESULTS

In this section we detail the results of the experiments carried out throughout the present work. Experiments involve not only a face recognition method, but also a database processing for enhancing performance. We present samples of the pre-processing results so as to have a better visualization of the condition under which the methods were tested. Figure 19 show frontalization outputs for lower angles and Fig. 20 displays what we obtained for diverse subjects under the same face rotation conditions.

A. Training and Prediction Time

Table IV shows average time for training the whole model on the data according to sample size on the three methods. VGG Face has also included time per epoch computation in seconds between brackets next to the total training time. Most of the CNN training involved between 100 and 150 epochs for the set number of samples. An extra training sequence of 500 epochs was carried over the entire dataset taking around 10 hours of training. It must be mentioned that low GPU DRAM forced to train on CPU, causing an increment in time metrics.

Figure 19: For lower rotation angles, back-projection reaches better results.

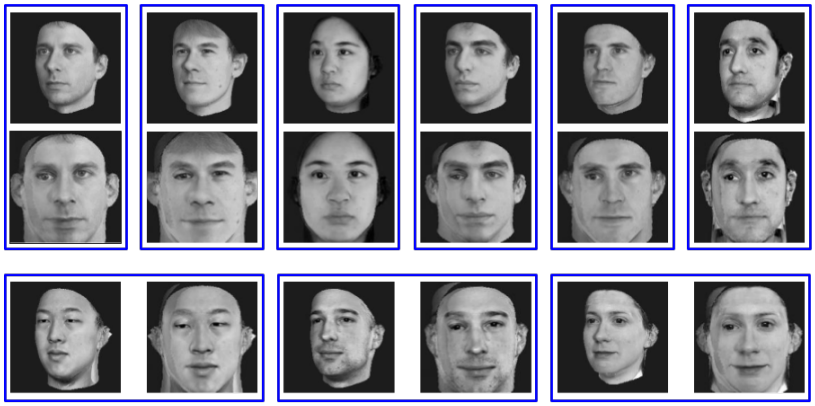

Figure 20: Outputs for the rest of the subjects under 20° face rotation.

Training Time

Samples per subject	Eigenfaces(s)	Fisherfaces(s)	VGG Face (min)
7	0.16	0.37	129.86 (77.92s)
35	1.09	1.46	177.28 (70.91s)
70	3.10	4.71	176,75 (70.7s)
140	11.26	24.40	180,24 (72,09s)
196	21.58	66.92	205,93 (82,37s)

Table IV: Training time for each sample size. VGG Face has overall time displayed in minutes, with time-per-epoch in seconds between brackets.

Notice that for the dataset used, training time for classical methods is negligible compared to that of a CNN. In most cases, one epoch alone takes as much time to train as the whole dataset for Eigenfaces or Fisherfaces. On the other hand, classical methods expose an exponential behavior in training time as sample size rises. As Fig. 21 shows, this behavior is accentuated in Fisherfaces due to the maximization of both between class and within class scatter.

On the testing side, Table V shows the average prediction time computed for each method. Time taken for an image prediction is a constant for each method, regardless of the training set size. Times are computed by running the performance assessment scheme detailed in section IV-E.

Prediction Time

Eigenfaces	Fisherfaces	VGG Face
1.0765 s	1.0517 s	7.1067 s

Table V: Average time for each method to recognize a face, pre-processing is excluded.

B. Performance versus Sample Size

The overall performance over sample size results obtained can be seen in Table VI. The top performance for each method is highlighted in bolds, being Fisherfaces the method with best overall scores for the given data. This method reached 100% accuracy with 20 samples per angle and 140 samples per subject, and performed over 95% with only 35 samples per subject.

