Face Recognition using Convolutional Neural Networks

EED498
(Spring 2021)

Submitted by

Vaibhav Sachdeva (1710110369)

Under supervision of

Prof. Madan Gopal

Distinguished Professor

Department of Electrical Engineering

Department of Electrical Engineering
School of Engineering

SHIV NADAR UNIVERSITY

Candidate Declaration

I hereby declare that the thesis entitled "Face Recogntion using Convolutional Neural Networks" submitted for the B Tech Degree program. This thesis has been written in my own words. I have adequately cited and referenced the original sources.

Vaibhav Sachdeva

Vaibhav Sachdeva

1710110369

Date: 30th April 2021

CERTIFICATE

It is certified that the work contained in the project report titled “Face Recognition using Convolutional Neural Networks,” by “Vaibhav Sachdeva,” has been carried out under my/our supervision and that this work has not been submitted elsewhere for a degree.

Dr. Madan Gopal
Electrical Engineering Department
School of Engineering
Shiv Nadar University
April, 2021

ABSTRACT

The emergence of high-resolution cameras and high-speed processors has spearheaded the research towards design of face recognition systems for numerous real-world applications, such as human machine interaction, access control, video surveillance, and security systems. Face Recognition is a common biometric authentication technique involving extraction of features from an underlying face, and feeding the extracted features to a classifier, to identify the corresponding individual. Today, Deep Learning methods for image recognition and classification are widely employed, and they ensure promising results. Convolutional Neural Networks (ConvNets) have the capability to automatically learn features at multiple abstraction levels, through back-propagation, with convolution, pooling, and fully connected layers. In this work, designing, and evaluating a Face Recognition model using ConvNets, will be the prime objective. Two different methodologies – model training through Traditional Convolutional Neural Networks, and model training through a Siamese Network will be discussed and compared. The training and evaluation of the model will be carried out on a suitable size (computational constraints) open-source face image dataset. Details about the tuning of CNN parameters to assess and enhance the proposed system's recognition accuracy will also be reported.

Keywords: Deep Learning, Face Recognition, Convolutional Neural Networks, Siamese Network

Table of Contents

Title **Page No.**

List of figures ..... **vi**

List of tables ..... **vii**

1 Introduction **1**

1.1 Motivation 1

1.2 Problem Statement 2

1.3 Dataset 2

2 Literature Review **4**

2.1 Literature Survey 4

2.2 Convolutional Neural Networks 5

2.2.1 Convolution Layer (Kernel) 6

2.2.2 Pooling Layer 9

2.2.3 Fully Connected Layer (Flatten) 10

2.2.4 Fully Connected Layer (Dense) 11

2.3 Siamese Neural Network 12

2.3.1 Working 13

2.4 Metrics for Model Evaluation 17

3 Work Done **19**

3.1 Methodology 1 19

3.1.1 Dataset Curation 19

3.1.2 Numerical Feature Set 22

3.1.3 Proposed ConvNet Architecture 23

3.1.4 Model Training 25

3.1.5 Hyperparameters 25

3.2 Methodology 2 27

3.2.1 Dataset Curation 27

3.2.2 Proposed Siamese Network Architecture 28

3.2.3 Face Verification 29

3.2.4 Face recognition 30

4 Results and Conclusion 32

4.1 Results for Methodology 1 33

4.2 Results for Methodology 2 34

4.3 Conclusion 36

5 Appendix 37

5.1 Source Code 37

Bibliography 38

List of Figures

1. Convolutional Neural Network Framework........6
2. Convolution Operation........6
3. Movement of kernel over an image........7
4. Convolution over a sample colored image (3 channels)........7
5. Types of Padding........9
6. Max and Average Pooling........10
7. Flattening of a 2D matrix........10
8. Softmax Activation Function........11
9. The Siamese Neural Network........13
10. LFW images as input to the Siamese Neural Network........13
11. Anchor, Positive and Negative image (a triplet) from the LFW dataset........15
12. Images from the LFW dataset........20
13. Distribution of images for 42 classes........21
14. RGB colorspace demonstration........22
15. Proposed CNN Architecture........24
16. Triplets formed after processing the LFW dataset........27
17. Face alignment on the LFW images........28
18. Distances between pair of images........29
19. Distance threshold v/s F1 score........30
20. Output of the classifier/face recognition model........31
21. F1 score of Methodology 1 for all 42 classes........33
22. F1 scores of different classes using the KNN classifier (methodology 2)........34
23. F1 scores of different classes using the SVM classifier (methodology 2)........35

List of Tables

1. Table 1 21
2. Table 2 33
3. Table 3 34
4. Table 4 34

Chapter 1
Introduction

To identify an individual, the face is the focus of primary attention. The human faces represent complex, multidimensional, and meaningful visual stimulants. The human capacity to identify faces is extremely remarkable, and to infer intelligence or character from facial appearance is astounding. The ability of individuals to determine the emotional state, the subject of attention, and the intent of others are essential skills for effective social interactions, in addition to the simple ability to identify. For all these purposes, face recognition has become a field of intense focus for the vision community, and has gained a large amount of attention in the past few years. In addition, Deep Learning-based approaches have achieved major performance improvements on numerous computer vision tasks, including face recognition, with the advancement of Deep Convolutional Neural Networks.

1.1 Motivation

Facial recognition is one of the most widespread and commonly used image processing technologies that has gained considerable importance in recent years due to the extensive use of biometric systems. Instead of using passwords, keys, PINs, cards etc., biometric-based strategies have arisen to replace human identification, as it is hard to remember them, they can be stolen, one can forget them, and they can be easily lost. Biological characteristics have numerous advantages compared to traditional identity recognition technologies, such as -

1. Reproducibility - biological characteristics are born with, cannot be changed, so biological characteristics of other people cannot be copied.
2. Availability - biological features as part of the human body, and will never be forgotten.
3. They are easy to use.

Biometric technology is easy to collaborate with computers and networks to realize automation management with the rapid growth of technology and artificial intelligence, and is quickly incorporated into the everyday life of people. Biometrics has drawn the attention of major companies and research institutes on the basis of the above advantages, and has effectively replaced conventional recognition technologies in almost every area. When we compare the variations between different biometrics, we can see that the cost of facial recognition is low, user acceptance is quick, and knowledge acquisition is fast. Facial recognition is the use of computer vision technology and related algorithms, to find faces from the pictures or videos, and then recognize the identity.

1.2 Problem Statement

Face recognition is a technique used for verification or identification of a person's identity by analyzing and relating patterns based on the person's facial features. A face recognition device/system with the input of an arbitrary image can check for the identity of different people present in the input image. In general, a facial recognition system consists of four broad modules-Identification, Alignment, Extraction of Features, and Matching, where localization and normalization are used. We can recognize thousands of faces learned throughout our lifetime and even recognize familiar faces at a glance even after years of separation. But this skill is not useful if there are changes in a person's faces, due to viewing conditions, expression, aging, and distractions such as glasses, beards or changes in hair style. With this project, through comprehensive analysis and interpretation of facial image samples using state of the art deep learning methods, robust face recognition algorithms will be explored. The prime objective is to develop a robust pipeline/model for facial image feature extraction using deep learning techniques for accurate user recognition.

1.3 Dataset

For the purpose of this project, we consider the LFW (Labelled Faces in the Wild) dataset, a database of face photographs designed for studying the problem of unconstrained face recognition. The data set contains more than 13,000 images of faces collected from the web. Each face has been labelled with the name of the person pictured. 1680 of the people pictured have two or more distinct photos in the data set. The only constraint on these faces is that they were detected by the Viola-Jones face detector, an outdated but powerful face detector.

There are four different sets of LFW images including the original and three different types of "aligned" images. The aligned images include "funnelled images", LFW-A (which uses an unpublished method of alignment), and "deep funnelled" images. Among these, LFW-A and the deep funnelled images produce superior results for most face verification algorithms over the original images and over the funnelled images. [1]

Chapter 2

Literature Review

2.1 Literature Survey

Face recognition was introduced in Chapter 1, addressing the use case and bright future of this technology. A huge amount of study and effort has been devoted to this field by several large universities and corporations. This report reviews research progress on the Labelled Faces in the Wild (LFW) dataset, which was developed to promote research for images taken in normal everyday settings (unconstrained face recognition). There have been numerous studies performed on Face Recognition using Deep Learning on the LFW database, some of which are -

Sun, Yi, et al. [2] proposed two deep learning architectures, working together as an ensemble, called the 'DeepID3', which was able to achieve an overall accuracy of 99.53% on the LFW dataset. DeepFace [3] is another deep learning network architecture consisting of a 9-layer deep CNN model with over 120M trainable parameters. It was trained on about 4 M face pictures captured for 4000 people, achieving an overall testing accuracy of 97.35% on the LFW dataset. A Siamese [4] Deep-Face network was also experimented in [3], where the L1-distance between two facial features was optimized. DeepID [5] proposed by authors Wang, et al. is yet another promising deep learning technique based on CNNs, which initially learns weights by the identification of faces, then performs feature extraction using the outputs of final hidden layer, and then generalizes this method for the facial verification task. This model was trained using the CelebFaces dataset, a very large database used for face recognition tasks, and was tested on the LFW dataset, achieving an accuracy 97.45%. FaceNet [6] is an additional CNN based deep learning architecture, proposed by Schroff, et al., which directly learns a mapping from face images to a compact Euclidean space where distances directly correspond to a measure of face similarity. The model was trained using a private database consisting of 200M face images of 8M unique individuals, and gives an accuracy of 99.63% when evaluated/tested on the LFW dataset.

2.2 Convolutional Neural Networks

Artificial Intelligence has witnessed comprehensive results in bridging the gap between human and computer abilities. To make great things possible, researchers and enthusiasts alike work on various facets of the area. The domain of Computer Vision is one of several such fields. The goal for this field is to allow machines to view the world as humans do, interpret it in a similar way and even use information for a variety of tasks, such as recognition of images and videos, image analysis and classification, reconstruction of media, recommendation systems, natural language processing, etc. With time, advances in Computer Vision with Deep Learning have been developed and refined, predominantly by one unique algorithm, a Convolutional Neural Network (ConvNet). A Deep Learning algorithm, Convolution Neural Network can take an input image, allocate significance (learnable weights and biases) to various aspects of the image and be able to differentiate one from the other. As compared to other classification algorithms, the pre-processing required in a ConvNet is much lower. Although filters are hand-engineered in primitive ways, but with enough experience, ConvNets have the ability to learn these features. The ConvNet architecture is comparable to that of the neuron connectivity pattern in the human brain and was inspired by the Visual Cortex organization. Individual neurons respond only in a small area of the visual field, known as the Receptive Field, to stimuli. To cover the entire visual region, a range of such fields overlap. Apart from powering vision in robots and self-driving vehicles, Convolutional Neural Networks have proven extremely efficient in recognizing faces and objects. From large datasets, they can easily derive high-level features or characteristics and can also become invariant to brightness, lighting, texture and contrast variations. ConvNets are a type of feed-forward neural networks that consist of filters or kernels or neurons that have weights or parameters and biases that can be learned. Each filter takes those inputs, performs convolution and, optionally, non-linearity follows it. A typical ConvNet framework can be seen as shown in Fig. 1. The architecture contains 3 major layers, namely Convolutional, Pooling, and Fully Connected layers.

