Comparative Analysis of Performance of Face Recognition using LBPH, Eigenfaces, Fischerfaces, SVM, and CNN algorithms.

Submitted in partial fulfillment of the requirements for the degree of

Bachelor of Technology

In

Computer Science and Engineering

By

Francis Alex Kuzhippallil	***18BCE2325***
Adith Kumar Menon	***18BCE2311***

Under the guidance of

Mr. Natarajan P

School of Computer Science and Engineering

VIT, Vellore.

November, 2021

TABLE OF CONTENTS

ABSTRACT

Face recognition is a technique that identifies or locates human faces in digital images. A typical example of face detection occurs when we take photographs through our smartphones, and it instantly detects faces in the picture. Face detection is different from Face recognition. Face detection detects merely the presence of faces in an image while facial recognition involves identifying whose face it is. Face Recognition has many applications in biometrics, surveillance systems, information security and law enforcement. The aim of this project is to implement face recognition using the Eigenface, Fisherface, SVM, CNN, LBPH algorithms and OpenCV (Open Source Computer Vision), a popular computer vision library. Furthermore developing a hybrid algorithm based on the fusion of the algorithms mentioned and comparing the performance of the algorithms in face recognition with accuracy as performance metrics. Lastly, hybrid algorithm is used to perform real-time detection and recognition of face, which might be used for security level applications.

Keywords:
CNN, LBPH, SVM, Eigenface, Fisherface, Hybrid, Real-time Detection, OpenCV.

1. INTRODUCTION

1.1 PROBLEM STATEMENT

Biometric techniques play a bigger and bigger role in nowadays research and development, because more and more applications find their place in people's life: Finger prints to login on your OS, to get in your workout center or to start your car engine aren't rare anymore. On higher (security) levels scans of the eye are used. Cameras are present on many public places to improve security and this works fine with motion detectors if it is not necessary to identify the person on the picture automatically. Overlooking the fact that not everybody agrees with the presence of the cameras it is a difficult problem to detect and recognize faces on given camera images. There are a lot of approaches to this problem, which have all advantages and disadvantages. The aim of this project is to get an idea of some (simple) methods and algorithms, how

faces can be detected in images and how they can be identified or matched with a given face database. Furthermore to perform live face detection and recognition.

1.2 ABOUT OPENCV

OpenCV (Open Source Computer Vision Library) is a library of programming functions mainly aimed at real-time computer vision. OpenCV (Open Source Computer Vision Library) is a library of programming functions mainly aimed at real-time computer vision. OpenCV Python is the python API for OpenCV. You can think of it as a python wrapper around the C++ implementation of OpenCV. OpenCV Python is not only fast (since the background consists of code written in C/C++) but is also easy to code and deploy (due to the Python wrapper in foreground). This makes it a great choice to perform computationally intensive programs. Face recognition is performed by using classifiers. A classifier is essentially an algorithm that decides whether a given image is positive (face) or negative (not a face). A classifier needs to be trained on thousands of images with and without faces. Fortunately, OpenCV already has two pre-trained classifiers, which can readily be used in a program. The two classifiers are: Haar Classifier and Local Binary Pattern (LBP) classifier. This project makes use of the Haar classifier

1.3 SOFTWARE USED

Hands-on knowledge of NumPy and Matplotlib is essential before working on the concepts of OpenCV. The following packages must be installed and running before installing OpenCV:

- Python (Anaconda and Jupyter Notebook)
- NumPy
-Matplotlib
-sklearn
-tensorflow

2. LITERATURE SURVEY

2.1 SURVEY OF EXISTING MODELS AND WORKS

[1] Aftab states, one of the major challenges faced in computer vision is face recognition. With face recognition being highly inevitable in law enforcement agencies, it is important to recognize faces efficiently against various adverse conditions such as blur, illumination, resolution and lighting. Aftab and team have used LBPH algorithm to identify human face from various angles even at low resolutions. The dataset used is LR500. The proposed system has been successfully been able to recognize faces correctly. This paper has limited itself to face detection and recognition. Furthermore, the proposed system can be suitable for increasing the security of the country and figuring the criminals.

[2] Akshata explains that face recognition identifies a human based on facets of face. Furthermore, LBPH is an efficient face recognition algorithm which extracts some portion of image initially. Further it converts the image to grayscale and the pixel value comparison of neighborhood and central is done. Moreover, GPU is being used, since it's much more powerful than CPU. Akshata have proposed frontal face and side profile face recognition using LBPH algorithm in GPU. The proposed system works perfectly in detecting frontal face and side profile face. In addition it is proved that the GPU processing time is much lesser than CPU and performance is much better than CPU.

[3] Agnihhotram reveals that face detection and recognition has been a challenging prospect in computer vision. This paper also uses LBPH and CNN for recognition of faces. They have used ORL and FEI dataset for their training and classification. The intended objective as proposed is achieved, with CNN performing better to LBPH is observed. Furthermore author states that this system can be used to track students for attendance and other security purposes as well.

[4] XueMei provides LBPH algorithm as the optimal solution to face recognition problem. Since it has the capabilities to recognize both front face and side face as well. In order to increase the performance of face

recognition under illumination, diversification, expression variation and attitude deflection, pixel neighborhood gray median LBPH is proposed (MLBPH). Here the gray value of the pixel is replaced by the median value of neighborhood sampling value. The dataset used for the research is FERET standard face database. Furthermore based on their experimental work, it is identified that MLBPH works superior to L BPH algorithm. The limitation of this paper is the problem of lens distortion.

[5] Mohannad states, one of the most interesting field for researchers is face recognition. Major motivation behind this enormous interest is the dire need to improve the accuracy of several real-time applications. Mohannad and team have proposed face recognition using Back propagation Neural Network. For this they have used T-Dataset, Yale and AT datasets as well. To show its improved accuracy, they compared the performance with LBPH algorithm. All together BPNN has achieved a tremendous result. The only limitation is that, they haven't compared the performance based on different neural networks such as CNN, LSTM etc.

[6] Vestina aims to build a real-time system that can help visually impaired to recognize people present in their surroundings. Input data taken is real time video taken from 16 mega pixel camera with 1280x 720. LBPH algorithm is used to increase the accuracy of recognition. Based on the experiment, system was able to increase the accuracy by 8.3% (87.2%). The limitation of the system is that it can recognize only one face at a time. However the system is able to identify faces from different heights using LBPH algorithm.

[7] Kshirsagar explains about face being a complex multidimensional visual model and recognizing faces is a difficult task. This paper follows face recognition based on coding and decoding of face image. Eigenfaces approach uses PCA for the recognition of the images. It is useful in finding the lower dimensional space. Based on the research it is found that Eigenfaces has been able to reduce the dimension and the network speed for recognition as well.

[8] Alina explains the requirement of difficult and extensive computations for coding and decoding the face images. This paper have used PCA for feature extraction and face recognition commonly known as

the Eigenfaces approach. To reduce the high computational cost, geometrical approximated PCA is being used. Distance measure taken is inverse Euclidean distance. High accuracy is obtained for face recognition using gaPCA, thereby representing viable alternative to classical statistical approach for computing the principal components. Parallelization of gaPCA can help in increasing the execution rates. That's the only limitation in the paper.

[9] Richard reveals that his paper corresponds to intelligent systems domain. Paper proposes facial recognition using Eigenfaces and PCA. Distance measure between facial characters chosen was Euclidean distance. Matlab was the tool used to do the classification. An accuracy of 91.43% was obtained when storing 3 components for each face and evaluating more users for training model.

[10] Ibnu reveals that Eigenfaces is algorithm based on PCA, which is mainly used to recognize faces. Eigenfaces is mainly useful in detecting faces after plastic surgery and combining them with facial image reconstruction techniques. Dataset used is grimace face image database owned by University of Essex, UK and Japanese Female Facial Expression (JAFFE) database. The average accuracy obtained in their research is 85% approx... The only limitation in this research is the lack of efficiency in identifying images using glasses.

[11] Rika states that Eigenface algorithm is the collection of eigenvectors used for face recognition through computers. Rika further explains that face recognition is part of image processing that recognizes faces from an image file in JPEG format. The proposed research is to build face recognition software using Eigenface algorithm. Furthermore an accuracy of 85% was obtained.

[12] Abdul states that security of information and physical property has been major problem in modern times. Face recognition aids in ensuring security and information access. This paper focusses on recognition of faces using Eigenfaces algorithm. ORL is the database used. Due to its increased user friendliness, it has gained more popularity than any other biometrics.

[13] Hyung proposes a face recognition technique that fuses elastic graph matching (EGM) and Fisherface

algorithm. Fischerfaces algorithm is robust about variations such as lightning, direction and facial expression. In comparison to the conventional methods, the proposed approach was able to obtain satisfactory results in rates and speeds perspectives. They were able to obtain a recognition rate of 99.3% with Yale database. Limitation is, Fischerfaces is not necessarily good enough for discriminating classes defined by a set of feature vectors.

[14] Banu explains the importance of Face recognition in computer vision. Banu proposes the usage of Fischerfaces and LBPH are implemented and analyzed using three distinct face image dataset and a real time video application. The accuracy, training and testing time for both the algorithms are measured using k-fold cross validation scheme. From the experiment, it is analyzed that Fischerfaces is good choice for real time face recognition applications. Whereas LBPH can handle dynamic face recognition scenarios.

[15] Lushen has combined Fischerfaces and one-against-rest classifiers based on support vector machine. Fisher linear discriminative rules are adopted to extract the optima features of face. One-against- rest classifiers are used to recognize the face images. Yale and ORL databases is being used. Accuracy of 97.5% (ORL database) and 97.8% (Yale) is being obtained. Lushen further proposes that Fischerfaces algorithm is superior to Eigenfaces on feature extraction.

[16] Peter proposed a face recognition which is insensitive to large variation in lighting direction and facial expression. They have used pattern classification approach. Projection method is based on Fisher's Linear Discriminant and produces well separated classes in a low dimensional subspace, even under severe variation in lighting and facial expressions. Yale is the database being used. Extensive experimental results demonstrate that Fisherface method has an error rate that are lower than Eigenface technique.

[17] Delpiah explains about various face recognition algorithms namely Adaboost, Eigenface PCA, and Fischerfaces. This research proposes how Fisherfaces and Eigenfaces perform in face recognition from web camera photos. The evaluation measure is accuracy. Furthermore it is found out that accuracy of Eigenface is 96% and accuracy of Fisherface algorithm is 97%. Also the Fisherface algorithm has quicker

time processing than Eigenface algorithm.