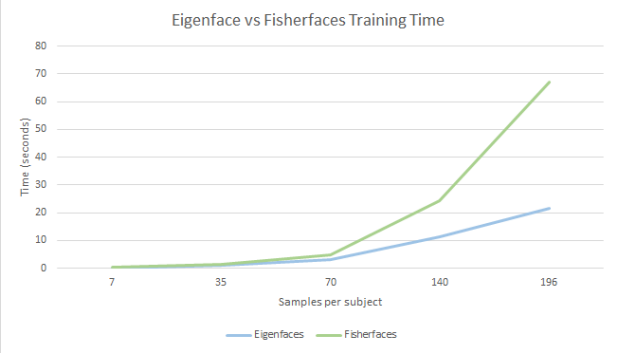

Figure 21: Training time escalates exponentially with sample size for methods that seek for a lower dimension projection to maximize scatter.

Overall Performance

Samples per subject	Eigenfaces %	Fisherfaces %	VGG Face %
7	42.14	60.85	48.4
35	73.43	97.9	52
70	81.28	98.87	**62.8**
140	92.42	**100**	65.2
196	**98.04**	100	38.4

Table VI: Global performance for each sample size. Notice the top score for each method in bold.

C. Performance over Face Rotation

Here we display a more detailed analysis of performance including true positive rates for each face rotation condition besides the sample size. Again, bold fonts signal the top scores for each method and each sample size. Figure 22 allows a better visualization of the influence of face rotation in recognition precision.

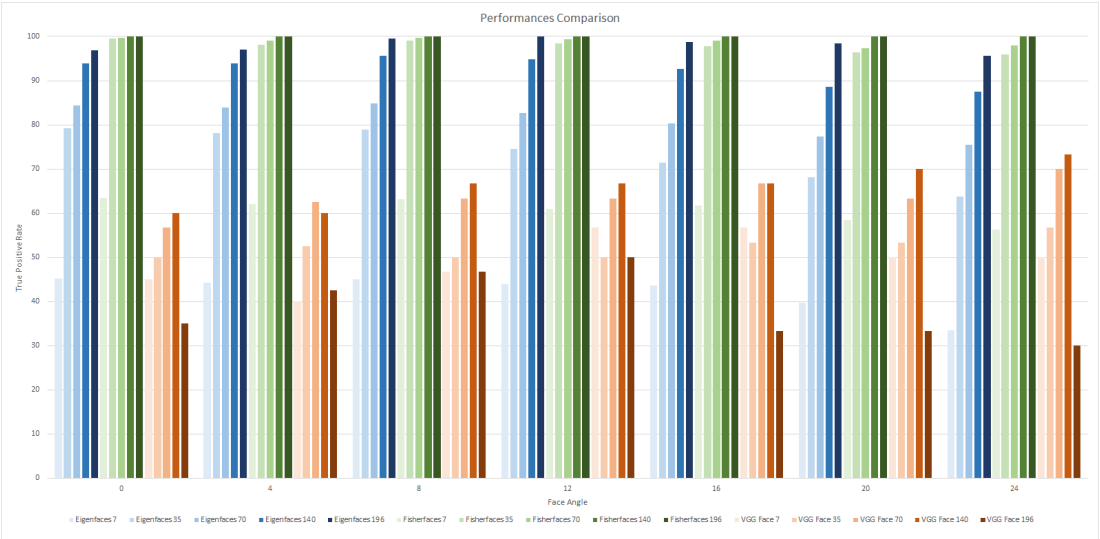

Figure 22: Comparison graph in which the data from Table VII can be visualized. Eigenfaces in blue tones, Fisherfaces in greens and VGG Face in browns.