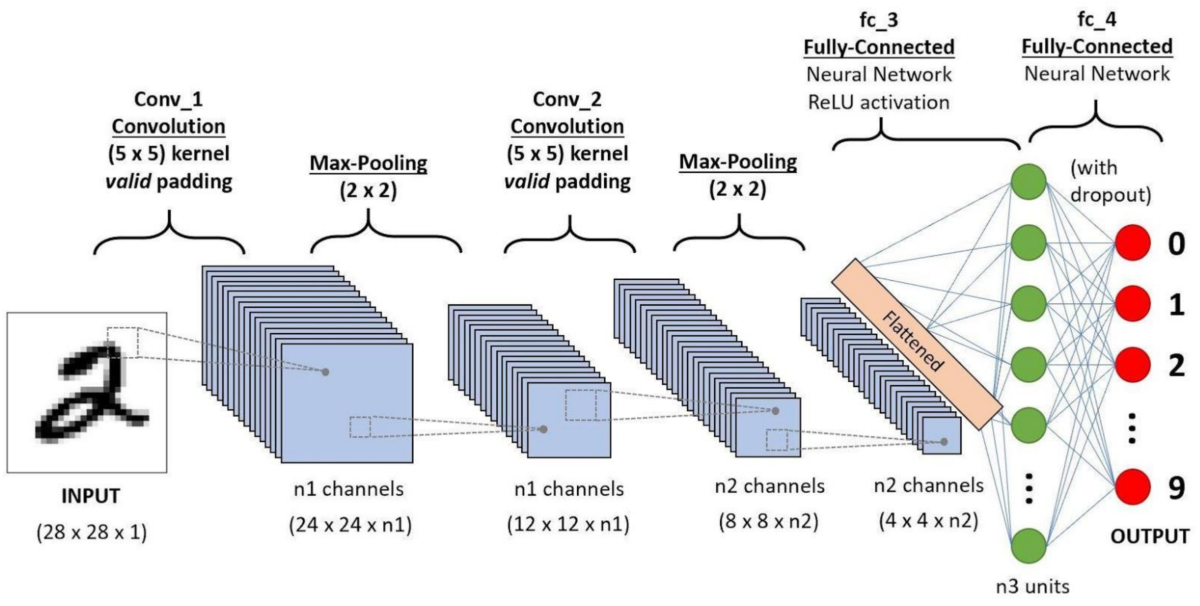


Fig. 1: Convolutional Neural Network Framework

2.2.1 Convolution Layer (Kernel)

Let us consider an image of dimension 5 x 5 x 1 (a grayscale image having 5 pixels in length and 5 pixels in width), represented by the leftmost matrix in Fig. 2, in order to demonstrate the functionality of the convolutional layer. Here, we use a 3 x 3 x 1 filter or kernel, represented by the blue matrix (next to input image), for the convolution operation.

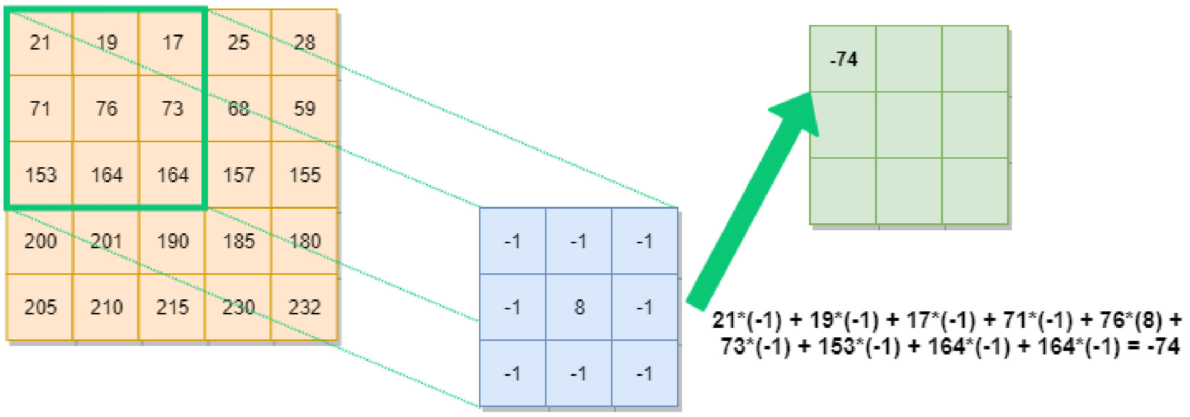


Fig. 2: Convolution Operation

The filter starts from the top-left corner of the input images and moves right with specific stride values until it traverses the entire width of the image. Post covering the width of the image, it then jumps down to the start (left edge) of the input image and repeats the entire process with the same stride value. In the above example, considering a stride of 1, the kernel shifts 9 times, each time performing a matrix multiplication operation between the kernel and the portion of the image over which the kernel is hovering. The movement of kernel over an image, having certain depth (more than 1 channel), is demonstrated in Fig. 3.

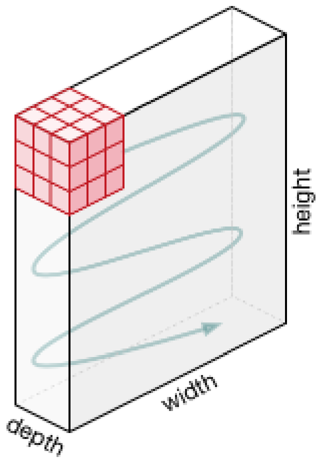
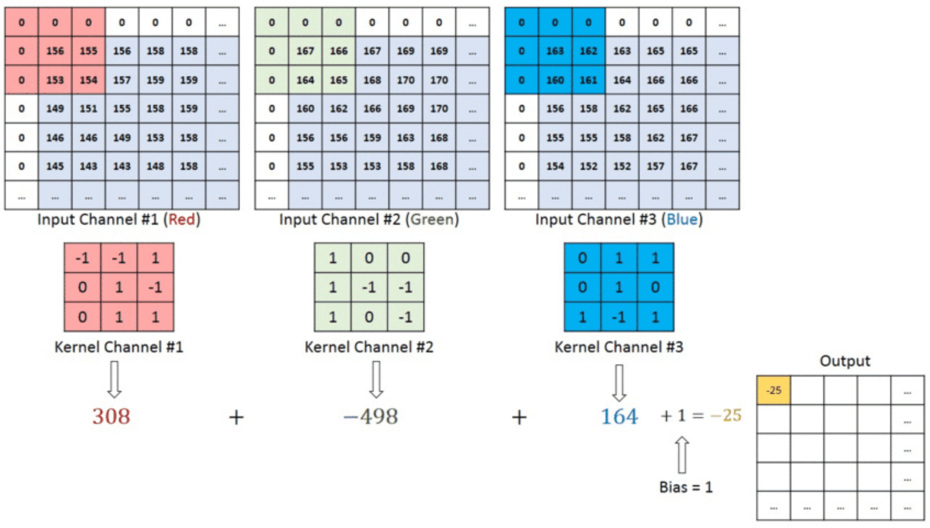


Fig. 3: Movement of kernel over an image

The convolved feature matrix is simply the product of multiplying the kernel values over the image pixel region by summing up the matrix. For colored images consisting of 3 channels, namely Red, Green, and Blue, the depth of the kernel or filter is the same as that of the image input. The operation of convolution over a sample colored image consisting of RGB channels is shown appropriately in Fig. 4. The flattened 1D channel output of the Convolved Feature Matrix is generated by the summation of all elements of the resulting matrix obtained from the kernel matrix multiplication and the input image (for all three channels).



0	0	0	0	0	0	...
0	156	155	156	158	158	...
0	153	154	157	159	159	...
0	149	151	155	158	159	...
0	146	146	149	153	158	...
0	145	143	143	148	158	...
...	...	...	...	...	...	...

Input Channel #1 (Red)

0	0	0	0	0	0	...
0	167	166	167	169	169	...
0	164	165	168	170	170	...
0	160	162	166	169	170	...
0	156	156	159	163	168	...
0	155	153	153	158	168	...
...	...	...	...	...	...	...

Input Channel #2 (Green)

0	0	0	0	0	0	...
0	163	162	163	165	165	...
0	160	161	164	166	166	...
0	156	158	162	165	166	...
0	155	155	158	162	167	...
0	154	152	152	157	167	...
...	...	...	...	...	...	...

Input Channel #3 (Blue)

-1	-1	1
0	1	-1
0	1	1

Kernel Channel #1

1	0	0
1	-1	-1
1	0	-1

Kernel Channel #2

0	1	1
0	1	0
1	-1	1

Kernel Channel #3

$308 + -498 + 164 + 1 = -25$

Bias = 1

Output

-25				...
				...
				...
				...
...	...	...	...	...

Fig. 4: Convolution over a sample colored image (3 channels)

The prime objective of the Convolution operation is to extract features from the input image. Multiple convolutional layers are stacked over one another in order to capture both high and low level features. Conventionally, the first convolution layer is responsible for capturing the low level features such as pixel intensity, pixel gradient orientation, and color. On adding more layers, the architecture adapts to the high level features such as objects and actions, giving us a network which has an exhaustive understanding of images in the dataset.
By default, each pixel in the image is not given an opportunity to be at the center of the filter, as the pixels on the edge of the input are only ever exposed to the edge of the filter. By starting the filter outside the frame of the image (adding pixels to the image both vertically and horizontally), it gives the pixels on the border of the image more of an opportunity for interacting with the filter, and in turn, an output feature map that has the same shape as the input image. The added pixel values have zero value, thus having no effect with the dot product operation when the filter is applied. The process of adding extra pixels is referred to as Padding. Following are the reasons to why padding is considered important -

1. It's easier to design networks if we preserve the height and width and don't have to worry too much about tensor dimensions when going from one layer to another because dimensions will just "work".
2. It allows us to design deeper networks. Without padding, reduction in volume size would reduce too quickly.
3. Padding actually improves performance by keeping information at the borders.

In a convolution operation, there are two types of results - one in which the convoluted feature decreases the dimensionality relative to the input, and the other in which the dimensionality is either increased or stays the same. In the case of the former, this is achieved by applying valid padding or, in the case of the latter, the same padding. Fig. 5 demonstrates the two types of padding.

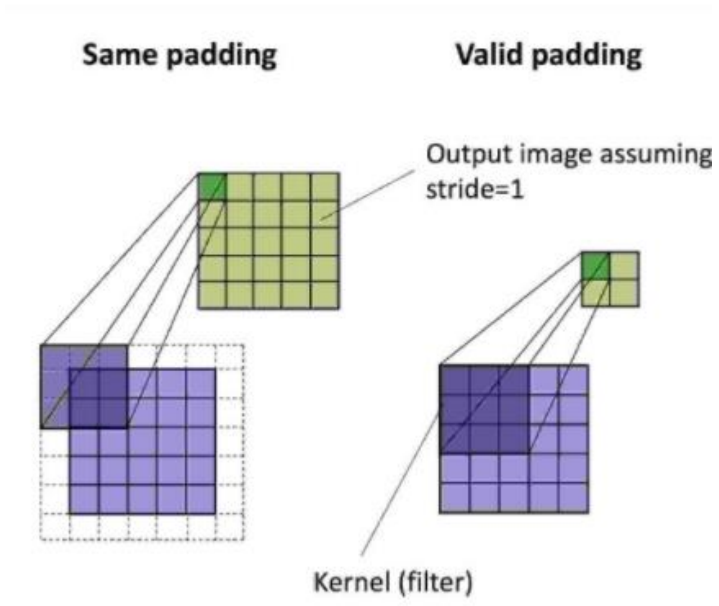


Fig. 5: Types of Padding

Valid Padding - If we augment the 5x5x1 image into a 6x6x1 image and then convolve it with the 3x3x1 kernel over it, we find that the convolved matrix turns out to be 5x5x1 in size.
Same Padding - If we perform the above convolution procedure without adding any padding, a matrix with kernel dimensions (3x3x1) itself is provided to us.

