Face2Face: an innovative way to face recognition
DL-IC 2018 Project

D'Amico Edoardo
Politecnico di Milano
damicoedoardo95@gmail.com

Gabbolini Giovanni
Politecnico di Milano
giovanni.gabbolini@gmail.com

Parroni Federico
Politecnico di Milano
federico.parroni@live.it

Abstract

In this paper we present a method to compare two images and get a similarity measure of these. Common usages of such a technique are several, first of all, authentication and face matching.
This system, called Face2Face, uses a deep convolutional neural network to map a couple of images into a vector of features which capture the similarity between the two images. Then, this vector is processed by the last fully-connected layers of the network which output a measure in the range [0,1] representing the probability of having depicted the same person on the two images. Training has been done by stacking couples of images one on top of the other, exploiting convolution filters in order to capture face similarities. This method boosts also the size of the training set: in fact, each image of the N in the training set can be coupled with all the others, so the training samples that can be obtained $\in O(N^2)$.
The main benefit of this mechanism are essentially the almost absence of hardware needed (only a camera) and the ease of use for the actual implementation.

1. Introduction

Nowadays, authentication is one of the most critical point of computer security. Authentication is a mechanism that allows a user to exhibit a proof of his identity in order to be recognized by a particular system. Aim of this project is to implement a biometric authentication software, based on face recognition, using a convolutional neural network. The only hardware requirement is a camera. The process of authentication is composed by the following steps:

- User records a set of photos of himself (sample photos) and saves them into the system
- When user wants to authenticate, the system asks for a photo
- If the similarity between the sample photos and the current one is greater than a fixed threshold, user is authenticated positively

The main advantage of this approach is the cost/performance tradeoff that we can achieve: currently, costs of commercial biometric scanners are in the order of hundreds of euros, while a suitable camera for this technique costs about 10-20€ and the probability of granting access to a wrong user is in the order of 1 over 1000.

This work should provide a different approach from the current state of the art to compare two images using a deep convolutional neural network, resulting in a lightweight and real-time process. The neural network takes as input two images and outputs a similarity value between 0 and 1, so it can be directly used without other intermediate softwares. The methodolody used to train the network requires a quite low number of photos to build a large amount of training samples, as described later in 4.

2. Related work

Our work is based on training a deep convolutional neural network, using a purely data driven approach, to extract features from couple of images than use it as input to a fully connected network to say wheter two images representing faces are of the same person or not. Similar work has been carried out some using directly deep convolutional neural network to give a response other using it as a support for the extraction of features used as input for SVMs.
The most relevant are reported below.

Schroff *et al.* [1] use a deep convolutional neural network to generate an embedding $f(x)$, from an image x into a feature space $\mathbb{R}^d$, such that the squared distance between all faces of the same identity is small, whereas the squared distance between a pair of face images from different identities is large. To achieve that has been used the so called 'triplet loss' that allows the faces for one identity to live on a manifold, while still enforcing the distance of two different faces to be maximum. The work

presented does not directly solve the problem of the face recognition but create a proper enviroment in which a simple K-NN classifier can solve the task.

Aleju [9] proposes a network presenting two branches, one per image. Each branch applies a few convolutions and ends in a fully connected layer. The outputs of both branches are then merged and further processed by another fully connected layer, before making the final yes-no-decision (whether both images show the same person).

Taigman *et al.* [10] propose a multi-stage approach that aligns faces to a general 3D shape model. A multi-class net-work is trained to perform the face recognition task on over four thousand identities. The authors also experimented with a so called Siamese network where they directly opti- mize the $L1$-distance between two face features. Their best performance on LFW (97.35%) has been obtained from an ensemble of three networks using different alignments and color channels. The predicted distances (non-linear SVM predictions) of those networks are combined using a non-linear SVM.

Zhenyao *et al.* [11] employ a deep network to warp faces into a canonical frontal view and then learn CNN that classifies each face as belonging to a known identity. For face verification, PCA on the network output in conjunction with an ensemble of SVMs is used.