Samples per subject	**Face rotation (°)**	Eigenfaces %	Fisherfaces %	VGG Face %
7	0	**45.22**	**63.39**	45
	4	44.22	61.99	40
	8	44.97	63.2	46.67
	12	43.89	60.87	**56.67**
	16	43.56	61.8	56.67
	20	39.64	58.5	50
	24	33.47	56.22	50
35	0	**79.19**	**99.5**	50
	4	78.1	98.13	52.5
	8	78.84	99.08	50
	12	74.59	98.42	50
	16	71.35	97.83	53.33
	20	68.19	96.33	53.33
	24	63.72	96	**56.67**
70	0	84.43	99.67	56.67
	4	83.88	99.08	62.5
	8	**84.9**	**99.67**	63.33
	12	82.66	99.33	63.33
	16	80.3	99	**66.67**
	20	77.31	97.33	63.33
	24	75.5	98	70
140	0	93.83	**100**	60
	4	93.83	100	60
	8	95.7	100	66.67
	12	**94.82**	100	66.67
	16	92.67	100	66.67
	20	88.58	100	70
	24	87.5	100	**73.33**
196	0	96.8	**100**	35
	4	97.05	100	42.5
	8	99.47	100	46.67
	12	**100**	100	**50**
	16	98.8	100	33.33
	20	98.5	100	33.33
	24	95.67	100	30

Table VII: True positive rates for each face rotation within every sample size.

VI. Conclusions

In this section we discuss the results obtained from the experiments, first analyzing the run time aspect, and then going into the precision performance. We expose our conclusions on this first stage of the smartgym project, and present future lines of work in accordance. We conclude that we have been able to have a preliminary insight to the performances of three different methods for face recognition, and how precision is affected by training sample size and face rotation.

A. Time Analysis

Due to inherent differences between classical and deep learning methods, VGG Face's training time is computed in a different way as seen in Table IV. Convolutional neural networks are designed to learn from massive data through extended periods of time, loading batches of images on each epoch to pass forward and then back propagate to compute the gradients. As the batch size is constant throughout the entire experimentation (set to 20 samples per epoch due to hardware limitations), the training time per epoch remains more or less constant. The major influence of the sample size over training time for the network lies in the amount of data this one can learn from the samples considering an adequate learning capacity, the more the better. Table IV evidences all this as it can be appreciated that time per epoch remains between 70 and 80 seconds, while the total training time (displayed in minutes) is proportional to the sample size because there is more data to learn from and takes more epochs for the network to have 'seen' every sample at least once.

Opposed to VGG Face, Eigenfaces and Fisherfaces compute a linear discrimination for classes using the projection of images into a lower dimensional space. We explained in Section III that the feasibility of this methods depends on solving a much smaller $M \times M$ matrix problem. This comes from the condition that the number of data points in the image space is less than the dimension of the space ($M < N^2$), which holds as long as the number of subjects and sample size are small. The problem escalates when one of the two or both grow as the dimensionality of the image space is incremented exponentially. In these cases, computation time rises because M approaches N^2, and so we see that behavior in Fig. 21. This condition is to be considered for the project's future experiments on performance for different number of individuals to recognize.

Prediction times achieved by the methods are of near real-time performance. It must be considered that neither of the codes has been optimized, a task that is left for future works on the implementations within FCB's basketball gymnasium. Moreover, possible optimization methods contemplate the use of face tracking to replace continuum face recognition once the person identity has been acquired, which could enhance the system's speed.

B. Precision Performance Analysis

Results show that for environments under restricted conditions with a low number of subjects, classical methods still outperform naive deep learning implementations. We said 'naive' as the VGG Face net was implemented as classifier (not feature extractor) with a class probabilities vector as output and not a single label. We have already mentioned that according to [15], state-of-the-art performance is achieved by combining Softmax loss training with Triplet loss training. This VGG Face implementation manifests prone to overfitting because of the low number of samples compared to the learning capacity of the CNN, and cannot reach the performance of the other methods with the training schedule applied. A net version with an extra layer to compute euclidean distance and output a single label was projected but its training has yet to be concluded.

It was originally intended to test gradual increments in the complexity of the network once the FCB dataset was made available. Not having completed data acquisition in the club's gymnasium, further architecture modifications were left for smartgym project's future stages.

