Dissertation on

Face Recognition Approaches using Different Machine Learning and Deep Learning Algorithms

Thesis submitted towards partial fulfillments of the requirements for the degree of

Masters of Technology in IT (Courseware Engineering)

Submitted by

CHAUDHURI MD TAUSIF

EXAMINATION ROLL NO.:**M4CWE21023**

UNIVERSITY REGISTRATION NO.:**150125** of **2019-2020**

Under the guidance of

Dr. Saswati Mukherjee
School of Education Technology

Course affiliated to

Faculty of Engineering and Technology
Jadavpur University
Kolkata-700032
India
2021

MTech IT (Courseware Engineering)
Course affiliated to
Faculty of Engineering and Technology
Jadavpur University
Kolkata, India

Certificate of Recommendation

This is to certify that the thesis entitled **"Face Recognition Approaches Different Machine Learning And Deep Learning Algorithms"** is a bona fide work carried out by **Chaudhuri Md Tausif** under our supervision and guidance for partial fulfilment of the requirements for the degree of **Masters of Technology in IT (Courseware Engineering)** in School of Education Technology, during the academic session 2019-2021.

Signed by,

……………………………………………………

SUPERVISOR

Dr. Saswati Mukherjee,
Asst. Professor,
School Of Education Technology
Jadavpur University, Kolkata – 700032

Countersigned by,

………………………	**………………………**
DIRECTOR	**DEAN**
School of Education Technology, Jadavpur University, Kolkata-700032	FISLM, Jadavpur University, Kolkata-700032

MTech IT (Courseware Engineering)
Course affiliated to
Faculty of Engineering and Technology
Jadavpur University
Kolkata, India

CERTIFICATE OF APPROVAL**

This forgoing thesis is hereby approved by as a creditable study of an engineering subject carried out and presented in a manner satisfactory to warranty its acceptance as a prerequisite to the degree for which it has been submitted. It is understood that by this approval the undersigned do not endorse or approve any statement or approve any statement made or opinion expressed or conclusion drawn therein but approve the thesis only for purpose for which it has been submitted.

Committee of Final Examination for Evaluation of Thesis

1. ……………………………………………………………..
2. ……………………………………………………………...
3. ………………………………………………………………
4. ……………………………………………………………….

** Only in the case the thesis is approved.

DECLARATION OF ORIGINALITY AND COMPLIANCE OF ACADEMICS ETHICS

I hereby declare that this thesis contains literature survey and original research work by the undersigned candidate, as part of his **Masters of Technology in IT (Courseware Engineering)** studies during academic session of 2019-2021. All the information in this document has been obtained and presented in accordance with academic rules and ethical conduct.

I also declare that, as required by this rule and conduct, I have fully cited and referred all material and results that are not original to this work.

Name: CHAUDHURI MD TAUSIF

Examination Roll No.: M4CWE21023

Registration No.: 150125 of 2019-20

Title of the Thesis: FACE RECOGNITION APPROACHES USING DIFFERENT MACHINE LEARNING AND DEEP LEARNING ALGORITHMS

Signature: Chaudhuri Md Tausif
8-9-2021 **Date: 8-9- 2021**

ACKNOWLEDGEMENT

I feel fortunate while presenting this dissertation at School of Education Technology, Jadavpur University, Kolkata, in the partial fulfillment of the requirement for the degree of Masters of Technology in IT (Courseware Engineering).

I hereby take this opportunity to show my gratitude towards my supervisor, Dr. Saswati Mukherjee, who has guided and helped me with all possible suggestions, support, aspiring advices and constructive criticism along with illuminating views on different issues of this dissertation which helped me throughout my work.

I would like to express my warm thanks to Prof. Matangini Chattopadhyay, Director of School of Education Technology for her timely encouragement, support and advices. I would also like to thank to Dr. Ranjan Parekh and Mr. Joydeep Mukherjee for their constant support during my entire course of work.

My Thanks and appreciation go to my classmates from MTech IT (Courseware Engineering) and MMD.

Chaudhuri Md Tausif
8-9-2021

Chaudhuri Md Tausif
MTech IT (Courseware Engineering)
School of Education Technology
Jadavpur University, Kolkata-700032

Dedicated To,
My Parents

And my guide,
Dr. Saswati Mukherjee

Contents

1. Introduction

Face Recognition is presumed to be an active research area that i.e., in case of image recognition, image processing, computer vision, in the field of neuroscience. We can explain face identification in that way if a person put any random image which is comprised of complicated, multidimensional and meaningful analeptic. It’s a very difficult task to detect face because there are many possibilities that is present in human face. Like expression of a human face, a face position, camera variability, condition of light, image pixel, pose of a human face etc.

Face Recognition consists of several steps like face detection, face analysis, image extraction, detection of image border.

2. Problem Statement

An efficient approach for measuring accuracy on Face Recognition using Different Machine Learning and Deep Learning Algorithm.

3. Objective

The objectives of this dissertation are as follows:

1. To get a better accuracy using different machine learning algorithms on Olivetti Research Laboratory dataset.

2. CNN algorithm is used to predict better accuracy over the other machine learning algorithms to get a better result on the training and test images.

4. Assumptions

The proposed method has following assumptions:

1. The method focused on ORL dataset having 40 folders and 10 images in each folder.

2. Four machine learning algorithms are (PCA, LDA,1-NN, SVM) and deep learning algorithm CNN are used on these images to find accuracy.

5. Scope

The proposed work is implemented by different machine learning algorithms just like 1-NN, SVM, Kernel SVM, PCA, LDA, Random Forest, Naive Bayes and deep learning algorithm Convolutional Neural Network to predict a better accuracy on ORL dataset. In each machine learning algorithm 5-fold cross validation is used to improve the accuracy and find the accuracy in percentage.

6. Background Concept

Some key concepts need to be discussed to further understand the details of the framework that has been implemented to predict the accuracy in face recognition. This section includes some basic concepts of eigen face, eigen values and vectors which is related to PCA and LDA machine learning algorithm.