3. Proposed approach

To accomplish our work we used a convolutional neural network. This kind of model is nowadays the most widely used for Image Recognition tasks. Many ways have been proposed to tackle the problem of stating similarity among different images, for example Schroff *et al.* [1]. Our solution is based on the idea of using directly convolutional layers to extract similarity, which is in our case expressed in terms of identity of people. In order to follow this idea, we used as input of our convolutional neural network two gray scale images stacked one upon the other. Then, two-dimensional convolutional layers filter those two levels images, transforming the initial input. The vector which comes from the flattenization of the last convolutional layer output represents an embedding of the initial input which would naturally tell the relation between the two images in terms of similarity among identities of people appearing in those. Then the vector of features is used as input of a fully connected neural network, which will, at the really end, output a number between 0 and 1, *i.e.* the probability of having two photos who depict the same person.

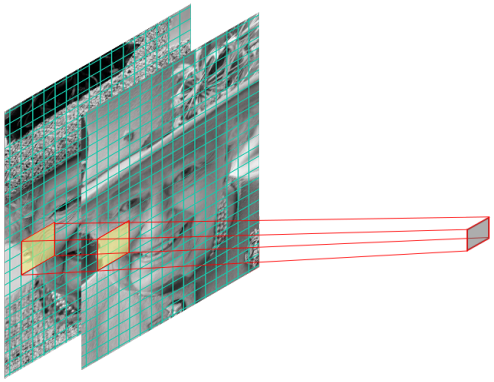

Figure 1: Convolution between two stacked image. Large convolution filters can help to capture similiarities.

3.1. Mathematics

We can consider training samples as i.i.d observations from a Bernoulli random variable. We want that our model approximates as well as possible this Bernoulli variable (t_n can assume only values 0 or 1):

$$P(t_n = 1|x_n, \mathbf{w}) = y(x_n) \tag{1}$$

where $y(x_n)$ is the output of the convolutional neural network for the n^{th} sample.
We want to maximize the likelihood of getting the right ouput. To do so, we have to choose the vector of coefficients **w** such that:

$$\underset{\mathbf{w}}{\operatorname{argmax}}\, J(\mathbf{w}) = \underset{\mathbf{w}}{\operatorname{argmax}} \prod_{n}^{N} {y_n}^{t_n}(1 - y_n)^{1-t_n} \tag{2}$$

Passing to the negative log, we get the standard binary crossentropy:

$$\underset{\mathbf{w}}{\operatorname{argmin}}\, J(\mathbf{w}) = -\sum_{n=1}^{N} \Big[t_n \log y_n + (1 - t_n) \log(1 - y_n) \Big] \tag{3}$$

3.2. Model Selection

The model selection has been one of the main challanges during our activity: we did not have other already existing works from which transfer the structure of the network, at least no other research product which was using the same approach to achieve the same goal. So first we decided to go for a Cross Validation among the possible models. Even if the amount of data that we managed to harvest (4.1.1) was enough to proceed with this idea, the methodology appeared to be computationally unfeasible. So we decided to go for a standerd validation approach. Of course we did not validate every possible model but instead we listed a set of fifteen networks of growing complexity. As a guideline for

Layer	Kernel	Output shape	# of params
Conv2D	3x3 (16)	(80, 80, 16)	304
MaxPooling2D	2x2	(40, 40, 16)	0
Dropout	p=0.1	(40, 40, 16)	0
Conv2D	3x3 (16)	(40, 40, 16)	2320
MaxPooling2D	2x2	(20, 20, 16)	0
Dropout	p=0.1	(20, 20, 4)	0
Conv2D	3x3 (16)	(20, 20, 16)	2320
MaxPooling2D	2x2	(10, 10, 16)	0
Dropout	p=0.1	(10, 10, 16)	0
Conv2D	3x3 (16)	(10, 10, 16)	2320
Flatten	(1600)	0	
Dense	128	(128)	204928
Dropout	p=0.1	(128)	0
Dense	128	(128)	16512
Dense	128	(128)	16512
Dense (softmax)	2	(2)	258
Total			245474