With regards to the classical methods, both Eigenfaces and Fisherfaces have reached over 90% scores with enough samples per subject. Fisherfaces being the top scorer, has demonstrated greater potential for further implementations with the camera set on FCB's facilities to obtain some field metrics. It must be pointed that this work's experiments aimed to do a first assessment of which methods to chose with a smartgym implementation in mind. The data used for testing is restricted to images of faces under controlled environment (restricted background, rotation and expression conditions). And so they must not be considered a definitive parameter to choose one method. Further test should be conducted, with more subjects, harder environment conditions and more important, field data straight from FCB's facilities were the final implementation should be deployed.

The present work allows to conclude that face recognition is everything but a closed task. The approach choice is highly dependent on the deployment target place. Whereas deep learning methods keep improving performance for in-the-wild environments, pushing boundaries on precision, the requisite of large data bases still makes classical methods more appealing for small scale implementations.

C. Future Lines of Work

The present work has opened lines that could be followed in order to advance the smartgym project. In first place, the construction of the FCB's database according to criteria established by the study of the face recognition databases in Section II. A protocol for ground truth and training samples acquisition was developed in order to provide valuable samples for the database. Said protocol is not further explained in the present work as the construction of the database has been left out for not being launched yet.

Parallel to the present methods study, another work line has been in development with the goal of setting up the data acquisition and field testing system. This work is explained in the Appendix A. This setup will be used for gathering the training dataset of the FCB players, and capture sample videos to develop and test implementation prototypes.

Secondly, tests that take into account the variation in the number of subjects on the database should be carried as a second stage for the performance measure. This becomes really important considering that each gymnasium facility in the FCB serves for around 100 athletes, although shifts reduce the number of subjects simultaneously within the gym to roughly 10 to 20.

VGG Face architecture as presented here could be revised, and trained with triplet loss to improve performance according to [15]. Also, studies on the robustness of the methods to face accessories and expressions should be considered, as the implementation will have to work continuously all year around. Time flow allows for face changes due to both aging (which might be subtle) and look changes (which tend to be more drastic though temporary), be it for look trends or even injuries.

Moreover, the future works under this smartgym project should analyze runtime and performance in FCB's facilities with simultaneous detections to define the final hardware requirements. This could include the evaluation of on-line training methods to cope with time influence in face features and avoid periodical complete re-trainings.

APPENDIX A
EXPERIMENTAL SETUP FOR FACE RECOGNITION - VERSION 1.0

This section gives a brief overview of the environment that was designed to implement an experimental setup for Face Recognition. The hardware of this system consists of 4 Raspberry Pi's that are used for the acquisition of the images that are sent to a server where they are processed. Next, an application was designed to capture videos of a person sitting in front of a camera. These videos can then be labeled by another application that was designed for this purpose. This process is called the creation of a Ground Truth which is needed to train the algorithms for Face Recognition. Finally, there is a process running on each Raspberry for Face Detection. The faces are sent to the server where they are analyzed by the Face Recognition algorithms.

A. Hardware

The hardware of the system consists of a Router (with Wifi), four Raspberry Pi's and a workstation (a PC) that acts as a server. Each Raspberry (Figure 23) acts as a client and connects to the server by Wifi. See Figure 24 for a schematic overview of the system.

Figure 23: Raspberry Pi 3 with camera.

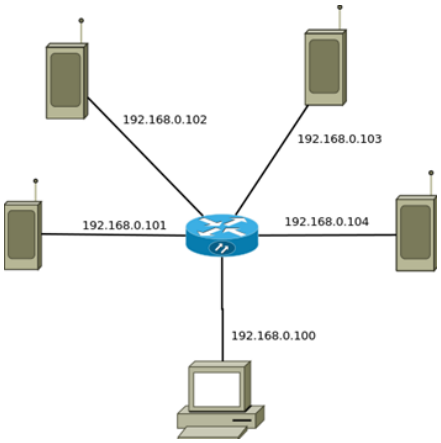

Figure 24: System architecture.