6.1 Eigen face

Eigen face method is very useful and important for face recognition and detection. It determines the variance of face. After that using machine learning approach (famous algorithm PCA is used in general way) to extract facial image data without reducing the space complexity and computation.

6.2 Eigen values and Eigen vectors

Eigen values are the special set of scalars which is associated with associated with the linear equation. It is generally used in matrix equation. In general words eigen values are scalars which is used to transform eigen vector.

The basic equation is

$$AX = \lambda X$$

Where λ is the is a eigen value of A.

And, Eigen vectors are vectors that do not change the direction when any linear transformation is applied. Briefly we can say that if A is a linear transformation from a vector space V and X is a vector in V (which is a non-zero vector), then V is the eigen vector of A if A(X) is the scalar multiple of X.

7. Outline of Dissertation

1. In Section 1 an introduction about the basic idea and concepts of Face Recognition has been given.

2. In Section 2 the problem statement has been given about the name of the dissertation.

3. Section 3 discusses the objective of the dissertation those are fulfilled and validated in 9 and 10.

4. Section 4 discusses the assumptions of the dissertation that are implemented later.

5. Section 5 discusses the scope of the algorithm.

6. Section 6 gives some idea of the background concept.

7. Section 7 depicts the outline of Dissertation.

8. Section 8 describes about the literature survey.

9. Section 9 presents the concept and problem analysis and description of algorithm.

10. Section 10 shows the Implementation and result of those algorithms.

11. Section 11 depicts the conclusion and future scope.

At the end of the dissertation, the references of other papers and in Appendix part, the implemented codes are attached.

8. Literature Survey

In this section, a review carried out on different technique is used on Face Recognition used by researchers. Various machine Learning and deep learning techniques are implemented in face recognition in order to predict better accuracy. Some prior works are discussed in this section.

8.1 2012-2020

Thee Chanyaswad et al. (2016) used a novel method for privacy preserving face recognition task using the DCA-eigenface subspace projection method. It is derived by the method that minimum dimension is required for maximum discriminant power of Discriminant Component Analysis.
In the three dataset is used here among them only the DCA-eigen face method is able to the recognition accuracy of 96.67% for Yale dataset, 97.06% for ORL dataset and 93.55% for Glasses Dataset.

Sayan Dev Sarkar et al. (2020) used ANN and image processing technique on face recognition. The proposed framework is combined on different feature extraction technique together with ANN was used to improve the accuracy of face recognition system. Out of many techniques only the PCA-ANN method or the LDA-ANN method performs absolutely sensational. The proposed technique tweaks 99 percent and 100 percent accuracy on different dataset.

Eko Setiwan et al. (2015) concluded some information about face recognition on low-power processor. K-Nearest Neighbour algorithms is used here to deliver best accuracy at K value is equal to 1. KNN demonstrated faster computation time compared to PCA and LDA. Overall the proposed method works great on low power processor.

Cunrui Wang et al. (2018) aims to extract salient feature via datamining for ethnicity recognition. In this proposed method are extracted from images which are used for ethnicity recognition and got the rate of recognition a little bit low. This is because of the facial features are bit different from that feature are extracted from different ethnicity facial images.
Hence in order to match of specific region in facial recognition those facial features are extracted from specific region on the ORL dataset and also fast sparse classification approach which is based on KNN algorithm is used for face recognition.

9. Concept and Problem Analysis

Facial recognition is a method of identifying and confirming every individual person’s identity by using their face. Generally, face recognition is used to identify and recognize faces of person from any image, or a video or real time approach just like OpenCV.
Face recognition is mainly used for security and law and most of the times research purpose.
In this proposed approach various machine learning (SVM, 1NN, Kernel SVM, Logistic regression, PCA, Naive Bayes, Random Forest) and deep learning algorithm CNN are implemented to compare the accuracies and to achieve better accuracy.

9.1 Data Collection

In this framework Olivetti research laboratory dataset, UK is used. The dataset contains 40 folders having 10 images in each folder. In some articles the photos were captured in different variant times, the light was slightly different, facial expressions (open eyes, closed eyes, smiles) and facial in details (glasses and without glasses) were all against dark. Photos has been taken similar backgrounds and articles are in the right position.

9.2 Data Preparation

In this proposed work the images are used to predict accuracy by using various machine learning and deep learning algorithm. By using this algorithm, features are extracted from testing sets and classifiers are trained. In order to improve accuracy 5-fold cross validation is used in every algorithm except CNN.

9.3 Descriptions of Algorithms

9.3.1 Support Vector Machine (SVM)

Support Vector Machine (SVM) is a supervised machine learning algorithm which is used for classification technique. SVM model set mainly based on labelled data.

SVM is a really good algorithm for image classification. An SVM model is basically is a representation of different classes in a hyperplane in a multidimensional space. Now the support vectors are the datapoints that are closest to the hyperplane. That's why this algorithm is known as Support Vector Machine.

The main goal of SVM algorithms is to divide the datasets into classes to achieve a maximum marginal hyperplane.

9.3.2 Principal Component Analysis (PCA)

Principal Component Analysis (PCA) is a famous machine learning algorithm which is widely used in exploratory data analysis and for making predictive models. It is a unsupervised machine learning algorithm generally used for dimensionality reduction.

Now the dimensionality reduction is one kind of learning technique used when the no. of features in a given dataset is too high. This technique reduces the dimensionality of the large dataset by transforming the large set of variables into smaller ones but still containing most of the information in the large set.

In this way, the information will not be lost and the features are reduced, and there will be fewer chances of overfitting the model.

9.3.3 K- Nearest Neighbour (KNN)

K-NN is basic classification algorithms in machine learning. It belongs to the supervised learning algorithm of machine learning models. k-NN is usually employed in search applications wherever we are looking for ***similar*** things. K-NN works in the way we measure similarity by making a vector illustration of the things, and then compare the vectors using an acceptable distance metric (for example the geometrician distance).

