CS419M - Intro to ML
Project Report

# Comparative Analysis of Face Recognition techniques using Deep Learning and Neural Networks

Team Tensor

| | |
|---|---|
| Atharva Raut | 190070050 |
| Akash Chodankar | 190100009 |
| Akshat Mehta | 190100011 |

Link to project on GitHub: [https://github.com/Team-Tensor/Face-Recognition](https://github.com/Team-Tensor/Face-Recognition)

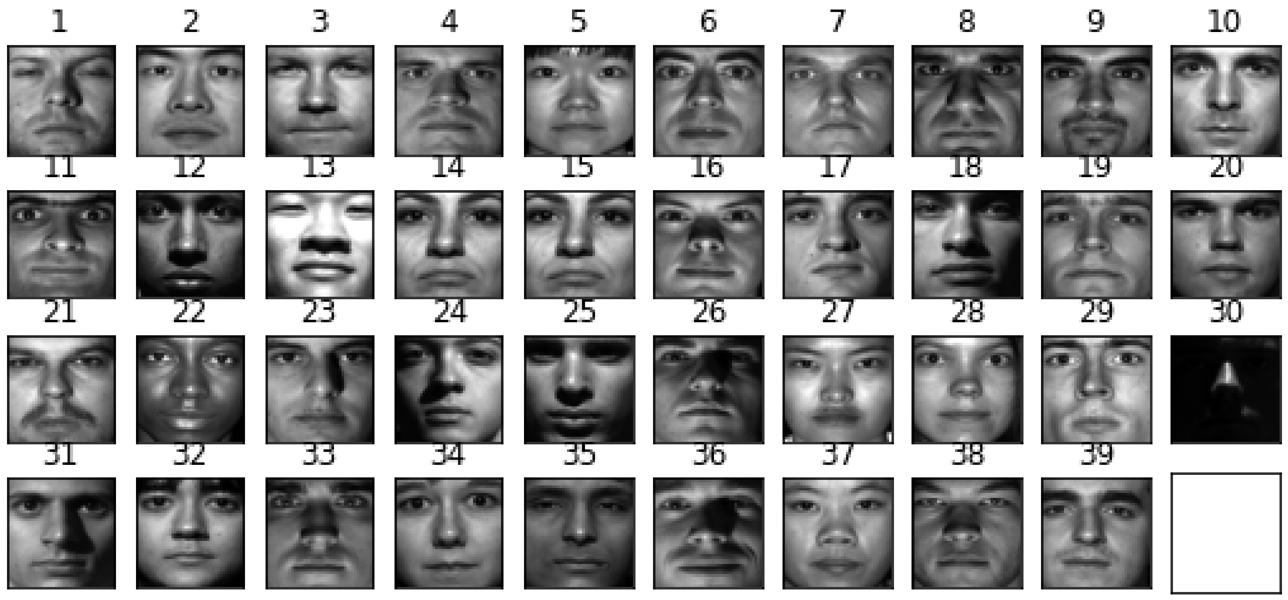
Instructor: Prof. Abir De, Department of Computer Science and Engineering

# Table of Contents

## 1. Project Overview

The project aims at exploring the different methods to build an accurate and robust face recognition model using Deep Learning and Artificial Neural Networks. The benchmarking tests were performed on 'The Extended Yale Face Database B' with cropped faces, containing a total of 2470 grayscale images, of 39 subjects under varying light conditions. The features were then extracted using common methods including raw pixel intensities, Principal Component Analysis and Convolutional Neural Networks. The extracted features were then used to train machine learning models like Support Vector Machine, Logistic Regression and Neural Networks. The results for all methods were analysed and a comparison of precision, recall and f1-score was done to select the best model.

The figure below shows an image corresponding to each of the subjects from the dataset under consideration



## 2. Problem Statement

Compare techniques for face recognition using different machine learning models and explore methods for feature extraction to build a classification model for accurately modelling the standardized dataset.

## 3. Literature Review

The first consideration before training any model is to select the format of input and output. The raw images were grayscale images of size 192x168, resized to 160x160 and unolled into 1D array for preprocessing.