[18] Maliha reveals the importance of Face detection in real time field of biometrics. Furthermore, Maliha and team proposes PCA facial recognition system. PCA is mainly used to decrease the dimensionality of the pixels and represent the data economically. For experimentation, they have built a camera based real-time face recognition on OpenCV, Eigenface, LBPH, and Python. The main challenge yet is to segregate different man and woman from face recognized and improve the accuracy of existing models.

[19] Daniel reveals that grasping objects is a highly used activity in the industry but is often done without the aid of vision, limiting the use of few controlled applications. This paper proposes a feature based called ORB algorithm, whose primary function is to extract features. Combined with RANSAC, both will find object position in the RGB image and find the pixel coordinates in the point cloud. In order to develop the system ROS and OpenCV are used. With ORB object detection can be done from multiple angles and positions. The only limitation in this paper is the poor estimate of the position and the total loss of its localization.

[20] Vinay explains that face recognition is the cutting edge fields of research and stands unwaveringly as the most challenging problem in domain in Computer Vision. In this paper, the authors have adopted ORB algorithm in fusion with RANSAC to address few crucial inadequacies and remove redundant key-points and noise. Furthermore it has been observed that current system has been able to achieve more accuracy than the common methods such as SIFT-RANSAC and SURF-RANSAC.

[21] Kewen proposes face recognition method based on CNN. This network consists of three convolution layers followed by two pooling layers out of which two fully connected layers and softmax regression layer Furthermore stochastic gradient descent algorithm is being used to train feature extractor and classifier. ORL is the database being used and accuracy is the performance metrics being used. Overall 99% accuracy was obtained for the face recognition. Also the network has excellent convergence and strong robustness.

[22] Suleman expresses that face recognition possess the importance to give biometric authentication which is essential in security. Furthermore, the authors proposes a framework for smart glasses that can recognize faces. Face recognition is achieved using CNN of deep learning technique. Transfer learning of a trained CNN model that is AlexNet is done for face recognition. Their research has obtained an accuracy of 98.5% accuracy in identifying 2500 variant images in a class. These smart glasses can work in security domain for authentication purposes.

[23] Bendjillali considers a face recognition system that is divided into three steps. First is the Viola-Jones face detection algorithm followed by facial image enhancement using Adaptive Histogram Equalization algorithm and feature learning for classification. ResNet50, VGG1 CNN's are used for feature learning followed by classification. Yale and CMU PIE face are used as databases. Furthermore Resnet50 architecture achieved an accuracy score of 97 and 98% for two databases respectively.

[24] Meena mentions about the challenges in face recognition task namely illumination, poses, expressions and background. Moreover, deep learning methods are widely employed and they have provided promising results for image recognition and classification. This paper proposes CNN with weights learned from pre-trained model VGG-16 on ImageNet database. Yale and AT&T databases are being used. Based on experiments being conducted, using neural networks have provided them with better results (96.5%) as compared to other face recognition algorithms.

[25] Kamlesh mentions that face recognition is the most common biometric technique. The author aims to design an autonomous system that recognizes faces based on surveillance and based on the criticality lock the secured region. Haarcascade algorithm is used to detect and extract the face from an image for training the system. CNN is being used to identify the faces. Moreover this system contains buzzer alarm and door locking system for improvising security and trapping the intruder.

[26] Di Wang mentions that face recognition has become the prominent field in artificial intelligence and computer vision. To improve the accuracy of the recognition, paper proposes face recognition based on

Convolution Neural Network. Results have shown that improved algorithm can be effectively applied to the dataset.

[27] Liping states that face recognition is a hot research topic in computer vision. However there are several challenges that need to be rendered, likes of illumination and occlusion which directly influences the accuracy of recognition. Hence, this paper proposes CNN based on Tensor Flow for face recognition. On experimentation, proposed method has obtained better recognition and robustness in complex environment.

[28] Yousef explains that this paper focuses on extension neural network as classification tool for face recognition. To extract the coefficients associated with the faces, Eigenfaces method I adopted. ENN is being used due to its high performance rate, less memory and processing requirements, higher learning speed and simple structure. On experimentation, proposed method has provided optimum value for learning rate, which broadly depends upon the data set being used as well.

[29] Priyanka states that attendance maintaining system is difficult process if it's done manually. Face recognition is a smart and automated way for managing the attendance. Techniques such as illumination variant, Viola and Jones algorithm and PCA are used to overcome intensity, head pose problem. Face input is taken from the web cam and cross checked with the student database. This smart system can be an effective way to maintain the attendance and records of students.

[30] Shrayan reveals the fact that, biometrics are used to characterize a person's DNA, hand geometry, confront and various emotions as well. Hereditary biometrics has been used to validate and distinguish people by examining their physical attributes. This paper proposes an application framework utilizing face certification checks. Fischerfaces and LBPH algorithm are used for face recognition. Overall the proposed aim is being achieved. This framework can be stretched out to perform recognition of harmed or disfigured faces by shaping 3D model that is invariant to head present and lighting changes.

2.2 GAPS IDENTIFIED

Following are the gaps which we have found out.

i) Most researchers have restricted their scope of research to three algorithms namely Eigenface, Fisherface and LBPH.

ii) The datasets used in majority of research papers are those released before 2015.

iii) There are only limited amount of research papers that have explained about best suitable image processing techniques for best accuracies.

iv) Very limited research options based on scope of hybrid algorithm and real time detection based on hybrid algorithm being explained.

3. OVERVIEW OF PROPOSED SYSTEM

3.1 MOTIVATION BEHIND CHOOSING THIS TOPIC

Biometric techniques play a bigger and bigger role in nowadays research and development, because more and more applications find their place in people's life: Finger prints to login on your OS, to get in your workout center or to start your car engine aren't rare anymore. On higher (security) levels scans of the eye are used. Cameras are present on many public places to improve security and this works fine with motion detectors if it is not necessary to identify the person on the picture automatically. Overlooking the fact that not everybody agrees with the presence of the cameras it is a difficult problem to detect and recognize faces on given camera images. There are a lot of approaches to this problem, which have all advantages and disadvantages. The aim of this project is to get an idea of some (simple) methods and algorithms, how faces can be detected in images and how they can be identified or matched with a given face database. Furthermore to perform live face detection and recognition.

3.2 METHODOLOGY ADOPTED

Methodology Adopted

A. Dataset Identification

Data selection process involves the need for selecting appropriate data for analysis and obtaining effective knowledge by performing diverse data mining techniques. The data used for project is Celebrity Dataset and Yale Face Dataset.

Dataset Name	Celebrity Dataset
Dataset Source	Google Images
Dataset Attributes	Images of Angelina Jolie, Mohanlal, Mammotty and Gal Gadot.
Dataset Type	The following data is image data. Each pixels of the image is converted into matrix format consisting of gray scale values.

Dataset Information for Facial Recognition

Dataset Name	Yale Face Database
Dataset Source	UCSD Computer Vision
Dataset URL	http://vision.ucsd.edu/content/yale-face-database
Dataset Attributes	The following data is image data. Each pixels of the image is converted into matrix format consisting of gray scale values.

Dataset Information for Facial Recognition

B. Dataset Exploration

Data exploration is an initial step of data analysis which inculcates summarizing the data and observing initial patterns in the data and attributes. Celebrity dataset consist of 4 different personalities' images and Yale dataset consist of 15 different personalities images out of which 5 have been chosen for our project.

C. Choosing the Suitable Image Processing Technique

While focusing on the efficiency of facial recognition, it is essential to choose the right preprocessing technique. Following are the image processing techniques used.

1) *Without Any Image Processing Techniques* - It refers to training, testing followed by making prediction on raw data as obtained from the data sources.

2) *Enhanced Laplacian Filter* – It is a second order image sharpening filter that have a stronger response to fine details, such as thin lines and isolated points. Enhanced Laplacian filter is used by applying the Laplacian mask on the image matrix, thereby sharpening it.

3) *CLAHE* – CLAHE operates on small regions in the image, called tiles, rather than the entire image. The neighboring tiles are then combined using bilinear interpolation to remove the artificial boundaries.

D. Facial Recognition Algorithms

The most pivotal part in correctly predicting faces is choosing the right facial recognition algorithms along with adequate training and testing. Following are the facial recognition algorithms used.

1) *CNN* - Convolutional Neural Networks (CNN) changed the way we used to learn images. It made it very easy! CNN mimics the way humans see images, by focusing on one portion of the image at a time and scanning the whole image. In our project we have used conv2d layer, maxpooling layer, flatten layer and finally dense layer.

2) *Eigenface* – An Eigenface is the name given to a set of eigenvectors when used in the computer vision problem of human face recognition. The eigenvectors are derived from the covariance matrix of the probability distribution over the high-dimensional vector space of face images. The Eigenfaces themselves form a basis set of all images used to construct the covariance matrix. This produces dimension reduction by allowing the smaller set of basis images to represent the original training images. Classification can be achieved by comparing how faces are represented by the basis set.

3) *Fisherface* – Fisherfaces algorithm extracts principle components that separates one individual from another. So, now an individual's features can't dominate another person's features. ... LDA is used to find a linear combination of features that separates two or more classes or objects.

4) *LBPH* - Local Binary Pattern (LBP) is a simple yet very efficient texture operator which labels the pixels of an image by thresholding the neighborhood of each pixel and considers the result as a binary number.

5) *SVM* - Support vector machines (SVMs) are a set of supervised learning methods used for classification, regression and outliers detection. The advantages of support vector machines are: Effective in high dimensional spaces. Still effective in cases where number of dimensions is greater than the number of samples.

6) *Hybrid* – This algorithm fuses all the five algorithms. Hybrid algorithms works on the concept of voting approach. Predictions made by all five algorithms are stored and the most frequent result is chosen from the lot. To ensure tackling the deadlock, two best accuracy algorithms are chosen and the result obtained is compared and verified.

E. Performance Metrics Used

1) *Accuracy* – This performance measure is calculated by performing ratio of correctly predictedobservation to the total number of observations.

F. Real Time Detection

For performing real time detection, I have created image dataset of my face and trained the data along with the celebrity dataset. Furthermore, Hybrid algorithm is used to perform the prediction when live webcam is on.

