Comparative Study of Different Face Recognition Algorithms

Aayush Shah
Information and Communication Technology Engineering
School of Engineering and Applied Science, Ahmedabad University
Ahmedabad, India
aayush.s.btechi16@ahduni.edu.in

Maitrik Shah
Information and Communication Technology Engineering
School of Engineering and Applied Science, Ahmedabad University
Ahmedabad, India
maitrik.s@ahduni.edu.in

Nihar Shah
Information and Communication Technology Engineering
School of Engineering and Applied Science, Ahmedabad University
Ahmedabad, India
nihar.s.btechi16@ahduni.edu.in

Soham Joshi
Information and Communication Technology Engineering
School of Engineering and Applied Science, Ahmedabad University
Ahmedabad, India
soham.j.btechi16@ahduni.edu.in

Yagnesh M. Bhadiyadra
Information and Communication Technology Engineering
School of Engineering and Applied Science, Ahmedabad University
Ahmedabad, India
yagnesh.b.btechi16@ahduni.edu.in

***Abstract*—In this paper, we take into account different approaches like Eigenfaces, Principal Component Analysis(PCA), Support Vector Machines(SVM), Artificial Neural Networks(ANN), Convolutional Neural Networks(CNN), K-Nearest Neighbour(KNN) for the problem of face recognition and compare all the approaches on the basis of different performance metrics such as Accuracy, Number of Iterations and Error Rate to see which technique is more feasible in real life. Then after we also recognize emotions in a face using the Support Vector Machines(SVM) and the Convolutional Neural Networks(CNN). The approaches we have considered treats Face Recognition and Emotions Recognition problem as two-dimensional recognition problem, the advantage being faces can be described by a small set of 2-D characteristics views.**

Keywords - Eigenfaces, Principal Component Analysis(PCA), Support Vector Machines(SVM), Convolutional Neural Networks(CNN), Face recognition, Emotions Recognition.

I. Introduction

Face recognition has wide ranges of applications such as identity authentication, access control, biometrics, and surveillance. Research activities in the area of Face Recognition have increased over the past few years. Emotions Recognition also finds many real-life applications in the field of Medicine Science, Patient Health Monitoring in the health-care department, Marketing, etc. In addition, "the variations between the images of the same face due to illumination and viewing direction are almost always larger than image variations due to change in face identity"[1]. This presents a challenge in the domain of both Face Recognition and Emotions Recognition. The central issues are what features to use to represent a face and classifying a new face image based on the chosen representation. Various relations(i.e., distance, angles) and other several properties are often used as descriptors of faces for recognition. Section 2 describes details of work done so far in the area and in Section 3, various methods which are implemented for Face Recognition are described in detail and working methodology of Emotion Recognition through Support Vector Machines(SVM) and Convolutional Neural Networks(CNN) is also elaborated.

The comparison results of all the different methods are further described in Section 4. The details of the results of Emotion Recognition is also described. Section 5 finally concludes the paper with future work possibilities and the results overview.

II. Related Work and motivation

Face Recognition is a very important problem in today's world and very advanced technologies are being used to address the problem. The 3-D Recognition as a security feature is now available in smartphones as well. The first solution to the problem was proposed in 1888 by Francis Galton[2]. He collected facial profiles as curves, calculated their norm and used the difference of the norm and the norm of the standard face profile collected as curves and classified input accordingly. This classification resulted in a vector of independent measures that could be compared with the other vectors in the database. The development in the field of face recognition has grown rapidly since the past few years, and one approach came out in the existence in 1997 was

suggesting three types of basic face recognition algorithms: frontal, profile and view-tolerant recognition. The first one, frontal recognition is the simple classic approach to face recognition, but better results can be observed using the second way(profiles based approach) which took into account some of the physics, geometry, and statistics, and as a single, stand-alone system, Profile schemes have a rather marginal significance for identification[3]. One very important innovation in the field of face detection was Haar-like classifier-based approach where the authors selected various rectangle based approaches and through those face, the detection was giving significance accuracy up to 93% when enough data was available to train the model[14].

Eigenfaces are one of the most taken-into-account approaches for this problem. It is known by many names including Karhunen-Loève Expansion, Eigenpicture, Eigenvector, and a Principal Component. Eigenfaces are mathematically principal components of the distribution of the faces or the eigenvectors of the covariance matrix of the set of images. One of the approaches suggested that "weights" vector for describing a face could be obtained by projecting the face onto a standard face picture(eigenpicture)[4, 5]. Reference [6] used eigenfaces and the technique proposed by Kirby and Sirovich[15] for face detection and recognition. Each face can exactly be represented by the linear combination of eigenfaces. The variation among the faces can be represented by eigenvectors.

