PROJECT REPORT ON

DETECTION AND RECOGNITION OF FACES USING MACHINE LEARNING

SUBMITTED BY:

SHREYANSH CHORDIA (171070013)
PRATHAMESH GURAV (171070019)
SAGAR SANAP (171070036)
PARTH WANI (171070052)

THIRD YEAR, B.TECH. COMPS
ACADEMIC YEAR 2019-20

UNDER THE GUIDANCE OF

PROF. SEEMA SHRAWANE & PROF. SUCHITA DANGE

DEPARTMENT OF COMPUTER ENGINEERING
VEERMATA JIJABAI TECHNOLOGICAL INSTITUTE, MUMBAI

THEORY

Face Detection

Face detection also called facial detection is an artificial intelligence (AI) based computer technology used to find and identify human faces in digital images.

Face detection applications use algorithms and ML to find human faces within larger images, which often incorporate other non-face objects such as landscapes, buildings and other human body parts like feet or hands. Face detection algorithms typically start by searching for human eyes -- one of the easiest features to detect. The algorithm might then attempt to detect eyebrows, the mouth, nose, nostrils and the iris. Once the algorithm concludes that it has found a facial region, it applies additional tests to confirm that it has, in fact, detected a face.

To help ensure accuracy, the algorithms need to be trained on large data sets incorporating hundreds of thousands of positive and negative images. The training improves the algorithms' ability to determine whether there are faces in an image and where they are.
The methods used in face detection can be knowledge-based, feature-based, template matching or appearance-based. Each has advantages and disadvantages:

- **Knowledge-based, or rule-based methods** -- describe a face based on rules. The challenge of this approach is the difficulty of coming up with well-defined rules.

- **Feature invariant methods** -- which use features such as a person's eyes or nose to detect a face can be negatively affected by noise and light.

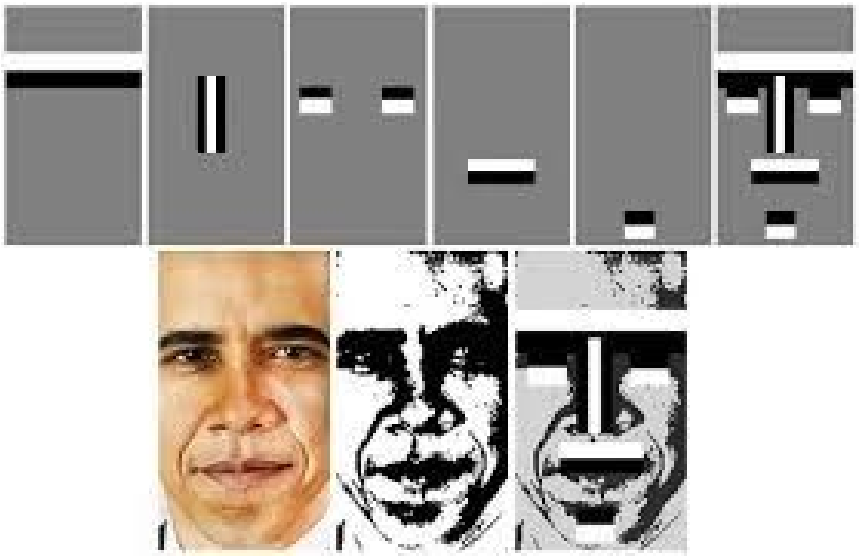
- **Template-matching methods** -- are based on comparing images with standard face patterns or features that have been stored previously and correlating the two to detect a face. Unfortunately these methods do not address variations in pose, scale and shape.

- **Appearance-based methods** -- employ statistical analysis and machine learning to find the relevant characteristics of face images. This method, also used in feature extraction for face recognition, is divided into sub-methods.

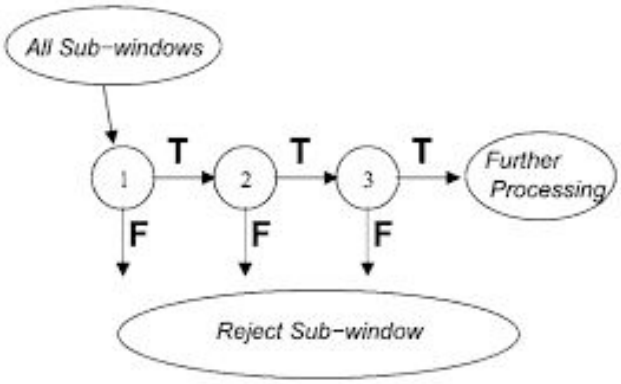
For our project, we focused on a feature invariant method for detection of faces that is widely known as **Viola Jones Algorithm.**

VIOLA & JONES ALGORITHM

Major improvements to face detection methodology came in 2001, when computer vision researchers Paul Viola and Michael Jones proposed a framework to detect faces in real time with high accuracy. The Viola-Jones framework is based on training a model to understand what is and is not a face. Once trained, the model extracts specific features, which are then stored in a file so that features from new images can be compared with the previously stored features at various stages. If the image under study passes through each stage of the feature comparison, then a face has been detected and operations can proceed.