3.3 MAJOR LIBRARIES IMPORTED

A. Pandas

Pandas is a Python package that provides fast, flexible, and expressive data structures designed to make working with structured (tabular, multidimensional, potentially heterogeneous) and time series data both easy and intuitive. It aims to be the fundamental high-level building block for doing practical, realworld data analysis in Python. Additionally, it has the broader goal of becoming the most powerful and flexible open source data analysis / manipulation tool available in any language

B. NumPy

NumPy is a Python library used for working with arrays. It also has functions for working in domain of linear algebra, Fourier transform, and matrices. NumPy was created in 2005 by Travis Oliphant. It is an open source project and you can use it freely. NumPy stands for Numerical Python.

C. Sklearn

Library that supports various algorithms like linear regression, Lasso regression, Elastic Net KNN etc. It also necessary to for pre-processing, estimating accuracy score and splitting the data into training and testing data. Scikit learn library makes machine learning in python much more robust and easy.

D. Matplotlib

Matplotlib is an amazing visualization library in Python for 2D plots of arrays. Matplotlib is a multi-platform data visualization library built on NumPy arrays and designed to work with the broader SciPy stack.

Each pyplot function makes some change to a figure: e.g., creates a figure, creates a plotting area in a figure, plots some lines in a plotting area, decorates the plot with labels, etc.

3.4 HARDWARE REQUIREMENTS (MINIMUM REQUIREMENTS)

Processor	Intel(R) Core(TM) i5-6500U CPU @ 2.50GHz($ CPU), ~ 2.60GHz
Graphic Card	AMD Radeon Graphics Processor
RAM	16.0 GB
ROM	512GB SSD storage1

3.5 NOVELTY OF OUR PROJECT

The main points which we have covered in our project work are as follows:

i. We are performing comparative analysis of five major facial recognition algorithms namely CNN, Eigenface, Fisherface, LBPH and SVM. We are drawing comparison based on traditional facial recognition algorithms along with deep learning and machine learning algorithms. This hasn't been observed in the previous research papers.

ii. We have developed a hybrid algorithms , which fuses the combination of all the five algorithms thereby increasing the overall efficiency of facial recognition

iii. Real- time face detection using the hybrid algorithm.

iv. Performing various image processing techniques as a part of data cleaning to increase the overall accuracy of facial recognition.

4. RESULT AND DISCUSSION

4.1 SCREENSHOTS OF CODE AND OUTPUT

LIBRARIES IMPORTED

In [1]:
```
import cv2
import os
import numpy as np
from PIL import Image as im
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Flatten
from tensorflow.keras.layers import Dense
from tensorflow.keras.layers import Conv2D
from tensorflow.keras.layers import MaxPooling2D
from tensorflow.keras.callbacks import TensorBoard
import matplotlib.pyplot as plt
%matplotlib inline
from warnings import filterwarnings
filterwarnings('ignore')
import tensorflow as tf
import pandas as pd
from scipy.stats import stats
import matplotlib.image as mpimg
import pandas as pd
from sklearn import svm
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report,accuracy_score,confusion_matrix
```

Importing Necessary Libraries

In [2]:
```
def detect_face(img):

    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    face_cascade = cv2.CascadeClassifier('C:\\Users\\Maria\\Desktop\\FRANCIS\\FALL SEM 21-22\\IMP\\our  project\\Dataset\\haarca
    faces = face_cascade.detectMultiScale(gray, scaleFactor=1.2, minNeighbors=0);

    if (len(faces) == 0):
        return None, None

    (x, y, w, h) = faces[0]

    return gray, faces[0]
```

Function to detect faces from image using Haarcascade classifier

```
]: def prepare_training_data(data_folder_path):

    dirs = os.listdir(data_folder_path)
    faces = []
    labels = []

    for dir_name in dirs:

        if not dir_name.startswith("s"):
            continue;

        label = int(dir_name.replace("s", ""))

        subject_dir_path = data_folder_path + "\\" + dir_name

        subject_images_names = os.listdir(subject_dir_path)

        for image_name in subject_images_names:

            if image_name.startswith("."):
                continue;

            image_path = subject_dir_path + "\\" + image_name

            image = cv2.imread(image_path)

            cv2.imshow("Training on image...", image)
            cv2.waitKey(1)

            face, rect = detect_face(image)

            if face is not None:

                faces.append(face)
                labels.append(label)

    cv2.destroyAllWindows()
    cv2.waitKey(1)
    cv2.destroyAllWindows()

    return faces, labels
```

Extracting Images from Directory

```
In [5]: def draw_rectangle(img, rect):
            (x, y, w, h) = rect
            cv2.rectangle(img, (x, y), (x+w, y+h), (0, 255, 0), 2)

        def draw_text(img, text, x, y):
            cv2.putText(img, text, (x, y), cv2.FONT_HERSHEY_PLAIN,1.3, (0, 255, 0), 2)
```

Drawing Rectangular Box Enclosing the Face

```
print("Preparing data...")
faces, labels = prepare_training_data("C:\\Users\\Maria\\Desktop\\FRANCIS\\FALL SEM 21-22\\IMP\\our  project\\Dataset\\Fischer\\
print("Data prepared")

print("Total faces: ", len(faces))
print("Total labels: ", len(labels))
```

```
Preparing data...
Data prepared
Total faces:  132
Total labels:  132
```

Passing the Directory Path

In [44]:
```
data=[]


for i in range(1,40):
    img1 = im.open('C:\\Users\\Maria\\Desktop\\FRANCIS\\FALL SEM 21-22\\IMP\\our  project\\Dataset\\training-data\\s1\\'+str(i)+
    image_arr1 = np.array(img1)
    data.append([image_arr1,0])


for i in range(1,46):
    img2 = im.open('C:\\Users\\Maria\\Desktop\\FRANCIS\\FALL SEM 21-22\\IMP\\our  project\\Dataset\\training-data\\s2\\'+str(i)+
    image_arr2 = np.array(img2)
    data.append([image_arr2,1])



for i in range(1,32):

    img3 = im.open('C:\\Users\\Maria\\Desktop\\FRANCIS\\FALL SEM 21-22\\IMP\\our  project\\Dataset\\training-data\\s3\\'+str(i)+
    image_arr3 = np.array(img3)
    data.append([image_arr3,2])



for i in range(1,30):

    img4 = im.open('C:\\Users\\Maria\\Desktop\\FRANCIS\\FALL SEM 21-22\\IMP\\our  project\\Dataset\\training-data\\s4\\'+str(i)+
    image_arr4 = np.array(img4)
    data.append([image_arr4,3])



print(data)
```

```
[[array([[[  0,   1,   0],
        [251, 253, 252],
        [156, 158, 157],
        ...,
        [177, 179, 178],
        [200, 202, 201],
        [ 98, 100,  99]],

       [[  0,   1,   0],
        [251, 253, 252],
        [156, 158, 157],
        ...,
```

Training Data for CNN

In [45]:
```
data2=[]


for i in range(1,15):
    img1 = im.open('C:\\Users\\Maria\\Desktop\\FRANCIS\\FALL SEM 21-22\\IMP\\our  project\\Dataset\\testing_data\\s1\\'+str(i)+'
    new_image1 = img1.resize((128,128))
    image_arr1 = np.array(new_image1)
    data2.append([image_arr1,0])



for i in range(1,11):
    img2 = im.open('C:\\Users\\Maria\\Desktop\\FRANCIS\\FALL SEM 21-22\\IMP\\our  project\\Dataset\\testing_data\\s2\\'+str(i)+'
    new_image2 = img2.resize((128,128))
    image_arr2 = np.array(new_image2)
    data2.append([image_arr2,1])


for i in range(1,11):

    img3 = im.open('C:\\Users\\Maria\\Desktop\\FRANCIS\\FALL SEM 21-22\\IMP\\our  project\\Dataset\\testing_data\\s3\\'+str(i)+'
    new_image3 = img3.resize((128,128))
    image_arr3 = np.array(new_image3)
    data2.append([image_arr3,2])



for i in range(1,11):

    img4 = im.open('C:\\Users\\Maria\\Desktop\\FRANCIS\\FALL SEM 21-22\\IMP\\our  project\\Dataset\\testing_data\\s4\\'+str(i)+'
    new_image4 = img4.resize((128,128))
    image_arr4 = np.array(new_image4)
    data2.append([image_arr4,3])



print(data2)
```

```
[[array([[[77, 37, 14],
        [80, 38, 15],
        [84, 40, 15],
        ...,
        [51, 22,  4],
        [51, 20,  4],
        [55, 22,  4]],
```

Testing Data for CNN

```
]: import random
   random.shuffle(data)
   random.shuffle(data2)

   X_train =[]
   y_train =[]
   for features, label in data:
     X_train.append(features)
     y_train.append(label)

   X_train = np.array(X_train).reshape(-1,128,128,3)


   X_test =[]
   y_test =[]
   for features, label in data2:
     X_test.append(features)
     y_test.append(label)

   X_test = np.array(X_test).reshape(-1,128,128,3)


   X_train = X_train.astype('float32')

   X_train /= 255

   from keras.utils import np_utils
   y_train = np_utils.to_categorical(y_train,4)
   y_test = np_utils.to_categorical(y_test,4)
```

Normalizing training and testing data for CNN

```
:  cnn = Sequential()
   cnn.add(Conv2D(32, (3, 3), activation="relu", input_shape=(128,128,3)))
   cnn.add(MaxPooling2D(pool_size = (2, 2)))
   cnn.add(Conv2D(32, (3, 3), activation="relu", input_shape=(128,128,3)))
   cnn.add(MaxPooling2D(pool_size = (2, 2)))
   cnn.add(Conv2D(32, (3, 3), activation="relu", input_shape=(128,128,3)))
   cnn.add(MaxPooling2D(pool_size = (2, 2)))
   cnn.add(Conv2D(64, (3, 3), activation="relu", input_shape=(128,128,3)))
   cnn.add(MaxPooling2D(pool_size = (2, 2)))
   cnn.add(Conv2D(64, (3, 3), activation="relu", input_shape=(128,128,3)))
   cnn.add(MaxPooling2D(pool_size = (2, 2)))
   cnn.add(Flatten())
   cnn.add(Dense(activation = 'relu', units = 128))
   cnn.add(Dense(activation = 'relu', units = 64))
   cnn.add(Dense(units=4, activation="softmax"))
   cnn.compile(optimizer = 'adam', loss = 'binary_crossentropy', metrics = ['accuracy'])
```

```
WARNING:tensorflow:From C:\Users\Maria\Anaconda3\New folder\lib\site-packages\tensorflow_core\python\ops\resource_variable_ops.
py:1630: calling BaseResourceVariable.__init__ (from tensorflow.python.ops.resource_variable_ops) with constraint is deprecated
and will be removed in a future version.
Instructions for updating:
If using Keras pass *_constraint arguments to layers.
```