It is typically utilized in data processing, pattern recognition, recommender systems and intrusion detection.

9.3.4 Logistic Regression

Though Logistic regression is named as regression but it is actually a classification model which belongs to the supervised learning model.
Logistic regression is a statistical method which is very important tool in machine learning. This algorithm is used to classify incoming data which is based on historical data.
A logistic regression model predicts a dependent data variable by analysing the relationships between one or more independent variable.

9.3.5 Random Forest Classification

Random forest is one type of machine learning method which is used to delve in both regression and classification method. Random Forest belongs to this ensemble method learning, which is a technique that is combined of many classifiers to provide solutions to complex problem.

Random forest is consisted of many decision tree and the forest generated by the random forest algorithm is trained by bagging or bootstrap aggregating.

A random forest enhances the limitations of a decision tree classifier, which mainly reduces the overfitting problem on a dataset.

9.3.6 Naive Bayes Classification

Naive Bayes algorithm is one of the most famous algorithms in the world of supervised learning algorithm. This algorithm is based on bayes theorem. And this algorithm is called Naive because each and every input variable is independent. This is very powerful and simple algorithm for predictive modelling.

Some popular examples in naïve bayes algorithms are sentiment analysis , classifying articles etc.

Bayes theorem provides a way of calculating posterior probability P(c|x) from P(c), P(x) and P(x|c). Look at the equation below:

$$\underset{\text{Posterior Probability}}{P(c\mid x)} = \frac{\overset{\text{Likelihood}}{P(x\mid c)}\,\overset{\text{Class Prior Probability}}{P(c)}}{\underset{\text{Predictor Prior Probability}}{P(x)}}$$

$$P(c\mid X) = P(x_1\mid c)\times P(x_2\mid c)\times\cdots\times P(x_n\mid c)\times P(c)$$

Above,

1. P(c|x) is the posterior probability of class (c, target) given predictor (x, attributes).
2. P(c) is the prior probability of class.
3. P(x|c) is the likelihood which is the probability of predictor given class.
4. P(x) is the prior probability of predictor.

9.3.7 Convolutional Neural Network (CNN)

CNN or Convolutional Neural Network is a very prominent algorithm in Deep Learning community. CNN falls under the class of supervised deep learning model. CNN is broadly used in face recognition as example that to detect an animal face weather it is a cat or dog. CNN contains several steps like convolution operation, padding, pooling, flattening. It consists of 4-8 layers with image processing tasks which is incorporated into the design. Image segmentation, feature extraction and classification in one processing module with minimal pre-processing tasks on the desired input image is performed by CNN algorithm.

Minimal domain knowledge of the problem at hand is sufficient to perform efficient pattern recognition tasks. With the conventional pattern recognition tasks in which prior knowledge of the problem at hand is really required in order to apply a suitable algorithm to extract the right features.

10. Implementation and Result

This section will describe the result and performance evaluation of the proposed method to predict the likelihood of Face recognition using different machine learning and deep learning algorithms.

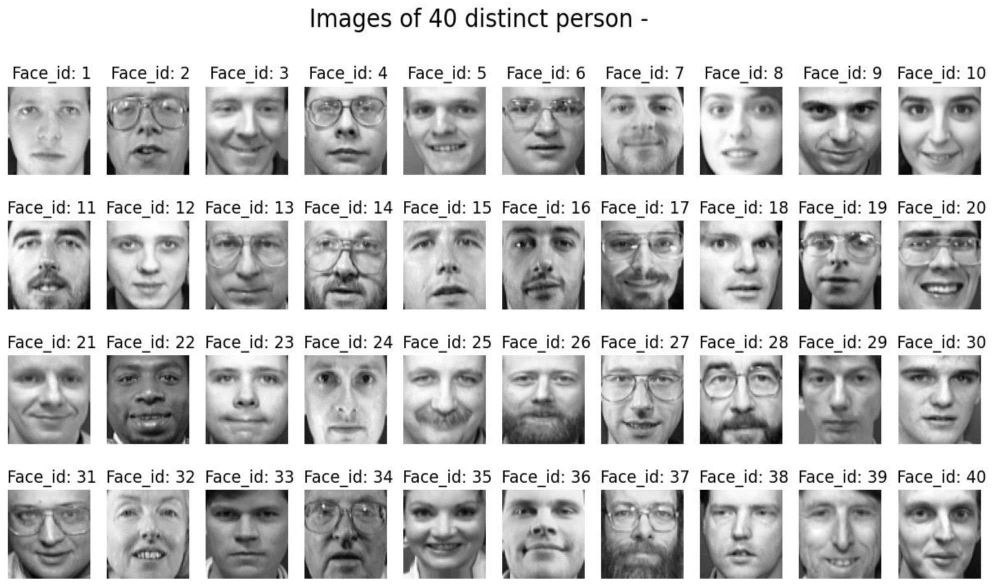
10.1 Experimental Analysis

5-fold cross validation technique is used to improve accuracy and to get accuracy in percentage. Different machine learning and deep learning algorithms are implemented to predict accuracy and to compare between those algorithms. Here those accuracies are listed below for each and every algorithm with 5 -fold cross validation and without 5-fold cross validation.