Table 1: Structure of the network

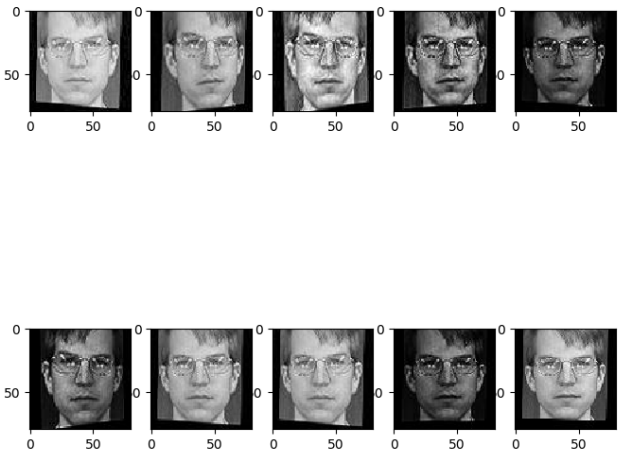

Figure 2: Example of nine augmented photos obtained from a starting one, which is depicted at the bottom-right. We can see the good variety of images obtained from our augmentation procedure.

this ranking we have mainly considered Bengio *et al.* [2] To have the most unbiased estimate of the true error from our validation set, we randomly split the data into validation and training set before the actual validation procedure, and before any of the candidate network was trained and evaluated on the data. When validating the various models, we used Adam Optimizer: this guaranteed to be free from the choice of the learning rate, removing one degree of freedom from our validation procedure. The resulting model is presented in Table 1.

All the models have been trained for threehundred epochs changing dinamically the training data: when the model started overfitting *i.e*. when the training and test error were starting to be uncorrelated, we swapped to a different training set, sampled from the whole set of training samples. Every sampled chunk counted roughly fifty thousand couples of images. This technique was on one hand necessary, because the whole dataset was not fitting on memory, and on the other hand turned out to be a powerful regularization tool.

4. Experiments

4.1. Datasets

4.1.1 Own Dataset

We gathered our own data through a web page [3] that we set up. Through it, people had the possibility to upload their own photos. As a result, we obtained a set of more than eighty folders of photos, each one belonging to a different person. On average, we managed to get ten photos per person. Those photos are characterized by really regular poses of the faces, because the contributors were explicitly asked in the website to shot the photos keeping the camera in from of them, while they were watching it, as they were unlocking a device.

Data Augmentation The quantity of data that we obtained is not even close to the usual amount used in large scale Deep Learning projects. So we proceeded with augmentation procedures of our training set. In particular we resorted to the Augumentor Python library [4]. This library allows to generate, starting from a set of images, other ones, obtained from various transformations of the former set. Those transformations can be customized and can be applied randomly to each image. It follows that each augmented image is the combination of the application of a random number of transformation specified by the developer, resulting in a great variety in the augmented set.

The transformations applied, with probabilities, are:

- Flip left-right with probability 0.5;
- Skew left-right with probability 1: the skew magnitude is random and varies from 0 to 0.2, which experimentally guarantees resulting images which are not deformed;
- Alter brightness with probability 1: we perform a non linear trasformation as suggested in [5], using parameters $\theta = 1$ and $\phi = 1$ respectively. In particular we used this transformation increasing the pixel bright-

ness and decreasing it with probability 0.5 in both cases;

- Adaptive equalization [6] with probability 0.5, using a clip limit of 0.01;

In particular the last two methods were employed in order to obtain a wider variety of lighting conditions, which turned out to be a characteristic that was lacking the most in our own dataset. Using this augmentation procedure, we crafted five brand new images from just one, obtaining fifty images per person on our dataset.