Detection of a face by searching for features (HAAR-LIKE FEATURES) in the image



A face gets detected if and only if all the features exist otherwise rejected

Face Recognition

Face recognition is the process of identifying a face, such that if the algorithm has learnt a face (can be done by different methods) then if the detected face is of the same person, the algorithm would recognize it, that is it would be able to identify it.

The 2 major techniques that have been implemented in this project for recognition of a face are

1. **Eigen Faces**

2. **Siamese Network**

It is important to note that *Siamese Network* is one of the finest methods for Face Recognition that uses a Deep Network and is able to achieve state-of-the-art results.

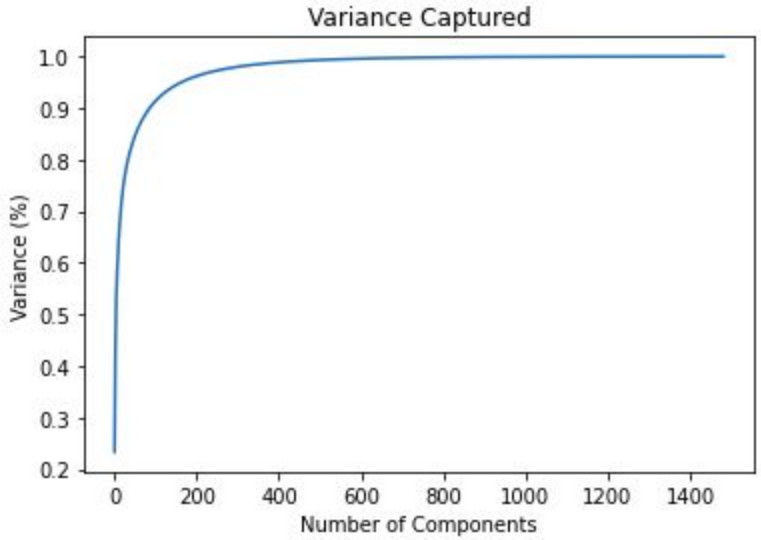
EIGENFACE METHOD

The EigenFace Method is simply the PCA (Principal component Analysis) algorithm.
In Machine Learning, PCA is used for reducing the dimensionality of our data by keeping as much variance as possible.
The PCA Algorithm does this generating EigenVectors that are of the size of our single sample data. The number of eigen-vectors that are generated by the algorithm are lesser than the number of features present in every sample.
Hence according to the hypothesis, every sample of the data is nothing but simply a Linear Combination of all the eigen-vectors that are generated.

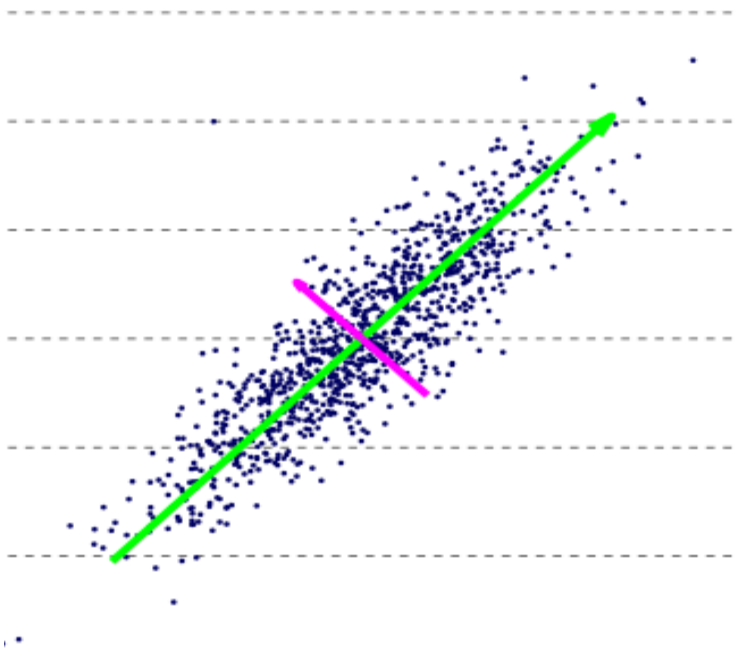
But do we require all the eigenvectors to represent the face?

Now here is where things change. The amount of variance that is captured by each eigen-vector can be identified by its eigen-values.
Such that, the higher the eigenvalues the greater is the amount of variance of the dataset that is captured by the eigen-vector.
Let us assume that we have a dataset of 3000 features. We run the PCA algorithm on this dataset for dimensionality reduction.
On visualising the eigen-vectors with their eigen-values, what we generally see is that eigen-vectors can effectively capture all up to 96% of all the variance of the data by using approximately 10% dimensionality of the dataset (for huge feature datasets).

So if we can effectively capture 96% variance of the a 3000 dimension dataset by just using 300 dimensions then why do we require so many dimensions.