For a naive approach, using pixel intensity values, the feature space has a size 25600. The major issue with this approach is the large dimensionality of the feature space, which also fails to capture the major content of features and may sometimes even cause overfitting on the training set. To overcome this challenge, a widely used method is Principal Component Analysis (PCA) that helps reduce the dimensionality, while capturing the maximum useful data from the features.

The paper 'Face Recognition using Principle Component Analysis' by Kyungnam Kim, was used to gain insights into PCA. The excerpt below highlights the key ideas from the paper.

### 3.1 Principal Component Analysis (PCA)

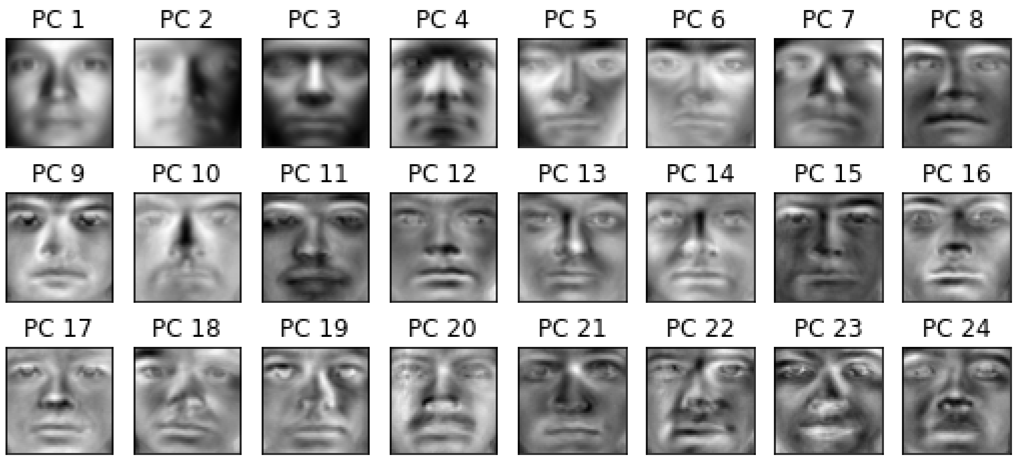
Process of computing the principal components and using them to perform a change of basis on the data, sometimes using only the first few principal components and ignoring the rest.

It is a method to practice feature extraction. When the input data to an algorithm is too large to be processed and it is suspected to be redundant then it can be reduced to a set of useful features.

Advantages of PCA are:

- Reduced number of features
- Less memory requirement
- Reduced overfitting
- Faster training times
- Preserve maximum variance components

Because PCA is a classical technique which can do something in the linear domain, applications having linear models are suitable, such as signal processing, image processing, system and control theory, communications, etc.



Visualization of the first few PCA decomposed components

A 2-D image can be represented as a 1-D vector by joining each row (or column) into a long thin vector. Let's suppose we have M vectors of size N (= rows of image × columns of image) representing a set of sampled images. $p_j$'s represent the pixel values

$$x_i = [p_1 \dots p_N]^T \qquad , i = 1, \dots, M$$

The images are mean centered by subtracting the mean image from each image vector. Let $m$ represent the mean image.

$$m = \frac{1}{M}\sum_{i=1}^{M} x_i$$

And let $w_i$'s be defined as mean centered image

$$w_i = x_i - m$$

Our goal is to find a set of $e_i$'s which have the largest possible projection onto each of the $w_i$'s. We wish to find a set of M orthonormal vectors $e_i$ for which the quantity

$$\lambda i = \frac{1}{M} \sum_{i=1}^{M} (e^{T}{}_{i}\, w_n)^2$$

is maximized with the orthonormality constraint

$$e^{T}{}_{l} e_k = \delta_{lk}$$

It has been shown that the $e_i$'s and $\lambda_i$'s are given by the eigenvectors and eigenvalues of the covariance matrix $C = WW^T$

W is a matrix composed of the column vectors wi placed side by side. The size of C is N×N which could be large. It is impractical to directly solve for the eigenvectors of C. A common theorem in linear algebra states that the vectors $e_i$ and scalars $\lambda_i$ can be obtained by solving for the eigenvectors and eigenvalues of the M×M matrix $W^TW$. Let di and μi be the eigenvectors and eigenvalues of $W^TW$, respectively.