Pentland, B. Moghaddam, and T. Starner extended the work done with the concept of eigenfaces and eigenvectors with the modular approach[7]. They considered various face components, such as eye, nose. They created a new eigenspace containing the above eigenfeatures(i.e., eigeneyes, eigennose, and eigenmouth). This system will be less susceptible to appearance changes as compared to the traditional eigenface method. This system achieved a recognition rate of approximately 95 percent on the FERET database of 7,562 images of approximately 3,000 individuals.

Many other algorithms were proposed other than eigenface based algorithms, Artificial Neural Networks(ANN) was one of them. It became popular due to its characteristics of non-linearity. The way through which one constructs the Artificial Neural Network is extremely important for the problem of face recognition. One of the first try to use Artificial Neural Networks for facial recognition was single layer adaptive network called WISARD in which a separate network for each stored individual was constructed[8]. A Convolutional Neural Networks(CNN) based face recognition approach was also proposed in 1997. With the use of a Decision-based Neural Network (DBNN) [9], a Probabilistic Decision-based Neural Network[PDBNN] was also proposed as a solution for the face recognition problem[10].

Edge detection was also considered as an approach for face identification. Many smartphone cameras nowadays use various kind of edge detection techniques for blur effects and better quality of photography. Edge information is insensitive to illumination effects. An edge detection based approach was used for face recognition in [11]. "Support Vector Machines(SVM)" named technique finds the hyperplane that separates the largest possible fraction of points of the same class on the same side while maximizing the distance from either class to the hyperplane given a set of points belonging to two classes. This hyperplane, according to [12], is called Optimal Separating Hyperplane (OSH) which minimizes the chances of misclassifying the examples which are in the training set as well as not in the training set. Reference [13] used Support Vector Machines(SVM) with a binary tree recognition strategy for face recognition problem.

III. IMPLEMENTATION

The process of face recognition starts with loading the dataset and extracting/detecting the face from the images. Face detection is an important part of face recognition as the first step of automatic face recognition. Because human faces are able to convey many different emotions such as happiness, sadness, interest, excitement, confusion, and intrigue, if you pay attention to one's face, you are able to develop an idea for what another person is thinking and what they might do next. For businesses, all of this information is extremely valuable and it can help with understanding how the intended target audience feels about the brand and its communication at all its contact points. Also by detecting a face, we can solve the problem of overfitting where our algorithm starts to memorize and not generalize the data by taking the background of images. we will remove the background and perform different image recognition methods on faces. Face detection has been done by hog face detector which is widely used face detection model. The model is built out of 5 HOG filters – front looking, left looking, right looking, front looking but rotated left, and a front looking but rotated right. Fastest method on CPU. Works very well for frontal and slightly non-frontal faces. It is a light-weight model as compared to the haar cascade face detector and DNN face detector. It works under small occlusion. After completing face detection on the available dataset, data augmentation is done by flipping the image, rotating image left and right and adding noise to the image. Data augmentation was required as we had a lot of parameters to train and to achieve high accuracy number of training samples had to be large. After data augmentation, various classifiers and models are used and their performance is compared.

1. K – Nearest Neighbour Classifier(KNN)

   K nearest neighbors is a simple algorithm that stores all available cases and classifies new cases based on a similarity measure (e.g., distance functions). Simply

put, the K–NN algorithm classifies unknown data points by finding the most common class among the K–closest examples. Each data point in the K-closest examples casts a vote and the category with the most votes wins

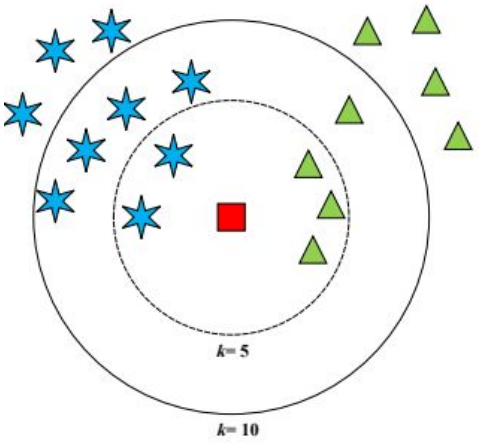

Figure 1. The principle diagram of the K-NN classification algorithm.

As shown in Figure, the red square represents the sample to be classified. It needs to be classified into a blue star or green triangle. It is obvious that it is classified to green triangle while k is set to 5 (smaller circle) since the probability of classifying it into a green triangle is 60 %, which is higher than that of classifying it to blue star (40 %). While k is set to 10 (bigger circle), the red square is classified into a blue star, since the probability of classifying it into a blue star is 60 %, higher than the probability of classifying it into a green triangle (40 %). In order to apply the k-nearest Neighbour classification, we need to define a distance metric or similarity function. Common choices include the Euclidean distance: $d(p, q) = \sqrt{\sum_{i=1}^{N} (qi - pi)^2}$ where N is the number of variables, and qi and pi are the values of the ith variable at points p and q respectively. Other distance metrics/similarity functions can be used depending on the type of data (the chi-squared distance is often used for distributions (histograms)). In our case, we have been using the Euclidean distance to compare images for similarity