In our project we used PCA to reduce dimensionality from 2914 features to 250 features



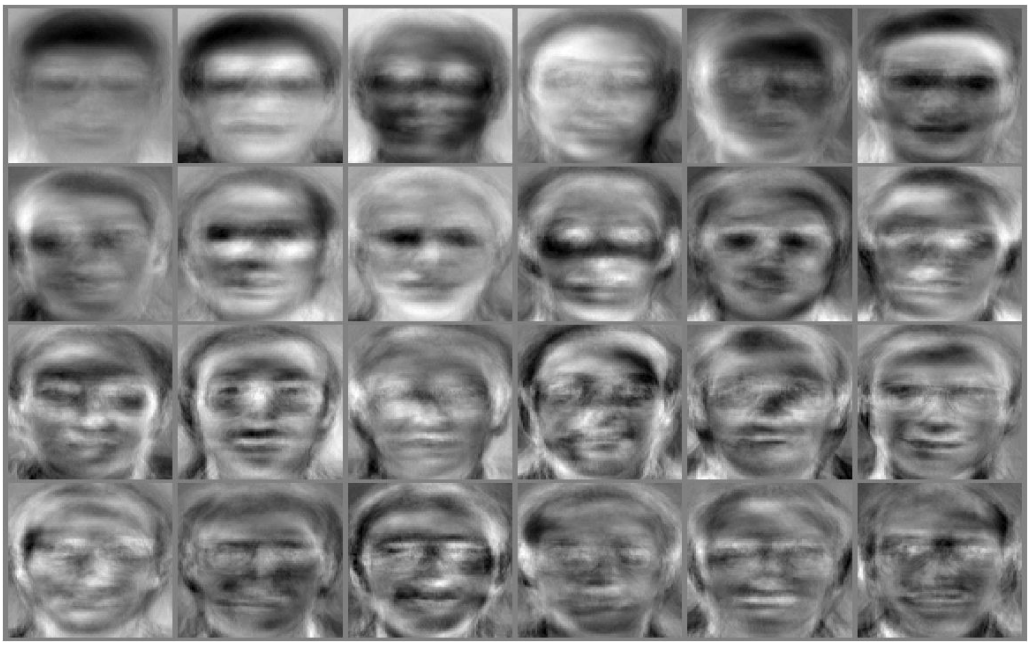
Visual representation of PCA algorithm

Hence by using PCA, we generate eigen-vectors, which are popularly known as eigen-faces, because when these eigen-vectors are visualised, we get certain important information. We realise that these eigen-faces individually look like ghostly faces, but each eigen-face highlights certain features.

For example we see that in some eigen-faces either there is a detailing in eye, in some eigen-face the lips are better structured and in some eigen-faces the nose might be well structured. This gives us an intuition that how a linear combination of all these faces can lead to an image whose 96% data is still captured.

$$Face\,X = \lambda 1\,(Nose) + \lambda 2\,(Eyes) + \lambda 3\,(Lips) + \lambda 4\,(Hair) + ...$$

Where the $Eyes, Nose, Lips, Hair$ etc. represent different eigen-faces. A particular feature cannot be represented easily by a single eigen-face. It is represented by a collection of eigen-vectors.

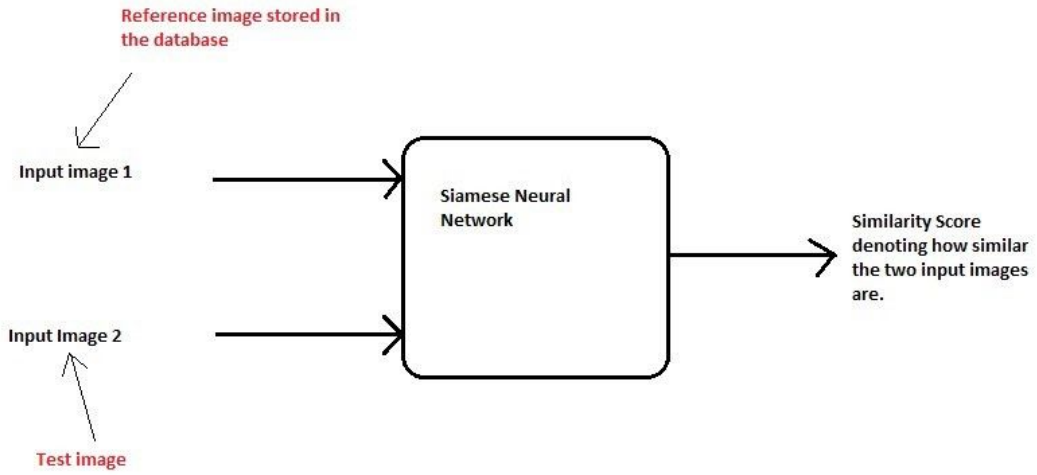


Eigen faces

SIAMESE NETWORK

Siamese Network is a one-shot learning network, that is, it does not require an explicit amount of training. Only by having a single image of a person, such a network can robustly recognize that person.
But this network requires a lot of training. Hence we have just implemented the network and have not trained it until perfection.
The siamese network generates embeddings of faces and compares for similarity (i.e. the cosine distance between the embeddings).