$$W^TWd_i = \mu_i d_i$$

By multiplying left to both sides by W

$$WW^T (W\, d_i) = \mu_i (W\, d_i)$$

which means that the first M − 1 eigenvectors ei and eigenvalues $\lambda_i$ of $WW^T$ are given by $Wd_i$ and $\mu_i$ , respectively. $Wd_i$ needs to be normalized in order to be equal to $e_i$ . Since we only sum up a finite number of image vectors, M, the rank of the covariance matrix cannot exceed M − 1 (The −1 comes from the subtraction of the mean vector m).

The eigenvectors corresponding to nonzero eigenvalues of the covariance matrix produce an orthonormal basis for the subspace within which most image data can be represented with a small amount of error. The eigenvectors are sorted from high to low according to their corresponding eigenvalues. The eigenvector associated with the largest eigenvalue is one that reflects the greatest variance in the image.

That is, the smallest eigenvalue is associated with the eigenvector that finds the least variance. A facial image can be projected onto M' (<<M) dimensions by computing

$$\Omega= [v_1 v_2 \ldots v_{M'}]^T$$

where $v_i = e^T_i w_i$ . $v_i$ is the ith coordinate of the facial image in the new space, which came to be the principal component. The vectors ei are also images, so called, eigenimages, or eigenfaces in our case. They can be viewed as images and indeed look like faces. So, Ω describes the contribution of each eigenface in representing the facial image by treating the eigenfaces as a basis set for facial images. The simplest method for determining which face class provides the best description of an input facial image is to find the face class k that minimizes the Euclidean distance.

$$\epsilon_k = ||\Omega - \Omega_k||$$

where $\Omega_k$ is a vector describing the k th face class. If $\epsilon_k$ is less than some predefined threshold $\theta_e$ , a face is classified as belonging to the class k.

## 4. Proposed Methods

As mentioned earlier, the first step before training a model is to decide what we expect from it. The inputs given to the model, i.e. features were selected from among the three choices:

1. Raw Pixel Intensity Values
2. Principal Component decomposed features
3. Features extracted using pre-trained model VGG16

The base images were 192x168 pixels, but were resized to 160x160 pixels to use with standardized models requiring square images. To reduce the time required to extract pixel values from the images in .pgm format, the 160x160 sized images were unrolled and stored to the csv file for subsequent usage.

For PCA, the library function from scikit-learn is to be used for transforming the features to the eigenspace. The number of components can be selected to capture the required cumulative variance from the components. A very low value would cause underfitting, whereas a high value might lead to overfitting.

The VGG16 model would be used to extract features using the concept of transfer learning and only the weights of the final layer are trained. Similar to the raw images, to save time, the extracted features using this model were stored into a csv file for future use and need not be processed at every run.

The models used for training include:

1. Logistic Regression
2. Support Vector Machine
3. Feed-forward Neural Network
4. Convolutional Neural Network

For Logistic Regression and SVM, the tests are done for inputs from all three feature sources.

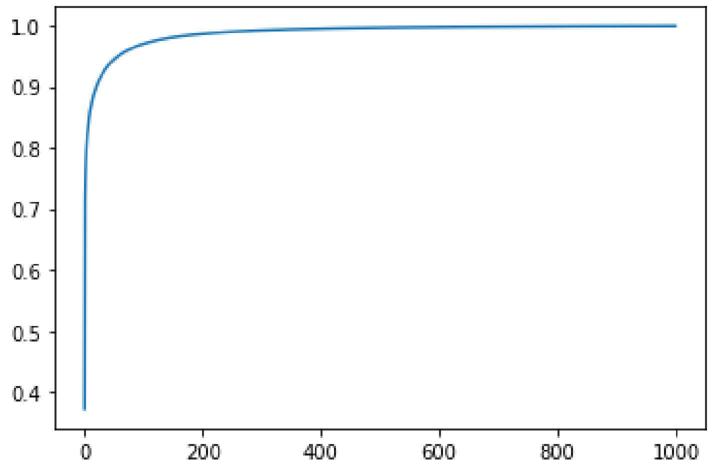
The PCA decomposed features are used for FFNN, whereas raw images are used for CNN.