2.2.2 Pooling Layer

The pooling layer is comparable to the convolution layer, which is responsible for reducing the convoluted feature's spatial size. Reducing the spatial size of the convolution layer output (the convolved feature matrix) helps to reduce the computational power needed to process the data. In addition, pooling is also useful for extracting rotational and positional invariants, which are the dominant characteristics of the image, thus preserving the process of effectively training the model.
Two types of pooling exist: Max Pooling and Average Pooling. Max Pooling returns the maximum value of the image portion covered by the kernel, while Average Pooling returns the average of all the values from the image portion covered by the kernel. Fig. 6 represents both Max and Average Pooling operations.

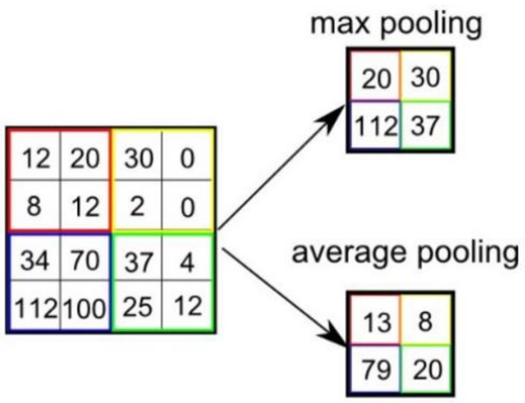


Fig. 6: Max and Average Pooling

Max Pooling operates much better than Average Pooling because it completely discards noisy activations and also performs de-noising along with the reduction of dimensionality, while Average Pooling only reduces dimensionality as a tool for suppressing noise.
The i-th layer of a Convolutional Neural Network is a combination of the Convolutional and the Pooling Layer. The number of such layers can be increased to capture low-level information much more, depending on the complexity of the images, but at the expense of more computational power. The model develops the capacity to comprehend the image features after moving through the convolutional and pooling layer. The output of the pooling layer is to be flattened for classification purposes, i.e. the output matrix is to be unrolled (converted to a single vector) and fed to a regular Neural Network containing fully connected layers.

2.2.3 Fully Connected Layer (Flatten)

At the end of the ConvNet, the Flatten Layer helps transform the output of the pooling layer into a single vector, which can then be used as an input, consisting of fully connected layers, to a traditional neural network. Fig. 7 represents the action of unrolling or flattening a 2D matrix of pixel values into a single vector.

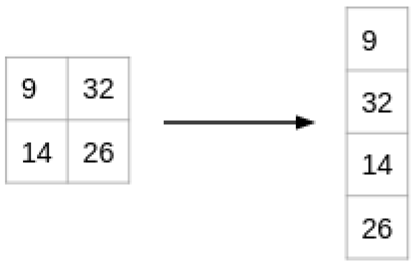


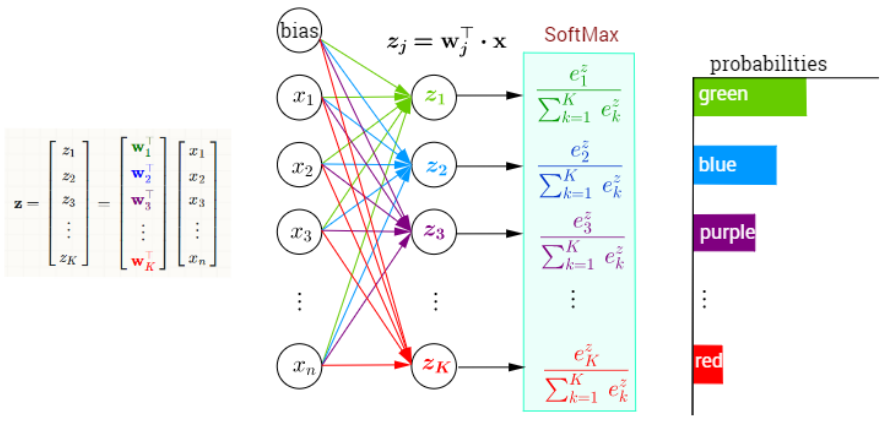
Fig. 7: Flattening of a 2D matrix

2.2.4 Fully Connected Layer (Dense)

Fully connected layers are an integral part of ConvNets, which have been proven very effective in recognizing and classifying images. The ConvNet process starts with convolution and pooling, breaking down the image into features, and independently analyzing them. The outcome of this process is fed into a neural network structure that is fully connected and drives the final decision on classification. As obtained from the flatten layer, the feature vector is passed on to the first layer of a multi-layer perceptron. The incorporation of this fully connected neuron layer allows to learn non-linear aggregations of high-level features from the output of convolutional layers. The feature vector is multiplied by weights and summed up by a bias term that is then passed through a function of nonlinear activation. This fully connected layer is succeeded with another layer of neurons, thus creating the Neural Network. In this neural network, the last layer includes neurons equal to the number of classes under which the data should be categorized. The 'softmax' function, a mathematical function that converts a number vector into a vector of probabilities where the probabilities of each value are proportional to the relative scale of each value in the vector, is the activation function used in the final layer.

$$\sigma(\vec{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

The above equation describes the activation function of softmax, which outputs the predicted probability of an input image belonging to a particular class. A probability score (probability of belonging to the class represented by the neuron) is generated by each neuron in the final layer and the final label prediction of the input image is assigned to that class of neuron that yields the highest predicted probability. Fig. 8 depicts the working of a Neural Network with softmax as the activation function in the last layer.



$\mathbf{z} = \begin{bmatrix} z_1 \\ z_2 \\ z_3 \\ \vdots \\ z_K \end{bmatrix} = \begin{bmatrix} \mathbf{w}_1^\top \\ \mathbf{w}_2^\top \\ \mathbf{w}_3^\top \\ \vdots \\ \mathbf{w}_K^\top \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \\ x_3 \\ \vdots \\ x_n \end{bmatrix}$
bias
x_1
x_2
x_3
x_n
$z_j = \mathbf{w}_j^\top \cdot \mathbf{x}$
z_1
z_2
z_3
z_K
SoftMax
$\frac{e_1^z}{\sum_{k=1}^K e_k^z}$
$\frac{e_2^z}{\sum_{k=1}^K e_k^z}$
$\frac{e_3^z}{\sum_{k=1}^K e_k^z}$
$\frac{e_K^z}{\sum_{k=1}^K e_k^z}$
probabilities
green
blue
purple
red

Fig. 8: Softmax activation function

We may therefore conclude that the facial recognition model consists of two main components: A Convolutional Neural Network that is used to extract features from the input image and a Neural Network that is used to classify the extracted features.

2.3 Siamese Neural Network

In the modern Deep Learning era, Neural networks are almost good at every task, but they rely on large datasets to perform well. But, for certain problems like face recognition, we can’t always rely on getting large datasets. Hence, in order to solve this kind of problem we have a new type of neural network architecture known as the Siamese Networks (sometimes called a twin neural network), which uses only a few number of training images to give good predictions. The ability to learn from very little data is what made Siamese Networks popular in recent years.

A Siamese Neural Network is a class of neural network architecture that consist of two identical subnetworks (two identical convolutional neural networks in case of face recognition), both joined together at the end using some energy function. The term ‘identical’ here means that the subnetworks have the same configuration i.e. the exact same parameters and weights. The Siamese Network finds the similarity/dissimilarity between input images by comparing their feature vectors, generally known as image encodings/embedding.

A Siamese network takes two different input images which are passed through two subnetworks having the same architecture, parameters, and weights. The two subnetworks are a mirror image of each other. Hence, any change to any subnetworks architecture, parameter, or weights is also applied to the other subnetwork. The two subnetworks output an encoding/embedding, which are used to calculate the difference between the two inputs. The Siamese network's objective is to classify if the two inputs are the same or different using a Similarity Score. The similarity score can be calculated using Binary cross-entropy, Contrastive function, or Triplet loss, which are techniques for the general distance metric learning approach. Siamese network is a **one-shot classifier** that uses discriminative features to generalize the unfamiliar categories from an unknown distribution.

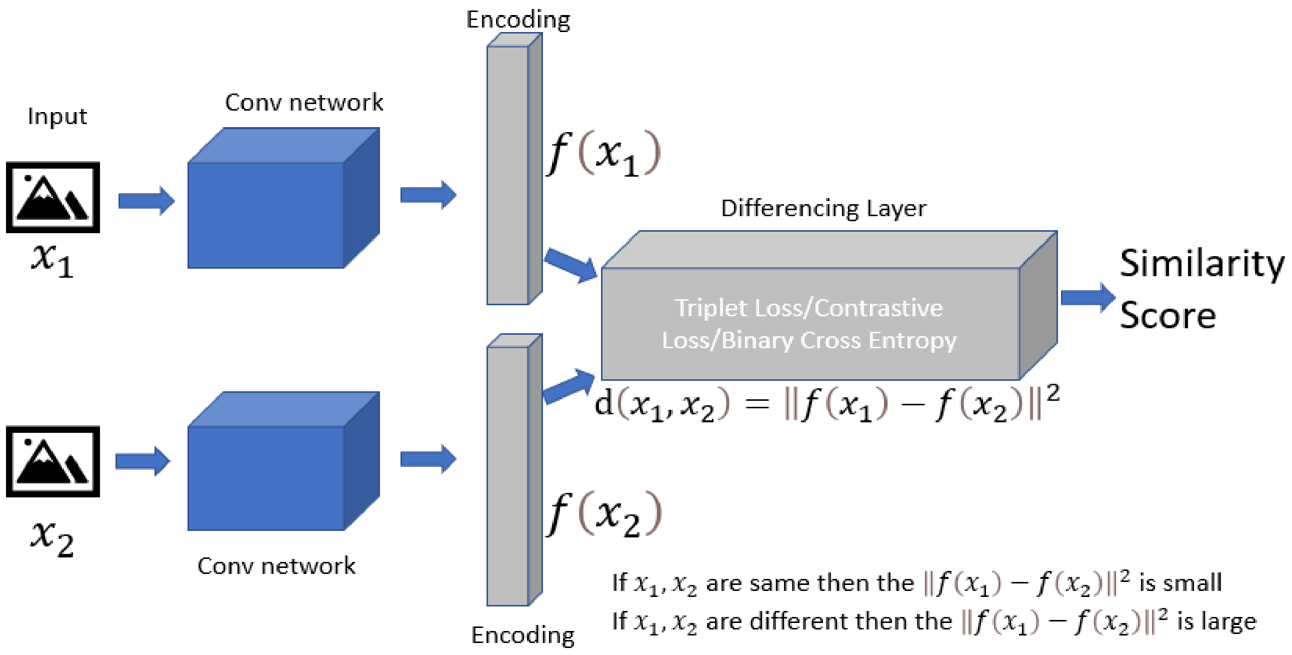

Input
x_1
Conv network
Encoding
$f(x_1)$
Differencing Layer
Triplet Loss/Contrastive Loss/Binary Cross Entropy
$d(x_1, x_2) = \|f(x_1) - f(x_2)\|^2$
Similarity Score
x_2
Conv network
$f(x_2)$
Encoding
If x_1, x_2 are same then the $\|f(x_1) - f(x_2)\|^2$ is small
If x_1, x_2 are different then the $\|f(x_1) - f(x_2)\|^2$ is large