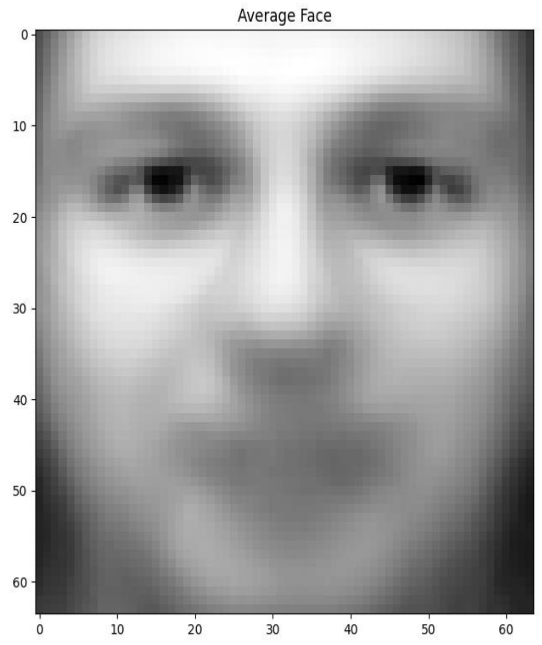
10.2 List of Accuracies

ML Algorithms	Accuracy	Accuracy in percentage with cross validation
Naive Bayes	0.8625	79.06
Random Forest	0.875	89.38
Kernel SVM	0.8875	92.81
SVM	0.9375	95.62
Logistic regression	0.9625	95.94
KNN (k=1)	0.95	90.00
PCA	0.975	95.94
CNN	0.94875	___