So here the intuition is that, the network generates similar embeddings of the similar faces, hence the cosine distance of the euclidean distance, any distance will be smaller for similar images while it will be larger for dissimilar faces.



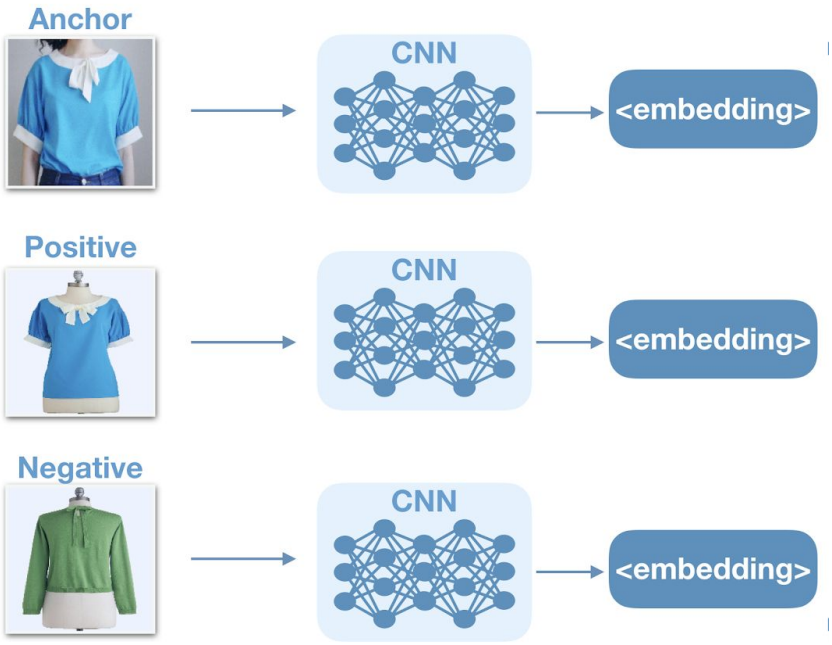
Siamese Network predicting similarity between faces

Data Sample

For this network, during the training phase we send in 3 images.

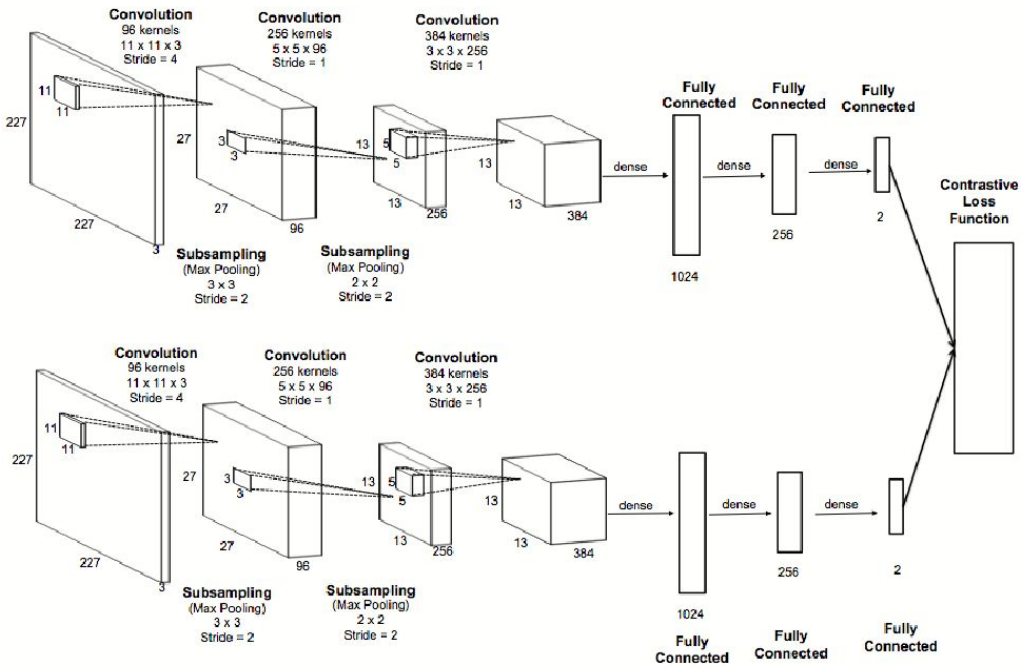
1. Anchor image
2. Positive image
3. Negative image

Such that, in every iteration, our aim is to reduce the distance between the embeddings of the anchor image and the positive image, and increase the distance between the embeddings of the anchor image and negative image.