The Raspberries use a modified Linux OS named Raspbian for streaming and the PC works with Ubuntu 16.04LTS. The server contains two different applications. On the one hand, there is an application that was designed to capture videos of a person sitting in front of the camera (Figure 25). These videos are grabbed with some meta data (viz. group, user, and a description) for later use. For this process to be fluent, instead of connecting the Raspberry by Wifi, it is recommended to use a direct Ethernet connection to the router.

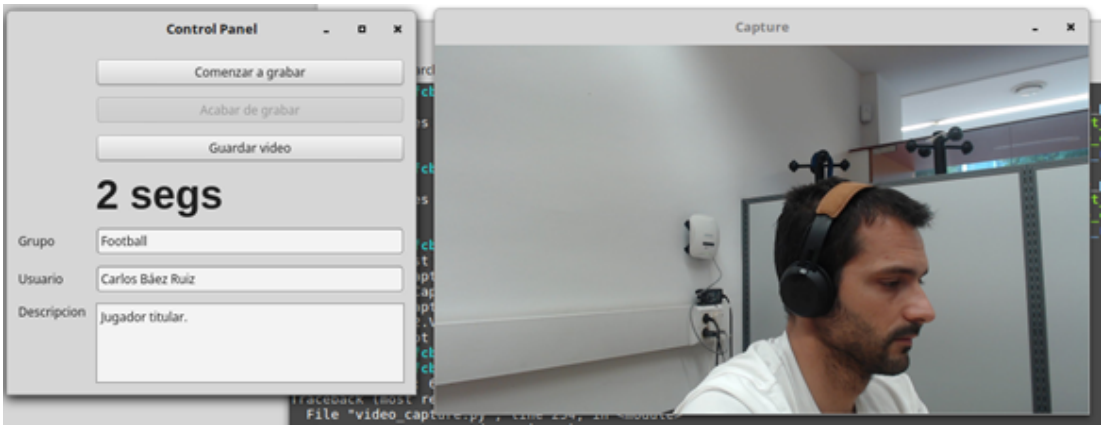

Figure 25: Video caption application demonstration.

On the other hand, there is an application that is designed for labeling videos. In this way a so called Ground Truth of the system is created. This GroundTruth is then used to train the Face Recognition Algorithms (Figure 26). The software installed on each Raspberry captures images from the "PiCam" camera and if this image contains a face, the face is cropped and sent to the server. The face will then be analyzed by the Recognition software on the server.