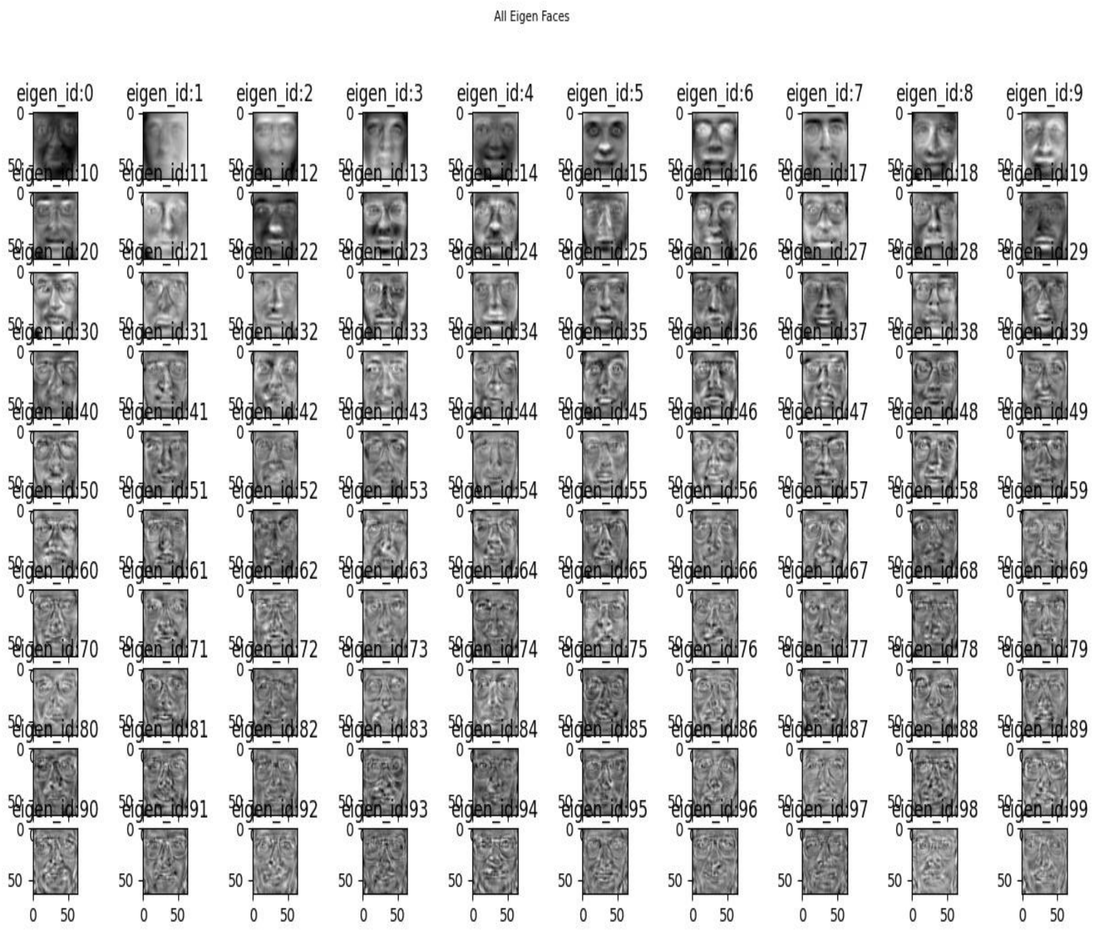
10.3 Images Of 40 distinct person



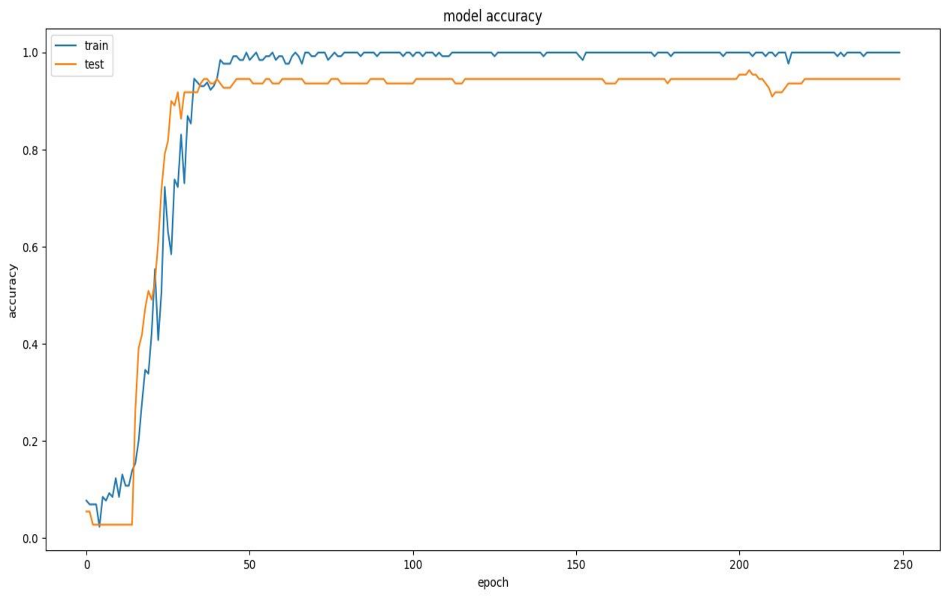
10.4 Average Face in PCA

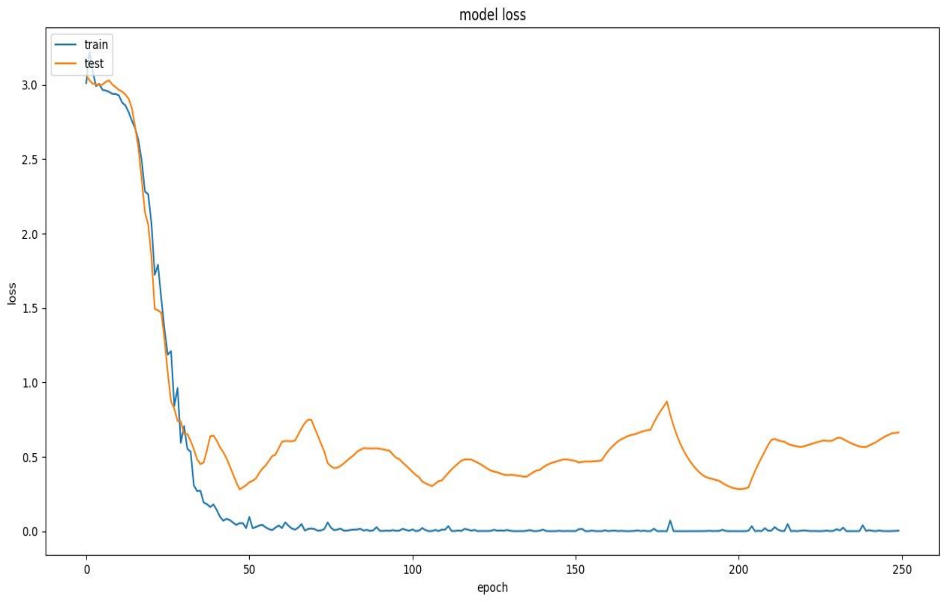


10.5 All Eigen Faces in PCA



10.6 Model Accuracy and Model Loss in CNN





11. Conclusion and Future Scope

This disseration aims to predict accuracy on fce recognition and to compare those accuracies using different machine learning algorithms.

The future of face recognition technology is bright. Forecaster opine that this technology is expected to grow at a formidable rate and will generate huge revenues in the coming years. Security and Surveillances are the major segments which will be deeply influenced. Other areas that are now welcoming it with open arms are private industries, public buildings and schools. In the long run, robots using facial recognition may also come to foray. They can be helpful in completing the tasks that are impractical or difficult for human beings to complete.

References

[1] Chanyaswad, J. M. Chang, P. Mittal and S. Y. Kung, “Discriminant-component eigen faces for privacy-preserving face recognition,” *2016 IEEE 26th International Workshop on Machine Learning for Signal Pre-processing (MLSP),* 2016, pp. 1-6

[2] S. D. Sarkar and A. Shenoy K.B., “Face Recognition using Artificial Neural Network and Feature Extraction,” *2020 7th International Conference on Signal Processing and Integrated Networks (SPIN),* 2020, pp. 417-422

[3] Setiawan, Eko, Muttaqin and Adharul, “Implementation of K-Nearest Neighbours Face Recognition on Low Power Processor,” *TELKOMNIKA (Telecommunication Computing Electronics and Control),* 2015, pp. 949-954

[4] Wang C, Zhang Q, Liu W, Liu Y, Miao L., “Facial feature discovery for ethnicity recognition,” *WIREs Data Mining Knowl Discov,* 2018, e1278

[5] Mohammad Mohsen Ahmadinejad, Elizabeth Sherly, "A Comparative Study on Kernel PCA and PCA Methods for Face Recognition", *International Journal of Science and Research (IJSR), Volume 5 Issue 5,* May 2016, pp.1844 – 1847

[6] Hao Wang, Yitong Wang, Zheng Zhou, Xing Ji, Dihong Gong, Jingchao Zhou, Zhifeng Li, and Wei Li, “Large Margin Cosine Loss for Deep Face Recognition,” *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR),* 2018, pp. 5265-5274

[7] T. Taniguchi, K. Furusawa, H. Liu, Y. Tanaka, K. Takenaka and T. Bando, “Determining Utterance Timing of a driving Agent with Double Articulation Analyzer,” in *IEEE Translation on Intelligent Transportation System*, vol. 17, no. 3, pp. 810-821, March 2016.

[8] S. Allagwail, O.S. Gedik and J. Ravedi, “Face Recognition with Symmetrical Face Training Samples Based on Local Binary Patterns and the Gabor Filter,” *Symmetry*, 2019, 11(2):157

Appendix

Implementing Different ML Algorithm

```
print(" Face Recognition Using Diff ML Algorithm
".center(125,"*"))
#importing libraries
import numpy as np
import matplotlib.pyplot as plt
from skimage.io import imshow
import matplotlib.image as mpimg

data= np.load('olivetti_faces.npy')
target = np.load('olivetti_faces_target.npy')

print("Data - ",data.shape)
print("Target - ",target.shape)

# Sample images of a subject
no_of_img = 10
plt.figure(figsize=(24,24))
for i in range(no_of_img):
    plt.subplot(1,10,i+1)
    x=data[i+40] # 4th subject
imshow(x)
plt.show()

#images of 40 distinct people
fig = plt.figure(figsize=(24,10))
cols = 10
rows = 4
for i in range(1, cols*rows+1):
    img = data[10*(i-1),:,:]
    fig.add_subplot(rows, cols, i)
    plt.imshow(img , cmap= plt.get_cmap('gray'))
    plt.title(" Face_id: {}".format(i),fontsize = 14)
    plt.axis("off")

plt.suptitle(" Images of 40 distinct person - ", fontsize= 20)
plt.show()

#Image data in matrix from is being converted to vector
y = target.reshape(-1,1) #storing target images in Y
x = data.reshape(data.shape[0], data.shape[1] * data.shape[2])
# reshaping and storing image in X
print(" X shape - ",x.shape)
print(" Y shape - ",y.shape)
print()
#splitting the dataset into training and testing dataset
```