Fig. 9: The Siamese Neural Network

2.3.1 Working

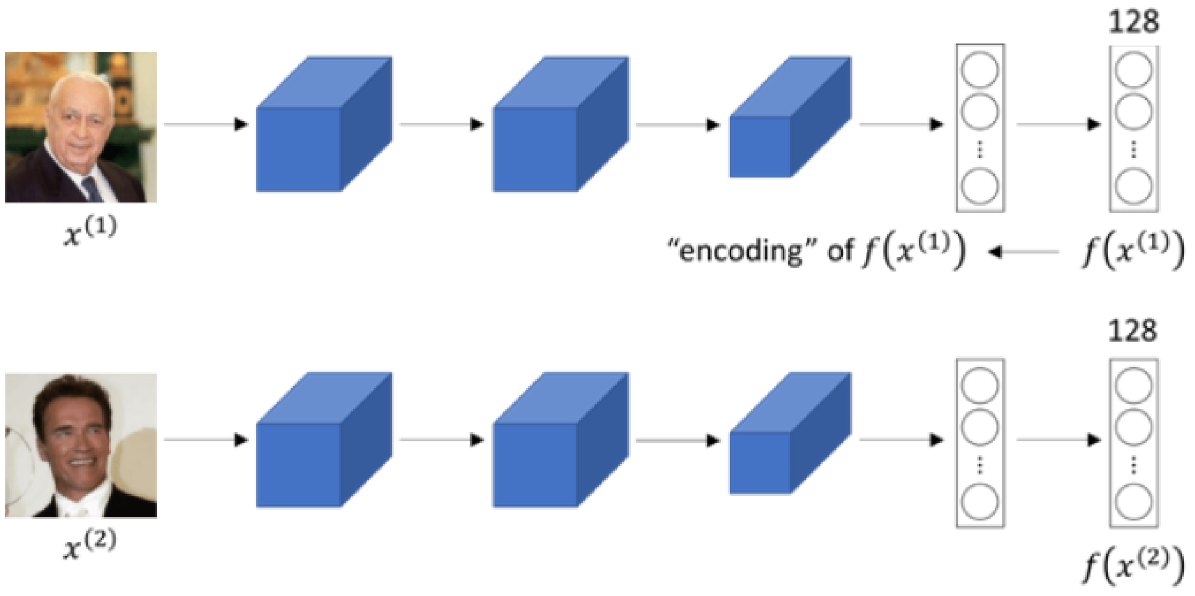


Fig. 10: LFW images as input to the Siamese Neural Network

In Fig. 10, the first subnetwork (a ConvNet) takes a face image ($x^{(1)}$) as input, and after passing the image through convolution, pooling and fully connected layers, outputs a vector representation of the input image (let us assume that we get a vector having 128 elements as the output of the ConvNet). This output feature vector is generally referred to as an encoding/embedding of the input image. Now, the second face image ($x^{(2)}$) is the one which is to be compared with the first face image, hence the second face image is passed through a network that is exactly same as the first network, i.e. it has the same parameters and weights. Now that we have two encodings/embeddings, f(x$^{(1)}$) and f(x$^{(2)}$), we compare these two to know how similar the two faces are. In particular, we want to learn a function that takes two images as input—one being the actual image and other the candidate image—and outputs the similarity between the two. The degree of similarity or difference between the two images is represented in terms of *d*, typically representing the distance between two embeddings. The two input images, if very similar, should output a lower value of *d*, and if different, should output a higher value of *d*. For this to happen, it is important to note that the embeddings of the two input images, f(x$^{(1)}$) and f(x$^{(2)}$), must be quite similar if both the input faces are same/similar, and if the faces are different, f(x$^{(1)}$) and f(x$^{(2)}$) should be very different. For making a prediction, if the degree of difference between the two images (*d*) is less than a threshold value (τ) (a hyper parameter), the prediction states that the two input images are of the same person, and if it is greater than τ, the prediction states the images are of two different people.

$$\mathrm{d(img1, img2)} = \text{degree of difference between images}$$

$$d\left(x^{(1)}, x^{(2)}\right) = \left\| f\left(x^{(1)}\right) - f\left(x^{(2)}\right) \right\|_2^2$$

$$\begin{array}{ll} \text{If } \mathrm{d\,(img1, img2)} \leq \tau & \text{“same”} \\ \qquad\qquad\qquad\quad > \tau & \text{“different”} \end{array}$$

In order to follow the above mentioned conditions, the parameters are to be learned in such a way that –

$$\text{If } x^{(i)}, x^{(j)} \text{are the same person,} \left\| f\left(x^{(i)}\right) - f\left(x^{(j)}\right) \right\|^2 \text{is small}$$
$$\text{If } x^{(i)}, x^{(j)} \text{are different persons,} \left\| f\left(x^{(i)}\right) - f\left(x^{(j)}\right) \right\|^2 \text{is large}$$

One way to learn the parameters of the Siamese network, which gives good embeddings for face images, is to define and apply **Gradient Descent on the Triplet Loss function**. In order to apply the triplet loss, we compare pairs of images, instead of individual images. In the Triplet loss terminology, we always look at three types of images (hence the name triplet loss), an anchor image (A), a positive image (P) and a negative image (N). An anchor and a positive image belong to the same person whereas a negative image represents a different individual. We want the distance between an anchor image (A) and a positive image (P) to be low, indicating that the person in the anchor and the positive image is the same. On the contrary, we want the negative image (N) when compared to the anchor image (A) to have a larger distance i.e. be much further apart.

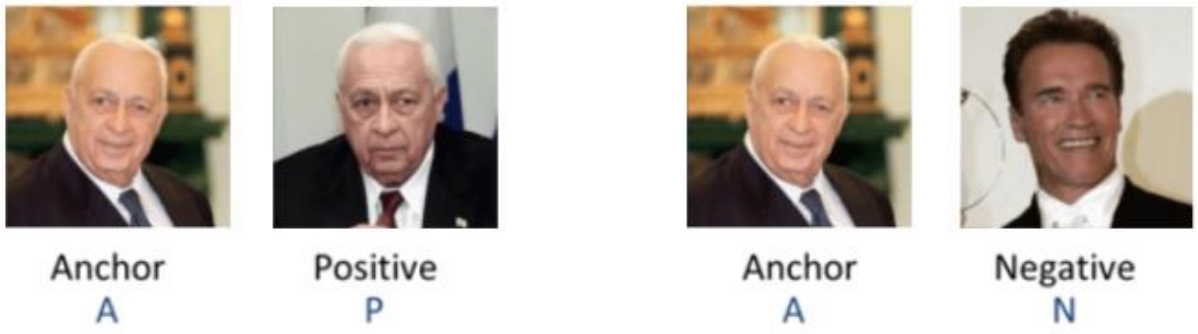


Fig. 11: Anchor, Positive and Negative image (a triplet) from the LFW dataset

To formalize the above stated points, we want our encodings (and hence the parameters of our neural network) to have the following property –

$$\|f(A) - f(P)\|^2 \leq \|f(A) - f(N)\|^2$$

$$d(A, P) \leq d(A, N)$$

After making slight changes to the above expression –

$$\|f(A) - f(P)\|^2 - \|f(A) - f(N)\|^2 \leq 0$$

One of the problem with the above equation is that, if *f* always outputs 0, the two distances become 0−0=0 and 0−0=0, and by saying *f* (image)=0, we can always satisfy the equation trivially. Thus, it is to be made sure that the Siamese network doesn't just output 0 for all the embeddings. Another way for the Siamese network to give a trivial output is if the encoding for every image is identical to the encoding of every other image, which again gives 0-0=0. In order to prevent the network from doing this, the objective function is to be modified. It should be modified in such a way that it need not be just less than or equal to 0, it should be a little lower than 0. In particular, it should be less than –α, where −α is another hyper parameter known as the margin (similar to SVM).

$$|f(A) - f(P)|^2 - |f(A) - f(N)|^2 \leq 0 - \alpha$$

The above expression (modified objective function) prevents the neural network from outputting the trivial solutions, since the difference between the distances now need to be lower than –α. By convention, we usually write plus +α on the left side of equation.

$$\|f(A) - f(P)\|^2 - \|f(A) - f(N)\|^2 + \alpha \leq 0$$

According to the above expression, d(A,N) should be much bigger than d(A,P). Hence, in order to achieve this, either d(A,P) can be pushed down or d(A,N) can be pushed up, so that there is a gap of the hyperparameter α between the two distances (between the negative and the anchor, and the anchor and the positive).

Finally, the triplet loss function can be defined as –

$$L(A, P, N) = max(\|f(A) - f(P)\|^2 - \|f(A) - f(N)\|^2, 0)$$

The reason for choosing the max function in the triplet loss function is, as long as we are able to achieve the objective of making $\|f(A)-f(P)\|^2-\|f(A)-f(N)\|^2$ equal to or less than 0, the loss on this particular sample is equal to 0. In contrast, if $\|f(A)-f(P)\|^2-\|f(A)-f(N)\|^2$ is greater than 0, then we get a positive loss, using the max function, which needs to be minimized. The expression above defines the loss function for a single triplet. But the cost function is to be defined over the entire dataset. Hence, using the loss function on a single triplet, the overall cost function for the dataset can be written as the sum of individual losses on different triplets, over the entire training set.

$$J = \sum_{i=1}^{M} L\left(A^{(i)}, P^{(i)}, N^{(i)}\right)$$

During model training, it is crucial to choose triplets whose negative pairs and positive pairs are quite hard to discriminate i.e. the difference of distances in the embedding space must be lower than the margin α. Else, the network is not able to learn useful embeddings. Hence, every training iteration should choose a new batch of triplets based on the embeddings learned in the previous iteration. In particular, we want to choose a triplet in such a way that d(A,P) is actually quite close to d(A,N), making the pairs hard to discriminate.

$$d(A,P) + \alpha \leq d(A,N)$$

$$d(A,P) \approx d(A,N)$$

In such a case, the learning algorithm has to try extra hard to push d(A,N) up. The result of choosing such triplets is that it increases the computational efficiency of the learning algorithm. If the triplets had been chosen randomly, then a lot of them would've been really easy to learn, and hence the gradient descent wouldn't have done anything because the Siamese network would've gotten them right pretty much every time. Therefore, by choosing hard triplets, the gradient descent has to work extra hard in order to push d(A,P) further away from d(A,N).

2.4 Metrics for Model Evaluation

The recognition and classification of images into separate classes of individual people can be understood as a multi-class classification problem, which, compared to a simple binary classification task, presents a greater challenge. Some of the metrics used commonly for assessing the efficiency of a multi-class classifier are below-

1. **Overall Accuracy (OA)** - The most intuitive performance measure is accuracy which is the ratio of correctly predicted observation to the total observations. One would assume that if a high accuracy is achieved, their model is good. Yes, accuracy is a great metric, but only when no class imbalance, i.e. the number of samples belonging to different classes are almost equal.

$$OA = \frac{T_p + T_n}{T_p + T_n + F_p + F_n}$$

In the above expression of overall accuracy, *Tp* represents the True Positives which are the number of correctly labelled positive samples, *Tn* represents the True Negatives which are the number of correctly labelled negative samples, *Fp* represents the False Positives which are number of negative samples incorrectly labelled as positive, and *Fn* represents the False Negatives which are the number of positive samples incorrectly labelled as negative.

