

# Face Recognition Project Report

## I. INTRODUCTION

Face Recognition is a very important, well-studied and a successful application of computer vision. Its importance stems from its wide applicability in many commercial and legal contexts; essentially everywhere the identification of individuals may be of interest. However, the task of automatic face recognition can be hindered by many factors limiting accuracy, which can be categorized as:

**1) Inter-face Factors:** Images of the same face can exhibit quite different features over time.

**a) Expression.** The momentary facial expression like anger or smile of an individual has the potential to deviate from the known 'signature' by the system

**b) Style.** The individual may have had a haircut, grown beards, put on a make-up, or wear accessories like glasses.

**c) Disguise.** The individual may be trying to hide his/her identity e.g. by making his/her hat cover most of his face

**2) Intra-face Factors:** Different faces could easily be confused.

**a) Lookalikes.** Many faces are quite similar to each other, like relatives or by chance.

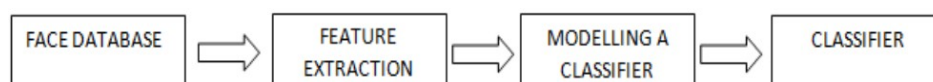
**b) Underrepresented sub-classes.** Especially a sub-class of people whose facial features resemble each other but are different from the majority of the people the designers of a system had anticipated create problems; such as the inability to distinguish among Chinese people of a system trained mainly on Europeans.

**3) Physical Factors:** Conditions surrounding both the face & the imaging system affect the image.

**a) Illumination.** The variability of the intensity and the direction of light create many interesting variations; like indoor/outdoor differences or a part of the face being shadowed.

**b) Pose.** The projection of the 3-D world to an image means that the 2-D representation of the same face is mathematically quite different from different view angles. Some face recognition systems in everyday use cannot even differentiate between a real face from a photograph of it.

The workflow for training an automatic face recognition system looks like the following:



We could broadly categorize different approaches related to feature extraction and modeling into three:

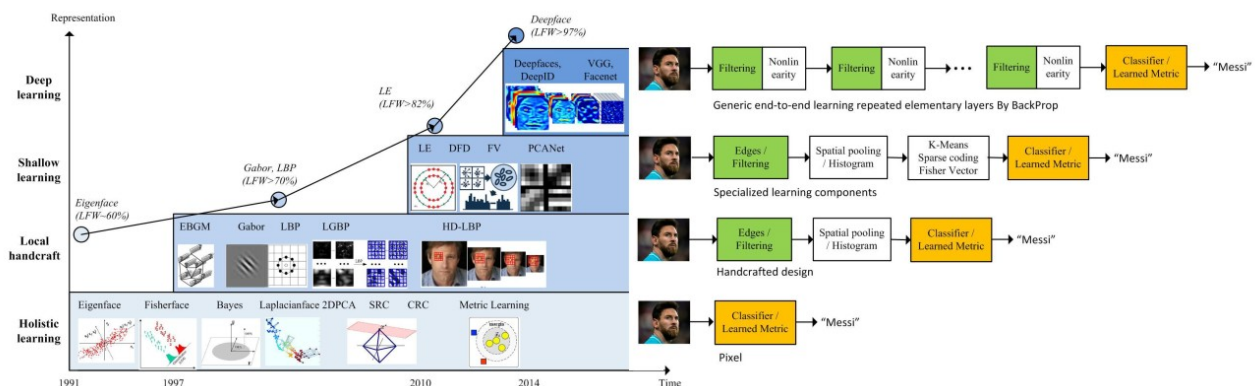
### 1) Classical Algorithm-based Methods

For example, Eigen Faces uses Principal Component Analysis, whereas Fisher Faces uses Linear Discriminant Analysis, Gabor filters bandpass filters to find the most distinguishing features. Shown below is a 2-dimensional Gabor representation of feature vector components of a face [1].



## 2) Classical Machine Learning Methods:

Approaches like Support Vector Machines and Decision Trees are very fast to train but require careful feature engineering to represent the underlying images. The latest trend is to increase automation by moving away from handcrafted design and specialized learning components to generic end-to-end repeated elementary layers using neural networks [2].

