FACE RECOGNITION

A PROJECT REPORT

for

DATA MINING TECHNIQUES (ITE2006)

in

B.Tech – Information Technology and Engineering

by

RISHI JAIN (19BIT0241)

VAIBHAVA PRIYA N D (19BIT0276)

ANUSHKA R (19BIT0279)

Under the Guidance of

Dr. SENTHILKUMAR N C

Associate Professor, SITE

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

School of Information Technology and Engineering

June, 2021

DECLARATION BY THE CANDIDATE

I/We hereby declare that the project report entitled **"Face Recognition"** submitted by me/us to Vellore Institute of Technology University, Vellore in partial fulfillment of the requirement for the award of the course **Data Mining Techniques (ITE2006)** is a record of bonafide project work carried out by us under the guidance of **Dr. Senthilkumar N C.** I/We further declare that the work reported in this project has not been submitted and will not be submitted, either in part or in full, for the award of any other course.

Place : Vellore Signature

Date : June 7, 2021

VIT®

Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

School of Information Technology & Engineering [SITE]

CERTIFICATE

This is to certify that the project report entitled **"Face Recognition"** submitted by **Anushka R (19BIT0279)** to Vellore Institute of Technology University, Vellore in partial fulfillment of the requirement for the award of the course **Data Mining Techniques (ITE2006)** is a record of bonafide work carried out by them under my guidance.

Dr. Senthilkumar N C
GUIDE
Associate Professor, SITE

Face Recognition

Abstract

The face is one of the easiest ways to distinguish the individual identity of each other.

Face recognition is a personal identification system that uses personal characteristics of a person to identify the person's identity. Human face recognition procedure basically consists of two phases, namely face detection, where this process takes place very rapidly in humans, except under conditions where the object is located at a short distance away, the next is the introduction, which recognizes a face as an individual.

The face analysis method of the proposed algorithm can recognize faces in real-time and analyze facial physiognomy which is composed of inherent physiological characteristics of humans, and fortunes with regard to human's life. The proposed algorithm can draw the person's avatar automatically based on face recognition.—Clustering face images according to their latent identity has two important applications: (i) grouping a collection of face images when no external labels are associated with images, and (ii) indexing for efficient large scale face retrieval.

Keywords: Face detection, Clustering, Face analysis .

I. INTRODUCTION

Face recognition is a method of identifying or verifying the identity of an individual using their face. Face recognition systems can be used to identify people in photos, video, or in real-time.

Generally, this identification is used to access an application, system or service. It is a method of biometric identification that uses that body measures, in this case face and head, to verify the identity of a person through its facial biometric pattern and data. The technology collects a set of unique biometric data of each person associated with their face and facial expression to identify, verify and/or authenticate a person

Compared to other biometric technologies, such as fingerprint and iris recognition, the availability of face recognition technology has its unique advantages, for example, the capture of face images usually does not interfere with people's normal activities. It is because of this that over the last decade, face recognition has become a popular research area in computer vision and one of the most successful applications of image analysis and understanding.

The applications of facial recognition range from a static, controlled "mug-shot" verification to a dynamic, uncontrolled face identification in a cluttered background (e.g., airport). The most popular approaches to face recognition are based on either (i) the location and shape of facial attributes, such as the eyes, eyebrows, nose, lips, and chin and their spatial relationships, or (ii) the overall (global) analysis of the face image that represents a face as a weighted combination of a number of canonical faces. While the verification performance of the face recognition systems that are commercially available is reasonable, they impose a number of restrictions on how the facial images are obtained, sometimes requiring a fixed and simple background or special illumination. These systems also have difficulty in recognizing a face from images captured from two drastically different views and under different illumination conditions. The face itself, without any contextual information, is a sufficient basis for recognizing a person from a large number of identities with a high level of confidence.

II. BACKGROUND

Earlier facial recognition technology was considered as an idea of science fiction. But in the past decade, facial recognition technology has not only become real — but it's widespread. Today, people can easily read articles and news stories about facial recognition everywhere. Here is the history of facial recognition technology and some ideas about its bright future. Facial recognition technology along with AI (Artificial Intelligence) and Deep Learning (DL) technology are benefiting several industries. These industries include law enforcement agencies, airports, mobile phone manufacturing companies, home appliance manufacturing companies, etc. Nowadays even retailers are using AI-based facial recognition technology to prevent violence and crime. Airports are getting a better-secured environment, mobile phone makers are using face recognition to bring the biometric security feature in the devices.

Facial recognition is more than 50 years old.A research team led by Woodrow W Bledsoe ran experiments between 1964 and 1966 to see whether 'programming computers' could recognize human faces.The team used a rudimentary scanner to map the person's hairline, eyes, and nose. The task of the computer was to find matches.It wasn't successful.Bledsoe said: "The face recognition problem is made difficult by the great variability in head rotation and tilt, lighting intensity and angle, facial expression, aging, etc."

The manually recorded metrics could be later saved within a database. And when the new photograph of an individual was entered into the system, it was able to get the most closely resembled image via database. During this period, face recognition was untouched by technology and computer processing power. Still, it was the first and foremost step taken by Bledsoe to prove that face recognition was a practical biometric.

III. Literature Survey

[1]. Review Of Existing Algorithms For Face Detection And Recognition

Mas Idayu Md Sabri ,December 2009

Five different methods in face detection and recognition have been reviewed, namely, PCA, LDA, Skin Colour, Wavelet and Artificial Neural Network. There are four parameters that are taken into account in this review, which are size and types of database, illumination tolerance, facial expressions variations and pose variations. No conclusion is made on which algorithm is the best for specific tasks. The performance of the algorithms depends on numerous factors to be taken into account. Instead of using these algorithms solely, they can be improved or enhanced to become a new method or hybrid method that yields a better performance.

[2]. Face Recognition Using Principle Component Analysis Kyungnam Kim ,2010

The eigenface system to perform face recognition gave the basic idea of PCA. Although the face recognition results were acceptable, the system only using eigenfaces might not be applicable as a real system. It needs to be more robust and to have other discriminant features. The results showed that the system gave correct identification for the known faces which are trained by the system. The noticeable fact is that the bigger the size of training set is, the less the reconstruction errors are.

[3]. Face Recognition Using Pca And Eigen Face Approach Abhishek Singh, Saurabh Kumar, May 2012

The efficiency was found to 75.83%.A person with different expression was collected as a dataset.It implemented the face recognition system using Principal Component Analysis and Eigen face approach . Observe that normal expressions are recognized as face efficiently because facial features are not changed much in that case and in other cases where facial features are changed efficiency is reduced in recognition.

[4]. Face Recognition Based On Vector Quantization Using Fuzzy Neuro Clustering

Author: Elizabeth B. Varghese, M. Wilscy

Year: 2013

A new Fuzzy Neuro approach to Face Recognition based on the Vector Quantization method is presented. A simple and efficient codebook design algorithm for face recognition using vector quantization is proposed. The codebook is created from two different codebooks. One codebook is created by the intensity variation among the pixels. The other codebook is created from the face images using Integrated Adaptive Fuzzy Clustering (IAFC). The performance of the proposed algorithm is demonstrated by using publicly available AT&T database, Yale database, Indian Face database and a small face database, DCSKU database.These two codebooks are combined to get the proposed codebook. From the results it is clear that the proposed method is more efficient than most of the existing Face Recognition systems. For practical applications of face recognition, not a simple recognition rate but a False Acceptance Rate (FAR) and a False Rejection Rate (FRR) are more important. Equal Error Rate (ERR) is calculated from the plots of FAR and FRR. The ERR for the proposed approach is low for all the test databases used. Hence the proposed system is better for both recognition and rejection than the method that uses the SOM codebook.