CNN Layers for Facial Recognition

```
cnn.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d (Conv2D)              (None, 126, 126, 32)      896
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 63, 63, 32)        0
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 61, 61, 32)        9248
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 30, 30, 32)        0
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 28, 28, 32)        9248
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 14, 14, 32)        0
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 12, 12, 64)        18496
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 6, 6, 64)          0
_________________________________________________________________
conv2d_4 (Conv2D)            (None, 4, 4, 64)          36928
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 2, 2, 64)          0
_________________________________________________________________
flatten (Flatten)            (None, 256)               0
_________________________________________________________________
dense (Dense)                (None, 128)               32896
_________________________________________________________________
dense_1 (Dense)              (None, 64)                8256
_________________________________________________________________
dense_2 (Dense)              (None, 4)                 260
=================================================================
Total params: 116,228
Trainable params: 116,228
Non-trainable params: 0
_________________________________________________________________
```

CNN Model Summary

```
hist= cnn.fit(X_train, y_train, batch_size =50, epochs =20, verbose = 1, validation_data = (X_test, y_test))
```

```
WARNING:tensorflow:From C:\Users\Maria\Anaconda3\New folder\lib\site-packages\tensorflow_core\python\ops\math_grad.py:1424: where (from tensorflow.python.ops.array_ops) is deprecated and will be removed in a future version.
Instructions for updating:
Use tf.where in 2.0, which has the same broadcast rule as np.where
```

CNN Training the Model

5) ML

In [63]:
```
neutral = []
categ=[]

kernel = np.array([[0, -1, 0],
                   [-1, 5,-1],
                   [0, -1, 0]])

for i in range(1,40):

    img1 = im.open('C:\\Users\\Maria\\Desktop\\FRANCIS\\FALL SEM 21-22\\IMP\\our  project\\Dataset\\Fischer\\org\\train_org\\s1\\

    img1 = img1.resize((128,128), im.ANTIALIAS)

    image_arr1 = np.array(img1)
    image_sharp1 = cv2.filter2D(src=image_arr1, ddepth=-1, kernel=kernel)

    img6=  image_sharp1.flatten()

    neutral.append(img6)
    categ.append(0)

for i in range(1,45):

    img2 = im.open('C:\\Users\\Maria\\Desktop\\FRANCIS\\FALL SEM 21-22\\IMP\\our  project\\Dataset\\Fischer\\org\\train_org\\s2\\

    img2 = img2.resize((128,128), im.ANTIALIAS)

    image_arr2 = np.array(img2)
    image_sharp2 = cv2.filter2D(src=image_arr2, ddepth=-1, kernel=kernel)

    img7 = image_sharp2.flatten()
```

ML Training Data

56]:
```
n2=[]
c2=[]

for i in range(1,10):

    img1 = im.open('C:\\Users\\Maria\\Desktop\\FRANCIS\\FALL SEM 21-22\\IMP\\our  project\\Dataset\\test _Fischer\\s1\\'+str(i)+
    img2 = im.open('C:\\Users\\Maria\\Desktop\\FRANCIS\\FALL SEM 21-22\\IMP\\our  project\\Dataset\\test _Fischer\\s2\\'+str(i)+
    img3 = im.open('C:\\Users\\Maria\\Desktop\\FRANCIS\\FALL SEM 21-22\\IMP\\our  project\\Dataset\\test _Fischer\\s3\\'+str(i)+
    img4 = im.open('C:\\Users\\Maria\\Desktop\\FRANCIS\\FALL SEM 21-22\\IMP\\our  project\\Dataset\\test _Fischer\\s4\\'+str(i)+


    img1 = img1.resize((128,128), im.ANTIALIAS)
    img2 = img2.resize((128,128), im.ANTIALIAS)
    img3 = img3.resize((128,128), im.ANTIALIAS)
    img4 = img4.resize((128,128), im.ANTIALIAS)


    image_arr1 = np.array(img1)
    image_sharp1 = cv2.filter2D(src=image_arr1, ddepth=-1, kernel=kernel)

    image_arr2 = np.array(img2)
    image_sharp2 = cv2.filter2D(src=image_arr2, ddepth=-1, kernel=kernel)


    image_arr3 = np.array(img3)
    image_sharp3 = cv2.filter2D(src=image_arr3, ddepth=-1, kernel=kernel)


    image_arr4 = np.array(img4)
    image_sharp4 = cv2.filter2D(src=image_arr4, ddepth=-1, kernel=kernel)



    img6=  image_sharp1.flatten()
    img7 = image_sharp2.flatten()
    img8 = image_sharp3.flatten()
    img9 =  image_sharp4.flatten()
```

ML Testing Data

```
In [65]: import pandas as pd
neut=np.array(neutral)

df=pd.DataFrame(neut)

target=np.array(categ)

df['Target']=target

x=df.iloc[:,:-1]
y=df.iloc[:,-1]
x_train=x
y_train=y
```

Dividing Data into Training Data in ML

```
n2=np.array(n2)
c2=np.array(c2)

df2=pd.DataFrame(n2)

target=np.array(c2)

df2['Target']=target

x=df2.iloc[:,:-1]
y=df2.iloc[:,-1]

x_test=x
y_test=y
```

Dividing Data into Testing Data in ML

```
from sklearn import svm
from sklearn.model_selection import GridSearchCV
param_grid={'C':[0.1,1,10,100],'gamma':[0.0001,0.001,0.1,1],'kernel':['rbf','poly']}
svc=svm.SVC(probability=True)
print("The training of the model is started, please wait for while as it may take few minutes to complete")
model=GridSearchCV(svc,param_grid)
model.fit(x_train,y_train)
print('The Model is trained well with the given images')
model.best_params_
```

```
The training of the model is started, please wait for while as it may take few minutes to complete
The Model is trained well with the given images

{'C': 0.1, 'gamma': 0.0001, 'kernel': 'poly'}
```

SVM Fitting of Data

```
face_recognizer = cv2.face.LBPHFaceRecognizer_create()
faces=np.array(faces)
labels=np.array(labels)

face_recognizer.train(faces,labels)

p=[]

for i in range(1,45):

    test_img1 = im.open("C:\\Users\\Maria\\Desktop\\FRANCIS\\FALL SEM 21-22\\IMP\\our  project\\Dataset\\Fischer\\sharpening\\te
    new_img1 = test_img1.resize((128,128))
    img_arr1=np.array(new_img1)

    img = img_arr1.copy()

    face, rect = detect_face(img)

    label= face_recognizer.predict(face)

    p.append(label[0])

    print(label[0])


    label_text = subjects[label[0]]


    draw_rectangle(img, rect)
    draw_text(img, label_text, rect[0], rect[1]-5)
```

LBPH Algorithm Training and Testing

```
face_recognizer = cv2.face.EigenFaceRecognizer_create()
faces=np.array(faces)
labels=np.array(labels)

face_recognizer.train(faces,labels)

p=[]

for i in range(1,45):

    test_img1 = im.open("C:\\Users\\Maria\\Desktop\\FRANCIS\\FALL SEM 21-22\\IMP\\our  project\\Dataset\\Fischer\\sharpening\\te
    new_img1 = test_img1.resize((128,128))
    img_arr1=np.array(new_img1)

    img = img_arr1.copy()

    face, rect = detect_face(img)

    label= face_recognizer.predict(face)

    p.append(label[0])

    print(label[0])


    label_text = subjects[label[0]]


    draw_rectangle(img, rect)
    draw_text(img, label_text, rect[0], rect[1]-5)
```

Eigenface Algorithm Training and Testing

```
face_recognizer = cv2.face.FisherFaceRecognizer_create()
faces=np.array(faces)
labels=np.array(labels)

face_recognizer.train(faces,labels)

p=[]

for i in range(1,45):

    test_img1 = im.open("C:\\Users\\Maria\\Desktop\\FRANCIS\\FALL SEM 21-22\\IMP\\our  project\\Dataset\\Fischer\\sharpening\\te
    new_img1 = test_img1.resize((128,128))
    img_arr1=np.array(new_img1)

    img = img_arr1.copy()

    face, rect = detect_face(img)

    label= face_recognizer.predict(face)

    p.append(label[0])

    print(label[0])


    label_text = subjects[label[0]]


    draw_rectangle(img, rect)
    draw_text(img, label_text, rect[0], rect[1]-5)
```

Fisherface Algorithm Training and Testing

```
import Augmentor
p = Augmentor.Pipeline("C:\\Users\\Maria\\Desktop\\FRANCIS\\FALL SEM 21-22\\IMP\\our  project\\yalefaces\\test\\s5")
p.rotate(probability=0.7, max_left_rotation=0.5, max_right_rotation=0.5)
p.sample(50)
```

```
Processing <PIL.Image.Image image mode=L size=320x243 at 0x25608970748>:   6%|     | 3/50 [00:00<00:04,  9.73 Samples/s]
```

Data Augmentation using Augmentor

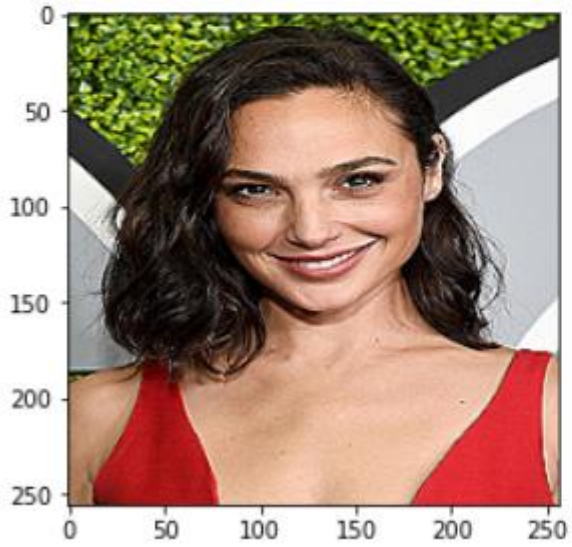
```
for i in range(1,45):
    img1 = im.open('C:\\Users\\Maria\\Desktop\\FRANCIS\\FALL SEM 21-22\\IMP\\our  project\\Dataset\\Fischer\\org\\train_org\\s2\
    new_image1 = img1.resize((256,256))
    data4=[]
    data4 = np.asarray(new_image1)
    gray_img1=cv2.cvtColor(data4,cv2.COLOR_BGR2GRAY)
    gray_img1_eqhist=cv2.equalizeHist(gray_img1)
    clahe=cv2.createCLAHE(clipLimit=40)
    gray_img1_clahe=clahe.apply(gray_img1_eqhist)
    data = im.fromarray(gray_img1_clahe)
    data.save(r'C:\\Users\\Maria\\Desktop\\FRANCIS\\FALL SEM 21-22\\IMP\\our  project\\Dataset\\Fischer\\clahe\\train_clahe\\s2\
```