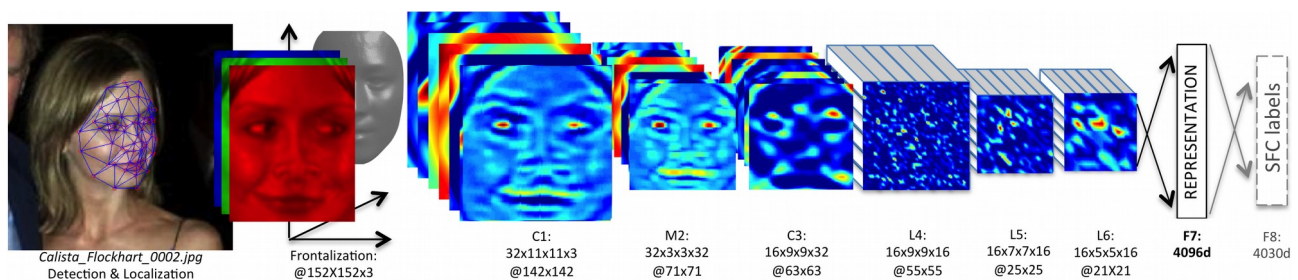


## 3) Deep Learning Methods:

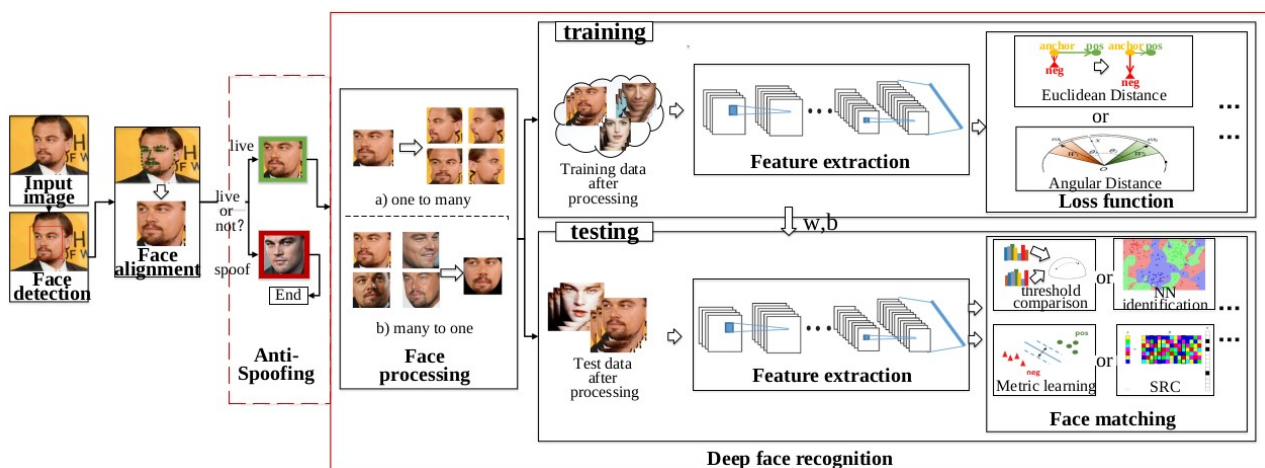
Ideally, a deep learning system is a fully trainable system beginning from raw image pixels, to the final output of recognized faces. However, sometimes deep neural networks are also fed with processed and summarized inputs like DCT coefficients to reduce the number of parameters needed for accurate classification, which also reduces the training time required. Most recent successful image recognition applications use many layers of Convolutional Neural Networks (CNNs) connected in creative ways.

In this project we have examined the use of deep learning in automatic face recognition tasks, and implemented as working code the key ideas taken from the most famous of them, namely DeepFace [3] and FaceNet [4]. We further note that the input to real-life automatic face recognition system is not a 2-D image but a real person showing up, which make the approaches mentioned above insufficient in the sense that the system must be able to distinguish a real person from his/her picture to be useful in most cases. Finally, the topic of reliability led us to investigate spoofing techniques, one of which is generating realistic artificial face images to specifications using Variational Auto-Encoders (VAEs) and Generative Adversarial Networks (GANs) [5]. We plan to work on a system distinguishing impostors created as a result of GANs' capability to generate deepfakes from real, authentic images as a follow-up [6].

## II. FACE RECOGNITION USING DEEP LEARNING



A CNN has the capacity to learn stylistic details like texture and color distribution in early layers, and more major representations of the input image in later layers. The major representation in the final layer can be considered as a compact encoding of the input image. This idea has successfully been employed in face recognition systems. A map of the whole process is shown below [3].