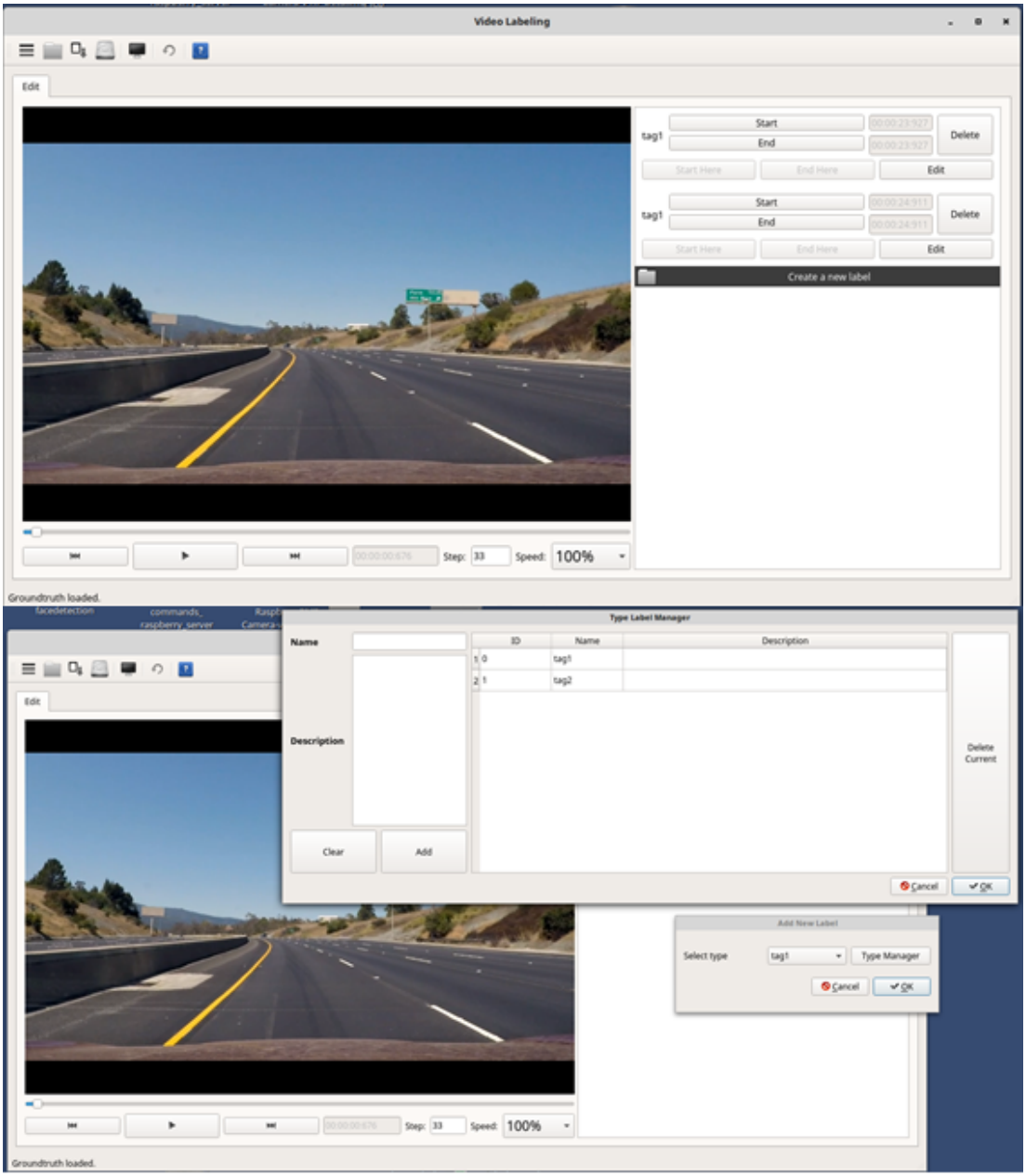

Figure 26: Application for Ground Truth annotation.

Once the system was set, the first tests for ground truth annotation were conducted as Figure 27 and 28 show. The first one displays the application initialization, and the second shows the faces captured.

Finally, there were conducted some tests on the whole system with the face recognition implementation feature added. For the tests the models used were the classical Eigenfaces and Fisherfaces as implemented in [27]. Figure 29 shows some of the results obtained during these tests.