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y,
test_size=0.2, random_state= 1)

print(" X_train : ",x_train.shape)
print(" X_test  : ", x_test.shape)
print(" Y_train : ", y_train.shape)
print(" Y_test  : ", y_test.shape)
print()

#storing accuracies of ML algorithms for comapring those
accuracies
list_name = []
list_accuracy = []

''' Applying 1-NN algorithm '''

#feature scaling
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
x_train = sc.fit_transform(x_train)
x_test = sc.transform(x_test)
print(x_train,"\n")
print(x_test,"\n")

''' training the KNN - 1NN model on training dataset '''
from sklearn.neighbors import KNeighborsClassifier
knnclassifier  = KNeighborsClassifier( n_neighbors = 1,
metric= "minkowski", p=2 )
knnclassifier.fit(x_train , y_train)

#predicting the test set
y_pred = knnclassifier.predict(x_test)
#
print(np.concatenate((y_pred.reshape(len(y_pred),1),y_test.res
hape(len(y_test),1)),1))
print()


#making a confusion matrix
from sklearn.metrics import confusion_matrix , accuracy_score
cm = confusion_matrix(y_test , y_pred)
print("confusion matrix - \n",cm)
print()
ac = accuracy_score(y_test , y_pred)
print("Accuracy Score in 1NN- \n",ac)
print()

""" Applying K fold cross validation for KNN model """
from sklearn.model_selection import cross_val_score
#accuracy obtained on each of the 5 test fold
```

```
KNN_accuracies = cross_val_score(estimator= knnclassifier, X =
x_train, y = y_train, cv = 5)
print(" Accuracy in 1NN after Kfold cross validation: {:.2f}
%".format(KNN_accuracies.mean()*100)) # multipling it by 100
to get the value in percentage
# print(" Standard Deviation: {:.2f}
%".format(accuracies.std()*100)) # multipling it by 100 to get
the value in percentage

list_name.append(" 1NN Algorithm")
list_accuracy.append(KNN_accuracies)


""" Training the Logistic Regression Model on Training set"""
from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression(random_state= 0)
classifier.fit(x_train,y_train)

""" making the Confusion matrix """
from sklearn.metrics import confusion_matrix , accuracy_score
y_pred = classifier.predict(x_test)
cm = confusion_matrix(y_test , y_pred)
print("confusion matrix - \n",cm)
print()

ac = accuracy_score(y_test , y_pred)
print("Accuracy Score in Logistic regression- \n",ac)
print()

""" Applying K fold cross validation for Logistic Regression
model """
from sklearn.model_selection import cross_val_score
#accuracy obtained on each of the 5 test fold
LR_accuracies = cross_val_score(estimator= classifier, X =
x_train, y = y_train, cv = 5)
print(" Accuracy in Logistic Regression after Kfold cross
validation: {:.2f} %".format(LR_accuracies.mean()*100)) #
multipling it by 100 to get the value in percentage
# print(" Standard Deviation: {:.2f}
%".format(accuracies.std()*100)) # multipling it by 100 to get
the value in percentage

list_name.append(" Logistic Regression Algorithm")
list_accuracy.append(LR_accuracies)


''' Training the SVM model on the training dataset '''
from sklearn.svm import SVC
SVMclassifier = SVC(kernel = "linear" , random_state= 0)
SVMclassifier.fit(x_train,y_train)
```

```
#predicting the test set
y_pred = SVMclassifier.predict(x_test)
#
print(np.concatenate((y_pred.reshape(len(y_pred),1),y_test.res
hape(len(y_test),1)),1))
print()

#making a confusion matrix
from sklearn.metrics import confusion_matrix , accuracy_score
cm = confusion_matrix(y_test , y_pred)
print("confusion matrix - \n",cm)
print()

ac = accuracy_score(y_test , y_pred)
print("Accuracy Score in SVM- \n",ac)
print()

""" Applying K fold cross validation for SVM model """
from sklearn.model_selection import cross_val_score
#accuracy obtained on each of the 5 test fold
SVM_accuracies = cross_val_score(estimator= SVMclassifier, X =
x_train, y = y_train, cv = 5)
print(" Accuracy in SVM after 5 fold cross validation: {:.2f}
%".format(SVM_accuracies.mean()*100))

list_name.append(" SVM Algorithm")
list_accuracy.append(SVM_accuracies)

''' Training the Kernel SVM model on the training dataset ->
kernel = rbf'''
from sklearn.svm import SVC
SVMclassifier = SVC(kernel = "rbf" , random_state= 0)
SVMclassifier.fit(x_train,y_train)

#predicting the test set
y_pred = SVMclassifier.predict(x_test)
#
print(np.concatenate((y_pred.reshape(len(y_pred),1),y_test.res
hape(len(y_test),1)),1))
print()

#making a confusion matrix
from sklearn.metrics import confusion_matrix , accuracy_score
cm = confusion_matrix(y_test , y_pred)
print("confusion matrix - \n",cm)
print()

ac = accuracy_score(y_test , y_pred)
print("Accuracy Score in Kernel SVM- \n",ac)
print()
```