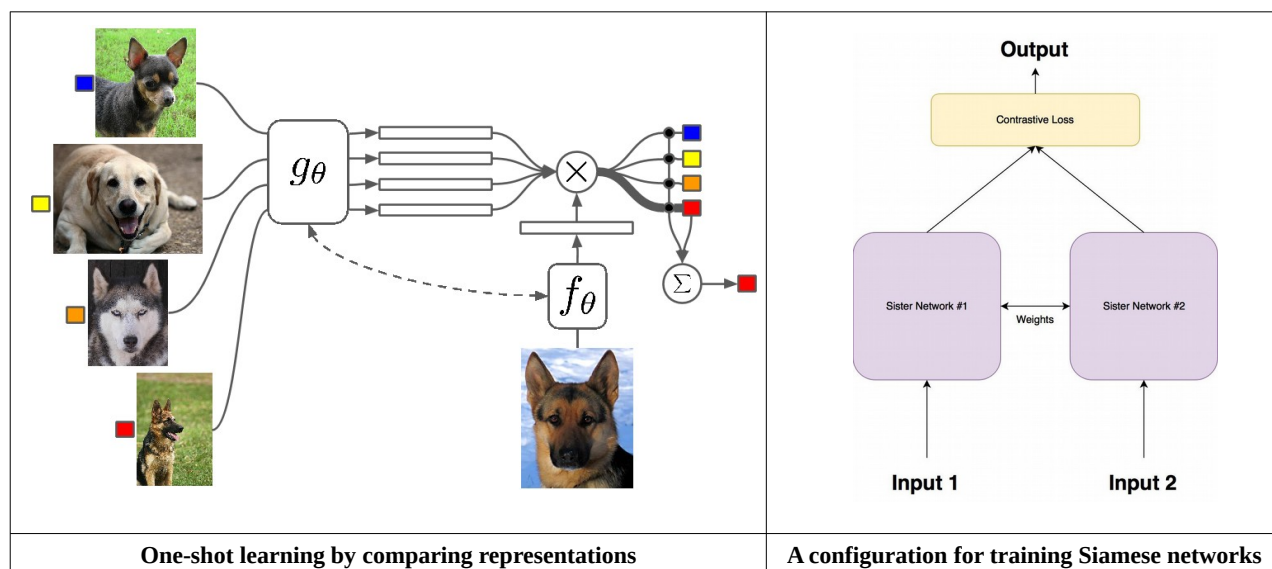


The detection of the window containing a face in the image is critical for considering only the identifying features of the person in question, and not the background noise. The next issue for a real-world system is to detect whether it is looking at a real person or somebody is showing the photo of another person to it. There are anti-spoofing measures like liveness & depth detection. A photograph neither has depth, nor does it move, nor radiate infrared energy like humans do.

The system needs to be trained with images labeled with the identity of the people to be recognized. It needs to learn features such that similar faces should yield vectors close to each other in some N-dimensional space. As indicated above, a CNN performing a classification is stripped of its softmax output layer, and the output of the last fully connected layer can be used as a unique feature vector, called encoding, for each image. The distance metric could then be Euclidean, or cosine, or any weighted average of differences in the feature dimensions.

Once trained, the system can compute the encoding for any image and compare a similarity measure between any two images. Ideally, the images of the same person should all have close encodings,

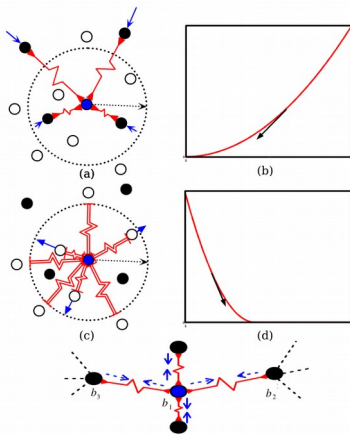
and images of different people should yield far-away encodings; usually in 128 dimensional space. Hence the key idea is to be able to learn and then calculate a single, unique, compact representation of each image where similarly looking entities have almost the same representation in the reduced space. Once we have this compact representation for any image, we can tell that two pictures are of the same entity by the closeness of their representations via the distance metric defined. This process is called one-shot learning [7]. The architecture employed is called a Siamese network, taking its name from biology to describe the shared parameters between the two networks.



Each of the sister Siamese networks is a CNN with exactly the same weights so that the encoding process for an image is the same in both. In practice the images need not be input in parallel, but the same neural network may be used twice to get the representations one after another. Therefore only one network is needed for the logical configuration above.

A Siamese network needs a loss function which would be high if its parameters were to generate close encodings for images of dissimilar entities, or distant encodings for images of the same entity. Through backpropagation, the loss would pull the same entity representations together in encoding space, and push dissimilar ones apart, as in the analogy below on the left.

One loss function that can achieve this is called contrastive loss, and works like this: If the Siamese network has received two images of the same face, then label  $Y=0$  and the contrastive loss is only the distance squared. Reducing the loss through backprop of  $\mathbf{W}$  would reduce the distance  $\mathbf{D}_w$ , which is what we need. For different faces, label  $Y=1$  and the distance term vanishes. The max operation is at least zero, which can be minimized only if  $\mathbf{D}_w > \text{margin } m$ . So the loss is trying to push the encodings of different entities to a distance of margin  $m$  at least.