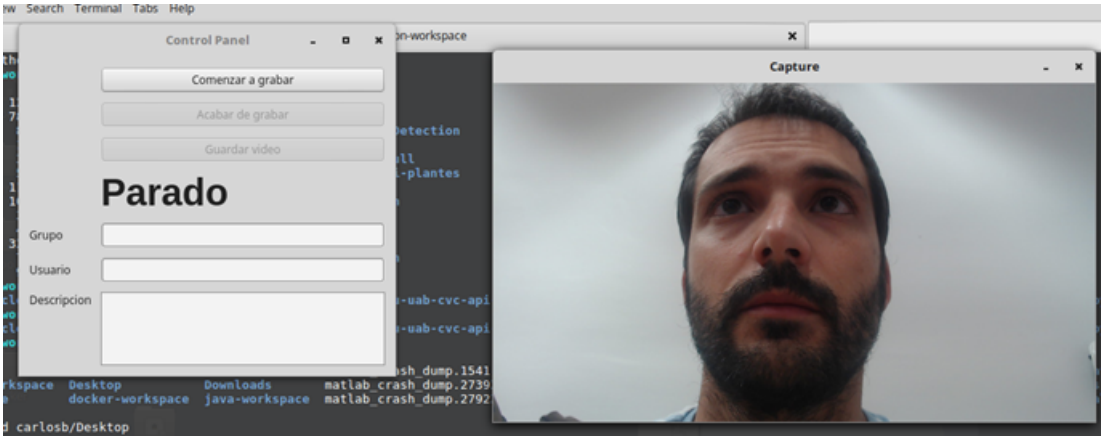

Figure 27: Application initialization for face capture.

Figure 28: Face images captured during tests on the ground truth generator app.

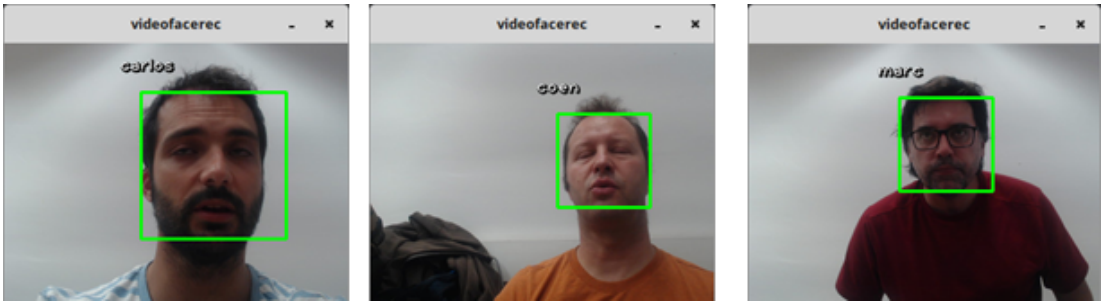

Figure 29: Face images captured during tests on the ground truth generator app.

ACKNOWLEDGMENTS

The author would like to thank F.C. Barcelona for their support and disposition to carry this collaboration project, in particular to Raúl Peláez Blanco and Joan Ramón Tarragó; Carlos Báez and Coen Antens from CVC for their assistance in the experimental setup for data acquisition and face recognition test.