Applying CLAHE to Images

```
img1=im.open('C:\\Users\\Maria\\Desktop\\FRANCIS\\FALL SEM 21-22\\IMP\\our  project\\Dataset\\testing\\16.jpg')
new_image = img1.resize((256,256))
kernel = np.array([[0, -1, 0],
                   [-1, 5,-1],
                   [0, -1, 0]])
plt.imshow(new_image)
```

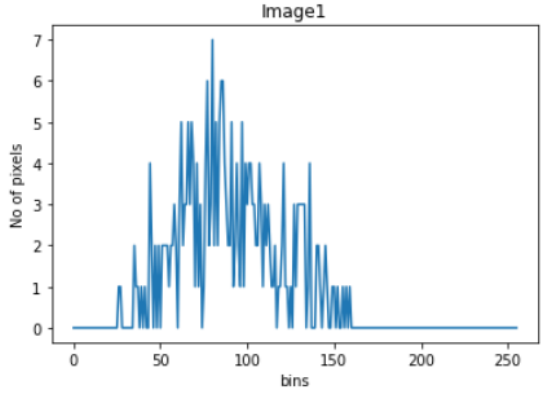
```
image_arr1 = np.array(new_image)
image_sharp1 = cv2.filter2D(src=image_arr1, ddepth=-1, kernel=kernel)
plt.imshow(image_sharp1)
```

```
<matplotlib.image.AxesImage at 0x2a8069a4c88>
```



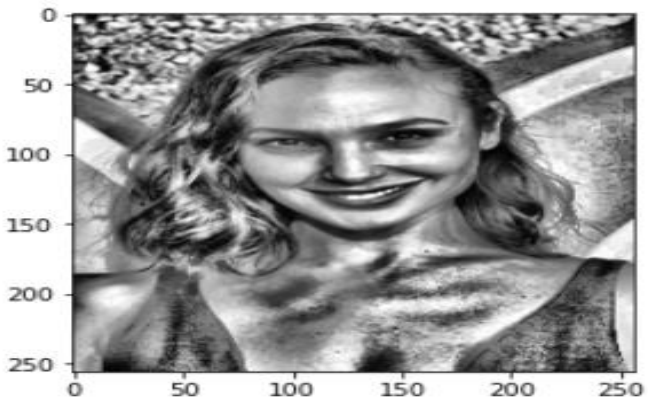
Sharpened Image using Enhanced Laplacian Filter

```
img1=im.open('C:\\Users\\Maria\\Desktop\\FRANCIS\\FALL SEM 21-22\\IMP\\our  project\\Dataset\\testing\\16.jpg')
new_image = img1.resize((256,256))
data=[]
data = np.asarray(new_image)
gray_img1=cv2.cvtColor(data,cv2.COLOR_BGR2GRAY)

hist1=cv2.calcHist(gray_img1,[0],None,[256],[0,256])
plt.title("Image1")
plt.xlabel('bins')
plt.ylabel("No of pixels")
plt.plot(hist1)
plt.show()
```



```
clahe=cv2.createCLAHE(clipLimit=40)
gray_img1_clahe=clahe.apply(gray_img1_eqhist)
plt.imshow(gray_img1_clahe,cmap='gray')
```

```
<matplotlib.image.AxesImage at 0x2a8045fa240>
```



Applying CLAHE on Image

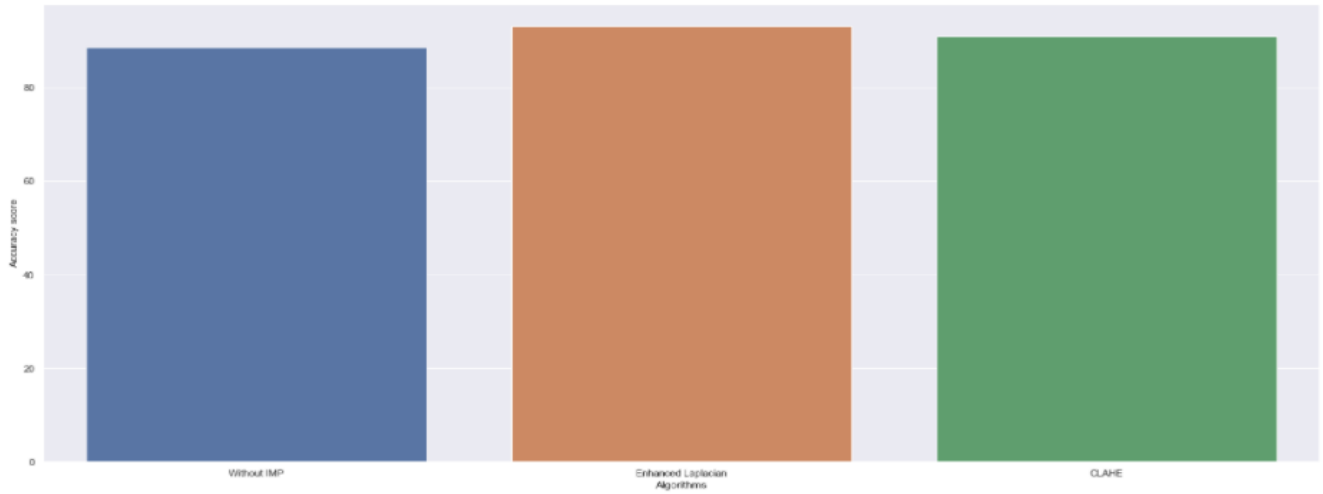
1) LBPH

```
scores = [acc_lb_w,acc_lb_lapl,acc_lb_clahe]
algorithms = ["Without IMP","Enhanced Laplacian","CLAHE"]

for i in range(len(algorithms)):
    print("The accuracy score achieved using "+algorithms[i]+" is: "+str(scores[i])+" %")
```

```
The accuracy score achieved using Without IMP is: 88.63636363636364 %
The accuracy score achieved using Enhanced Laplacian is: 93.18181818181819 %
The accuracy score achieved using CLAHE is: 90.9090909090909 %
```

```
import seaborn as sns
sns.set(rc={'figure.figsize':(25,10)})
plt.xlabel("Algorithms")
plt.ylabel("Accuracy score")

sns.barplot(algorithms,scores)
```

```
<AxesSubplot:xlabel='Algorithms', ylabel='Accuracy score'>
```



LBPH Result on Applying Various IMP Techniques

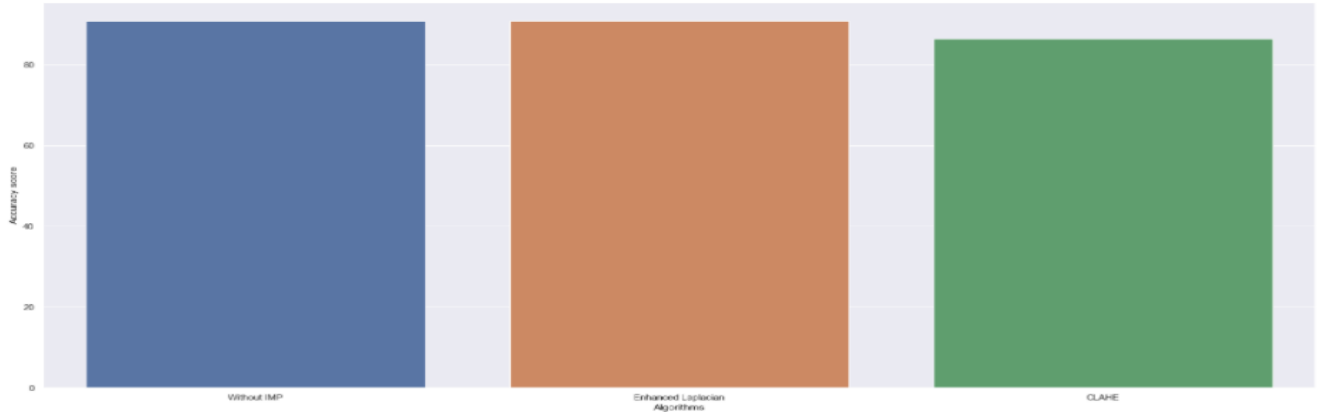
2) Eigen Face

```
scores = [acc_eigen_w,acc_eigen_lapl,acc_eigen_clahe]
algorithms = ["Without IMP","Enhanced Laplacian","CLAHE"]

for i in range(len(algorithms)):
    print("The accuracy score achieved using "+algorithms[i]+" is: "+str(scores[i])+" %")
```

```
The accuracy score achieved using Without IMP is: 90.9090909090909 %
The accuracy score achieved using Enhanced Laplacian is: 90.9090909090909 %
The accuracy score achieved using CLAHE is: 86.36363636363636 %
```

```
import seaborn as sns
sns.set(rc={'figure.figsize':(25,10)})
plt.xlabel("Algorithms")
plt.ylabel("Accuracy score")

sns.barplot(algorithms,scores)
```

```
<AxesSubplot:xlabel='Algorithms', ylabel='Accuracy score'>
```