	$L(W, Y, \vec{X}_1, \vec{X}_2) = (1 - Y) \frac{1}{2} (D_W)^2 + (Y) \frac{1}{2} \{ \max(0, m - D_W) \}^2$ <p>Where  <b>W</b>: weights of the network  <b>X<sub>1</sub>, X<sub>2</sub></b>: two input images  <b>D<sub>w</sub></b>: distance between the two images under network params <b>W</b>  <b>m</b>: margin, or the radius of sphere within which to push away dissimilar input images  <b>Y</b>: 0 if <b>X<sub>1</sub></b> and <b>X<sub>2</sub></b> show the same face; 1 if different faces.</p>
Spring analogy for loss function	Contrastive loss function definition

There is another widely used alternative to contrastive loss, namely triplet loss, which may be more cost effective computationally. Its name stems from the process of selecting the input in triplets instead of pairs. The triplet is 3-tuples of (an Anchor A, Positive P, Negative N) where Positive is another image of the same person as the Anchor, and Negative is a different person than the Anchor. Triplet loss is simply the difference between the distance(A, P) and distance(A, N) plus a margin clamped at minimum of zero. This would try to adjust network parameters **W** in a way to reduce the distance between A<sub>i</sub>'s and P<sub>i</sub>'s while increasing the distance between A<sub>i</sub>'s and N<sub>i</sub>'s; which is exactly what we seek. The margin m prevents the equality of the (A<sub>i</sub>, P<sub>i</sub>) and (A<sub>i</sub>, N<sub>i</sub>) distances. Hence, **Triplet Loss** = max(  $\sum_i \text{distance}(A_i, P_i) - \text{distance}(A_i, N_i) + m, 0$  ) [4]

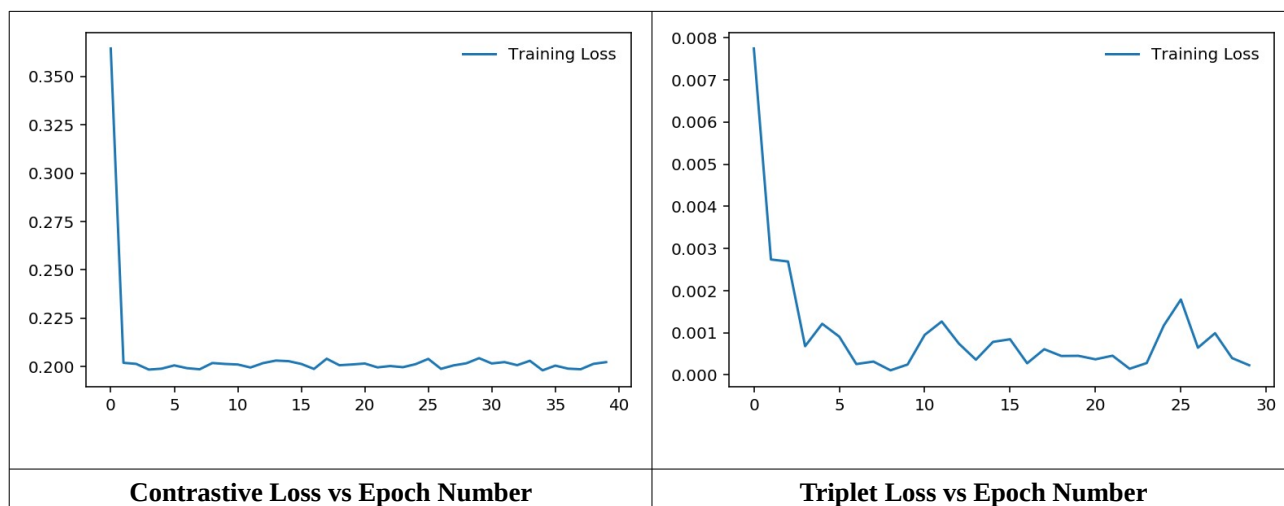
### III. FACE RECOGNITION EXPERIMENTS AND RESULTS

For the automatic face recognition task, we used the Faces94 dataset from University of Essex and selected 17 females each having 20 photos. The photos are frontal face pictures, showing slight variation in details like the closing of eyes or smiling. Generating random pairs from the dataset for the contrastive loss function, and triplets for the triplet loss function were both implemented in PyTorch, along with the functions themselves.