[5]. On Using a Pre-clustering Technique to Optimize LDA-Based Classifiers for

Appearance-Based Face Recognition

Authors: Sang-Woon Kim and Robert P.W. Duin

Year:2013

It proposes a new method that improves the performance of LDA-based classification by simply increasing the number of (sub)-classes through clustering a few classes of the training set prior to the execution of LDA. This is based on the fact that the eigenspace of the training set consists of the range space and the null space, and that the dimensionality of the range space increases as the number of classes increases. Therefore, when constructing the transformation matrix, through minimizing the null space, the loss of discriminative information resulting from this space can be minimized. To select the classes to be clustered, in the present paper, the intraset distance is employed as a criterion and the k-means clustering is performed to divide

them. The experimental results for an artificial data set of XOR-type samples and a well-known benchmark face database of Yale demonstrate that the classification efficiency of the proposed method could be improved.

[6]. Use Of Haar Cascade Classifier For Face Tracking System In Real Time Video.

Authors: K .H. Wanjale, Amit Bhoomkar, Ajay Kulkarni, Somnath Gosavi

Year:2013

We present information about the face detection and tracking system with real time video as an input. The working method of this system is entirely divided into two main modules. The face recognition and detection from the video is the first module while the tracking is the second module. To detect the face in the image, Face Name Graph Matching algorithm is used. This algorithm involves various methods such as the Haar-Cascade method, Open-cv libraries etc. Face clustering algorithm is used for tracking the face in the video. This system is mainly designed for security purposes. Video recorded in public areas for known persons or suspicious activities are used as an input for the system. System will identify and track one or more people captured in that video, this is taken as an output of this system. This paper involves working on a face tracking system, algorithms which are involved and results of the system.

[7]. Face Detection and Recognition Using K-means and Neural Network Methods

Author: Ashoka S.B.

Publication year: 2014

Proposed a novel algorithm for automatically detecting human faces in digital still color images, under non constrained scene conditions, such as presence of a complex background and uncontrolled illumination, where most of the local facial feature based methods are not stable. A face detector which had been developed in order to index a huge amount of video and images data and to cope with high-speed requirements. Only I frames (Intraframe transform coding) were analyzed from MPEG streams as they wanted to avoid costly decompression. Color segmentation of these I frames was performed at MPEG macro-block level(16 16 pixels).

Skin color filtering was performed, providing a macro-block binary mask which was segmented into non-overlapping rectangular regions containing contiguous regions of skin color macro blocks (binary mask segments areas). Then, the algorithm searched for the largest possible candidate face areas and iteratively reduced their size in order to scan the entire possible scheme has been substantially enhanced. It performs first color clustering of the original image, in order to extract a set of dominant colors and quantize the image according to this reduced set of colors. Then, a chrominance-based segmentation using an improvement of the method is performed.The finale set of codebook vectors is obtained using an iterative algorithm that originates from pattern recognition, The K-means algorithm. Also known as the Linde-Buzo- Gray algorithm.

[8]. Face Recognition System

Sayali Ghadge, Sana Khan, Sonam Vadsaria, 2014

PCA and Grayscale algorithms are used for this system after developing different algorithms i.e Eigenspace, Independent Component Analysis, Elastic bunch Graph Matching, Linear Discriminant Analysis.PCA was found better than grayscale algorithm in it accuracy.it focused on real time face recognition.

[9]. Comparison Of Different Face Recognition Algorithms

Pavan Pratap Chauhan, Vishal Kumar Lath, Mr. Praveen Rai April 2016

The paper has presented a comparison of different types of face recognition algorithms like PCA, LDA algorithm. The overall performance for verification of image by using these two algorithms we concluded that the LDA gives better performance in comparison with the PCA algorithm. The main difference between these algorithms when we perform Analysis and experimental results indicates that the PCA works well when the lightning variation is small. LDA works give better accuracy in facial expression. It was also noticed that PCA are time consuming as compared with the LDA algorithm.

[10]. Face Recognition: Issues, Methods And Alternative Applications Waldemar Wójcik, Konrad Gromaszek, Muhtar Junisbekov July 2016

Paper presented a brief Face Recognition: Issues, Methods and Alternative Applications survey of issues methods and applications in the area of face recognition. Mentioned methods used for face recognition including Classical face recognition algorithms, Artificial Neural Networks in face recognition, Gabor wavelet-based solutions, Face descriptor-based methods, 3D-based face recognition, . Video-based face recognition also showed the advantages of each method.

[11].Research on 3D face recognition method in cloud environment based on semi supervised clustering algorithm

Authors: Cuixia Li & Yingjun Tan& Dingbiao Wang & Peijie Ma

Year:2016

The semi supervised clustering algorithm is introduced to perform the cluster judgement to decide if the sample is labeled 3D face sample. According to the class probability or membership degree of 3D face samples to preselect partial samples, labeled 3D face samples training classifier is established, after the second-selection is processed for the preselected 3D face samples according to the classification confidence, the selected results and the predicted labels are added into labeled samples. The iterations are applied to the semi-supervised clustering algorithm to cluster the original labeled samples, new labeled samples and label the rest of the unlabeled samples. The iterative process is repeated until all samples have been marked, and the labeled samples are identified with 3D face data. The experiment combined the proposed method with cloud computing platform, the experimental results show that the proposed method has high recognition accuracy and efficiency.

[12]. Local Clustering Patterns in Polar Coordinate for Face Recognition.

Authors: Chih-Wei Lin1(B) and Kuan-Yin Lu

Year:2016

They proposed a novel local pattern descriptor called the Local Clustering Pattern (LCP) with low computational cost operating in the polar coordinate system for recognizing face. The local derivative variations with multi-direction are considered and that are integrated on the pairwise combinatorial direction. To generate the discriminative local pattern, the features of local

derivative variations are transformed into the polar coordinate system by generating the characteristics of distance (r) and angle (θ). LCP is an ensemble of several decisions from the clustering algorithm for each pixel in the polar coordinate system (P.C.S.). Differs from the existing local pattern descriptors, such as local binary pattern (LBP), local derivation pattern (LDP), and local tetra pattern (LTrP), LCP generates the discriminative local clustering pattern with low-order derivative space and low computational cost which are stable in the process of face recognition. The performance of the proposed method is compared with LBP, LDP, LTrP on the Extended Yale B and CAS-PEAL databases.

[13]. Face Recognition Using K-Means and RBFN

Author: Abhjeet Sekhon, Dr. Pankaj Agarwal

Year: IJCSMC, Vol. 6, Issue. 2, February 2017

Face recognition errand by using k-means with radial basis function network technique is used. The facial recognition system has proposed to recognize the faces which are enrolled in the database and which are not the parts of the database and dealt with AT&T database which join 400 pictures of 40 individuals. The appropriateness and exactness of K-means and Radial basis function network for face recognition errand is comprehended. With the assistance of HOG extraction strategy,prepared a data set of face feature values. To implement face recognition tasks , two artificial neural network techniques have been implemented: k-Means And RBFN. The proposed work was experimented on AT & T database. k-means algorithm implemented to make clusters of faces in the database. 10 different poses of one person were taken in the database. Clusters of the same images in one cluster were made. But in some cases k-means put the wrong face in the wrong clusters. So the efficiency level of k-means in clustering of faces in the database. It was observed that K-means algorithm can be used for face classification or making clusters of similar type faces in the database but independent face recognition errand cannot be conceivable by k-means. Large number of centroids increases the efficiency of k-means. Radial basis function network with k-means is accurate for face recognition tasks. But it is more accurate in small databases instead of large databases.