Eigenface Result on Applying Various IMP Techniques

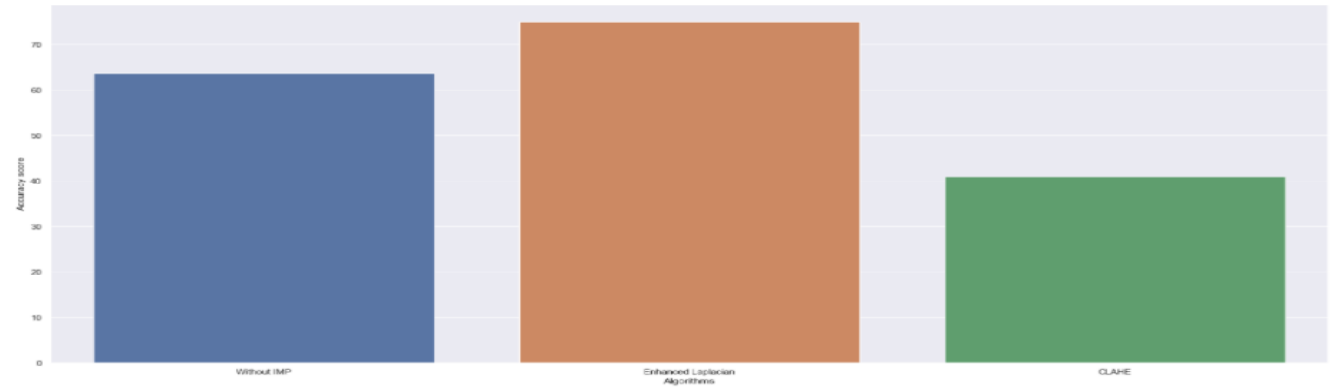
3) Fisher

```
scores = [acc_fisher_w,acc_fisher_lapl,acc_fisher_clahe]
algorithms = ["Without IMP","Enhanced Laplacian","CLAHE"]

for i in range(len(algorithms)):
    print("The accuracy score achieved using "+algorithms[i]+" is: "+str(scores[i])+" %")
```

```
The accuracy score achieved using Without IMP is: 63.63636363636363 %
The accuracy score achieved using Enhanced Laplacian is: 75.0 %
The accuracy score achieved using CLAHE is: 40.90909090909091 %
```

```
import seaborn as sns
sns.set(rc={'figure.figsize':(25,10)})
plt.xlabel("Algorithms")
plt.ylabel("Accuracy score")

sns.barplot(algorithms,scores)
```

```
<AxesSubplot:xlabel='Algorithms', ylabel='Accuracy score'>
```



Fisherface Result on Applying Various IMP Techniques

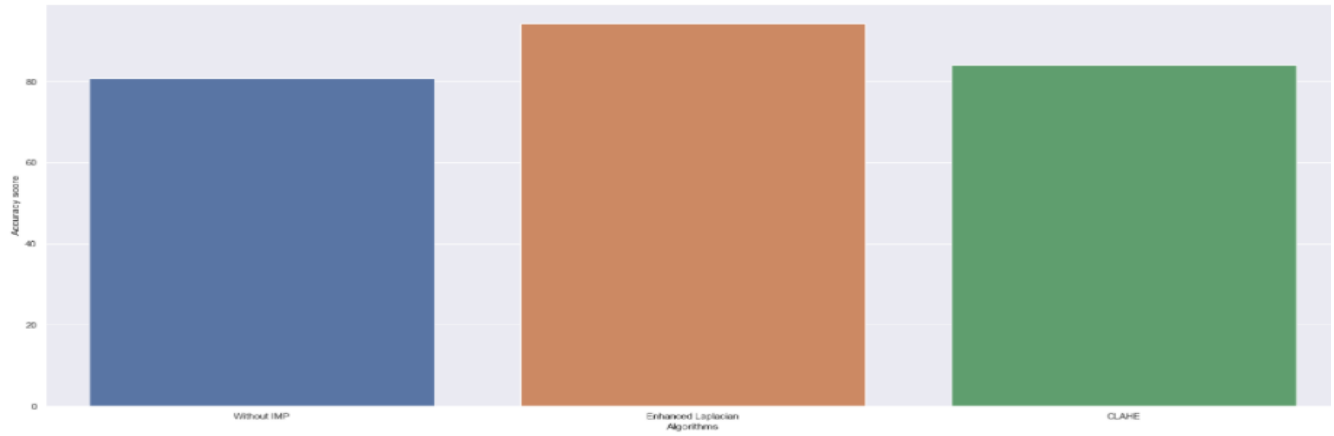
4) CNN

```
scores = [score_cnn_w,score_cnn_lapl,score_cnn_clahe]
algorithms = ["Without IMP","Enhanced Laplacian","CLAHE"]

for i in range(len(algorithms)):
    print("The accuracy score achieved using "+algorithms[i]+" is: "+str(scores[i])+" %")
```

```
The accuracy score achieved using Without IMP is: 80.68181872367859 %
The accuracy score achieved using Enhanced Laplacian is: 94.31818127632141 %
The accuracy score achieved using CLAHE is: 84.09090638160706 %
```

```
import seaborn as sns
sns.set(rc={'figure.figsize':(25,10)})
plt.xlabel("Algorithms")
plt.ylabel("Accuracy score")

sns.barplot(algorithms,scores)
```

```
<AxesSubplot:xlabel='Algorithms', ylabel='Accuracy score'>
```



CNN Result on Applying Various IMP Techniques

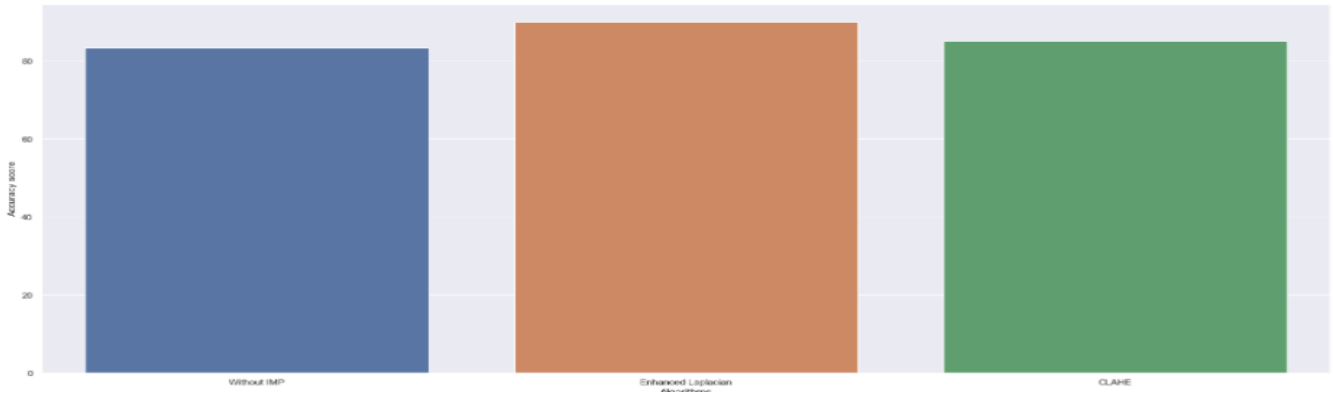
5) ML

```
scores = [acc_ml_w,acc_ml_lapl,acc_ml_clahe]
algorithms = ["Without IMP","Enhanced Laplacian","CLAHE"]

for i in range(len(algorithms)):
    print("The accuracy score achieved using "+algorithms[i]+" is: "+str(scores[i])+" %")
```

```
The accuracy score achieved using Without IMP is: 83.33333333333334 %
The accuracy score achieved using Enhanced Laplacian is: 90.0 %
The accuracy score achieved using CLAHE is: 85.0 %
```

```
import seaborn as sns
sns.set(rc={'figure.figsize':(25,10)})
plt.xlabel("Algorithms")
plt.ylabel("Accuracy score")

sns.barplot(algorithms,scores)
```

```
<AxesSubplot:xlabel='Algorithms', ylabel='Accuracy score'>
```



SVM Result on Applying Various IMP Techniques

```
import collections
g=[1,2,3,4,1,2,3,4,1,2,3,4,1,2,3,4,1,2,3,4,1,2,3,4,1,2,3,4,1,2,3,4,1,2,3,4,1,2,3,4,1,2,3,4,1,2,3,4]

c=0
for i in range(44):

    p_hy=[]

    p_hy.append(p_cnn[i]+1)

    p_hy.append(p_ml[i]+1)

    p_hy.append(p_lb[i])

    p_hy.append(p_eigen[i])

    p_hy.append(p_fisher[i])

    sm=0

    frequency = collections.Counter(p_hy)

    most_common = max(p_hy, key = p_hy.count)

    fm=frequency[most_common]

    del frequency[most_common]

    for j in range(1,5):

       if(frequency[j]==fm):

          sm=j


    if(sm!=0):

        if(sm==p_cnn[i]+1):

            num=sm

        elif(sm==p_cnn[i]):

            num=sm
        else:

            num=most_common

     else:

          num=most_common


     print("After iteration"+str(i))

     print(p_hy)

     print(num)

     if(num==g[i]):

         c=c+1

print("Accuracy Achieved is "+str(c*100/44)+"%")

acc_h=c*100/44
```

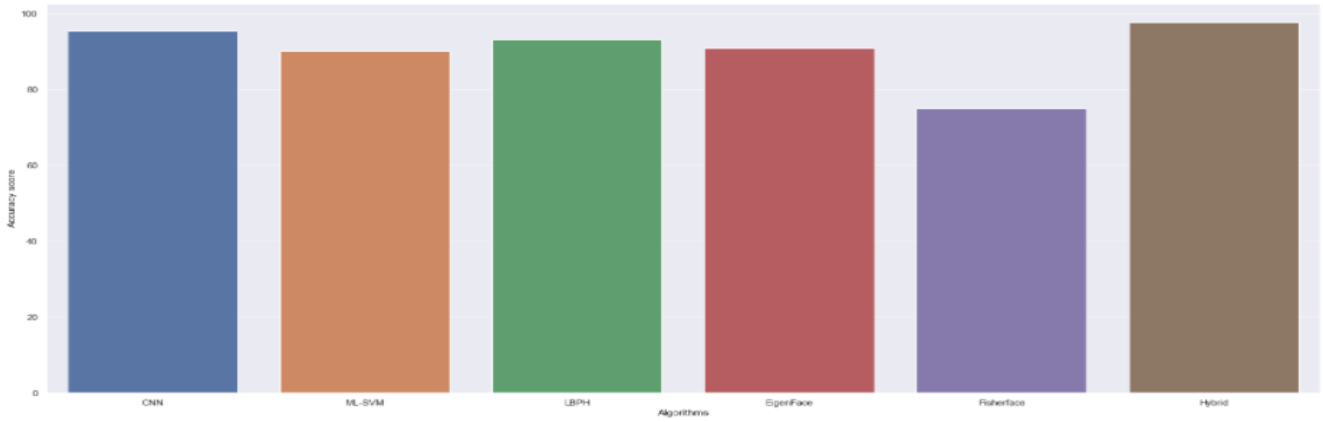
Hybrid Algorithm

```
scores = [score_cnn,acc_ml,acc_lb,acc_eigen,acc_fisher,acc_h]
algorithms = ["CNN","ML-SVM","LBPH","EigenFace","Fisherface","Hybrid"]

for i in range(len(algorithms)):
    print("The accuracy score achieved using "+algorithms[i]+" is: "+str(scores[i])+" %")
```

```
The accuracy score achieved using CNN is: 95.45454382896423 %
The accuracy score achieved using ML-SVM is: 90.0 %
The accuracy score achieved using LBPH is: 93.18181818181819 %
The accuracy score achieved using EigenFace is: 90.9090909090909 %
The accuracy score achieved using Fisherface is: 75.0 %
The accuracy score achieved using Hybrid is: 97.72727272727273 %
```

```
sns.set(rc={'figure.figsize':(25,10)})
plt.xlabel("Algorithms")
plt.ylabel("Accuracy score")

sns.barplot(algorithms,scores)
```

```
<AxesSubplot:xlabel='Algorithms', ylabel='Accuracy score'>
```