2. Convolutional Neural Network(CNN)

The Convolutional Neural Networks (CNN) are very similar to ordinary Neural Networks. They are made up of neurons that have learnable weights and biases. Each neuron receives some inputs, performs a dot product and optionally follows it with a non-linearity. The whole network still expresses a single differentiable score function: from the raw image pixels on one end to class scores at the other. They still have a loss function (e.g. SVM/ Softmax) on the last (fully-connected) layer and all the tips/tricks we have developed for learning regular Neural Networks still apply

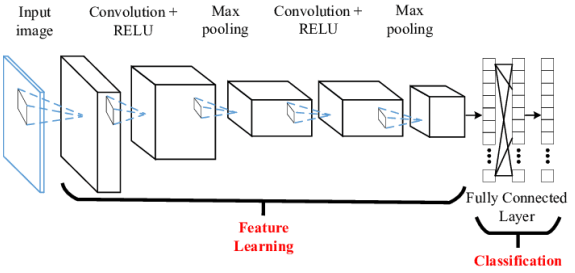

Figure 2. The Convolutional Neural Network (CNN).

The CNN consists of multiple layers. Each layer takes a multi-dimensional array of numbers as input and produces another multidimensional array of numbers as output (which then becomes the input of the next layer). When classifying images, the input to the first layer is the input image (32 × 32), while the output of the final layer is a set of likelihoods of the different categories (i.e., 1 × 1 × 10 numbers if there are 10 categories). A simple CNN is a sequence of layers, and every layer of a CNN transforms one volume of activations to another through a differentiable function. We have used three main types of layers to build CNN architectures: Convolution (CONV) Layer, Pooling Layer, and Fully-Connected Layer (exactly as seen in regular Neural Networks). We have stacked these layers to form a full CNN architecture:

• INPUT [32 × 32] holds the raw pixel values of the image, in this case, an image of width 32, height 32.

• CONV layer computes the output of neurons that are connected to local regions in the input, each computing a dot product between their weights and a small region they are connected to in the input volume. This may result in volume such as [32 × 32 × 12] if we decided to use 12 filters

• POOL layer performs a down-sampling operation along the spatial dimensions (width, height), resulting in volume such as [16 × 16 × 12].

Pooling layer downsamples the volume spatially, independently in each depth slice of the input volume. In this example, the input volume of size [224 × 224 × 64] is pooled with filter size 2, stride 2 into output volume of size [112 × 112 × 64]. Notice that the volume depth is preserved. The most common downsampling operation is the max, giving rise to max pooling. That is, each max is taken over 4

numbers (little 2 × 2 square). In this way, CNN transforms the original image layer by layer from the original pixel values to the final class scores. Note that some layers contain parameters and others don't. In particular, the CONV/FC layers perform transformations that are a function of not only the activations in the input volume but of the parameters (the weights and biases of the neurons) as well. On the other hand, the RELU/POOL layers will implement a fixed function. The parameters in the CONV/FC layers have been trained with gradient descent so that the class scores that the CNN computes are consistent with the labels in the training set for each image. A CNN architecture is in the simplest case a list of Layers that transform the image volume into an output volume (e.g. holding the class scores):

• There are a few distinct types of Layers (e.g. CONV/FC/RELU/POOL are by far the most popular).

• Each Layer accepts an input 3D volume and transforms it into an output 3D volume through a differentiable function.

• Each Layer may (CONV/FC) or may not have (RELU/POOL) parameters.

• Each Layer may (CONV/FC/POOL) or may not have (RELU) additional hyperparameters.