Data preprocessing The data that we managed to harvest and the results of the augmentation procedure were affected by noise and really low regularity. To hope to get good results, we needed to have all the images in a standardized form factor. To accomplish this idea, we set up a pipeline useful to apply to all the images those following steps:

1. Find the face in the image and crop it;
2. Resize the image to a standard size;
3. Rotate the image in order to get the person eyes in a fixed position;

Finding a face in an image and pick the bounding box which includes it is a really known task in Computer Vision and really efficient and reliable algorithms exist. We resorted to the HOG Algorithm [7]. The resulution is set to the standard size of 80x80 pixels. A more tough problem is rotating an image so that the eyes are aligned at the same position, however a detailed description of how to tackle this problem can be found in [8]

Couple creation At this point, we have a considerable number of folders, one for each identity, each one containing a set of, on average, fifty augmented photos. Still, we are missing our actual dataset, which is composed of couples of images and a labels: if the two photos come from the same folder, then the label will be 0, otherwise 1. Since we are considering couples, the number of training samples grows not linearly with the number of photos, but quadratically: for any photo in a folder, we can pair it with all the other photos on the folder and with an equal number of photos picked from other random folders. In formulas, a folder with N photos will give $2(N-1)^2$ training samples. Using this trick, at the really end we managed to have a balanced dataset with more than one million samples.

4.1.2 Academic Datasets

Labelled Faces in the Wild (LFW) We have tested the performance of our network in this well known dataset

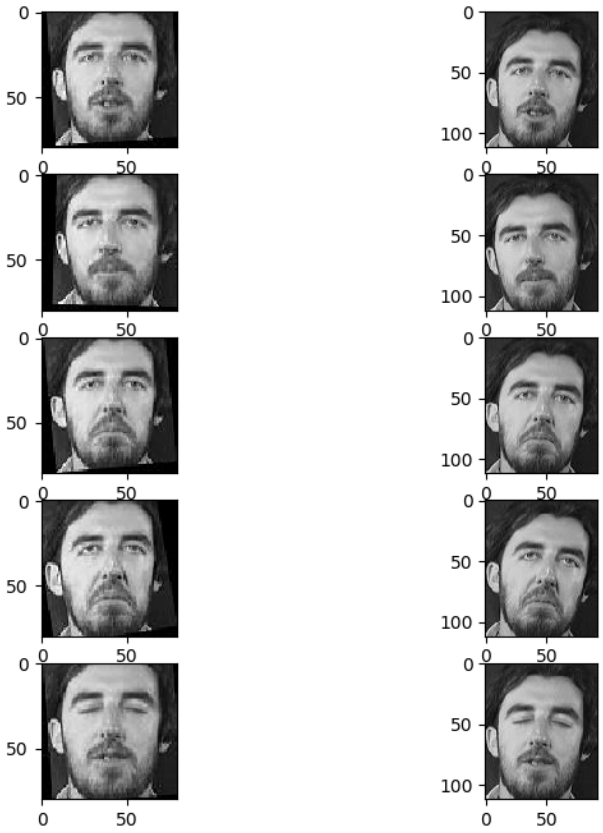

Figure 3: Examples of the application of the pipeline. On the left we see the processed images, on the right we have the original ones. We see the difference of sizing, the eye alignment and the face that has been cropped

(standard de-facto academic dataset for face verification). We have built the couples and assigned the labels as we did in our own dataset, and then evaluated the accuracy.

4.2. Evaluation

We have evaluated our method on a test set that accounts for approximately the 20% of the intial dataset that we gathered up, so the test data has the same distribution of our training set, but disjoint identities.

A substantial difference between the training and test set images is that the test set has not been augmented so that the evaluation of the performance will be unbiased as much as possible with respect to the training set, augmentation has been performed only on the training set (see next subsection).

We have evaluated the method on the face verification task (say wheter two face images are representing the same person) using as prediction the output of the net $Y(i, j)$

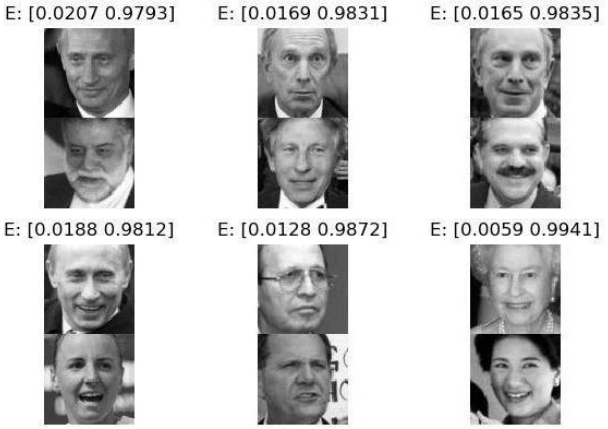

Figure 4: Examples of false positive training samples, obtained with a threshold equals to 0.975. On top of each couple we have the prediction of the network *i.e*. the second number, but all those images were labelled as 0, infact they depict the same person