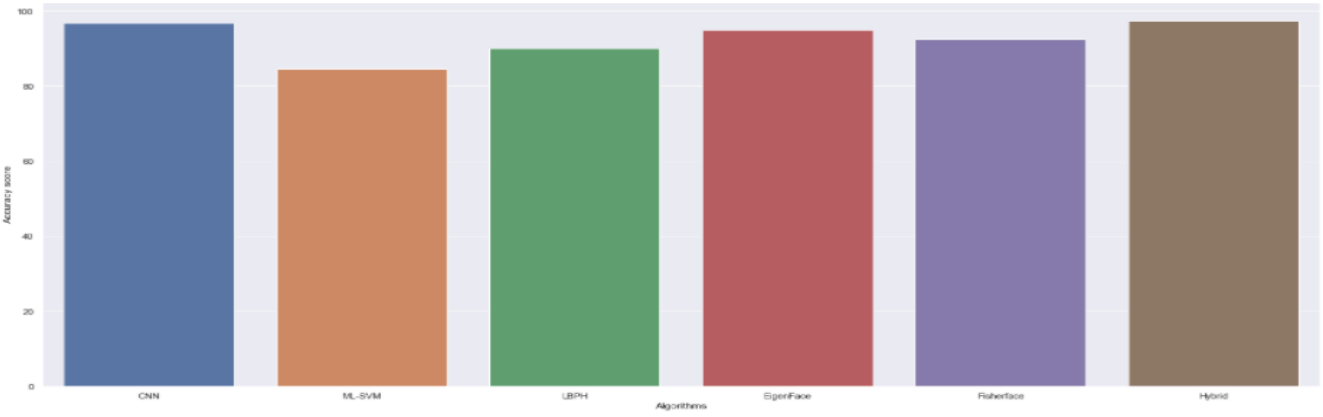
Facial Recognition Performance on Celebrity Dataset

```
scores = [score_cnn,acc_ml,acc_lb,acc_eigen,acc_fisher,acc_h]
algorithms = ["CNN","ML-SVM","LBPH","EigenFace","Fisherface","Hybrid"]

for i in range(len(algorithms)):
    print("The accuracy score achieved using "+algorithms[i]+" is: "+str(scores[i])+" %")
```

```
The accuracy score achieved using CNN is: 96.92307710647583 %
The accuracy score achieved using ML-SVM is: 84.61538461538461 %
The accuracy score achieved using LBPH is: 90.0 %
The accuracy score achieved using EigenFace is: 95.0 %
The accuracy score achieved using Fisherface is: 92.5 %
The accuracy score achieved using Hybrid is: 97.5 %
```

```
sns.set(rc={'figure.figsize':(25,10)})
plt.xlabel("Algorithms")
plt.ylabel("Accuracy score")

sns.barplot(algorithms,scores)
```

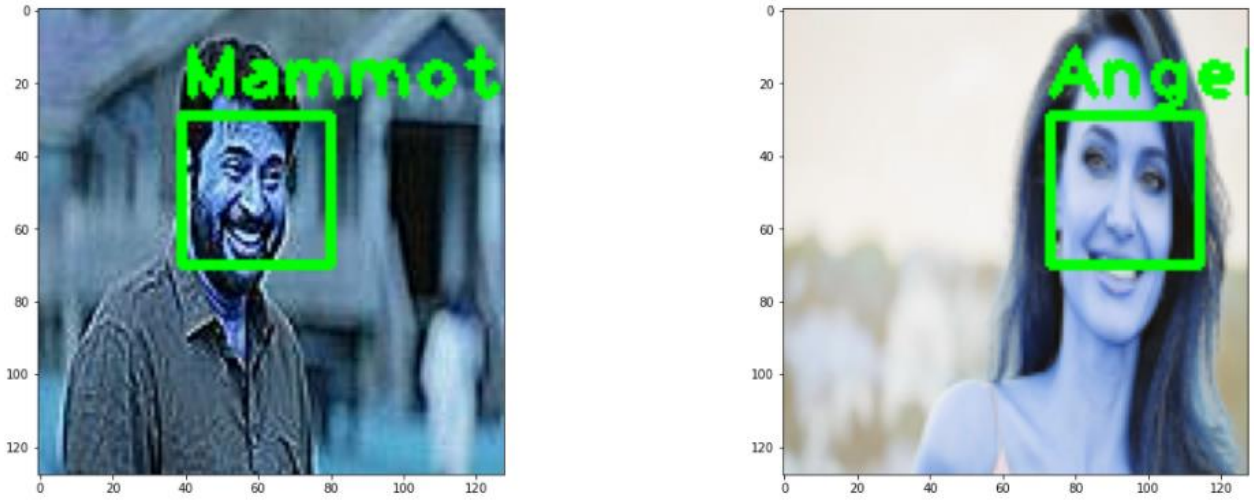
```
<AxesSubplot:xlabel='Algorithms', ylabel='Accuracy score'>
```



Facial Recognition Performance on Yale Dataset

```
cv2.waitKey(0)
cv2.destroyAllWindows()
```

```
Predicting images...
3
1
Prediction complete
```



Prediction Snippet of Celebrity Dataset

Live Face Recognition Using Hybrid Algorithm

In [48]:

```
from collections import Counter

face_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_frontalface_default.xml')
eye_cascade = cv2.CascadeClassifier(cv2.data.haarcascades + 'haarcascade_eye.xml')

video = cv2.VideoCapture(0);

while True:
    check, frame = video.read();
    faces = face_cascade.detectMultiScale(frame,scaleFactor=1.1, minNeighbors=5);
    for x,y,w,h in faces:
        frame = cv2.rectangle(frame, (x,y), (x+w,y+h), (0,255,0), 3);

        img=frame

    cv2.imshow('Face Detector', frame);

    key = cv2.waitKey(1);

    if key == ord('q'):
        break;

video.release();
cv2.destroyAllWindows();


data = im.fromarray(img)
img_cn=data.resize((256,256))
img_cn = np.array(img_cn)
img_cn = img_cn/255
img_cn = np.expand_dims(img_cn, axis=0)
prediction = cnn.predict(img_cn, batch_size=None,steps=1)
x=max(prediction)
y=max(x)
pos=0
for j in range(0,4):
    if(y==x[j]):
        pos=j
```

```
p_cnn=pos

d=[]
img_ml=data.resize((128,128))
img_ml=np.array(img_ml)
image_sharp1 = cv2.filter2D(src=img_ml, ddepth=-1, kernel=kernel)
img6=  image_sharp1.flatten()
d.append(img6)
x_test1=pd.DataFrame(d)
y_pred=model.predict(x_test1)
p_ml=y_pred[0]


img=data.resize((256,256))
img = np.array(img)

face, rect = detect_face(img)

label= face_recognizer.predict(face)
label1= face_recognizer1.predict(face)
label2= face_recognizer2.predict(face)

p=[]

p.append(label[0])
p.append(label1[0])
p.append(label2[0])
p.append(p_ml+1)
p.append(p_cnn+1)


occurence_count = Counter(p)
num= occurence_count.most_common(1)[0][0]

print(num)

label_text = subjects[num]


video = cv2.VideoCapture(0);
```

```
video = cv2.VideoCapture(0);


while True:
    check, frame = video.read();
    faces = face_cascade.detectMultiScale(frame,scaleFactor=1.1, minNeighbors=5);
    for x,y,w,h in faces:
        frame = cv2.rectangle(frame, (x,y), (x+w,y+h), (0,255,0), 3);
        cv2.putText(frame,label_text,(x, y), cv2.FONT_HERSHEY_PLAIN,1.3, (0, 255, 0), 2)

    cv2.imshow('Face Detector', frame);

    key = cv2.waitKey(1);

    if key == ord('q'):
        break;

video.release();
cv2.destroyAllWindows();
```

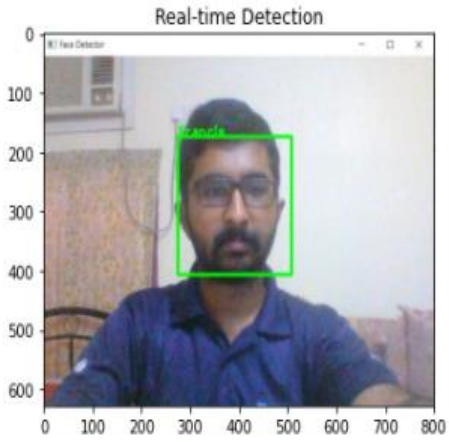
```
WARNING:tensorflow:When passing input data as arrays, do not specify `steps_per_epoch`/`steps` argument. Please use `batch_size
` instead.
```

Code for Real-time Detection using Hybrid Algorithm

Result Screenshot

```
]: image_result= im.open("C:\\Users\\Maria\\Desktop\\FRANCIS\\FALL SEM 21-22\\IMP\\our  project\\result.jpg")

   plt.imshow(image_result)

   plt.title("Real-time Detection")
```

```
]: Text(0.5, 1.0, 'Real-time Detection')
```



Live Detection Screenshot

4.2 EVALUATION OF RESULT

Data	Algorithms ->	CNN	SVM	LBPH	Eigenface	Fisherface	Hybrid
Celebrity	**Accuracy Score**	95.45%	90.0%	93.18%	90.90%	75.0%	97.72%
Yale	**Accuracy Score**	96.92%	84.61%	90%	95.0%	92.5%	97.5%

Accuracy Score Comparison for Predictions

5. CONCLUSION AND FUTURE WORKS

The aim of this project is to explore about the different algorithms available for facial recognition such as Eigenface, Fisherface, and LBPH and also dive in to the scope of facial recognition using CNN and SVM as well. Furthermore, we were able to train the algorithms with celebrity dataset and Yale face dataset images, and successfully test the data as well. CNN was the pick of the algorithms which consistently performed with high accuracies in both the cases. In order to take our project a step higher, we developed a hybrid algorithm fusing the prediction capabilities of all the five algorithms mentioned above. As expected, hybrid algorithms outshines all the algorithms and performs highly effective in recognizing faces for both datasets. Finally we have performed real time detection using the hybrid algorithm. As a part of future works, this concept can be transformed into a system for any real world applications which requires facial recognition.