REFERENCES

[1] G. B. Huang, M. Ramesh, T. Berg, and E. Learned-miller, "Labeled faces in the wild: A database for studying face recognition in unconstrained environments," 2007.
[2] F. Schroff, D. Kalenichenko, and J. Philbin, "Facenet: A unified embedding for face recognition and clustering." in *CVPR*. IEEE Computer Society, 2015, pp. 815–823.
[3] Y. Sun, D. Liang, X. Wang, and X. Tang, "Deepid3: Face recognition with very deep neural networks." *CoRR*, vol. abs/1502.00873, 2015.
[4] C. Zhang, X. Liang, and T. Matsuyama, "Generic learning-based ensemble framework for small sample size face recognition in multi-camera networks," *Sensors*, vol. 14, no. 12, pp. 23 509–23 538, 2014. [Online]. Available: https://doi.org/10.3390/s141223509
[5] P. J. Phillips, S. Z. Der, P. J. Rauss, and O. Z. Der, *FERET (face recognition technology) recognition algorithm development and test results*. Army Research Laboratory Adelphi, MD, 1996.
[6] M. Grgic, K. Delac, and S. Grgic, "Scface – surveillance cameras face database," *Multimedia Tools and Applications*, vol. 51, no. 3, pp. 863–879, 2011.
[7] B. Heisele, B. Weyrauch, V. Blanz, and J. Huang, "Component-based face recognition with 3d morphable models," *2012 IEEE Computer Society Conference on Computer Vision and Pattern Recognition Workshops*, vol. 05, p. 85, 2004.
[8] Q. Le, M. Ranzato, R. Monga, M. Devin, K. Chen, G. Corrado, J. Dean, and A. Ng, "Building high-level features using large scale unsupervised learning," in *Proceedings of the 29th International Conference on Machine Learning (ICML-12)*, ser. ICML '12, J. Langford and J. Pineau, Eds. New York, NY, USA: Omnipress, Jul. 2012, pp. 81–88.
[9] A. Krizhevsky, I. Sutskever, and G. E. Hinton, "Imagenet classification with deep convolutional neural networks," in *Advances in Neural Information Processing Systems*, vol. 25, 2012.
[10] 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*. IEEE, Jun. 2014, pp. 1701–1708.
[11] P. J. Phillips, P. J. Flynn, W. T. Scruggs, K. W. Bowyer, J. Chang, K. Hoffman, J. Marques, J. Min, and W. J. Worek, "Overview of the face recognition grand challenge." in *CVPR (1)*. IEEE Computer Society, 2005, pp. 947–954.
[12] T. Ahonen, A. Hadid, and M. Pietikainen, "Face Description with Local Binary Patterns: Application to Face Recognition," *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, vol. 28, no. 12, pp. 2037–2041, Dec. 2006.
[13] S. Balaban, "Deep learning and face recognition: the state of the art," in *SPIE Defense+ Security*. International Society for Optics and Photonics, 2015, pp. 94 570B–94 570B.
[14] E. Zhou, Z. Cao, and Q. Yin, "Naive-deep face recognition: Touching the limit of LFW benchmark or not?" *CoRR*, vol. abs/1501.04690, 2015.
[15] O. M. Parkhi, A. Vedaldi, and A. Zisserman, "Deep face recognition." in *BMVC*, X. Xie, M. W. Jones, and G. K. L. Tam, Eds. BMVA Press, 2015, pp. 41.1–41.12.
[16] Z. Zhu, P. Luo, X. Wang, and X. Tang, "Recover canonical-view faces in the wild with deep neural networks." *CoRR*, vol. abs/1404.3543, 2014.
[17] X. Cao, D. P. Wipf, F. Wen, G. Duan, and J. Sun, "A practical transfer learning algorithm for face verification." in *ICCV*. IEEE Computer Society, 2013, pp. 3208–3215.
[18] D. Chen, X. Cao, L. Wang, F. Wen, and J. Sun, "Bayesian face revisited: A joint formulation." in *ECCV (3)*, ser. Lecture Notes in Computer Science, A. W. Fitzgibbon, S. Lazebnik, P. Perona, Y. Sato, and C. Schmid, Eds., vol. 7574. Springer, 2012, pp. 566–579.
[19] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *CoRR*, vol. abs/1409.1556, 2014.
[20] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions." in *CVPR*. IEEE Computer Society, 2015, pp. 1–9.
[21] D. Chen, X. Cao, L. Wang, F. Wen, and J. Sun, "Bayesian face revisited: A joint formulation." in *ECCV (3)*, ser. Lecture Notes in Computer Science, A. W. Fitzgibbon, S. Lazebnik, P. Perona, Y. Sato, and C. Schmid, Eds., vol. 7574. Springer, 2012, pp. 566–579.
[22] M. Turk and A. Pentland, "Eigenfaces for recognition," *J. Cognitive Neuroscience*, vol. 3, no. 1, pp. 71–86, Jan. 1991.
[23] P. N. Belhumeur, J. P. Hespanha, and D. J. Kriegman, "Eigenfaces vs. fisherfaces: Recognition using class specific linear projection." pp. 711–720, 1997.
[24] R. O. Duda and P. E. Hart, *Pattern Classification and Scene Analysis*, ser. A Wiley Interscience Publication. Wiley, 1973.
[25] T. Hassner, S. Harel, E. Paz, and R. Enbar, "Effective face frontalization in unconstrained images." *CoRR*, vol. abs/1411.7964, 2014.
[26] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, vol. 1, 2001, pp. I–511–I–518 vol.1.
[27] P. Wagner. Facerec. [Online]. Available: https://github.com/bytefish/facerec