Anchor image, Positive image and negative image

Architecture of the network



Architecture of Siamese Network

layer	size-in	size-out	kernel	param	FLPS
conv1	220×220×3	110×110×64	7×7×3, 2	9K	115M
pool1	110×110×64	55×55×64	3×3×64, 2	0	
rnorm1	55×55×64	55×55×64		0	
conv2a	55×55×64	55×55×64	1×1×64, 1	4K	13M
conv2	55×55×64	55×55×192	3×3×64, 1	111K	335M
rnorm2	55×55×192	55×55×192		0	
pool2	55×55×192	28×28×192	3×3×192, 2	0	
conv3a	28×28×192	28×28×192	1×1×192, 1	37K	29M
conv3	28×28×192	28×28×384	3×3×192, 1	664K	521M
pool3	28×28×384	14×14×384	3×3×384, 2	0	
conv4a	14×14×384	14×14×384	1×1×384, 1	148K	29M
conv4	14×14×384	14×14×256	3×3×384, 1	885K	173M
conv5a	14×14×256	14×14×256	1×1×256, 1	66K	13M
conv5	14×14×256	14×14×256	3×3×256, 1	590K	116M
conv6a	14×14×256	14×14×256	1×1×256, 1	66K	13M
conv6	14×14×256	14×14×256	3×3×256, 1	590K	116M
pool4	14×14×256	7×7×256	3×3×256, 2	0	
concat	7×7×256	7×7×256		0	
fc1	7×7×256	1×32×128	maxout p=2	103M	103M
fc2	1×32×128	1×32×128	maxout p=2	34M	34M
fc7128	1×32×128	1×1×128		524K	0.5M
L2	1×1×128	1×1×128		0	
total				140M	1.6B

Similar Architecture used in FaceNet

Triple Loss Function

The loss function that we use here is not a ‘very often’ used loss function. A special loss function is used for training the network, that is the triple loss function.

This function calculates how far we are from gaining state-of-the-art.

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

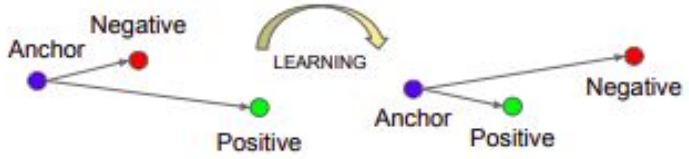
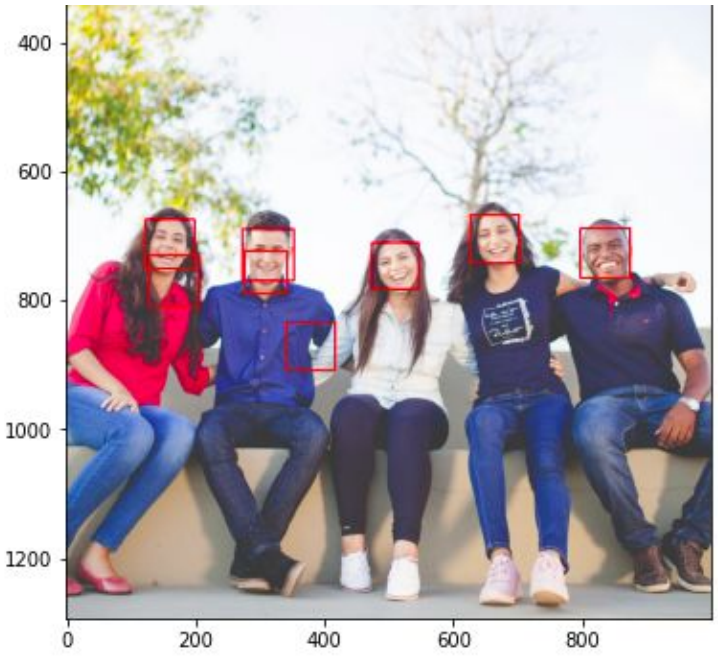


Figure 3. The **Triplet Loss** minimizes the distance between an *anchor* and a *positive*, both of which have the same identity, and maximizes the distance between the *anchor* and a *negative* of a different identity.