[14]. Face Identification and Clustering

Author: Atul Dhingra

Year:2017

The role of visual attributes using an agglomerative clustering algorithm to whittle down the search area where the number of classes is high to improve the performance of clustering. They observed that as they added more attributes, the clustering performance increases overall. The role of clustering in aggregating templates in a 1:N open set protocol using multi-shot video as a probe. They observed that by increasing the number of clusters, the performance increases with respect to the baseline and reaches a peak, after which increasing the number of clusters causes the performance to degrade. Experiments are conducted using recently introduced unconstrained IARPA Janus IJB-A, CS2, and CS3 face recognition datasets.

[15]. Face Recognition System With Face Detection

M.Vineetha Sai G.Varalakshmi, G.Bala Kumar, J.Prasad, 2017

The system with manual face detection and automatic face recognition did not have a recognition accuracy over 90%, due to the limited number of eigenfaces that were used for the PCA transform. an eye detection technique, was implemented to further normalise the segmented face image, performance will increase to levels comparable to the manual face detection and recognition system. The fully automated frontal view face detection system displayed virtually perfect accuracy.

[16]. Deep Face Recognition

Omkar M. Parkhi, Andrea Vedaldi, Andrew Zisserman, 2017

The accuracy was above 85% in all test cases and 99.63% was found for FaceNet[17] + Alignmen method. CNN architectures for face identification and verification are used . Labeled Faces in the Wild dataset (LFW) and YouTube Faces (YTF) are found efficient for data set.state

of art comparison was done for LFW AND YTF and LFW was found to to be more accurate.it showed that a deep CNN, without any embellishments but with appropriate training, can achieve results comparable to the state of the art.

[17]. Research On Face Feature Extraction Based On K-mean Algorithm

Author: Pengcheng Wei1, Zhen Zhou , Li Li and Jiao Jiang

Year:2018

Face recognition technology is an application technology of information security, which is a kind of multidisciplinary technology, such as comprehensive mathematics, pattern recognition, and biological characteristics. With the development of technology applications, the requirements for accuracy and anticounterfeiting of face recognition are also increasing. In this paper, the K-mean algorithm is used to analyze the face features. Firstly, the biometric features of the face are extracted, and then the K-mean method is used to cluster the face features. Lastly, the SVM method is used to classify. At present, face recognition can achieve higher accuracy under control and cooperation conditions, but under non-control and non-coordination conditions, face recognition is still a very challenging topic. Facial features are easily affected by factors such as

illumination and expression, which leads to a sharp decline in recognition rate. Therefore, it is of great practical significance to develop a feature extraction algorithm with high robustness and ability to extract features with better representation ability. In this paper, different data sets of different facial expressions, light, and scars are designed. The K-mean method is used to cluster different data sets, and SVM is used for classification research. The comparative analysis results show that the method designed by this paper can be improved. The recognition rate of different data sets, especially for expression data sets, has a different expression recognition rate of 91%. The results show that using the K-mean method can greatly improve the classification effect.

[18]. Face Detection And Recognition Using K Means And Enhanced K-means Algorithm With Ann

Dr. Hanumanthappa M, Dr. Ashoka S B, June 2018

The dataset is 200 images of the KDEF (Karolinska Directed Emotional Face Database) database. A novel approach of building artificial neural networks for face recognition is being applied by means of providing training to the neural network by extracted feature set rather than pattern recognition which reduces the overhead and complexity and makes the machine intelligent enough to sustain the fault tolerances. The K-Means algorithm for facial expression recognition gives better results than the K-Means algorithm.

[19]. Facial Expression Recognition Using Data Mining Algorithm

November 2018

This system used a data mining statistical approach using information gain and gini index in order to to recognize the correct emotion. Accuracy of the experiment is 80.9%. The JAFFE database is used as a dataset and template matching is done based on FCP's calculation.

[20]. Enhanced Face recognition system: Integration of Collaborative Representation based Classification (CRC) _KNN

Author: Vinodpuri Rampuri Gosavi, Anil Kishanrao Deshmane, Ganesh Shahuba Sable Journal: International Journal of Electronics, Communications, and Measurement Engineering, January 2019

Face analysis is an efficient method to detect and verify the faces. In this research article, the experiment is performed on the Yale database and the results are acquired from the stimulation tool MATLAB. The performance parameters are accurate, processing time, random noise and random occlusion. The obtained solutions are better as compared with the performance of Sparse Representation based Classification (SRC) in terms of accuracy and the procedure of the classification is performed in a short period of time. CRC with KNN improves the overall efficacy of the face recognition system. A comparison of performance is described and it is proven that the proposed method results give the enhancement in the overall performance of face recognition and accuracy value is 99%. The performance parameters are accurate, processing time, random noise and random occlusion.

[21].A Discrete-Time Projection Neural Network for Sparse Signal Reconstruction with Application to Face Recognition

Author: Bingrong Xu, Qingshan Liu and Tingwen Huang

Journal: IEEE Transactions on Neural networks and learning systems, January 2019

This paper deals with sparse signal reconstruction by designing a discrete-time projection neural network. To solve the L1-minimization problem, an iterative algorithm is proposed based on the discrete time projection neural network. A projection neural network based iterative method for sparse signal reconstruction with application to face recognition is used

The stability and convergence of the neural network is proven by using the Lyapunov method. Experimental results on the signal reconstruction show that the proposed algorithm is robust to different sparsity levels and amplitudes of signals. Furthermore, various public data sets are used to test the performance of the proposed algorithm for face recognition, which demonstrates that the PNNBIM not only achieves better recognition results compared with other algorithms.

[22]. Deep Unified Model for Face Recognition Based on Convolution Neural Network and Edge Computing

Author: Muhammad zeeshan khan, Saad harous, Saleet ul Hassan, Muhammad Usman Ghani Khan, Razi Iqbal and Shahid Mumtaz

Journal: IEEE Access, May 2019

This paper proposes an algorithm for face detection and recognition based on convolution neural networks (CNN), which outperform the traditional techniques. In order to validate the efficiency of the proposed algorithm, a smart classroom for the student's attendance using face recognition was proposed. The face recognition system is trained on publicly available labeled faces in the wild (LFW) dataset. The system can detect approximately 35 faces and recognizes 30 out of them from the single image of 40 students. The proposed

system achieved 97.9% accuracy on the testing data. Moreover, generated data by smart classrooms is computed and transmitted through an IoT-based architecture using edge computing. A comparative performance study shows that our architecture outperforms in terms of data latency and real time response

[23]. Face Recognition using Deep Neural Network across Variations in Pose and illumination.
Author: S.Meenakshi, M. Siva Jothi, D. Murugan.

Journal: International Journal of Recent Technology and Engineering (IJRTE), June 2019.