2. **Precision (P)** - Precision is defined as the ratio of positive observations that are correctly predicted to the total positive observations that are predicted.

$$P = \frac{T_p}{T_p + F_p}$$

3. **Recall (R)** - The number of true positives over the total number of true positives and false negatives is known as recall.

$$R = \frac{T_p}{T_p + F_n}$$

4. **F1 score (F1)** - The harmonic mean/weighted average of both Precision (P) and Recall (R) is the F1 score.

$$F1 = 2.\frac{P \times R}{P + R}$$

Chapter 3

Work Done

Two different methodologies have been used for training the face recognition model. One of them is training through a Traditional Convolutional Neural Network (methodology 1) and the other one is training through a Siamese Neural Network (methodology 2). Important steps involved in training the model using both the methodologies have been discussed in detail.

3.1 Methodology 1

The first methodology involves training the model by means of Traditional Convolutional Neural Network. Theoretical concepts pertaining to Convolutional Neural Networks and their success in the past has been discussed in detail in Chapter-2. The necessary steps involved in training the model using the first methodology are as follows–

3.1.1 Dataset Curation

Labelled Faces in the Wild (LFW) is a facial photography database developed to research the topic of unconstrained facial recognition. This database was developed and maintained by researchers at the University of Massachusetts, Amherst. The Viola Jones face detector detected 13,233 images of 5,749 individuals and retrieved them from the internet. In the dataset, 1,680 of the individuals pictured have two or more distinct images. There are four different sets of LFW images in the original database and three different types of "aligned" images as well. In contrast to the other image types, deep-funneled images provided superior results for most face verification algorithms, according to the researchers. Hence, the dataset used for this methodology is the deep-funneled version. Fig. 12 depicts images of random individuals chosen from the original LFW dataset.

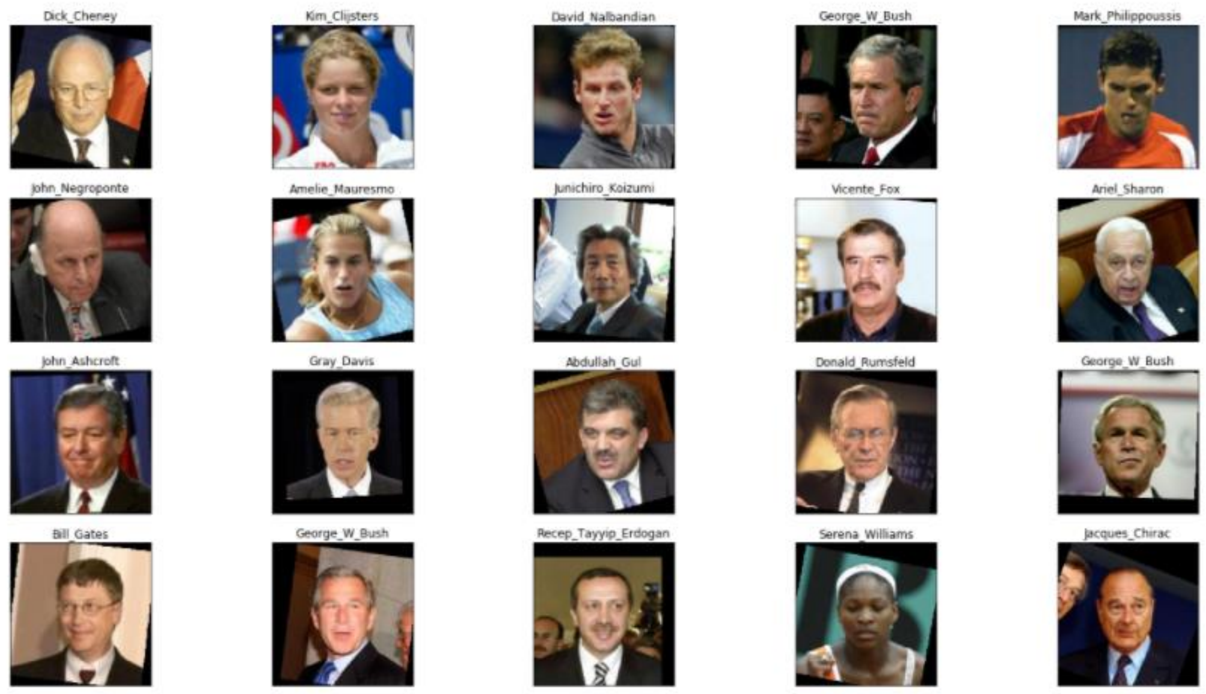


Fig. 12: Images from the LFW dataset

For training the model, a certain number of images pertaining to an individual are required. More the number of images belonging to a certain individual, better will be the performance of the model in recognizing that particular individual. The LFW dataset does not have equal number of images for all unique individuals. Upon exploring the dataset, it was observed that there are only 42 individuals having more than 25 unique images. Hence, taking 25 photographs as the bare minimum, the multi-class classification problems was tackled, making it a 42 class classification problem. There are a total of 2588 images belonging to 42 individuals. All individuals do not have the same number of unique images associated to them. There is a class imbalance, details of which are given in Table 1 –

Name	**Number of images**
Ariel Sharon	77
Colin Powell	236
Donald Rumsfeld	121
George W Bush	530
Gerhard Schroeder	109
Hugo Chavez	71
Jacques Chirac	52
Jean Chretien	55
John Ashcroft	53
Junichiro Koizumi	60
Serena Williams	52
Tony Blair	144
Alejandro Toledo	39

Alvaro Uribe	35
Andre Agassi	36
Bill Clinton	29
David Beckham	31
Gloria Macapagal Arroyo	44
Gray Davis	26
Guillermo Coria	30
Hans Blix	39
Jack Straw	28
Jennifer Capriati	42
John Negroponte	31
Juan Carlos Ferrero	28
Kofi Annan	32
Laura Bush	41
Lleyton Hewitt	41
Luiz Inacio Lula da Silva	48
Mahmoud Abbas	29
Megawati Sukarnoputri	33
Nestor Kirchner	37
Recep Tayyip Erdogan	30
Ricardo Lagos	27
Roh Moo-hyun	32
Rudolph Giuliani	26
Silvio Berlusconi	33
Tom Daschle	25
Tom Ridge	33
Vicente Fox	32
Vladimir Putin	49

Table 1

Fig. 13 gives a clear representation of the difference in the number of images belonging to unique individuals i.e. the class imbalance.

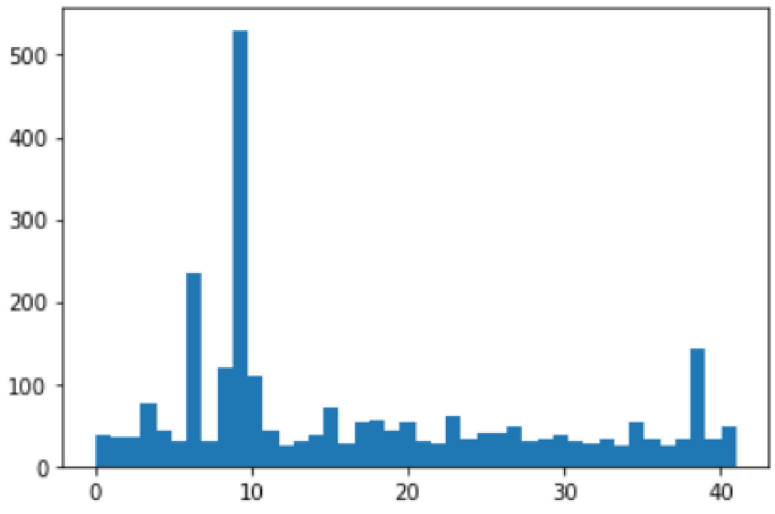


Fig. 13: Distribution of images for 42 classes

The curated LFW dataset was then split into the following components for training and evaluation of model performance on unseen data –

1. Train Dataset - The portion of data used for fitting of the model.
2. Validation Dataset - The portion of data used to facilitate unbiased evaluation of fitted model and also used to tune model hyper-parameters.
3. Test Dataset - The proportion of data used to facilitate an unbiased evaluation of final tuned model on test samples that the model has never seen before.

The dataset split follows the 70:20:10 ratio for training, validation and testing purposes respectively.

3.1.2 Numerical Feature Set

Machine Learning or Deep Learning models cannot take input as images themselves. They require input data in a numerical format. Hence, a fundamental step is to pre-process the images i.e. convert them to a numerical format before feeding them to the model for training. Image samples have to be first converted into a numerical format before feeding the training set of our dataset into the CNN for feature extraction and classification. Images or pictures prevail in various colour spaces such as RGB, HSV, CMYK or simple Grayscale. For this project, we deal with images having Red, Green and Blue channels, and Fig. 14 demonstrates the RGB colour space for a sample image depicting Red, Green and Blue colour channels respectively.

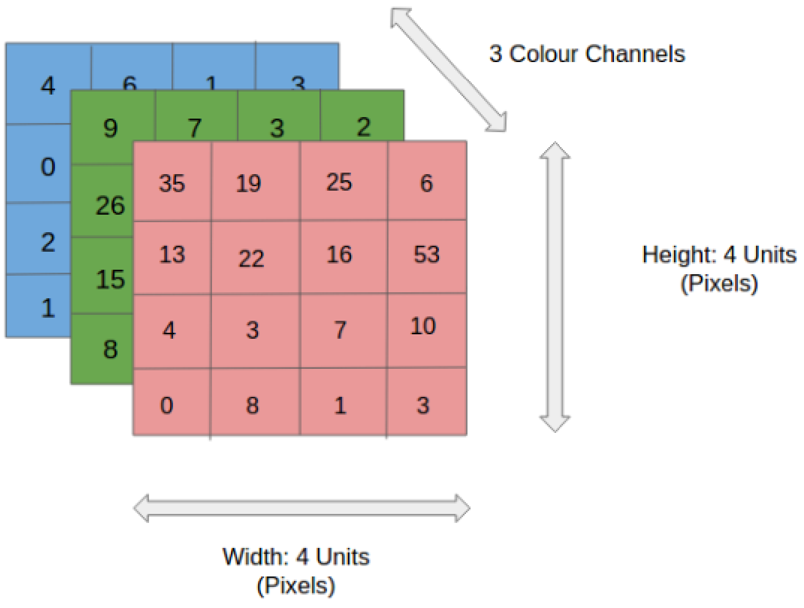


Fig. 14: RGB colorspace demonstration

The sample image whose RGB color space has been depicted in Fig. 12 has a dimension of 4x4 pixels, with 4 pixels as height and 4 pixels as the length of the image. Thus, each channel has a dimension of 4x4 pixels and there exists three such channels: Red, Green, and Blue. Therefore, an input image can be considered as a consolidation of three 2D-matrices of pixel values individually for every color channel. These 2D matrices of numeric pixel values are then unrolled (as an input tensor where each pixel contains three channel values) and are sequentially fed into the input layer of our ConvNet for sequential processing through the proposed Deep Learning pipeline. Thus, every pixel in an image can be thought of as a tensor of three values in RGB color space. This numerical form of data will then be fed to the machine which receives a tensor matrix as the input.