TRAINING, RESULTS & ACCURACY

1. Detection using Viola Jones Algorithm :



Viola Jones algorithm detecting faces with a few errors

2. Recognition of faces using classification algorithms :

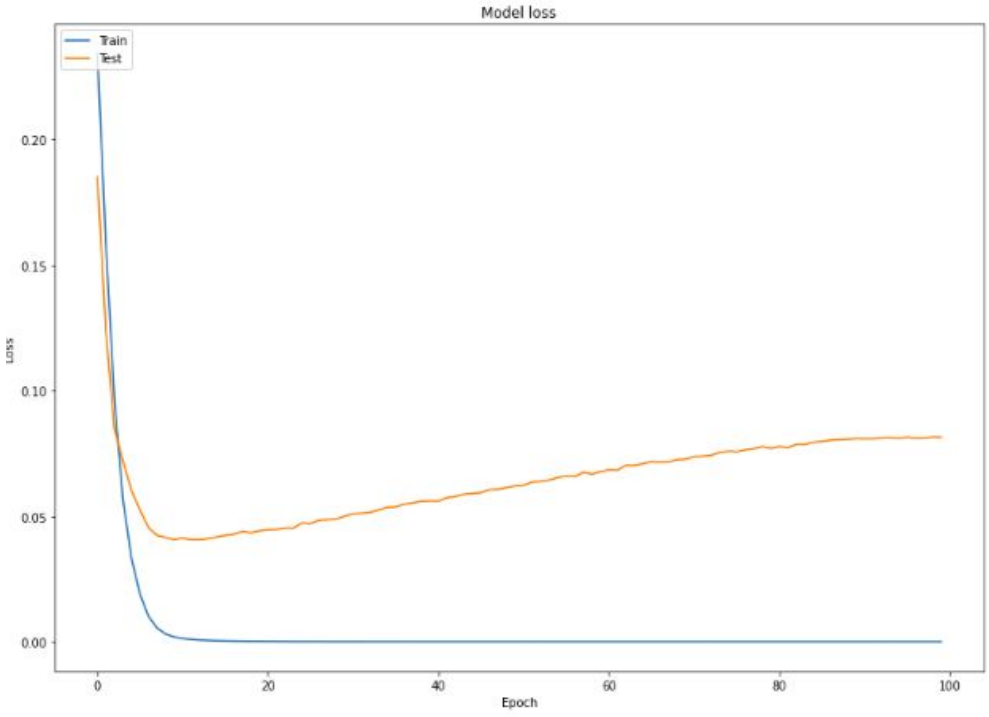
CLASSIFIER	ACCURACY	VARIANCE IN RESULT
Logistic Regression	0.627807	(0.036481)
NB Gaussian Classifier	0.605882	(0.019837)
Support Vector Machine	0.628342	(0.029630)
RandomForest Classifier	0.647059	(0.019793)

3. Details of the EigenFace Method :

Architecture :