representing the $probability$ of having the same person on the two images. All faces pairs (i, j) of the same identity are denoted with N_{same}, whereas all pairs of different identities are denoted with N_{diff}.

We define the set of all *true accepts* as

$$TA(thld) = \{(i, j) \in N_{same}, withY(i, j) > thld\} \quad (4)$$

these are the face pairs (i, j) that were correctly classified at treshold $thld$. Similarly we can define the *False accepts* as

$$FA(thld) = \{(i, j) \in N_{diff}, withY(i, j) > thld\} \quad (5)$$

is the set of all pairs that was misclassified as *same*, this figure of merit has a main role on the evaluation of our work since the method has been created with the aim of being used as a security check, and so false accepts cannot be tolerated.

Obtained performance On our test set we have reached

test accuracy = 0.915 with *threshold*=0.5

To further evaluate the performance, also the ROC curve and the AUC parameter has been analyzed (Figure 5).

The AUC obtained is 0.97 and from the ROC curve we can extract important information on which value the treshold of the model has to be set depending on the utilization purpose of the net. Since the model has been developed for security reasons, a possible idea is to tolerate at most a 0.1% of false positive. To achieve that the treshold has

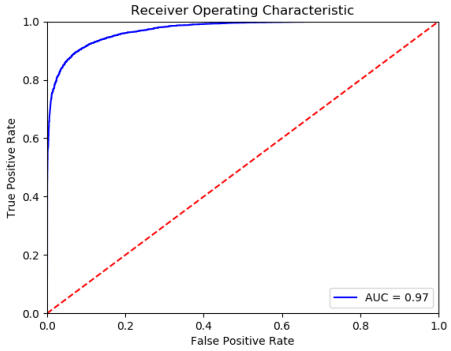

Figure 5: ROC curve of the model.

Treshold	Accuracy	Precision
0.5	0.915	0.9278
0.975	0.845	0.9985

Table 2: Perfomance comparison between the case of threshold 0.5 and 0.975 on the Test set.

to be set to 0.975 as suggested by the ROC curve, so that the network will recognize a couple of faces (i, j) as the same person only when $Y(i, j) > 0.975$. The performance between two different tresholds can be compared using the confusion matrices (Figure 6) taking into account also the

$$Precision(thld) = \frac{TA(thld)}{TA(thld) + FA(thld)}. \quad (6)$$

The accuracies obtained changing the two tresholds are comparable but the $Precision$ is considerably higher when the treshold is set to 0.975 as can be seen from Table 2.

LFW evaluation On the Labelled face in the wild dataset we have reached a

test accuracy = 0.78 with *threshold*=0.5

This performance can be justified by the fact that the faces in the dataset are not gathered up to be used as training data for authentication purpose in fact most of the images are depicting a face in an appropriate pose (ı.e the person is not looking at the camera). A better accuracy and precision can be reached by raising the treshold to 0.975 as shown on Table 3. As we see, the accuracy is not really high, but the precision is: only

4.3. Usage example

We used our trained model to verify the live test performance in a simple application. First, the app acquires a