3.1.3 Proposed ConvNet Architecture

The proposed framework has been suitably demonstrated in a diagram format given in Fig. 15. The input dimension of sample images for the proposed model is 154 x 154 x 3 pixels. The proposed pipeline consists of ten convolutional layers with increasing number of filters (from 12 to 48), and same padding in alternate layers. Every second layer has kernels of size (2, 2) with strides (2, 2) and serves the purpose of a trainable pooling layer since it reduces spatial dimensionality. After that, a Dropout for regularization is added, which sets 50% of randomly chosen activations to 0. Then, the last convolutional layer is added for dimensionality expansion to match the number of classes (42 in this case) that we predict. Finally, a global average polling is applied to squash spatial dimensions. The same could've been done with another convolutional layer or a fully connected layer, however, it increases the number of weights to train and, in turn, a risk of overfitting. The final layer of the proposed framework contains 42 neurons and each neuron outputs the probability of the input image belonging to the class it represents using the softmax activation function as described in the previous sections.

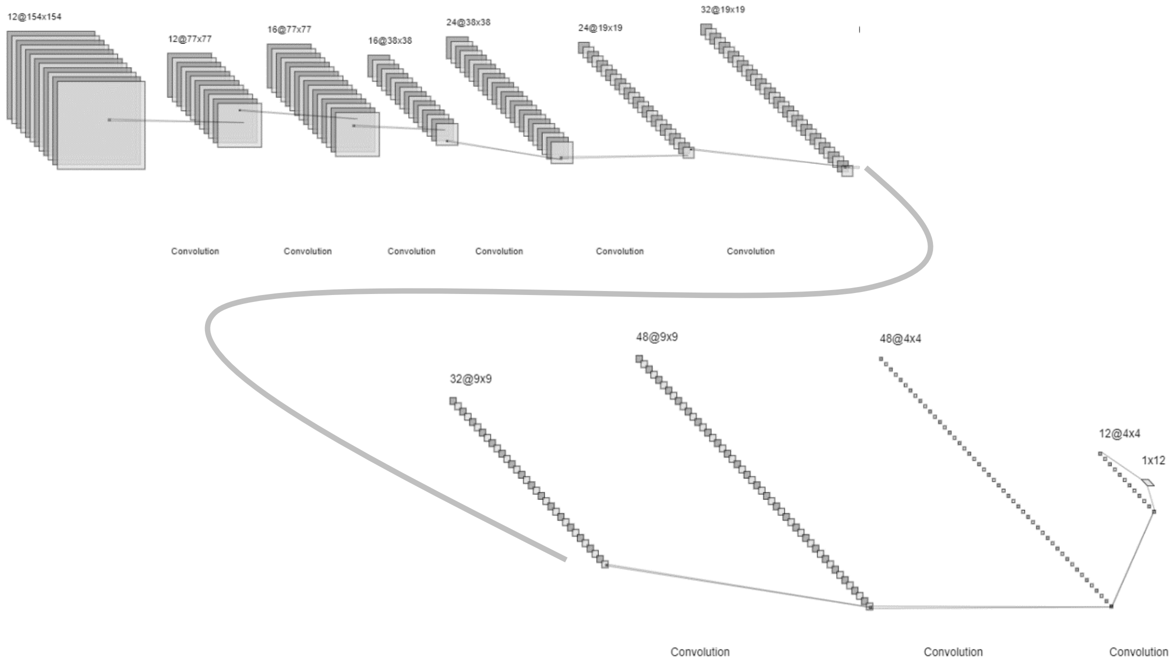


Fig. 15: Proposed CNN architecture

The above architecture uses Global Average Pooling in the last layer. Global pooling samples the entire feature map to a single value instead of down-sampling patches of the input feature map. This would be the same as setting the pool size to the size of the input feature map. In this model, global pooling is used to aggressively summarize the existence of a feature in an image, as well as an alternative to the use of a fully connected layer to transition from feature maps to an output prediction of the model.

The input image that is fed into the ConvNet classification model has a dimension of 154 x 154 x 3 pixels as discussed previously. The first convolutional layer that is responsible for capturing low-level features of the image uses filters (kernel) of size 3 x 3 pixels each that executes the convolution operation thus outputting an image pixel values of dimension 154 x 154 x 12, since same padding is used, which is then passed on to the subsequent layers. The next convolutional layer uses 12 filters, having a dimension of 2 x 2. While convoluting, a stride of 2 has also been used which serves the same purpose as a pooling layer and thus reduces the spatial size. In a similar fashion, the succeeding convolutional layers in the proposed architecture extract high-level features that is used in the classification of the input image. The final output from the last convolutional layer has a dimension of 4 x 4 x 42 that is converted to a single vector of dimension 1 x 42, using Global Average Pooling. This feature vector is then passed on to the softmax output layer for classification.

3.1.4 Model Training

Various architectures and model parameter settings were experimented. The initial rounds of experimentation consisted of a single convolutional layer and a single Max-Pool layer followed by fully connected dense and output layers. This initial architecture did not result in appreciable accuracy metrics due to the presence of a single convolutional layer. As discussed in previous sections, the first convolutional layers in a ConvNet are responsible for low-level feature extraction that extracts the generic characteristics of a human face such as eyes, nose, mouth and facial outline. The low-level features are common to all 42 unique individual classes and thus, the model was not able to learn distinguishing features required to correctly classify different individuals. The addition of convolutional layers further improved the classification performance metrics due to more degrees of freedom to capture high and low-level features that are distinguishable.

Furthermore, it is important to note that a large increase in the number of convolutional layers may lead to vanishing gradients and increase in computational complexity and thus, careful monitoring of test data accuracy while increasing the number of convolutional layers is important. Through various rounds of testing, the optimal ConvNet architecture given in Fig. 13 was chosen, which resulted in promising performance results.

3.1.5 Hyperparameters

Tuning hyperparameters for deep neural network is quite difficult, as it is slow to train a deep neural network, and there are numerous parameters to configure/optimize. Following are some of the hyperparamters which were tuned for the proposed architecture –

1. **Learning rate -** The learning rate is a hyperparameter that controls how much to change the model in response to the estimated error, each time the model weights are updated. Choosing the learning rate is difficult since a value that is too small can lead to a long training phase that may get stuck, while a value that is too high can lead to learning a sub-optimal set of weights too quickly or an inconsistent training process. In the proposed architecture, a fixed learning rate of 0.001 proved out to be optimum, thus giving the best results.

2. **Number of epochs -** Number of epochs are defined as the number of times we are training our model or the number of times we are going back and reuse the dataset to update weights once the whole training is done. It is advised not to keep the number very high as it causes the model to over fit and costs poor testing accuracy. The value should not be too less either as we don’t want our learning to stop while our model has not converged yet. This parameter is rather an empirical one and is selected based on hit and trial methods and can be reduced once for high validation accuracy, you start getting lower testing accuracy. In this project, the model was trained for 600 epochs.

3. **Batch size -** The batch size defines the number of samples that will be propagated through the network in one iteration. ConvNets are sensitive to batch sizes. In the proposed model, a batch size of 256 is used.

4. **Activation function** - The activation function in the neural network is responsible for transforming the summed weighted input from the node into the node activation or output for that input. The rectified linear activation function or ReLU is a piecewise linear function that will output the input directly if it is positive, otherwise it will output zero. For several forms of neural networks, it has become the default activation function because it is easier to train a model that uses it and often achieves better performance.

5. **Dropout** - In order to prevent overfitting in deep neural networks, dropout is a preferable regularization strategy. According to the desired probability, the approach simply drops off units in the neural network. A default value of 0.5 is a good choice for testing.

6. **Optimizer** - The optimizer is responsible for minimizing the objective function of the neural network. A commonly selected optimizer is stochastic gradient descent (SGD), which has proved itself as an efficient and effective optimization method for a large number of published machine learning systems. SGD may, however, be very sensitive to the choice of learning rate. To eliminate the shortcomings of SGD, other gradient based optimization algorithms have been proposed. Adam is an optimization algorithm that can be used to update network weights iteratively based on training data instead of the classical stochastic gradient descent technique. Adam is an adaptive learning rate optimization algorithm that’s been designed specifically for training deep neural networks [reference].

It can be looked at as a combination of RMSprop and Stochastic Gradient Descent with momentum. It uses the squared gradients to scale the learning rate like RMSprop and by using the moving average of the gradient instead of the gradient itself like SGD with momentum, taking advantage of momentum. Adam optimizer has given promising results in literature and thus we will use this as our optimizer.

3.2 Methodology 2

The second methodology involves training the model by means of a Siamese Neural Network. Theoretical concepts pertaining to Siamese Neural Networks and their success in the past has been discussed in detail in Chapter-2. The necessary steps involved in training the model using the second methodology are as follows–

3.2.1 Dataset Curation

For training the model using a Siamese Neural Network, the same subset of the deep-funneled variant of the LFW dataset (as used in methodology 1), comprising of 2588 images belonging to 42 individuals has been used. The only difference is that, in case of a Siamese Neural Network, the images are not directly fed to the network. The images are fed in terms of triplets, comprising of an anchor (A), a positive (P), and a negative (N) image. Hence, the dataset was processed, in order to convert it into a dataset of triplets. The following image represents the dataset obtained after processing –

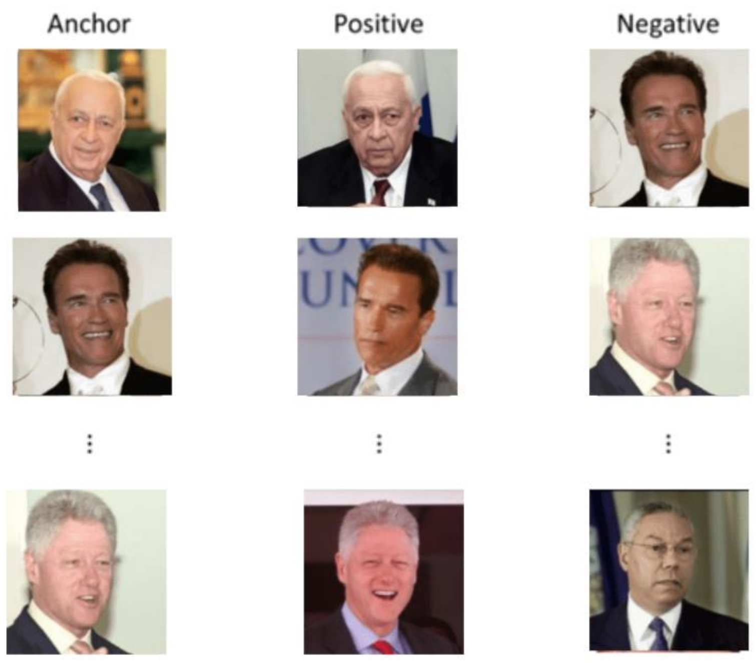


Fig. 16: Triplets formed after processing the LFW dataset

3.2.2 Proposed Siamese Network Architecture

As discussed previously in Chapter-2, a Siamese Neural Network is a class of neural network architecture that consist of two identical convolutional neural networks joined together at the end using an energy function (triplet loss function in this case). The term ‘identical’ here means that the two subnetworks have the same exact same parameters and weights. Any change to any subnetwork architecture, parameters, or weights is also applied to the other subnetwork. In order to have a better comparison of the two methodologies, the same convolutional neural network architecture (as used in methodology 1) has been used in this methodology as well, except for a few changes in the dimensions of the input image and the number of neurons/units in the last layer. The dimension of the input images for the proposed Siamese network model is 96 x 96 x 3 pixels. The dimension of the images has been reduced from 154 x 154 x 3 to 96 x 96 x 3 with the help of Face Alignment. It is a common practice to feed aligned face images (images that comprise of only the face of the person, and no surroundings) to the Siamese network. The Siamese network gives better results when aligned face images are fed, in comparison to when normal non-aligned images are fed as inputs. In-built libraries, such as Dlib for face detection and OpenCV for image transformation, and cropping to produce aligned 96 x 96 RGB face images were used. Fig. 17 depicts how the faces are detected in an image and then aligned to a dimension of 96 x 96 x 3.