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
dense_1 (Dense)              (None, 50)                12550
_________________________________________________________________
dense_2 (Dense)              (None, 25)                1275
_________________________________________________________________
dense_3 (Dense)              (None, 12)                312
=================================================================
Total params: 14,137
Trainable params: 14,137
Non-trainable params: 0
_________________________________________________________________
```

Training :



Loss v/s Epochs

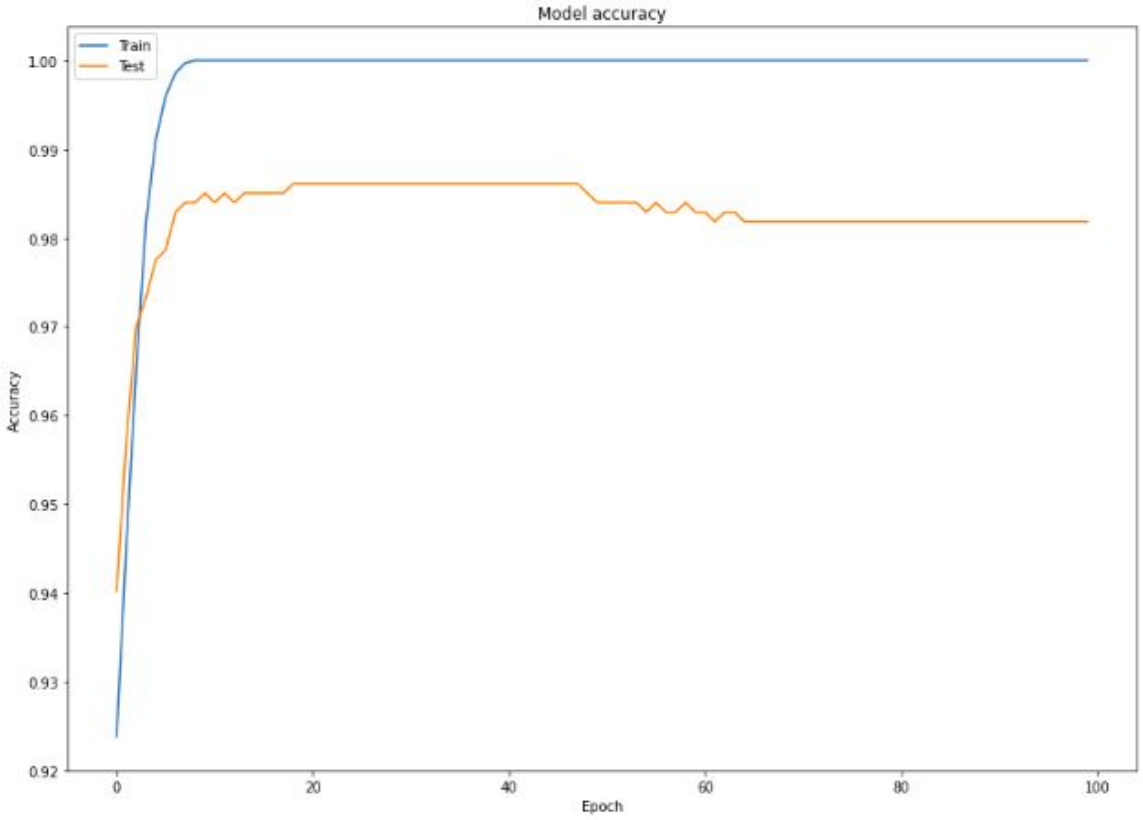
```
Epoch 95/100
1482/1482 [==============================] - 1s 619us/step - loss: 1.0168e-07 -
acc: 1.0000 - val_loss: 0.0809 - val_acc: 0.9818
Epoch 96/100
1482/1482 [==============================] - 1s 620us/step - loss: 1.0165e-07 -
acc: 1.0000 - val_loss: 0.0814 - val_acc: 0.9818
Epoch 97/100
1482/1482 [==============================] - 1s 633us/step - loss: 1.0164e-07 -
acc: 1.0000 - val_loss: 0.0809 - val_acc: 0.9818
Epoch 98/100
1482/1482 [==============================] - 1s 617us/step - loss: 1.0161e-07 -
acc: 1.0000 - val_loss: 0.0811 - val_acc: 0.9818
Epoch 99/100
1482/1482 [==============================] - 1s 640us/step - loss: 1.0161e-07 -
acc: 1.0000 - val_loss: 0.0815 - val_acc: 0.9818
Epoch 100/100
1482/1482 [==============================] - 1s 633us/step - loss: 1.0161e-07 -
acc: 1.0000 - val_loss: 0.0814 - val_acc: 0.9818
```

Last 5 epochs of training

Accuracy : 0.8846153846153846

	precision	recall	f1-score	support
0	1.00	0.75	0.86	4
1	0.85	0.92	0.88	12
2	1.00	0.83	0.91	6
3	0.93	0.96	0.94	26
4	1.00	1.00	1.00	5
5	1.00	0.67	0.80	3
6	0.67	0.67	0.67	3
7	1.00	1.00	1.00	3
8	1.00	0.67	0.80	3
9	0.67	0.67	0.67	3
10	1.00	0.67	0.80	3
11	0.70	1.00	0.82	7
micro avg	0.88	0.88	0.88	78
macro avg	0.90	0.82	0.85	78
weighted avg	0.90	0.88	0.88	78
samples avg	0.88	0.88	0.88	78

Accuracy Matrix over the 12 classes



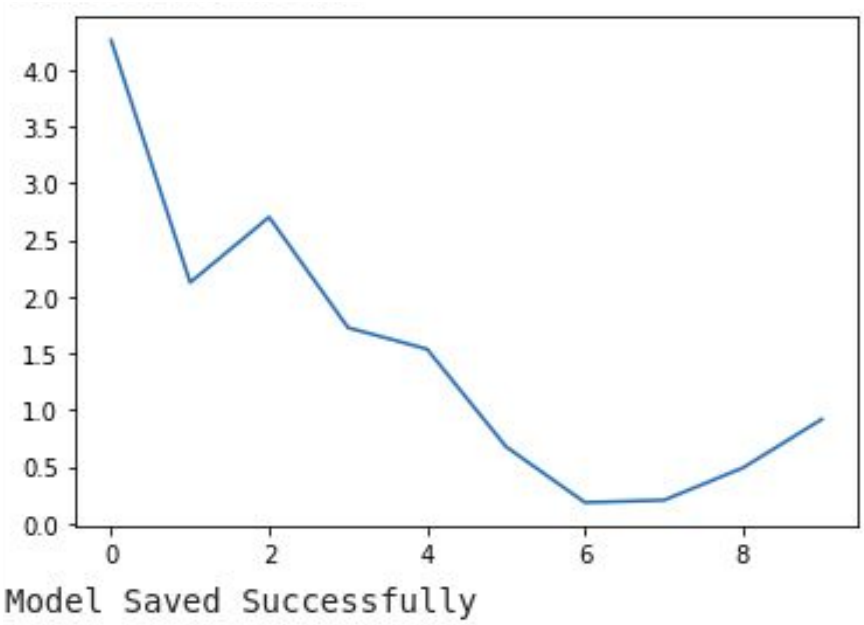
Accuracy v/s Epochs

3. Siamese Network

Training :