ORL face dataset is used as the dataset. Deep neural network data mining functionality is used here and the accuracy is 98.75%. Merits is Convolutional Neural Network (CNN) is chosen. The CNN architecture is designed for handling various situations like variation of poses and illumination variation. In this paper, a new method for face recognition using deep neural networks is presented. A CNN is designed with three convolution layers and a sub sampling layer. First convolution layer, C1, detects the edges from the face image. Second convolution layer, C2, detects the simple shape using edge features obtained in the first layer. Higher level features are extracted at the third convolution layer. The overall efficiency is obtained using different architectures.

[24]. Deep face Recognition for Biometric Authentication

Author: Maheen Zulfiqar, Fatima Syed,Muhammad Jaleed
Khan

Journal: International conference on Electrical, communication and computer Engineering, July 2019

KNN Viola-Jones algorithm and PCA is used in this paper. 98.76% recognition of a suspect in smart city video streams captured in public areas can be promptly reported to the concerned authority for a quick action. Security issues such as presentation attack, adversarial attack and template attack are developing to threaten the security of deep face recognition systems. Presentation attack with 3D silicone mask, which exhibits skin-like appearance and facial motion, challenges current anti-spoofing methods. Although

adversarial perturbation detection and mitigation methods are recently proposed, the root cause of adversarial vulnerability is unclear and thus new types of adversarial attacks are still upgraded continuously. The stolen deep feature template can be used to recover its facial appearance, and how to generate a cancelable template without loss of accuracy is another important issue.

[25]. Research on Inception Module Incorporated Siamese Convolutional Neural Networks to Realize Face Recognition

Author: Xian-Feng Xu, Li Zhang, Chen-Dong Duan and Yong
Lu Journal: IEEE Access, December 2019

The performance of face recognition can be affected by a series of uncontrollable factors, such as illumination, expression, posture and occlusion, which restricts its real-world applications. Therefore, improving the robustness of face recognition to environmental changes became an urgent problem. To have a better robustness against external environment interference, increase the processing speed of the data sets, and solve problems such as over fitting caused by fewer data sets, an improved Inception Module Incorporated Siamese Convolutional Neural Networks (IMISCNN) is proposed in this paper. A cyclical learning rate strategy is also introduced in IMISCNN for better model convergence. Inception Module Incorporated Siamese Convolutional Neural Networks (IMISCNN) is developed based on effective reduction of external interference and better features extraction by adopting the Siamese network structure.

[26]. Illumination-robust face recognition based on deep convolution al neural networks

architectures

Author: Bendjillali Ridha Ilyas, Khaled Merit, Mohammed Bellingham, Abdelmalik

taleb ahmed

Journal: Indonesian Journal of Electrical Engineering and Computer Science, 2019

Yale Face Database and CMU PIE Database is used as the dataset. ViolaJones algorithm is used to efficiently detect various parts of the human faces such as mouth, eyes, nose, eyebrows, 10

eyebrows, lips, ears, etc. The Viola-Jones face detection algorithm, facial image enhancement using Modified Contrast Limited Adaptive Histogram Equalization algorithm (M-CLAHE), and feature learning for classification. The face recognition system is performed in three main steps: (1) Facial image enhancement, (2) Face detection, (3) Extraction of facial features and classification. Finally, the comparison with the other methods on both databases shows the robustness and effectiveness of the proposed approach. The Inception-v3 architecture has achieved an accuracy rate of 99, 44% and 99, 89% respectively.

[27]. Face Recognition Systems: A Survey

Yassin Kortli, Maher Jridi 1, Ayman Al Falou 1 and Mohamed Atri

Published- 7 January 2020

The main purpose of techniques such as HOG, LBP, Gabor filters, BRIEF, SURF, and SIFT is to discover distinctive features, which can be divided into two parts: (1) local appearance-based techniques, which are used to extract local features when the face image is divided into small regions (including HOG, LBP, Gabor filters, and correlation filters); and (2) key-points-based techniques, which are used to detect the points of interest in the face image, after which features' extraction is localized based on these points, including BRIEF, SURF, and SIFT. In the context of face recognition, local techniques only treat certain facial features, which make them very sensitive to facial expressions and occlusions [4,14,37,50–53]. The relative robustness is the main advantage of these feature-based local techniques. Additionally, they take into account the peculiarity of the face as a natural form to recognize a reduced number of parameters. Another advantage is that they have a high compaction capacity and a high comparison speed. The main disadvantages of these methods are the difficulty of automating the detection of facial features and the fact that the person responsible for the implementation of these systems must make an arbitrary decision on really important points.

[28]. Face Image Recognition Based on Convolutional Neural Network Author: Guangxin Lou, Hongzhen Shi

Journal: China Communications, February 2020

Convolution neural network, as a common model structure in machine learning, has achieved excellent results in image enhancement, image fusion, image processing, and image recognition and so on, and is favored by experts and scholars. The discarded image information is recycled and utilized on the original convolutional neural network model, and the improved model can further improve the performance of machine self-learning and classification., compared with the five algorithms, the image recognition rate of the three data sets in this paper is outstanding, about 5.27% higher than LCA algorithm, about 1.5% higher than Eigen face, about 0.7% higher than 2DPCA, and about 0.51% higher than Fisher face. The improved image recognition rate algorithm is 0.27% higher than that of convolution neural networks. The improved model has a good recognition effect on ORL face image. Compared with other algorithms, VGG16 multi-layer fusion convolution model can further improve the recognition rate of images.

[29]. Incremental methods in face recognition

Suresh Madhvan & Nitin Kumar

Published 12 August 2020

The contribution of this paper is three-fold: (a) a novel taxonomy of the Incremental methods have been proposed (b) a review of the face datasets used in Incremental face recognition have been carried out and (c) a performance analysis of the Incremental face recognition methods over various face datasets is also presented. Important conclusions have been drawn that will help the researchers in making suitable choices amongst various methods and dataset.

[30]. Improving Face Recognition by Clustering Unlabeled Faces in the Wild

Author: Aruni RoyChowdhury, Xiang Yu, Kihyuk Sohn, Erik Learned-Miller, and Manmohan Chandraker .

Year: 2020

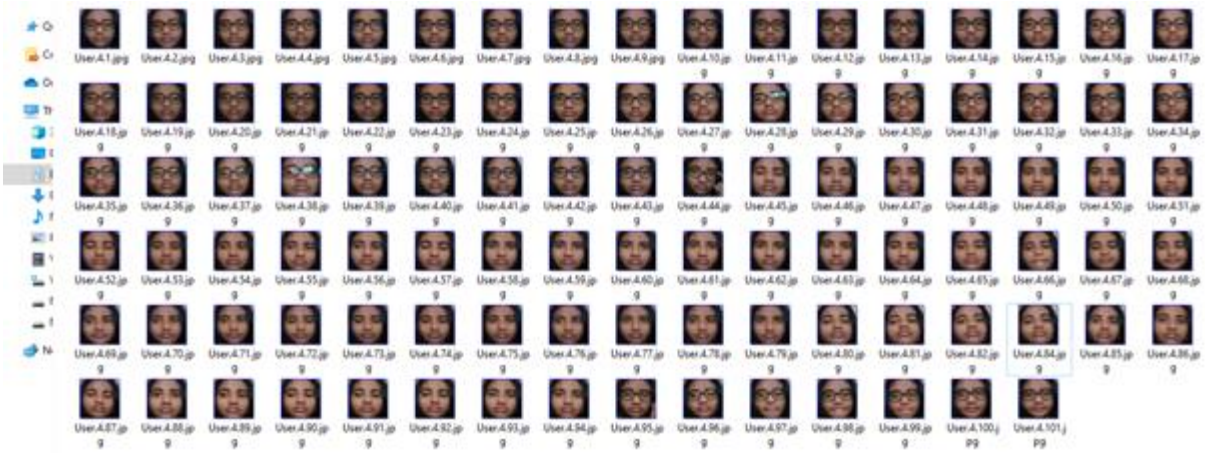
A novel identity separation method based on extreme value theory. It is formulated as an out-of-distribution detection algorithm, and greatly reduces the problems caused by overlapping-

identity label noise. Considering cluster assignments as pseudo-labels, they must also overcome the labeling noise from clustering errors. They proposed a modulation of the cosine loss, where the modulation weights correspond to an estimate of clustering uncertainty. Extensive experiments on both controlled and real settings demonstrate the method's consistent improvements over supervised baselines, e.g., 11.6% improvement on IJB-A verification.