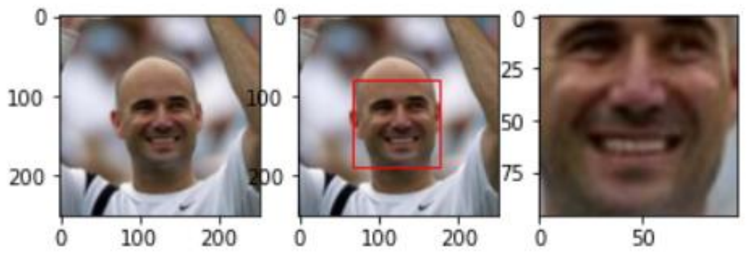


Fig. 17 Face Alignment on the LFW images

The two identical subnetworks of the Siamese Network consist of ten convolutional layers with increasing number of filters (from 12 to 48), and same padding in alternate layers. Every second layer has kernels of size (2, 2) with strides (2, 2) and serves the purpose of a trainable pooling layer since it reduces spatial dimensionality. After that, a Dropout for regularization is added, which sets 50% of randomly chosen activations to 0. The final layer of the proposed framework is a fully connected layer having 128 units followed by an L2 normalization layer on top of the convolutional base.

3.2.3 Face Verification

In Face Verification, given two images, we need to predict whether they belong to the same person or not. The simplest way to achieve this is to compare the two images pixel-by-pixel. If the distance between the two raw images is less than a certain chosen threshold, the two images might belong to the same person. It is obvious that this technique would give really poor results, because the pixel values change drastically due to changes in lighting, minor changes in head position, orientation of the person's face, and so on. Thus, instead of using the raw images, we learn an encoding/embedding, so that element-wise comparison of the two embeddings give more accurate judgements as to whether two pictures are of the same person or not. After training the proposed model, using the triplet loss function as the objective function, a 128 element vector is obtained as the output from both the subnetworks. The Euclidean distance between the two 128 element vectors is calculated, and if the distance is below the threshold τ, then the two images belong to the same individual, else they belong to different individuals. Fig. 18 demonstrates the output (distance between the embeddings of two images) of the proposed Siamese Network.

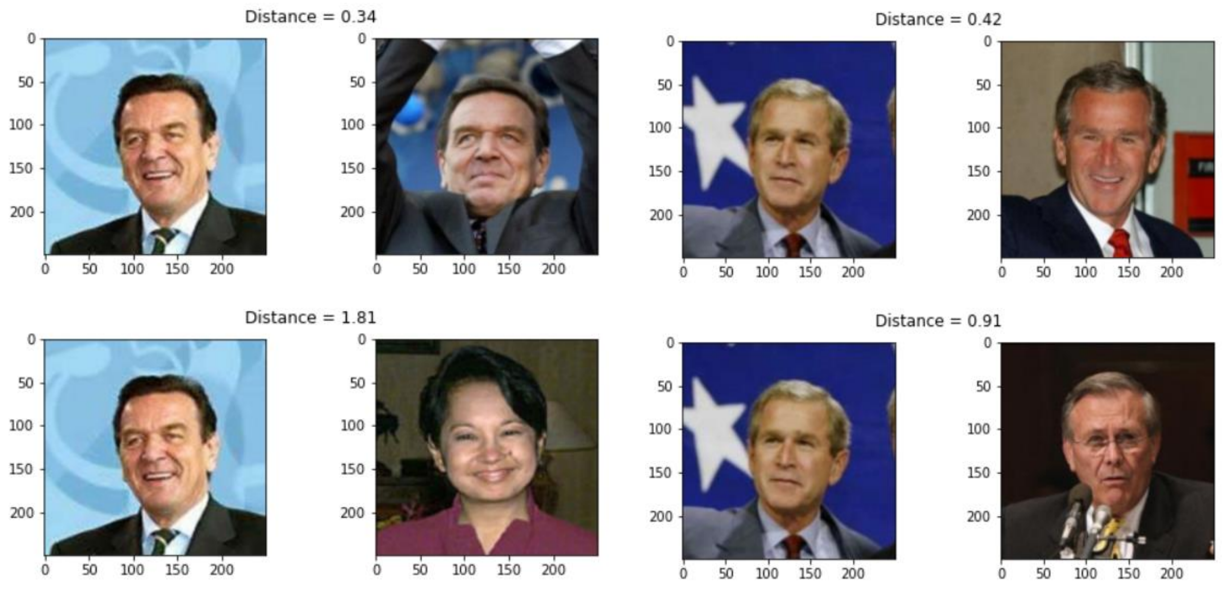


Fig. 18: Distances between pair of images

It can be clearly seen that the distance between the face images belonging to the same individual is low, (indicating that they belong to the same individual) and the distance between the face images belonging to different individuals is high (indicating that they belong to different individuals). Hence, it can be said that the proposed architecture correctly calculates the distances between two images, and is able to recognize if it is the same person in the two images or not.

The most crucial part in the face verification task is choosing the correct threshold distance τ. The performance of the model depends greatly on the chosen threshold. If not correctly chosen, the model would give poor results. To find the optimum value of τ, the performance of the model for verifying faces should be compared on a range of distance threshold values. At a given threshold, all possible embedding vector pairs are classified as either same identity or different identity, and are compared to the ground truth. Since we're dealing with skewed classes (much more negative pairs than positive pairs), it makes sense to use the F1 score as evaluation metric instead of accuracy. The following curve demonstrates the F1 scores over a range of distance thresholds, ranging from 0.3 to 1.0.

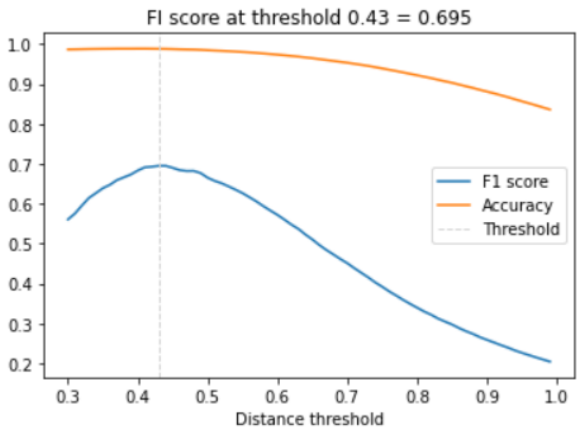


Fig. 19 Distance threshold v/s F1 score

It can be clearly seen in Fig.19 that the maximum F1 score is obtained at a distance threshold of 0.43, and hence it can be said that 0.43 is the optimal value for distance threshold. Hence, for the succeeding steps, $\tau = 0.43$ will be used as the distance threshold i.e. if the distance between the embeddings is lower than 0.43, then the two embeddings represent the same individual, else they represent different individuals.

3.2.4 Face Recognition

In some applications, rather than identifying if the person in the image is the same person or not, we need to recognize the individual i.e. given a face image, we need to tell the name of the person in the image. After obtaining the optimal value of the distance threshold τ, the face verification problem can now be extended to a face recognition problem. Face recognition is now as easy as calculating the distances between all the encodings in the database and an input encoding. The input image is assigned the label of the database entry which has the smallest distance, in case it is less than τ i.e. the distance threshold, or is labeled unknown otherwise. This technique also supports one-shot learning, as adding a single entry of a new person/identity would be sufficient to recognize new samples of that particular individual.

A more robust approach is to label the input using the top K scoring entries in the database which is essentially a KNN classification with a Euclidean distance metric. On the other hand, a Support Vector Machine (linear SVM) can also be trained, and can be used to classify/label i.e. recognize new inputs. In this project, both the SVM and KNN classifiers were trained, and their testing accuracies were compared (results in section 4.2). For training the two classifiers, 50% of the data was used, whereas rest of the 50% data was used for testing. After training the two classifiers, new input images were fed to them, and the outputs were as follows –

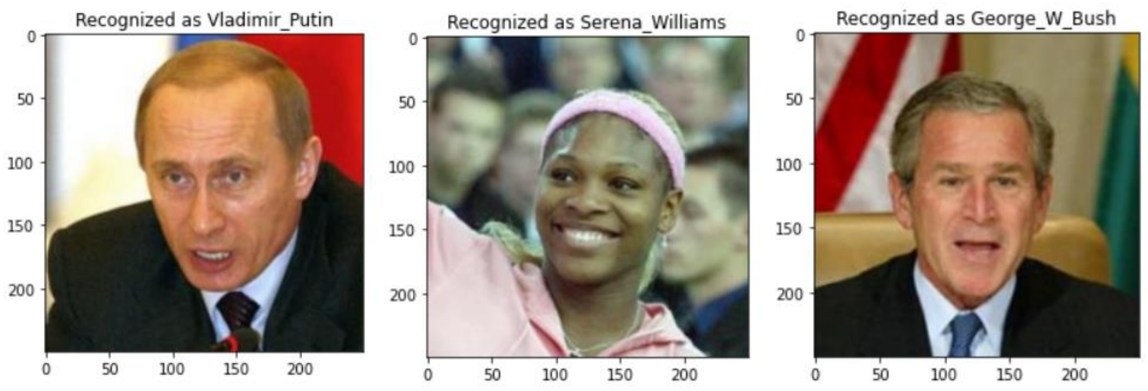

Recognized as Vladimir_Putin
Recognized as Serena_Williams
Recognized as George_W_Bush
0
50
100
150
200

Fig. 20 Outputs of the Classifier/Face Recognition Model

Chapter 4
Results and Conclusion

The classification of images into 42 separate groups of unique individuals as a multi-class classification problem poses a greater challenge compared to a simple binary classification task. Thus, this study targets a more complex classification problem and tests the two proposed frameworks (using methodology 1 and methodology 2) for their ability to correctly recognize an individual's identity from the database.

Since there exists a class imbalance in the dataset, F1 score is used as an evaluation metric. F1 score is the weighted average of precision and recall, and is a good choice for the imbalanced classification scenario. The range of F1 is in [0, 1], where 1 is perfect classification and 0 is total failure. In multi-class classification problems, there are four types of averaging that can be done on precision, recall, and F1 scores –

1. **Macro Average** - Calculates metrics for each label, and finds their unweighted mean. This does not take label imbalance into account.

2. **Micro Average** - Calculates metrics globally by counting the total true positives, false negatives and false positives.

3. **Weighted Average** - Calculates metrics for each label, and finds their average weighted by support (the number of true instances for each label). This alters 'macro' to account for label imbalance; it can result in an F-score that is not between precision and recall.

4. **Samples Average** - Calculate metrics for each instance, and find their average (only meaningful for multi-label classification where this differs from accuracy score).

4.1 Results for Methodology 1

Considering the above definitions, it is clear that the weighted average of F1 scores for different classes should be used as the evaluation metric. Table 2 reports the macro, micro, weighted and samples precision, recall and F1 score for the 42 class classification problem.