REFERENCES

[1] A. Ahmed, J. Guo, F. Ali, F. Deeba and A. Ahmed, "LBPH based improved face recognition at low resolution," 2018 International Conference on Artificial Intelligence and Big Data (ICAIBD), 2018, pp. 144-147, doi: 10.1109/ICAIBD.2018.8396183.

[2] A. U. Naik and N. Guinde, "LBPH Algorithm for Frontal and Side Profile Face Recognition on GPU," 2020 Third International Conference on Smart Systems and Inventive Technology (ICSSIT), 2020, pp. 776-779, doi: 10.1109/ICSSIT48917.2020.9214228.

[3] A. V. Sripriya, M. Geethika and V. Radhesyam, "Real Time Detection and Recognition of Human Faces," 2020 4th International Conference on Intelligent Computing and Control Systems (ICICCS), 2020, pp. 703-708, doi: 10.1109/ICICCS48265.2020.9121084.

[4] X. Zhao and C. Wei, "A real-time face recognition system based on the improved LBPH algorithm," 2017 IEEE 2nd International Conference on Signal and Image Processing (ICSIP), 2017, pp. 72-76, doi: 10.1109/SIPROCESS.2017.8124508.

[5] M. A. Abuzneid and A. Mahmood, "Enhanced Human Face Recognition Using LBPH Descriptor, Multi-KNN, and Back-Propagation Neural Network," in IEEE Access, vol. 6, pp. 20641-20651, 2018, doi: 10.1109/ACCESS.2018.2825310.

[6] V. Aza, Indrabayu and I. S. Areni, "Face Recognition Using Local Binary Pattern Histogram for Visually Impaired People," 2019 International Seminar on Application for Technology of Information and Communication (iSemantic), 2019, pp. 241-245, doi: 10.1109/ISEMANTIC.2019.8884216.

[7] V. P. Kshirsagar, M. R. Baviskar and M. E. Gaikwad, "Face recognition using Eigenfaces," 2011 3rd International Conference on Computer Research and Development, 2011, pp. 302-306, doi: 10.1109/ICCRD.2011.5764137.

[8] A. L. Machidon, O. M. Machidon and P. L. Ogrutan, "Face Recognition Using Eigenfaces, Geometrical PCA Approximation and Neural Networks," 2019 42nd International Conference on Telecommunications and Signal Processing (TSP), 2019, pp. 80-83, doi: 10.1109/TSP.2019.8768864.

[9] R. Mejía-Campos, D. Néjer-Haro, S. Recalde-Avincho, P. Rosero-Montalvo and D. Peluffo-Ordóñez, "Face Detection and Classification Using Eigenfaces and Principal Component Analysis: Preliminary Results," 2017 International Conference on Information Systems and Computer Science (INCISCOS), 2017, pp. 309-315, doi: 10.1109/INCISCOS.2017.59.

[10] I. U. Wahyu Mulyono, D. R. Ignatius Moses Setiadi, A. Susanto, E. H. Rachmawanto, A. Fahmi and Muljono, "Performance Analysis of Face Recognition using Eigenface Approach," 2019 International Seminar on Application for Technology of Information and Communication (iSemantic), 2019, pp. 1-5, doi: 10.1109/ISEMANTIC.2019.8884225.

[11] R. Rosnelly, M. S. Simanjuntak, A. Clinton Sitepu, M. Azhari, S. Kosasi and Husen, "Face Recognition Using Eigenface Algorithm on Laptop Camera," 2020 8th International Conference on Cyber and IT Service Management (CITSM), 2020, pp. 1-4, doi: 10.1109/CITSM50537.2020.9268907.

[12] A. S. Khan and L. K. Alizai, "Introduction to Face Detection Using Eigenfaces," 2006 International

Conference on Emerging Technologies, 2006, pp. 128-132, doi: 10.1109/ICET.2006.335908.

[13] Hyung-Ji Lee, Wan-Su Lee and Jae-Ho Chung, "Face recognition using Fisherface algorithm and elastic graph matching," Proceedings 2001 International Conference on Image Processing (Cat. No.01CH37205), 2001, pp. 998-1001 vol.1, doi: 10.1109/ICIP.2001.959216.

[14] B. W. Yohanes, R. Diaz Airlangga and I. Setyawan, "Real Time Face Recognition Comparison Using Fisherfaces and Local Binary Pattern," 2018 4th International Conference on Science and Technology (ICST), 2018, pp. 1-5, doi: 10.1109/ICSTC.2018.8528608.

[15] Lushen Wu, Peimin Wu, Fanwen Meng and Weijing Yu, "Study on face recognition with combined of fisher algorithm and support vector machine," The 2nd International Conference on Information Science and Engineering, 2010, pp. 5444-5447, doi: 10.1109/ICISE.2010.5691587.

[16] P. N. Belhumeur, J. P. Hespanha and D. J. Kriegman, "Eigenfaces vs. Fisherfaces: recognition using class specific linear projection," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 19, no. 7, pp. 711-720, July 1997, doi: 10.1109/34.598228.

[17] D. Wahyuningsih, C. Kirana, R. Sulaiman, Hamidah and Triwanto, "Comparison Of The Performance Of Eigenface And Fisherface Algorithm In The Face Recognition Process," 2019 7th International Conference on Cyber and IT Service Management (CITSM), 2019, pp. 1-5, doi: 10.1109/CITSM47753.2019.8965345.

[18] M. Khan, S. Chakraborty, R. Astya and S. Khepra, "Face Detection and Recognition Using OpenCV," 2019 International Conference on Computing, Communication, and Intelligent Systems (ICCCIS), 2019, pp. 116-119, doi: 10.1109/ICCCIS48478.2019.8974493.

[19] A. G. S. Conceição, D. M. Oliveira and M. P. Carvalho, "ORB Algorithm Applied to Detection, Location and Grasping Objects," 2018 Latin American Robotic Symposium, 2018 Brazilian Symposium on Robotics (SBR) and 2018 Workshop on Robotics in Education (WRE), 2018, pp. 176-181, doi: 10.1109/LARS/SBR/WRE.2018.00040.

[20] Vinay A., Avani S. Rao, Vinay S. Shekhar, Akshay Kumar C., K N Balasubramanya Murthy, S. Natarajan, Feature Extraction using ORB-RANSAC for Face Recognition, Procedia Computer Science,Volume70,2015,Pages174-184,ISSN1877-0509.

[21] K. Yan, S. Huang, Y. Song, W. Liu and N. Fan, "Face recognition based on convolution neural network," 2017 36th Chinese Control Conference (CCC), 2017, pp. 4077-4081, doi: 10.23919/ChiCC.2017.8027997.

[22] S. Khan, M. H. Javed, E. Ahmed, S. A. A. Shah and S. U. Ali, "Facial Recognition using Convolutional Neural Networks and Implementation on Smart Glasses," 2019 International Conference on Information Science and Communication Technology (ICISCT), 2019, pp. 1-6, doi: 10.1109/CISCT.2019.8777442.

[23] B. R. Ilyas, B. Mohammed, M. Khaled and K. Miloud, "Enhanced Face Recognition System Based on Deep CNN," 2019 6th International Conference on Image and Signal Processing and their Applications (ISPA), 2019, pp. 1-6, doi: 10.1109/ISPA48434.2019.8966797.

[24] R. M. Prakash, N. Thenmoezhi and M. Gayathri, "Face Recognition with Convolutional Neural Network and Transfer Learning," 2019 International Conference on Smart Systems and Inventive Technology (ICSSIT), 2019, pp. 861-864, doi: 10.1109/ICSSIT46314.2019.8987899.

[25] V. K. N. K. Pai, M. Balrai, S. Mogaveera and D. Aeloor, "Face Recognition Using Convolutional Neural Networks," 2018 2nd International Conference on Trends in Electronics and Informatics (ICOEI), 2018, pp. 165-170, doi: 10.1109/ICOEI.2018.8553969.

[26] D. Wang, H. Yu, D. Wang and G. Li, "Face Recognition System Based on CNN," 2020 International Conference on Computer Information and Big Data Applications (CIBDA), 2020, pp. 470-473, doi: 10.1109/CIBDA50819.2020.00111.

[27] L. Yuan, Z. Qu, Y. Zhao, H. Zhang and Q. Nian, "A convolutional neural network based on TensorFlow for face recognition," 2017 IEEE 2nd Advanced Information Technology, Electronic and

Automation Control Conference (IAEAC), 2017, pp. 525-529, doi: 10.1109/IAEAC.2017.8054070.

[28] Y. Shatnawi, M. Alsmirat and M. Al-Ayyoub, "Face Recognition using Eigen-Faces and Extension Neural Network," 2019 IEEE/ACS 16th International Conference on Computer Systems and Applications (AICCSA), 2019, pp. 1-7, doi: 10.1109/AICCSA47632.2019.9035343.

[29] P. Wagh, R. Thakare, J. Chaudhari and S. Patil, "Attendance system based on face recognition using eigen face and PCA algorithms," 2015 International Conference on Green Computing and Internet of Things (ICGCIoT), 2015, pp. 303-308, doi: 10.1109/ICGCIoT.2015.7380478.

[30] S. Mukhopadhyay and S. Sharma, "Real Time Facial Expression and Emotion Recognition using Eigen faces, LBPH and Fisher Algorithms," 2020 10th International Conference on Cloud Computing, Data Science & Engineering (Confluence), 2020, pp. 212-220, doi: 10.1109/Confluence47617.2020.9057985.

APPENDIX

https://drive.google.com/drive/folders/1KbnQ0PF7EaAcGAaHD3HT0tCV9L_BhmUO?usp=sharing

The following file contains the code of Facial Recognition. All the files are .ipynb files.