```
""" Applying K fold cross validation for Kernel SVM model """
from sklearn.model_selection import cross_val_score
#accuracy obtained on each of the 5 test fold
KSVM_accuracies = cross_val_score(estimator= SVMclassifier, X
= x_train, y = y_train, cv = 5)
print(" Accuracy in Kernel SVM after 5 fold cross validation:
{:.2f} %".format(KSVM_accuracies.mean()*100))

list_name.append(" Kernel SVM Algorithm")
list_accuracy.append(KSVM_accuracies)

#Accuracies of every ML algorithm is shown in list
# import pandas as pd
# df = pd.DataFrame({'METHOD': list_name, 'ACCURACY (%)':
list_accuracy})
# # df = df.sort_values(by= ['ACCURACY (%)'])
# df = df.reset_index(drop=True)
# df.head()
# print(df)
print()

''' Applying Random Forest Classification '''
#Training Random Forest Classification in Training model
from sklearn.ensemble import RandomForestClassifier
RFclassifier = RandomForestClassifier(n_estimators= 100 ,
criterion= 'entropy' , random_state= 0 )
RFclassifier.fit(x_train , y_train)

# #predicting a new result
# print(RFclassifier.predict(sc.transform([[30,87000]])))
# print()

#predicting Test set
y_pred = RFclassifier.predict(x_test)
#
print(np.concatenate((y_pred.reshape(len(y_pred),1),y_test.res
hape(len(y_test),1)),1))
print()

#making a confusion matrix
from sklearn.metrics import confusion_matrix , accuracy_score
cm = confusion_matrix(y_test , y_pred)
print("confusion matrix for Random Forest- \n",cm)
print()
ac = accuracy_score(y_test , y_pred)
print("Accuracy Score for Random Forest- \n",ac)
print()

""" Applying K fold cross validation in Random Forest """
from sklearn.model_selection import cross_val_score
#accuracy obtained on each of the 5 test fold
```

```
RF_accuracies = cross_val_score(estimator= RFclassifier, X =
x_train, y = y_train, cv = 5)
print(" Accuracy for Random Forest after applying KFold cross
validation: {:.2f} %".format(RF_accuracies.mean()*100)) #
multipling it by 100 to get the value in percentage
# print(" Standard Deviation: {:.2f}
%".format(accuracies.std()*100)) # multipling it by 100 to get
the value in percentage

''' Applying Gaussian NB '''
#training Naive Bayes model on training dataset
from sklearn.naive_bayes import GaussianNB
NBclassifier = GaussianNB()
NBclassifier.fit(x_train , y_train)

#predicting Test set
y_pred = NBclassifier.predict(x_test)
#
print(np.concatenate((y_pred.reshape(len(y_pred),1),y_test.res
hape(len(y_test),1)),1))
print()

#making a confusion matrix
from sklearn.metrics import confusion_matrix , accuracy_score
cm = confusion_matrix(y_test , y_pred)
print("confusion matrix - \n",cm)
print()
ac = accuracy_score(y_test , y_pred)
print("Accuracy Score- \n",ac)
print()

""" Applying K fold cross validation in Naive Bayes """
from sklearn.model_selection import cross_val_score
#accuracy obtained on each of the 5 test fold
NB_accuracies = cross_val_score(estimator= NBclassifier, X =
x_train, y = y_train, cv = 5)
print(" Accuracy for Naive Bayes after applying KFold cross
validation: {:.2f} %".format(NB_accuracies.mean()*100)) #
multipling it by 100 to get the value in percentage
# print(" Standard Deviation: {:.2f}
%".format(accuracies.std()*100)) # multipling it by 100 to get
the value in percentage
```

Implementing PCA

```
print(" PCA_OLivetti ".center(125,"*"))

#importing libraries
import numpy as np
import matplotlib.pyplot as plt
from skimage.io import imshow
import matplotlib.image as mpimg

data= np.load('olivetti_faces.npy')
target = np.load('olivetti_faces_target.npy')

print("Data - ",data.shape)
print("Target - ",target.shape)

# Sample images of a subject
no_of_img = 10
plt.figure(figsize=(24,24))
for i in range(no_of_img):
    plt.subplot(1,10,i+1)
    x=data[i+40] # 4th subject
imshow(x)
plt.show()

#images of 40 distinct people
fig = plt.figure(figsize=(24,10))
cols = 10
rows = 4
for i in range(1, cols*rows+1):
    img = data[10*(i-1),:,:]
    fig.add_subplot(rows, cols, i)
    plt.imshow(img , cmap= plt.get_cmap('gray'))
    plt.title(" Face_id: {}".format(i),fontsize = 14)
    plt.axis("off")

plt.suptitle(" Images of 40 distinct person - ", fontsize= 20)
plt.show()

#Image data in matrix from is being converted to vector
y = target.reshape(-1,1) #storing target images in Y
x = data.reshape(data.shape[0], data.shape[1] * data.shape[2])
# reshaping and storing image in X
print(" X shape - ",x.shape)
print(" Y shape - ",y.shape)
print()
#splitting the dataset into training and testing dataset
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y,
test_size=0.2, random_state= 1)
```

```
print(" X_train : ",x_train.shape)
print(" X_test  : ", x_test.shape)
print(" Y_train : ", y_train.shape)
print(" Y_test  : ", y_test.shape)
print()

from sklearn.decomposition import PCA
pca = PCA(100)
X_train_pca = pca.fit_transform(x_train)
X_test_pca = pca.transform(x_test)

print('Original dataset:',x_train.shape)
print('Dataset after applying PCA:',X_train_pca.shape)
print('No of PCs/Eigen Faces:',len(pca.components_))
print('Eigen Face Dimension:',pca.components_.shape)
print('Variance
Captured:',np.sum(pca.explained_variance_ratio_))
print()

# Average face of the samples
plt.subplots(1,1,figsize=(8,8))
plt.imshow(pca.mean_.reshape((64,64)), cmap="gray")
plt.title('Average Face')
plt.show()