The experimental results show consistent performance gains across various scenarios and provide insights into the practice of large-scale face recognition with unlabeled data – (1) we require comparable volumes of labeled and unlabeled data to see significant performance gains, especially when several million labeled samples are available; (2) overlapped identities between labeled and unlabeled data is a major concern and needs to be handled in real-world scenarios; (3) along with large amounts of unlabeled data, greater gains are observed if the new data shows certain domain gap w.r.t. the labeled set; (4) incorporating scalable measures of clustering uncertainty on the pseudo-labels is helpful in dealing with label noise. Overall, learning from unlabeled faces is shown to be an effective approach to further improve face recognition performance.

IV. DATASET DESCRIPTION & SAMPLE DATA

We have used customized dataset from pc's webcam



V. PROPOSED ALGORITHM WITH FLOWCHART

Architecture:

The architecture of the System consists of three modules, namely:

- Enrolment Module
- Database
- Identification Module

Enrolment Module

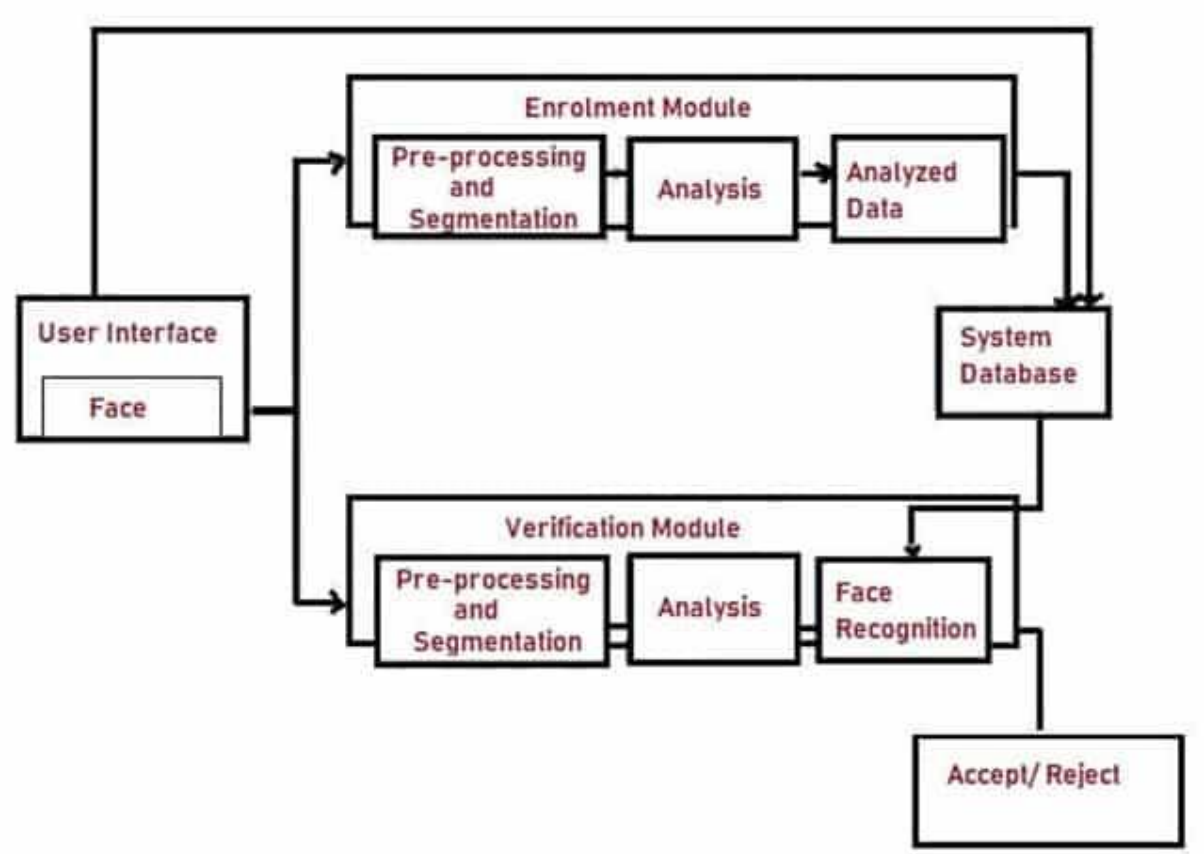
It scans and captures analog or digital image of a living being.

Database

An entity which handles compression, processing, storage and also accounts for comparison of the captured data with stored data.

Identification Module

This module interfaces with the application system.



Architecture of Facial Recognition System

Facial Recognition process generally includes three stages:

- Face Detection
- Feature Extraction
- Face Recognition

Face Detection

It accepts the image as an input and checks if 'Face' appears in the image and calculates its position on the image. The output of this stage is 'Patches' which contains 'Face' and Face alignment is done which acts as pre-processing stage for Feature Extraction.

Feature Extraction

Face Patch is transformed in to a set of Fiducial Points corresponding to their locations or it is transformed into vectors with specific dimension.

Face Recognition

This step includes recognition of Face from the database. When the system receives Face image, it undergoes Face Detection and Feature Extraction process. Then, the features are compared with each Face in the Database using the nodal points on the Face.