3. Principal Component Analysis

Principal components analysis (PCA) is a technique that can be used to simplify a dataset. It is a linear transformation that chooses a new coordinate system for the data set such that the greatest variance by any projection of set comes to lie on axis (first principal component), the second on the second axis, and so on. PCA can be used for reducing dimensionality in a dataset while retaining those characteristics of the dataset that contribute most to its variance, by keeping lower-order principal components and ignoring higher-order ones. The idea is that such low-order components often contain the "most important" aspects of the data. The task of facial recognition is discriminating input signals (image data) into several classes (persons). The input signals are highly noisy (e.g. the noise is caused by differing lighting conditions, pose, etc.), yet the input images are not completely random and in spite of their differences there are patterns which occur in an input signal. Such patterns, which can be observed in all signals could be - in the domain of facial recognition - the presence of some objects (eyes, nose, mouth) in any face as well as relative distances between these objects. These characteristic features are called eigenfaces in the facial recognition domain (or principal components generally). They can be extracted out of original image data by means of the mathematical tool called Principal Component Analysis (PCA). By means of PCA, one can transform each original image of the training set into a corresponding eigenface. original image. If one uses all the eigenfaces extracted from original images, one can reconstruct the original images from the eigenfaces exactly. But one can also use only a part of the eigenfaces. Then the approximation of the image. However, losses due to omitting some of the eigenfaces can be minimized. This happens by choosing only the most important features (eigenfaces). The omission of eigenfaces is necessary due to the scarcity. Thus the purpose of PCA reduces the large dimensionality of space (observed variables) to the smaller intrinsic dimensionality of feature space (independent variables), which are needed to describe the data economically. This is the case when there is a strong correlation between observed variables. To generate a set of eigenfaces, a large set of digitized images of human faces, taken under the same lighting conditions, are normalized to line up the eyes and mouths. They are then all resampled at the same pixel resolution (say m×n), and then treated as mn-dimensional vectors whose components are the values of their pixels. The eigenvectors of the covariance matrix of the statistical distribution of face image vectors are then extracted. Since the eigenvectors belonging to the same vector space as face images, they can be viewed as if they were m×n images: hence the name eigenfaces. Other eigenfaces are hard to categorize and look rather strange. When properly weighted, eigenfaces can be summed together to create an approximate gray-scale rendering of a human face. Remarkably few eigenvector terms are needed to give a fair likeness of most people's faces, so eigenfaces provide a means of applying data compression to faces for identification purposes. It is possible not only to extract the face from eigenfaces given a set of weights but also to go the opposite way. This opposite way would be to extract the weights from eigenfaces and the face to be recognized. These weights tell nothing less, as the amount by which the face in question differs from "typical" faces represented by the eigenfaces. Therefore, using these weights one can determine two important things:
Determine if the image in question is a face at all. In the case, the weights of the image differ too much from the weights of face images (i.e. images, from which we know for sure that they are faces) the image probably is not a face.

Similar faces (images) possess similar features (eigenfaces) to similar degrees (weights). If one extracts weights from all the images available, the images could be grouped to clusters. That is, all images having similar weights are likely to be similar faces.

4. Support Vector Machines(SVM)

Support Vector Machines(SVM) is a discriminative classifier formally defined by a separating hyperplane. In other words, given labeled training data (supervised learning), the algorithm outputs an optimal hyperplane which categorizes new examples. A Support Vector Machine (SVM) performs classification by constructing an N-dimensional hyperplane that optimally separates the data into two categories. SVM models are closely related to neural networks. In fact, an SVM model using a sigmoid kernel function is equivalent to a two-layer, perceptron neural network. Using a kernel function, SVM's are an alternative training method for polynomial, radial basis function, and multi-layer perceptron classifiers in which the weights of the network are found by solving a quadratic programming problem with linear constraints, rather than by solving a non-convex, unconstrained minimization problem as in standard neural network training. A set of features that describes one case (i.e., a row of predictor values) is called a vector. So the goal of SVM modeling is to find the optimal hyperplane that separates clusters of the vector in such a way that cases with one category of the target variable are on one side of the plane and cases with the other category are on the other side of the plane. The vectors near the hyperplane are the support vectors.

IV. Experimental Results

Here, various kind of figures show the detailed results of face recognition and accuracy step by step.

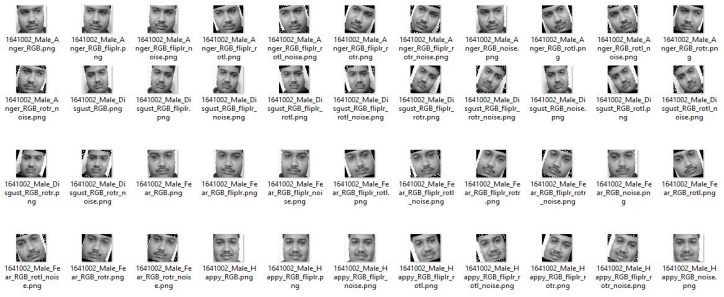

Figure 3. Augmented Data

```
In [42]: plt.imshow(X[0],cmap = "gray")
         #len(X)

Out[42]: <matplotlib.image.AxesImage at 0x244bce61b3
```

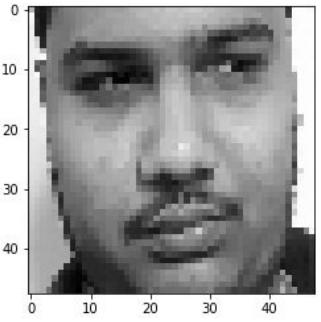

Figure 4. Detected Face Using Hog Face Detector

```
In [36]: eigenfaces = pca.components_.reshape((n_components, 50, 50))
         #plt.imshow(eigenfaces[0],cmap = "gray")
         plt.imshow(eigenfaces[1],cmap = "gray")

Out[36]: <matplotlib.image.AxesImage at 0x2441a057c88>
```