	Precision	**Recall**	**F1 Score**
Micro Avg.	0.88	0.88	0.88
Macro Avg.	0.85	0.84	0.84
Weighted Avg.	0.88	0.88	0.88
Samples Avg.	0.88	0.88	0.88

Table 2: Results of Methodology 1 for 42 class classification

Fig. 21 depicts the graph of F1 scores for different classes in the 42 class classification problem. The green line in the following figure represents the weighted average F1 score.

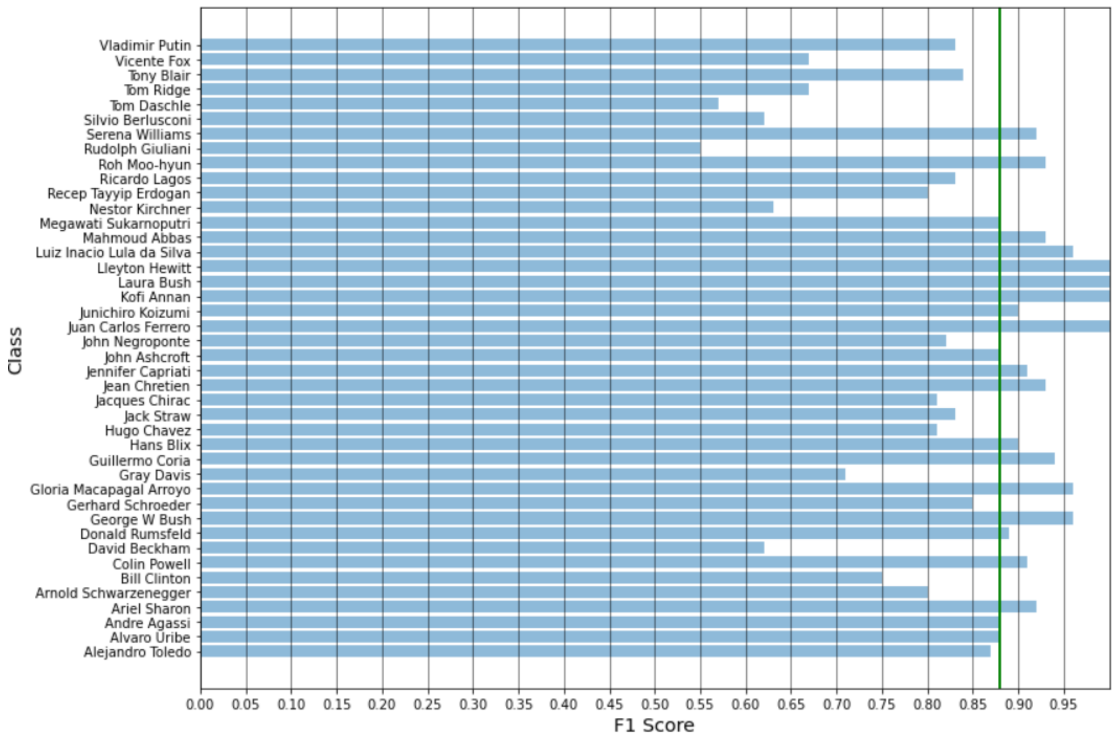


Fig. 21: F1 score of Methodology 1 for all 42 classes

4.2 Results for Methodology 2

Since we are using the same class imbalanced dataset in methodology 2 (as used in methodology 1), we'll be using the weighted F1 score as the evaluation metric. Table 3 and Table 4 report the weighted and macro precision, recall and F1 score for the KNN, and the SVM classifier respectively.

	Precision	**Recall**	**F1 Score**
Macro Avg.	0.92	0.90	0.91
Weighted Avg.	0.95	0.94	0.94

Table 3: Results of KNN classifier (methodology 2) for 42 class classification

	Precision	**Recall**	**F1 Score**
Macro Avg.	0.95	0.94	0.95
Weighted Avg.	0.98	0.98	0.98

Table 4: Results of SVM classifier (methodology 2) for 42 class classification

Fig. 22 and Fig. 23 depict the graph of F1 scores obtained for different classes in the 42 class classification problem, using the KNN, and the SVM classifier respectively. The green line in the following figures represents the weighted average F1 score.

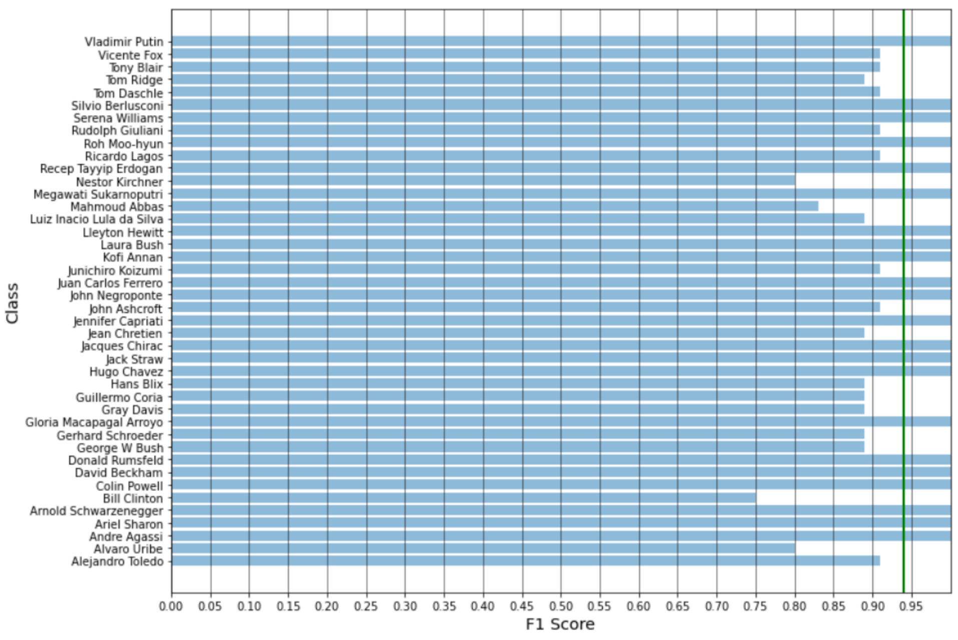


Fig. 22: F1 scores of different classes using the KNN classifier (methodology 2)

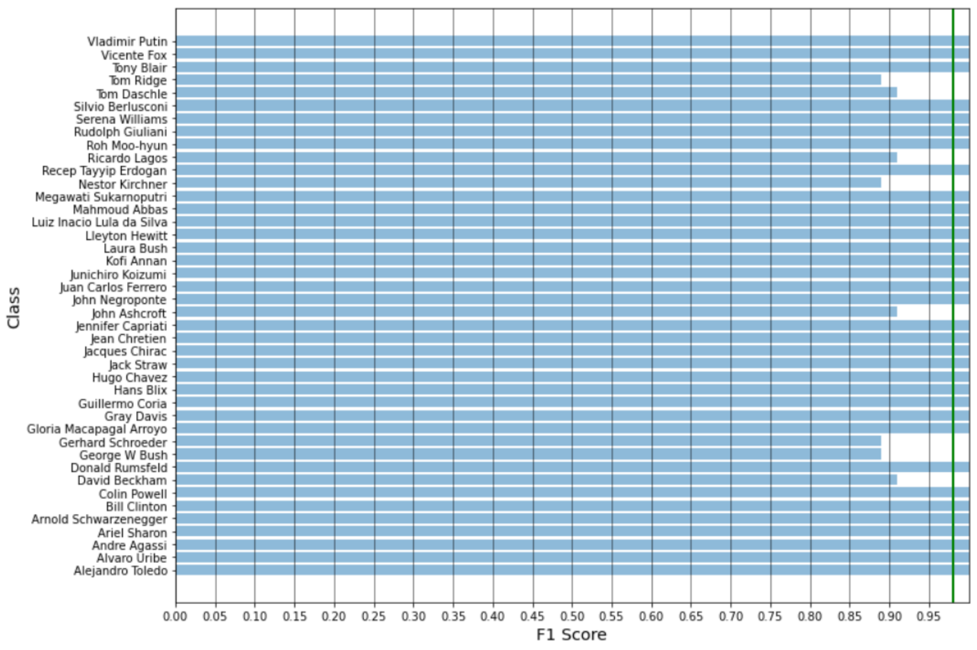


Fig. 23: F1 scores of different classes using the SVM classifier (methodology 2)

It can be clearly seen that the SVM classifier performs better than the KNN classifier (since it has a higher F1 score) for the 42 class classification problem. Hence, we will consider the results of the SVM classifier for comparison with the results obtained in Methodology 1.

The experimentation and analysis of the proposed framework for facial recognition has been implemented using Jupyter Notebook and has been coded in Python using TensorFlow, Sklearn, and Keras libraries.

4.3 Conclusion

On comparing the results of Methodology 1 with the results of Methodology 2, it is clear that Methodology 2 (model training through a Siamese Neural Network) giver much better results, in terms of a higher F1 score. But, it should be noticed that a subset, comprising of 2588 images, belonging to 42 unique individuals was used for training the two models. The LFW dataset comprises of more 13,233 images belonging to 5,749 unique individuals. The reason for choosing a subset of such a large dataset was to adhere to the computational constraints. Thus, it can be said that the Siamese Network performs better than Traditional Convolutional Neural Networks for this particular subset of the dataset. One cannot generalize or conclude that one methodology performs better than the other, since the two models were trained on a subset of a bigger dataset.

Furthermore, in literature, researchers have trained face recognition machines on millions of images pertaining to millions of unique individuals. They have used one dataset for training the machine and a completely different dataset for testing the trained model. The LFW dataset, if used completely (all 13,233 images) for training and testing purposes, will still not give concrete results, by which one can assume one methodology to be better than the other. Both the methodologies have given exceptionally good results in literature. The use of one methodology over the other boils down to the application for which the machine is to be used. For some applications, training a traditional convolutional neural network might be a better choice than training a Siamese Network, and vice versa. Both the methodologies have their respective pros and cons, and hence no methodology is superior/inferior than the other. This work/project focuses more on exploring/discussing the two methodologies in detail, than achieving conclusive results.

Chapter 5

Appendix

5.1 Source Code

The source code for the project along with the data can be found on this Link.

Bibliography

[1] http://vis-www.cs.umass.edu/lfw/

[2] Sun, Yi, et al. "Deepid3: Face recognition with very deep neural networks." arXiv preprint arXiv:1502.00873 (2015).

[3] Taigman, Yaniv, et al. "Deepface: Closing the gap to human-level performance in face verification." Proceedings of the IEEE conference on computer vision and pattern recognition. 2014.

[4] Bromley, J.; Bentz, J.W.; Bottou, L.; Guyon, I.; LeCun, Y.; Moore, C.; Sackinger, E.; Shah, R. Signature verification using a siamese time delay neural network. Int. J. Pattern Recognit. Artif. Intell. 1993, 7, 669–688.

[5] Sun, Yi, Xiaogang Wang, and Xiaoou Tang. "Deep learning face representation from predicting 10,000 classes." Proceedings of the IEEE conference on computer vision and pattern recognition. 2014..

[6] Schroff, Florian, Dmitry Kalenichenko, and James Philbin. "Facenet: A unified embedding for face recognition and clustering." Proceedings of the IEEE conference on computer vision and pattern recognition. 2015