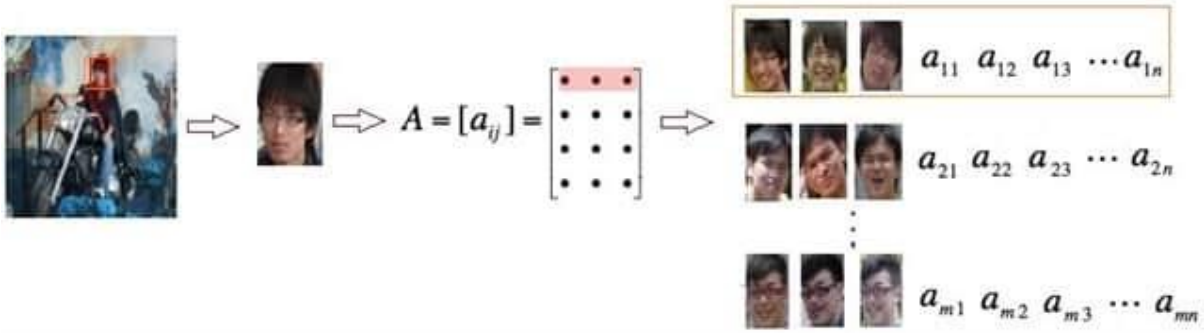
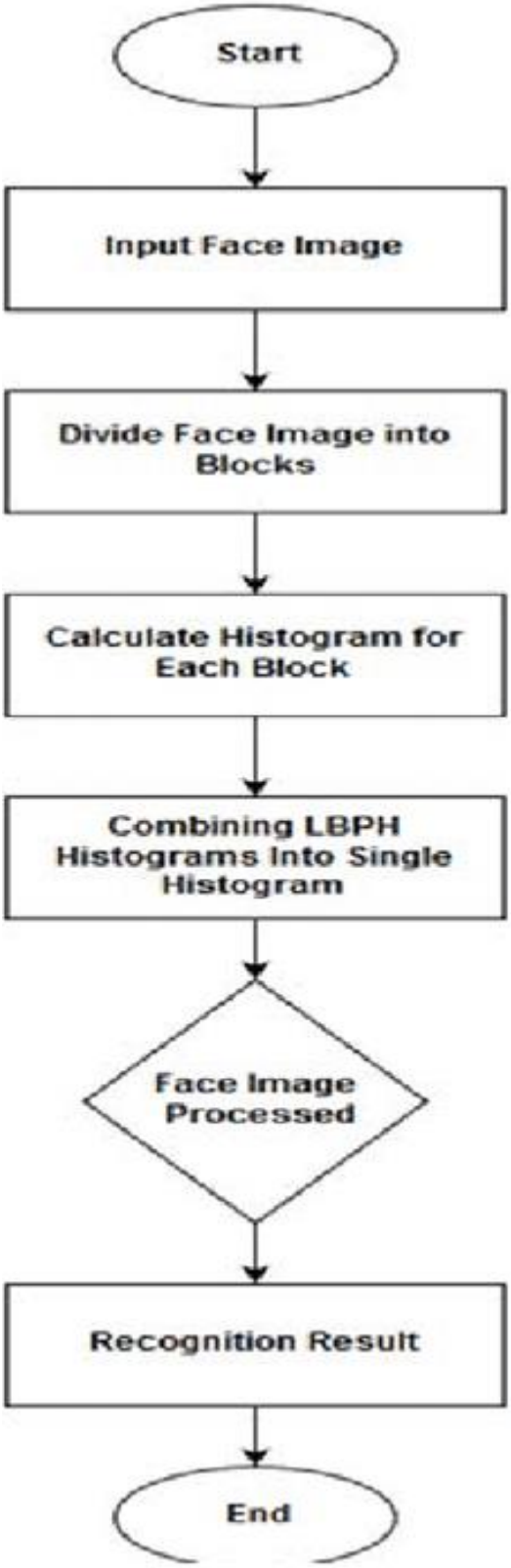


Fig. 4 – Different Stages of Facial Recognition System

Identification and Verification procedure is carried out where the system makes a probable identification of the Image and is verified whether the probability is True or False. i.e. comparison of the input vectors with the stored vectors in the Database occurs using different classification techniques


Start
Input Face Image
Divide Face Image into Blocks
Calculate Histogram for Each Block
Combining LBPH Histograms Into Single Histogram
Face Image Processed
Recognition Result
End

Proposed Method:

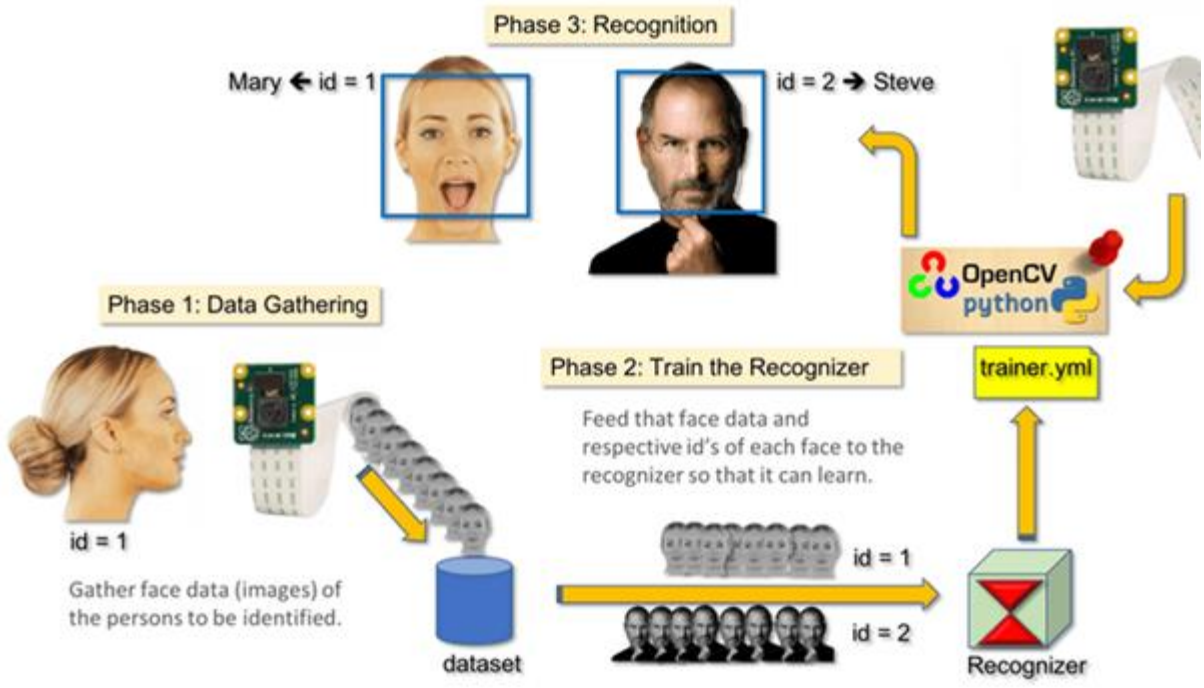
In this project we used,LBPH ,Haar-cascade algorithm,CNN,clustering (DBSCAN)to cluster the dataset

It is going to group similar vectors in an other word similar faces at the first time, and every group of similar faces of course represents a known person that we want to recognize afterwards

Clustering-based domain adaptation method designed for face recognition task in which the source and target domain do not share any classes. Our method effectively learns the discriminative target feature by aligning the feature domain globally, and, at the meantime, distinguishing the target clusters locally. Specifically, it first learns a more reliable representation for clustering by minimizing global domain discrepancy to reduce domain gaps, and then applies simplified spectral clustering method to generate pseudo-labels in the domain-invariant feature space, and finally learns discriminative target representation.