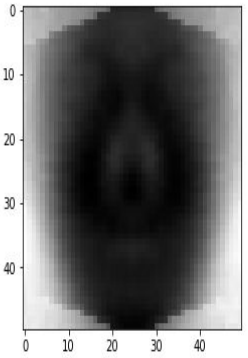

Figure 5. Face Recognition using Eigenface-PCA

```
Found 499 correct labels
```

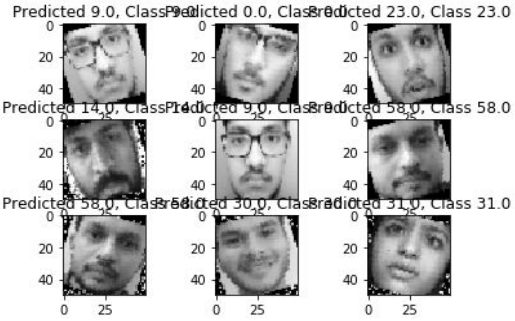

Figure 6. Recognized faces using Eigenfaces

```
Fitting the classifier to the training set
Iteration 1, loss = 3.48506415
Validation score: 0.427646
Iteration 2, loss = 1.96835697
Validation score: 0.760259
Iteration 3, loss = 0.99804072
Validation score: 0.850972
Iteration 4, loss = 0.52243670
Validation score: 0.907127
Iteration 5, loss = 0.30470809
Validation score: 0.935205
Iteration 6, loss = 0.19396099
Validation score: 0.961123
Iteration 7, loss = 0.13228477
Validation score: 0.963283
Iteration 8, loss = 0.09439031
Validation score: 0.971922
Iteration 9, loss = 0.07056873
Validation score: 0.971922
Iteration 10, loss = 0.05498233
Validation score: 0.974082
Iteration 11, loss = 0.04424470
Validation score: 0.974082
Iteration 12, loss = 0.03645268
Validation score: 0.971922
Iteration 13, loss = 0.03058131
Validation score: 0.974082
Validation score did not improve more than tol=0.000100 for two consecutive epochs. Stopping.
```

Figure 7. PCA Model Fitting for Number of Iterations

```
accuracy = ( (100 * count) / len(pred_Y))
print(accuracy)
```

```
83.46303501945525
```

```
In [18]: correct = np.where(pred_Y==test_Y)[0]
         #print ("Found %d correct labels" % len(correct))
         print ("Found %d correct labels" % len(correct))
         for i, correct in enumerate(correct[40:49]):
             plt.subplot(3,3,i+1)
             plt.imshow(test_X[correct].reshape(50,50), cmap='gray', interp
             plt.title("Predicted {}, Class {}".format(pred_Y[correct], tes
             #plt.tight_layout()
         #for i, correct in enumerate(correct[:9]):
             #plt.subplot(3,3,i+1)
             #plt.imshow(cv2.resize(test_X[correct],(50,50)), cmap='gray',
             #plt.title("Predicted {}, Class {}".format(pred_Y[correct], te
             #plt.tight_layout()
```

```
Found 429 correct labels
```

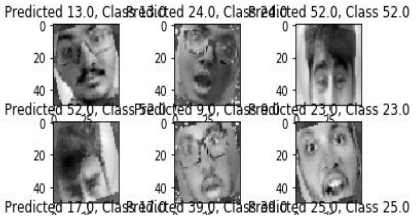

Figure 8. Face Recognition using KNN and Accuracy calculation.

```
accuracy = ( (100 * count) / len(pred_Y))
print(accuracy)
```

```
99.0272373540856
```

```
In [20]: correct = np.where(pred_Y==test_Y)[0]
         print ("Found %d correct labels" % len(correct))
         #print ("Found %d correct labels" % len(correct))
         for i, correct in enumerate(correct[30:39]):
             plt.subplot(3,3,i+1)
             plt.imshow(test_X[correct].reshape(50,50), cmap='
             plt.title("Predicted {}, Class {}".format(pred_Y[
             #plt.tight_layout()
```

```
Found 509 correct labels
```

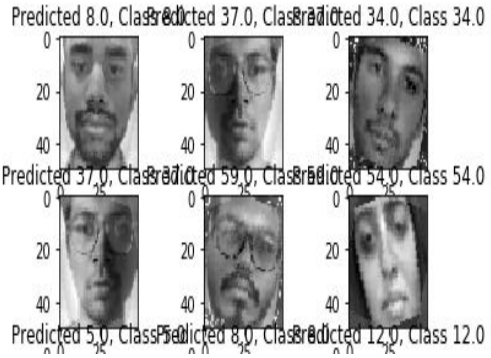

Figure 9. Face Recognition using SVM

```
In [23]: print('Test loss:', test_eval[0])
         print('Test accuracy:', test_eval[1])
```

```
Test loss: 0.14373024576253002
Test accuracy: 0.9688715953307393
```