```
4.257913796340718
2.1230578001807716
2.699061393737793
1.7247612897087545
1.5363181969698738
0.67587490642772
0.18471094440011418
0.20510726816513958
0.4904266806209789
0.9190531969070435
```

Reduction in loss with epochs



Visualisation of Training

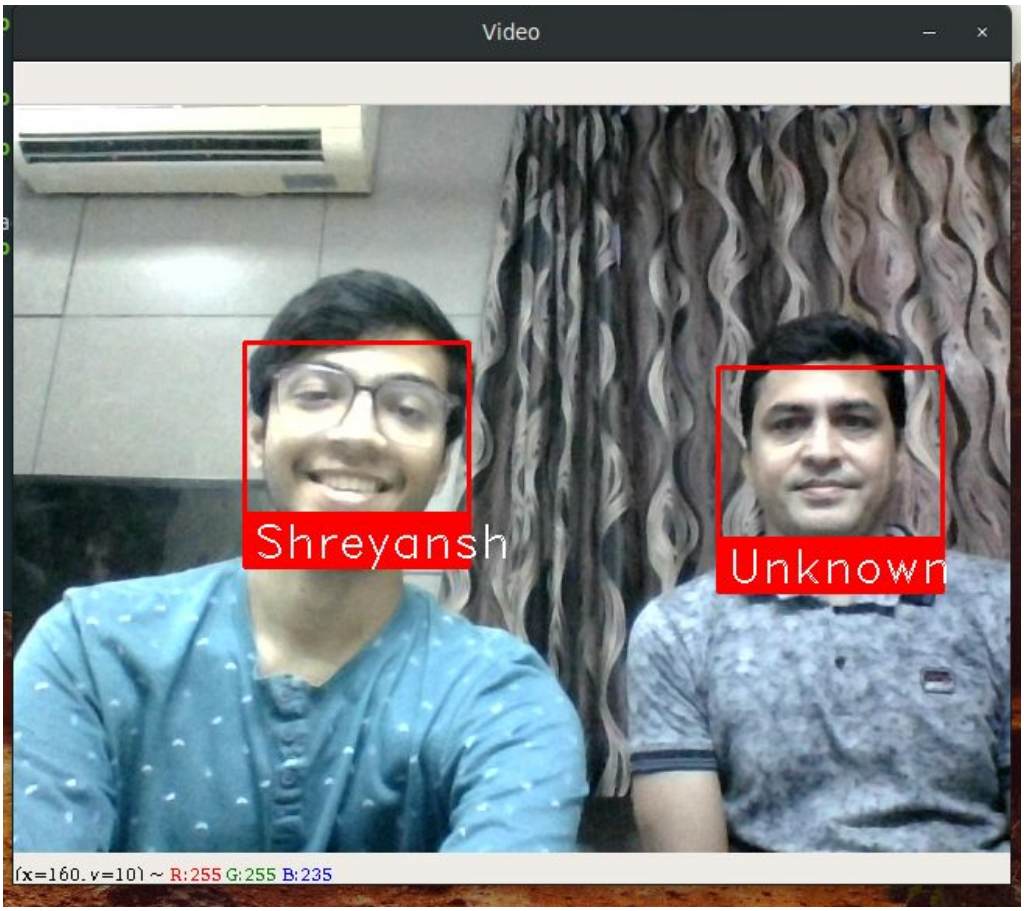
4. Using Pre Defined face-recognition module in Python

Here we perform one shot learning.

I add my following picture in the dataset and then test the face-recognition module for one shot learning.

Loading my picture to dataset as Shreyansh

Testing :



Face-recognition module recognizing faces loaded in the dataset

CONCLUSION

We have implemented the Viola Jones Algorithm for detection of faces, and for recognition of faces we implemented different algorithms and different neural networks in our quest of achieving state-of-the-art. The algorithms that we examined were :

1. Logistic Regression
2. NB Gaussian Classifier
3. Support Vector Machine
4. Random Forest Classifier
5. PCA algorithm (EigenFace Method)
6. Siamese Network

We haven't been able to achieve state-of-the-art results because we lack the amount of time and computation power required to build such robust models.

FURTHER IMPROVEMENTS

Face Detection :

Although the Viola-Jones framework is still popular for recognizing faces in real-time applications, it has limitations. For example, the framework might not work if a face is covered with a mask or scarf, or if a face is not properly oriented, then the algorithm might not be able to find it.
To help eliminate the drawbacks of the Viola-Jones framework and improve face detection, other algorithms -- such as region-based convolutional neural networks (R-CNN) and Single Shot Detector (SSD) -- have been developed to help improve processes. We would like to research these networks as well.

Face Recognition :

We would like to try transfer learning on ImageNet and FaceNet to see if we can gain even better results.

REFERENCES

https://www.cs.cmu.edu/~efros/courses/LBMV07/Papers/viola-cvpr-01.pdf

https://youtu.be/FgakZw6K1QQ

https://medium.com/datadriveninvestor/principal-components-analysis-pca-71cc9d43d9fb

https://www.cs.cmu.edu/~rsalakhu/papers/oneshot1.pdf

https://arxiv.org/pdf/1503.03832.pdf

https://medium.com/@mohitsaini_54300/train-facenet-with-triplet-loss-for-real-time-face-recognition-a39e2f4472c3

https://medium.com/@athul929/building-a-facial-recognition-system-with-facenet-b9c249c2388a

https://arxiv.org/pdf/1503.03832.pdf

https://www.researchgate.net/publication/332677329_Machine_Learning_Based_Approach_for_Person_Identification_in_Group_Photos