## 5. Models

### 5.1 Logistic regression

Normalized PCA features gave the best performance for the logistic regression classifier, trained with a maximum of 1000 iterations.

The number of components from PCA decomposition to be taken as features is to be selected based on the required amount of variance in the feature space. The graph below shows the cumulative variance captured by the components against the total number of components.



It can be seen that 1000 components are able to capture almost the entire variance in the dataset. Taking any more features than this doesn't improve performance and is redundant. It is clear that this is a huge reduction in dimensionality as compared to the actual pixel values of the image (25600 features).

### 5.2 Support Vector Machine

The inputs were tested with raw images, PCA decomposed features and VGG16 extracted features. The best results were obtained with PCA.

The hyperparameters were tuned using GridSearchCV and optimum values were selected as follow:

- Kernel: Radial Basis Function (RBF)
- C: 1000 (penalty for misclassified points)
- $\gamma$: 0.01 (inversely varies with distance of influence)

## 5.3 Feed Forward Neural Network

A feed forward neural network with 2 hidden layers with watch layer having 128 neurons was used. The nonlinearity was taken care of by the ReLU activation function which was attached after every fully connected layer of the neural network. The input layer had 1000 features which were selected out of 25600 features using PCA. A batch normalization layer was attached before the input layer. The model was trained on the entire training dataset (i.e without batches), was evaluated upon using the cross entropy loss function and optimized using SGD with a learning rate of 0.003.

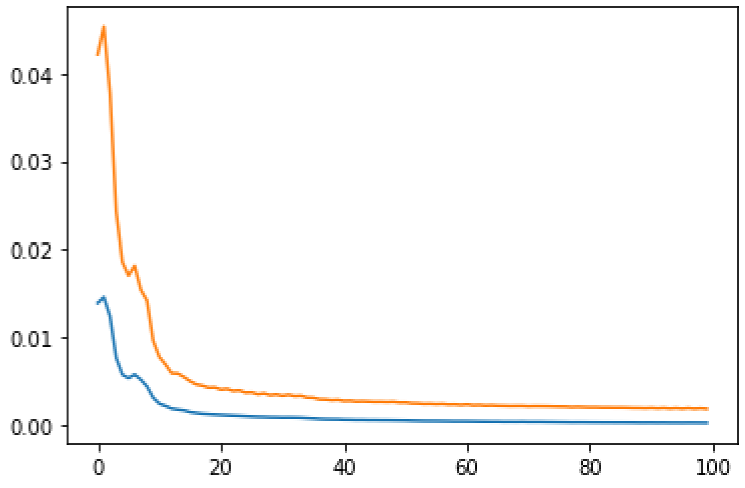
Summary:-

Batch normalization(1000) -> input layer(1000) -> hidden 1(128) -> hidden 2(128) -> output(39) -> softmax

Optimizer:- SGD with α=0.003

Evaluation criteria:- cross entropy loss

Training curve:



## 5.4 Convolutional Neural Network

The CNN architecture used here was developed using the CNN_S described in the paper cited below [1]. The network takes in a grayscale image in the form of a tensor with (m,1,160,160) dimensions, where m stands for the number of training examples. The architecture consists of a series convolution and max pooling operations which is repeated 3 times before passing the data into 3 fully connected layers and in the end through a softmax layer. The non-linearity is captured using the ReLU activation function. The training was conducted to adhere to the principal of early stopping.

The specification of the layers are as follows(in the sequence in which they appear):

1. Conv1 : - (1,12,5)
2. Max Pool1 : -(2)
3. Conv2 : -(12,24,4) , stride=2
4. Max Pool2 : -(2)
5. Conv3 : -(24,32,3), stride=2
6. Max Pool3 : -(2)
7. Fully connected 1(fc1): - (512,120)
8. Fully connected 2(fc2): - (120,84)
9. Fully connected 3(fc3): - (84,39)
10. Softmax

Where (a,b,c) in the convolution layers refers to the (input channels, output channels, kernel size)

The initial CNN architecture which was tried was a similar model without striding. Also only a single fully connected layer was used. The model was expected to capture the facial features well as was comprehended from the research paper[1].