Figure 10. Face Recognition accuracy using CNN

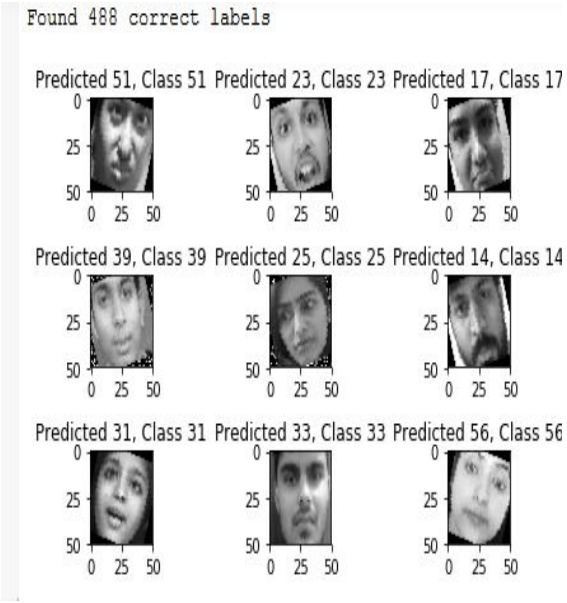

Figure 10. CNN Face Recognition : correct labels

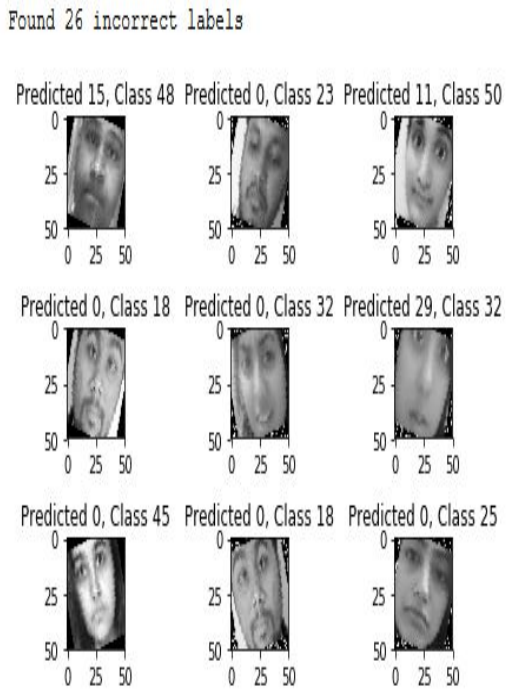

Figure 11. CNN Face Recognition : incorrect labels

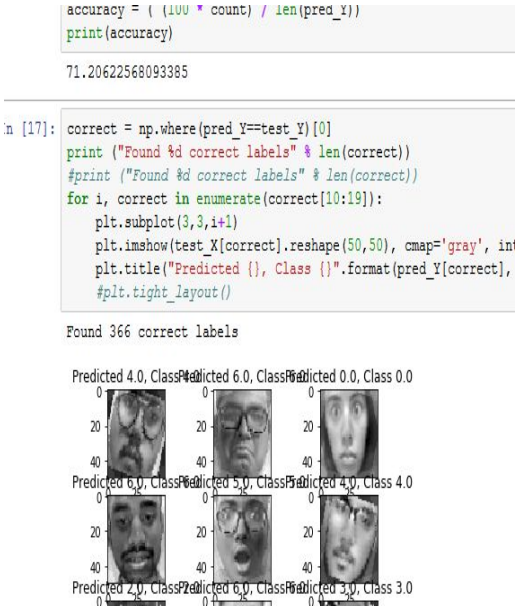

Figure 12. Emotion Recognition : SVM

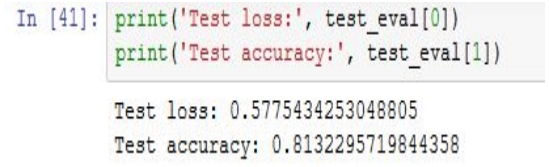

Figure 13. The Emotion Recognition accuracy of CNN

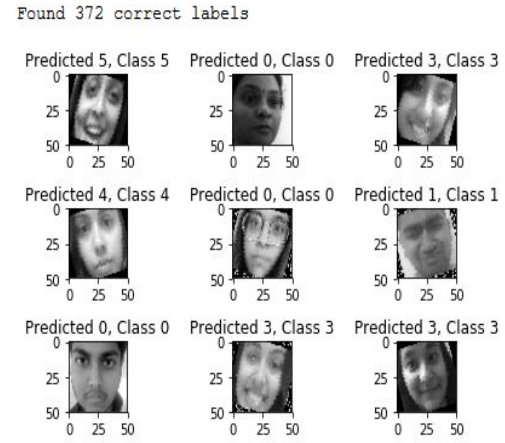

Figure 14. CNN Emotion Recognition : correct labels

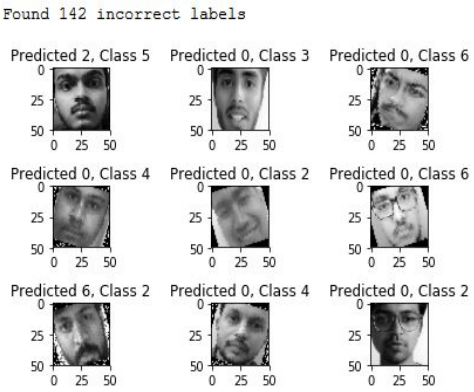

Figure 15. CNN Emotion Recognition : incorrect labels

	precision	recall	f1-score	support
Class 0	0.43	0.98	0.60	81
Class 1	0.92	0.69	0.79	64
Class 2	0.91	0.60	0.72	82
Class 3	0.97	0.85	0.90	71
Class 4	0.83	0.59	0.69	73
Class 5	0.92	0.77	0.84	60
Class 6	0.80	0.61	0.69	83
avg / total	0.81	0.72	0.74	514

Figure 16. CNN Emotion Recognition Classification Report

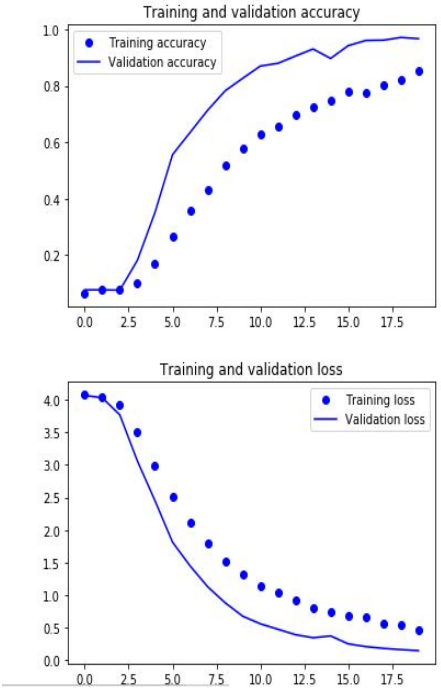

Figure 17. Training and validation accuracy and loss for Face Recognition using CNN

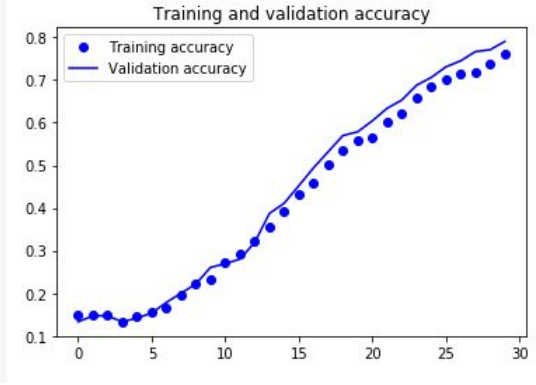

Figure 18. Training and validation accuracy for Emotion Recognition using CNN

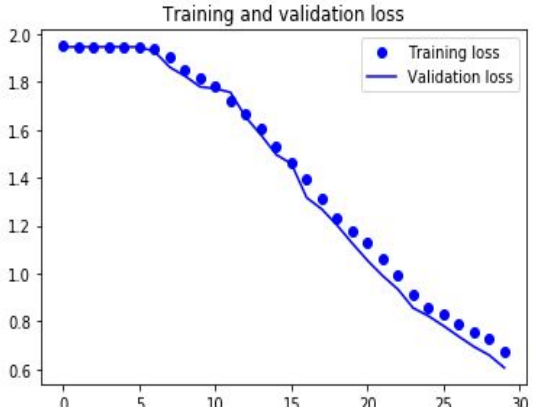

Figure 19. Training and validation loss for Emotion Recognition using CNN

- Performance of Different Methods On Dataset:-

TABLE I.
PERFORMANCE OF DIFFERENT METHODS ON DATASET FOR FACE RECOGNITION

Method	Accuracy
KNN	83.46%
SVM	99.027%
CNN	96.88%
PCA	97.407%

TABLE II.
PERFORMANCE OF DIFFERENT METHODS ON DATASET FOR EMOTION RECOGNITION

Method	Accuracy
SVM	71.206%
CNN	81.32%

V. Conclusion

Various different models like SVM, KNN, CNN, and PCA were tested on the dataset obtained by capturing frontal faces of all the classmates.In face recognition, it was observed that PCA performs the best because PCA was used to extract features from the images of the dataset and then the eigenfaces obtained were trained by an MLP(Perceptron) Classifier using a neural network.Also, the training and validation graph obtained from CNN shows that accuracy increases as the number of epochs increases. As well as the loss decreases as a number of epochs increases. The same can be observed from the graphs obtained during emotion recognition from CNN. In emotion recognition as the number of parameters increased CNN performed the best and Dropout layer used in CNN also decreased the chances of overfitting in the model were reduced.For future works Gabor filter, algorithm, etc can be used on the models and performance can be measured.