As a design decision, with the aim of increasing both the accuracy and the speed of training of our model, we inserted a ResNet CNN trained for the ImageNet classification task inside the Siamese Network. We did not freeze the already trained ResNet parameters and continued their training for the face recognition use case. However, a deep CNN seem to overfit our small dataset so we selected an easily trainable, small ResNet with only 18 layers. Furthermore, we encode face images as vectors of dimension 64. Smaller number of dimensions tended to reduce representational power of the encoders; whereas with higher dimensions the training became more difficult in the sense that same-class (positive) samples could not come close together with this amount of data in high-dimensional spaces.

Since triplet loss is more computationally involved, we run that model for 30 epochs whereas we run the model with contrastive loss for 40 epochs. In both cases, the training loss stabilizes within 5 epochs, as can be seen from the graphs below.





3 of the 17 people were held out from the training in order to be used in the validation of our model. The Euclidean distance between each image pair shows the degree of similarity calculated by our models. Ideally, two pictures of the same person have very low dissimilarity (distance), whereas images of two different people have high dissimilarity scores.


We note that for the contrastive model, setting a threshold slightly above 0.20 would enable us to distinguish images of the same face from images of other faces:

#### Distances (dissimilarities) calculated by the Contrastive Loss Function



The threshold is higher for the model with triplet loss, around 1.0:

### Distances (dissimilarities) calculated by the Triplet Loss Function

 Dissimilarity: 0.85	 Dissimilarity: 2.29	 Dissimilarity: 2.32	 Dissimilarity: 0.68	 Dissimilarity: 2.05
 Dissimilarity: 2.02	 Dissimilarity: 0.94	 Dissimilarity: 2.25	 Dissimilarity: 2.11	 Dissimilarity: 0.76
 Dissimilarity: 2.18	 Dissimilarity: 2.23	 Dissimilarity: 0.83	 Dissimilarity: 3.26	 Dissimilarity: 3.45
 Dissimilarity: 0.82	 Dissimilarity: 2.23	 Dissimilarity: 2.47	 Dissimilarity: 0.00	 Dissimilarity: 2.05
 Dissimilarity: 2.05	 Dissimilarity: 0.82	 Dissimilarity: 2.15	 Dissimilarity: 2.22	

Although the results are quite promising even on a tiny dataset, the accuracy of face recognition systems are already very high for professional systems. For example, the openly available Dlib library reports an accuracy of 99.38% on the standard Labeled Faces in the Wild benchmark [10]. The next logical improvement would be related to defense against spoofing by detecting deepfakes, liveliness, or depth.

## REFERENCES

- [1] Barbu, “Gabor Filter-Based Face Recognition Technique”, Proceedings Of The Romanian Academy, Series A, Volume 11, Number 3/2010, pp. 277–283.  
<https://acad.ro/sectii2002/proceedings/doc2010-3/12-Barbu.pdf>
- [2] Wang, Deng, “Deep Face Recognition: A Survey”, 2018. <https://arxiv.org/abs/1804.06655>
- [3] Taigman, Yang , Ranzato, “DeepFace: Closing the Gap to Human-Level Performance in Face Verification”, 2014. [https://www.cs.toronto.edu/~ranzato/publications/taigman\\_cvpr14.pdf](https://www.cs.toronto.edu/~ranzato/publications/taigman_cvpr14.pdf)
- [4] Schroff, Kalenichenko, Philbin, “FaceNet: A Unified Embedding for Face Recognition and Clustering”, 2015. <https://arxiv.org/abs/1503.03832>
- [5] Liu, Breuel, Kautz, “Unsupervised Image-to-Image Translation Networks”, 2017.  
<https://arxiv.org/abs/1703.00848>
- [6] Karras, Laine, Aila, “A Style-Based Generator Architecture for Generative Adversarial Networks”, 2018. <https://arxiv.org/abs/1812.04948>
- [7] Koch, Richard Zemel, Salakhutdinov, “Siamese Neural Networks for One-shot Image Recognition”, ICML 2015. <https://www.cs.cmu.edu/~rsalakhu/papers/oneshot1.pdf>
- [8] Raia Hadsell, Sumit Chopra, Yann LeCun, “Dimensionality Reduction by Learning an Invariant Mapping”, 2006. <http://yann.lecun.com/exdb/publis/pdf/hadsell-chopra-lecun-06.pdf>
- [9] Faces94 Database, University of Essex. <https://cswww.essex.ac.uk/mv/allfaces/index.html>
- [10] Dlib C++ Library Blog. <http://blog.dlib.net/2017/02/high-quality-face-recognition-with-deep.html>