There are 3 very distinct phases:

- Face Detection and Data Gathering
- Train the Recognizer
- Face Recognition

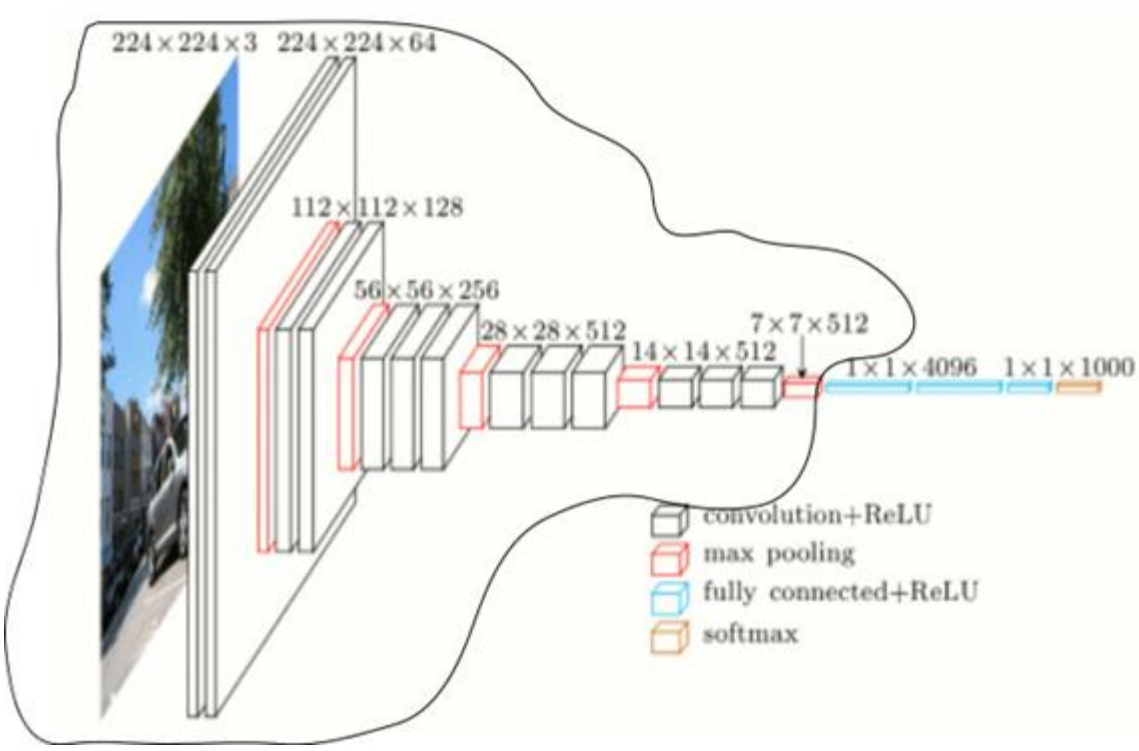


CNN:

Convolutional neural networks refer to a sub-category of neural networks: they, therefore, have all the characteristics of neural networks. However, CNN is specifically designed to process input images. Their architecture is then more specific: it is composed of two main blocks.

The first block makes the particularity of this type of neural network since it functions as a feature extractor. To do this, it performs template matching by applying convolution filtering operations. The first layer filters the image with several convolution kernels and returns "**feature maps**", which are then normalized (with an activation function) and/or resized.

This process can be repeated several times: we filter the features maps obtained with new kernels, which gives us new features maps to normalize and resize, and we can filter again, and so on. Finally, the values of the last feature maps are concatenated into a vector. This vector defines the output of the first block and the input of the second.

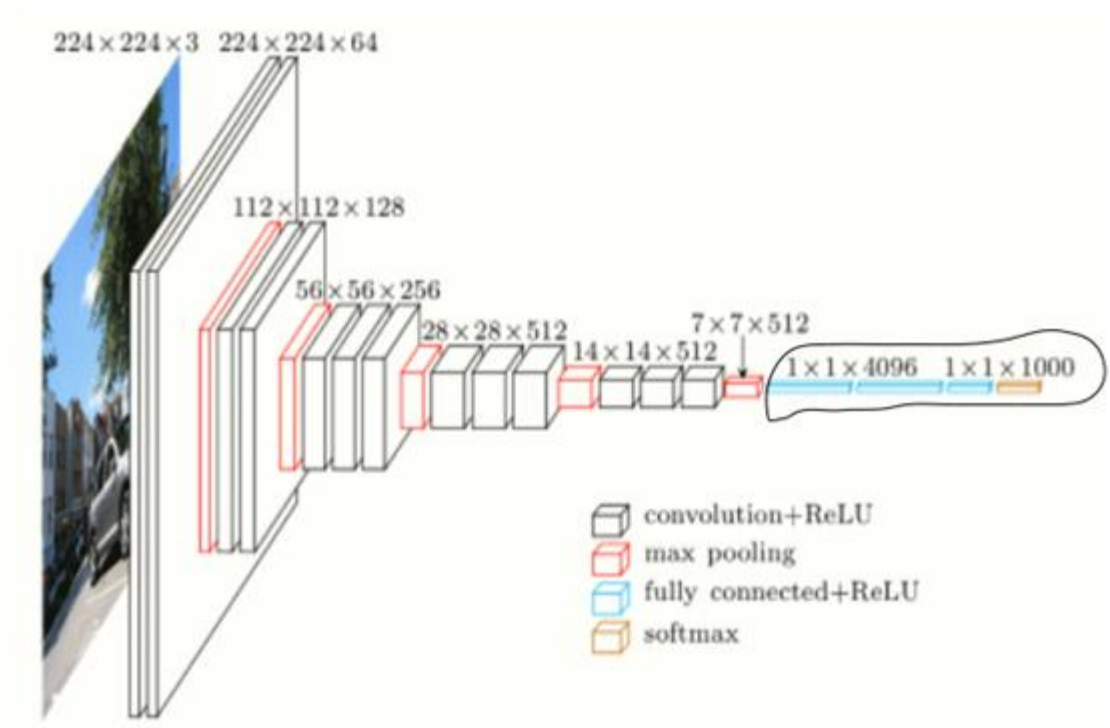


The first block is encircled in black

The second block is not characteristic of a CNN: it is in fact at the end of all the neural networks used for classification. The input vector values are transformed (with several linear combinations and activation functions) to return a new vector to the output. This last vector

contains as many elements as there are classes: element i represents the probability that the image belongs to class i. Each element is therefore between 0 and 1, and the sum of all is worth 1. These probabilities are calculated by the last layer of this block (and therefore of the network), which uses a **logistic function** (binary classification) or a **softmax function** (multi-class classification) as an activation function.

As with ordinary neural networks, the parameters of the layers are determined by gradient backpropagation: the **cross-entropy** is minimized during the training phase. But in the case of CNN, these parameters refer in particular to the image features.



The second block is encircled in black

The different layers of a CNN

There are four types of layers for a convolutional neural network: the **convolutional** layer, the **pooling** layer, the **ReLU correction** layer and the **fully-connected** layer.

The convolutional layer

The convolutional layer is the key component of convolutional neural networks, and is always at least their first layer.

Its purpose is to detect the presence of a set of features in the images received as input. This is done by convolution filtering: the principle is to "drag" a window representing the feature on the image, and to calculate the convolution product between the feature and each portion of the scanned image. A feature is then seen as a filter: the two terms are equivalent in this context.

The convolutional layer thus receives several images as input, and calculates the convolution of each of them with each filter. The filters correspond exactly to the features we want to find in the images.

We get for each pair (image, filter) a **feature map**, which tells us where the features are in the image: the higher the value, the more the corresponding place in the image resembles the feature.

Unlike traditional methods, features are not pre-defined according to a particular formalism (for example SIFT), but learned by the network during the training phase! Filter kernels refer to the convolution layer weights. **They are initialized and then updated by backpropagation using gradient descent**.

The pooling layer

This type of layer is often placed between two layers of convolution: it receives several feature maps and applies the pooling operation to each of them.

The pooling operation consists in **reducing the size** of the images while **preserving their important characteristics**.

To do this, we cut the image into regular cells, then we keep the maximum value within each cell. In practice, small square cells are often used to avoid losing too much information. The most common choices are **2x2 adjacent cells** that **don't overlap**, or **3x3 cells**, separated from each other by a step of 2 pixels (thus **overlapping**).