VI. References

[1] P.N. Belhumeur, J.P. Hespanha, and D.J. Kriegman, “Eigenfaces vs. Fisherfaces: Recognition Using Class Specific Linear Projection,” European Conf. Computer Vision, 1996, pp. 45-58.

[2] Francis Galton, “Personal identification and description,” In Nature, pp. 173-177, June 21, 1888.

[3] T. Fromherz, P. Stucki, M. Bichsel, “A survey of face recognition,” MML Technical Report, No 97.01, Dept. of Computer Science, University of Zurich, Zurich, 1997.

[4] L. Sirovich and M. Kirby, “Low-Dimensional procedure for the characterisation of human faces,” J. Optical Soc. of Am., vol. 4, pp. 519-524, 1987.

[5] M. Kirby and L. Sirovich, “Application of the Karhunen- Loève procedure for the characterisation of human faces,” IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 12, pp. 831-835, Dec. 1990.

[6] M. Turk and A. Pentland, “Eigenfaces for recognition,” J. Cognitive Neuroscience, vol. 3, pp. 71-86, 1991.

[7] A. Pentland, B. Moghaddam, and T. Starner, “View-Based and modular eigenspaces for face recognition,” Proc. IEEE CS Conf. Computer Vision and Pattern Recognition, pp. 84-91, 1994.

[8] T.J. Stonham, “Practical face recognition and verification with WISARD,” Aspects of Face Processing, pp. 426-441, 1984.

[9] S.H. Lin, S.Y. Kung, and L.J. Lin, “Face recognition/detection by probabilistic decision-based neural network,” IEEE Trans. Neural Networks, vol. 8, pp. 114-132, 1997.

[10] S.Y. Kung and J.S. Taur, “Decision-Based neural networks with signal/image classification applications,” IEEE Trans. Neural Networks, vol. 6, pp. 170-181, 1995.

[11] B. Takács, “Comparing face images using the modified hausdorff distance,” Pattern Recognition, vol. 31, pp. 1873-1881, 1998.

[12] V.N. Vapnik, “The nature of statistical learning theory,” New York: Springverlag, 1995.

[13] G. Guo, S.Z. Li, and K. Chan, “Face recognition by support vector machines,” In proc. IEEE International Conference on Automatic Face and Gesture Recognition, pp. 196-201, 2000.

[14] Viola, Paul, and Michael Jones. "Rapid object detection using a boosted cascade of simple features." CVPR (1) 1 (2001): 511-518.

[15] L. Sirovich and M. Kirby, “Low-Dimensional Procedure for the characterization of human faces,” J. Optical Soc. of Am., vol. 4, pp. 519-524, 1987.

[16] P.J. Phillips, “Support vector machines applied to face recognition,” Processing system 11, 1999.

[17] Huang J., X. Shao, and H. Wechsler, "Face pose discrimination using support vector machines,” 14th International Conference on Pattern Recognition, (ICPR), Brisbane, Queensland, Aus, 1998.

[18] W. Zhao and R. Chellappa, “SFS based view synthesis for robust face recognition,” Proc. Int’l Conf. Automatic Face and Gesture Recognition, pp. 285-292, 2000.

[19] T. Vetter and V. Blanz, “Estimating coloured 3D face models from fingle images: An example based approach,” Proc. Conf. Computer Vision (ECCV ’98), vol. II, 1998.

[20] I.J. Cox, J. Ghosn, and P.N. Yianios, “Feature-Based face recognition using mixture-distance,” Computer Vision and Pattern Recognition, 1996.

[21] S. Tamura, H. Kawa, and H. Mitsumoto, “Male/Female identification from 8_6 very low resolution face images by neural network,” Pattern Recognition, vol. 29, pp. 331-335, 1996.

[22] K.K. Sung and T. Poggio, “Learning human face detection in cluttered scenes,” Computer Analysis of Image and patterns, pp. 432-439, 1995.

[23] G.J. Edwards, T.F. Cootes, and C.J. Taylor, “Face recognition using active appearance models,” In ECCV, 1998.

[24] John A. Black, M. Gargesha, K. Kahol, P. Kuchi, Sethuraman Panchanathan,” A Framework for performance evaluation of face International Journal of Signal Processing Volume 2 Number 2 recognition algorithms,” in Proceedings of the International Conference on ITCOM, Internet Multimedia Systems II, 2002.

[25] P.J. Phillips, P. Grother, R.J. Michaels, D.M. Blackburn, E. Tabassi, and M. Bone, “Face recognition Vendor Test 2002: Evaluation Report,” NISTIR 6965, NAT. Inst. Of Standards and Technology 2003.

[26] Y. Kaya and K. Kobayashi, “A basic study on human face recognition,” Frontiers of Pattern Recognition, S. Watanabe, ed., pp. 265, 1972.