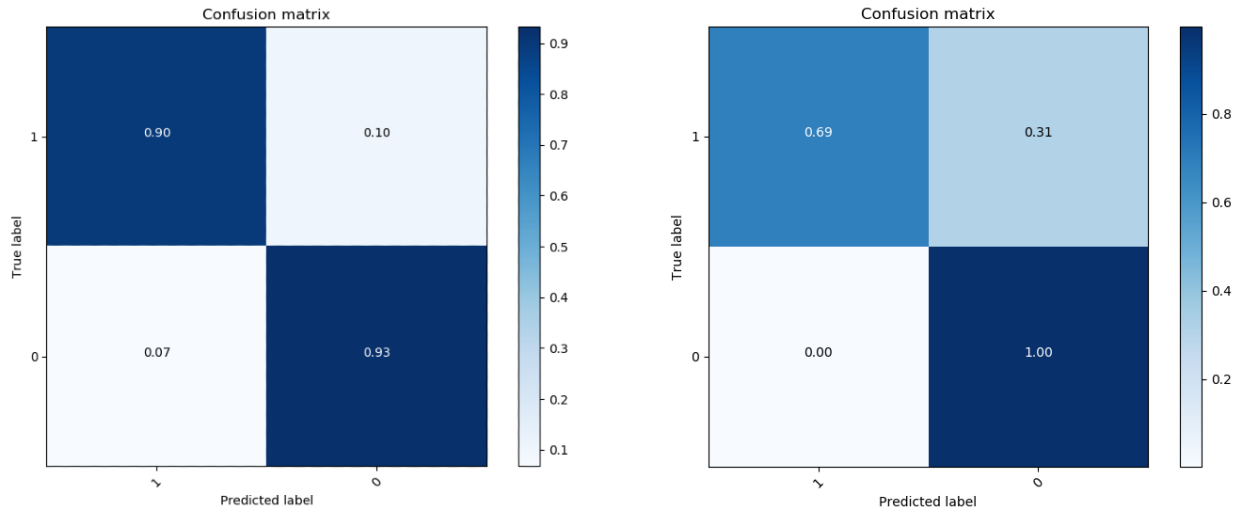

Figure 6: comparison between the confusion matrix with two different tresholds 0.5(on the left) and 0.975

Treshold	Accuracy	Precision
0.5	0.745	0.7474
0.975	0.775	0.9824

Table 3: Perfomance comparison between the case of threshold 0.5 and 0.975 on the LFW dataset.

sample image of the user, then it starts a live session. During that, it captures frames from the camera. For each of them, a frame-sample image couple is created and is given as input to the model, that outputs the prediction in a real-time manner. From our tests using general purpose laptops, the maximum number of predictions that can me made is around 8 predictions/sec.

5. Conclusion

We provide a model useful to guarantee authentication in general purpose applications, with a good tradeoff between implementation cost and performance that can be used in real-time scenarios, as shown by our experiment. The advantage of our method is that it can be used as a blackbox: given two images, it will output a probability that the faces depicted belong to the same person. Future work can be done in training the same selected network for more epochs and with more identities than the ones that we manage to gather. Moreover, a lower number of false positives can be achieved setting up an ensemble of models and then acting in a majority voting way.

Acknowledgements

We would like to thank all the people that contributed with their own photos to our project, which made our work concretely possible.

References

[1] Florian Schroff, Dmitry Kalenichenko and James Philbin *FaceNet: A unified embedding for face recognition and clustering*. 1, 2

[2] Yoshua Bengio *Practical Recommendations for Gradient-Based Training of Deep Architectures* 3

[3] *http://keyblade95.altervista.org/faceupload/* 3

[4] *https://github.com/mdbloice/Augmentor* 3

[5] *https://stackoverflow.com/questions/19363293/whats-the-fastest-way-to-increase-color-image-contrast-with-opencv-in-python-c* 3

[6] *https://towardsdatascience.com/image-augmentation-for-deep-learning-using-keras-and-histogram-equalization-9329f6ae5085* 4

[7] *https://www.learnopencv.com/histogram-of-oriented-gradients/* 4

[8] *https://www.pyimagesearch.com/2017/05/22/face-alignment-with-opencv-and-python/* 4

[9] *https://github.com/aleju/face-comparer* 2

[10] *Y. Taigman, M. Yang, M. Ranzato, and L. Wolf. Deepface: Closing the gap to human-level performance in face verification.* 2

[11] *Z. Zhu, P. Luo, X. Wang, and X. Tang. Recover canonical view faces in the wild with deep neural networks.* 2

[12] *https://github.com/keyblade95/DeepLearningProject/*