#showing all eigen faces
number_of_eigenfaces=len(pca.components_)
eigen_faces=pca.components_.reshape((number_of_eigenfaces,
data.shape[1], data.shape[2]))

columns=10
rows=int(number_of_eigenfaces/columns)
fig, axarr=plt.subplots(nrows=rows, ncols=columns,
figsize=(24,24))
axarr=axarr.flatten()
for i in range(number_of_eigenfaces):
    axarr[i].imshow(eigen_faces[i],cmap="gray")

    axarr[i].set_title("eigen_id:{}".format(i))
plt.suptitle("All Eigen Faces".format(10*"=", 10*"="),fontsize
= 8)
plt.show()

from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
x_train = sc.fit_transform(x_train)
x_test = sc.transform(x_test)

""" Training the Logistic Regression Model on Training set"""
from sklearn.linear_model import LogisticRegression
classifier = LogisticRegression(random_state= 0)
```

```
classifier.fit(X_train_pca,y_train)

""" making the Confusion matrix """
from sklearn.metrics import confusion_matrix , accuracy_score
y_pred = classifier.predict(X_test_pca)
cm = confusion_matrix(y_test , y_pred)
print("confusion matrix - \n",cm)
print()

ac = accuracy_score(y_test , y_pred)
print("Accuracy Score- \n",ac)
print()

""" Applying K fold cross validation for Logistic Regression
model """
from sklearn.model_selection import cross_val_score
#accuracy obtained on each of the 5 test fold
LR_accuracies = cross_val_score(estimator= classifier, X =
X_train_pca, y = y_train, cv = 5)
print(" Accuracy in Logistic Regression after Kfold cross
validation: {:.2f} %".format(LR_accuracies.mean()*100)) #
multipling
```

Implementing CNN

```
#importing the libraries
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import tensorflow as tf
from keras.preprocessing.image import ImageDataGenerator

physical_devices = tf.config.list_physical_devices('GPU')
tf.config.experimental.set_memory_growth(physical_devices[0],
True)

""" Step2 - Loading the Dataset """
#loading the dataset
data_set = np.load("ORL_faces.npz")

#loading the "Train" images
x_train = data_set['trainX']
#normalizing every image
x_train = np.array(x_train ,dtype="float32")/255

x_test = data_set['testX']
x_test = np.array(x_test, dtype="float32")/255

#loading the label of images
y_train = data_set['trainY']
y_test = data_set['testY']

#showing the train and test images in DataFormat
print("X_train: {}".format(x_train[:]))
print()
print("Y_train shape: {}".format(y_train))
print()
print("X_test shape: {}".format(x_test.shape))
print()

""" step3- Splitting the dataset """
from sklearn.model_selection import train_test_split
x_train, x_valid, y_train, y_valid =
train_test_split(x_train,y_train,random_state= 0,test_size=
110)

""" Step4 - Resizing the images """
im_rows = 112
im_cols = 92
batch_size = 512
im_shape = (im_rows, im_cols, 1)

#changing the size of images
x_train = x_train.reshape(x_train.shape[0], *im_shape)
```

```
x_test = x_test.reshape(x_test.shape[0], *im_shape)
x_valid = x_valid.reshape(x_valid.shape[0], *im_shape)
print("y_train shape: {}".format(y_train.shape[0]))
print("y_test shape: {}".format(y_test.shape))
print()

""" Building the CNN model """
from keras.preprocessing.image import ImageDataGenerator
#initializing the CNN
cnn = tf.keras.models.Sequential()
#adding the 1st Convolutional layer
cnn.add(tf.keras.layers.Conv2D(filters=36, kernel_size=7,
activation='relu', input_shape= im_shape))
#adding 1st pooling layer
cnn.add(tf.keras.layers.MaxPool2D(pool_size=2))
#adding 1st convolutional layer
cnn.add(tf.keras.layers.Conv2D(filters=54, kernel_size=5,
activation='relu', input_shape= im_shape))
#adding 2nd max pooling layer
cnn.add(tf.keras.layers.MaxPool2D(pool_size=2))
# Falttening
cnn.add(tf.keras.layers.Flatten())
#1st fully connected layer
cnn.add(tf.keras.layers.Dense(2024, activation='relu'))
#adding 1st dropout layer
cnn.add(tf.keras.layers.Dropout(0.5))
#2nd fully connected layer
cnn.add(tf.keras.layers.Dense(1024, activation='relu'))
#2nd dropout layer
cnn.add(tf.keras.layers.Dropout(0.5))
#3rd fully connected layer
cnn.add(tf.keras.layers.Dense(512, activation='relu'))
#3rd dropout layer
cnn.add(tf.keras.layers.Dropout(0.5))
#20 is the number of outputs
#final full connected layer
cnn.add(tf.keras.layers.Dense(20, activation='softmax'))

#Compiling the CNN
cnn.compile(optimizer='adam',
loss='sparse_categorical_crossentropy', metrics=['accuracy'])

""" Step 6 - Training the model """
history = cnn.fit( np.array(x_train), np.array(y_train),
batch_size=512,
epochs=250, verbose=2,
validation_data=(np.array(x_valid),np.array(y_valid)))

""" evaluating the score """

score = cnn.evaluate(np.array(x_test), np.array(y_test),
verbose= 0)
print('test los {:.4f}'.format(score[0]))
print('test accuracy {:.4f}'.format(score[1]))
print()

""" Plotting the result """
#list all the data in history
import itertools

print(history.history.keys())
print()

#summarizing history for accuracy
# summarizing history for accuracy
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()

# summarizing history for loss
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()

""" Predicting the accuracy """
from sklearn.metrics import confusion_matrix, accuracy_score
predicted_result = np.array(cnn.predict(x_test))
y_pred = cnn.predict_classes(x_test)
Accuracy = accuracy_score(y_test, y_pred)
print("accuracy : ", Accuracy)
print()
cm = confusion_matrix(y_test , y_pred)
print("confusion matrix - \n",cm)
print()
```