But we faced the following issues: -

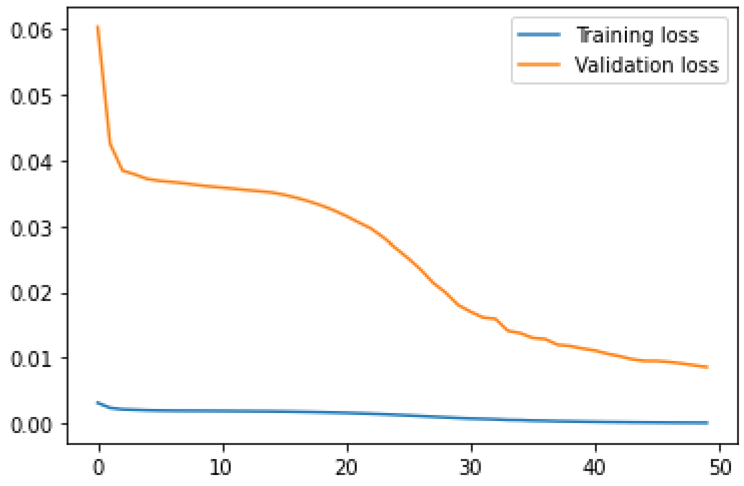
1. Inability of the model to train on the GPU without batches.
2. The CPU runtime crashed after a few iterations due to the vast number of features expanded at a single instance as batches were not used.
3. Sudden decrease of the dimensions from 512 to 39 might have resulted in the loss of some important features.

In response to these, the following improvisations were made: -

1. Strides were added to 2 convolutional layers in order to decrease the number of features in every layer at a much faster rate.
2. The convolved data was allowed to pass through 2 additional fully connected layers, gradually decreasing its size.

After the above mentioned changes the network was able work on the entire training and validation data without batches.

Training curve:



## 6. Results

The table below shows a summary of the result across different performance metrics and training time. The results are taken as the best scores obtained on the above-mentioned dataset with a train-test split of 3:1

| **Model (with features)** | **Training Time** | **Precision** | **Recall** | **F1-score** |
|---|---|---|---|---|
| Logistic Regression (with PCA) | 0.75s | 0.98 | 0.97 | 0.97 |
| Logistic Regression (with VGG16) | 29.26s | 0.89 | 0.88 | 0.87 |
| SVM (with PCA) | 9.49s | 0.95 | 0.94 | 0.95 |
| SVM (with VGG16) | 163.97s | 0.98 | 0.56 | 0.68 |
| FFNN (with PCA) | 1.59s | 0.84 | 0.81 | 0.82 |
| CNN (with raw images) | 10.24s | 0.84 | 0.84 | 0.83 |

The performance of VGG16 for all models is poor as compared to the standardized benchmark on ImageNet dataset. This could be attributed to the fact that it is primarily meant for Large-scale image recognition (on datasets containing 1M+ images), whereas our dataset is relatively very small. Thus, PCA served as a better alternative for Face Recognition over smaller datasets.

Among different optimizing models, Logistic Regression (for multi-class classification) gave the best results, followed closely by SVM.

The model for CNN showed improvement over different variations of the intermediate convolution and pooling layers. The model was tuned to the best possible configuration in the limited timeframe and could be improved by tweaking the number of layers, convolution layer parameters(kernel size, stride) etc.

## 7. References

- When Face Recognition Meets with Deep Learning: an Evaluation of Convolutional Neural Networks for Face Recognition by Guosheng Hu, Yongxin Yang, Dong Yi, Josef Kittler, William Christmas, Stan Z. Li, Timothy Hospedales. [1]
- Very Deep Convolutional Networks for Large-Scale Image Recognition by Karen Simonyan, Andrew Zisserman. [2]
- Face Recognition using Principle Component Analysis by Kyungnam Kim. [3]