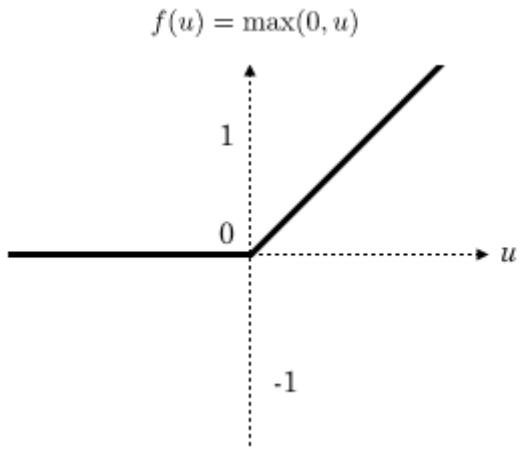
We get in output the same number of feature maps as input, but these are much smaller.

The pooling layer **reduces the number of parameters and calculations in the network**. This improves the efficiency of the network and avoids over-learning.

The maximum values are spotted less accurately in the feature maps obtained after pooling than in those received in input which is a big advantage.

The ReLU correction layer

ReLU (Rectified Linear Units) refers to the real non-linear function defined by *ReLU(x)=max(0,x)*. Visually, it looks like the following:



The ReLU correction layer replaces all negative values received as inputs by zeros. It acts as an **activation function**.

The fully-connected layer

The fully-connected layer is always the last layer of a neural network, convolutional or not — so it is not characteristic of a CNN.

This type of layer receives an input vector and produces a new output vector. To do this, it applies a **linear combination** and **then possibly an activation function** to the input values received.

The last fully-connected layer classifies the image as an input to the network: it returns a vector of size N, where N is the number of classes in our image classification problem. Each element of the vector indicates the probability for the input image to belong to a class.

To calculate the probabilities, the fully-connected layer, therefore, multiplies each input element by weight, makes the sum, and then applies an activation function (logistic if N=2, softmax if N>2). This is equivalent to multiplying the input vector by the matrix containing the

weights. The fact that each input value is connected with all output values explains the term fully-connected.

The convolutional neural network learns weight values in the same way as it learns the convolution layer filters: during the training phase, by **backpropagation of the gradient**.

The fully connected layer determines the relationship between the position of features in the image and a class. Indeed, the input table being the result of the previous layer, it corresponds to a feature map for a given feature: **the high values indicate the location** (more or less precise depending on the pooling) **of this feature in the image**. If the location of a feature at a certain point in the image is characteristic of a certain class, then the corresponding value in the table is given significant weight.

The parmetrization of the layers

A convolutional neural network differs from another by the way the layers are stacked, but also parameterized.

The layers of convolution and pooling have indeed hyperparameters, that is to say parameters whose you must first define the value.

The size of the output feature maps of the convolution and pooling layers depends on the hyperparameters.

Each image (or feature map) is W×H×D, where W is its width in pixels, H is its height in pixels and D the number of channels (1 for a black and white image, 3 for a colour image).

The convolutional layer has four hyperparameters:

1. The number of filters K

2. The size F filters: each filter is of dimensions F×F×D pixels.

3. The S step with which you drag the window corresponding to the filter on the image. For example, a step of 1 means moving the window one pixel at a time.

4. The Zero-padding P: add a black contour of P pixels thickness to the input image of the layer. Without this contour, the exit dimensions are smaller. Thus, the more convolutional layers are

stacked with P=0, the smaller the input image of the network is. We lose a lot of information quickly, which makes the task of extracting features difficult.

For each input image of size W×H×D, the pooling layer returns a matrix of dimensions Wc×Hc×Dc, where:

$$W_C = \frac{W-F+2P}{S} + 1$$

$$H_C = \frac{H-F+2P}{S} + 1$$

$$D_C = K$$

Choosing **P=F-1/2 and S=1** gives feature maps of the same width and height as those received in the input.

The pooling layer has two hyperparameters:

1. The size F of the cells: the image is divided into square cells of size F×F pixels.

2. The S step: cells are separated from each other by S pixels.

For each input image of size W×H×D, the pooling layer returns a matrix of dimensions Wp×Hp×Dp, where:

$$W_P = \frac{W-F}{S} + 1$$

$$H_P = \frac{H-F}{S} + 1$$

$$D_P = D$$

Just like stacking, the choice of hyperparameters is made according to a classic scheme:

- For the convolution layer, the filters are small and dragged on the image one pixel at a time. The zero-padding value is chosen so that the width and height of the input volume are not changed at the output. In general, we then choose F=3,P=1,S=1 or F=5,P=2,S=1
- For pooling layer, **F=2 and S=2** is a wise choice. This **eliminates 75% of the input pixels**. We can also choose F=3 and S=2: in this case, the cells overlap. Choosing larger cells causes too much loss of information and results in less good results in practice

Haar Cascade Algorithm:

It is a machine learning based approach where a cascade function is trained from a lot of positive and negative images. It is then used to detect objects in other images.

Here we will work with face detection. Initially, the algorithm needs a lot of positive images (images of faces) and negative images (images without faces) to train the classifier. Then we need to extract features from it. For this, Haar features shown in the below image are used. They are just like our convolutional kernel. Each feature is a single value obtained by subtracting sum of pixels under the white rectangle from sum of pixels under the black rectangle.

It applies each and every feature on all the training images. For each feature, it finds the best threshold which will classify the faces to positive and negative. Obviously, there will be errors or misclassifications. We select the features with minimum error rate, which means they are the features that most accurately classify the face and non-face images. (The process is not as simple as this. Each image is given an equal weight in the beginning. After each classification, weights of misclassified images are increased. Then the same process is done. New error rates are calculated. Also new weights. The process is continued until the required accuracy or error rate is achieved or the required number of features are found).

The final classifier is a weighted sum of these weak classifiers. It is called weak because it alone can't classify the image, but together with others forms a strong classifier.

LBPH algorithm:

Using the LBP combined with histograms we can represent the face images with a simple data vector.

1. **Parameters**: the LBPH uses 4 parameters:

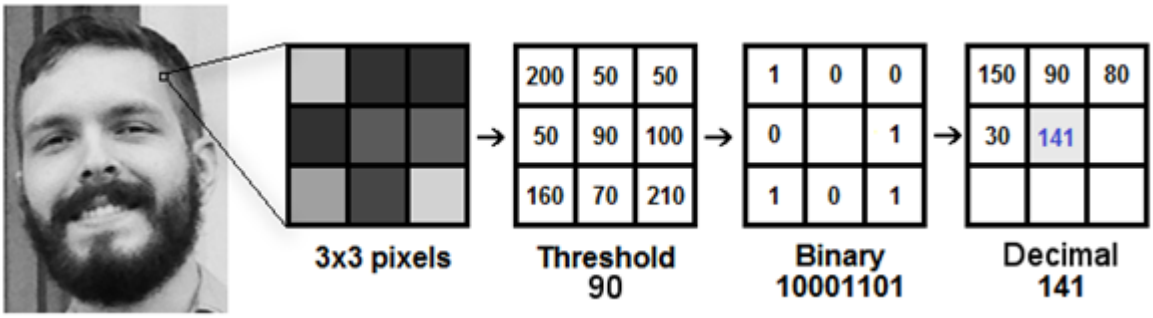
- **Radius**: the radius is used to build the circular local binary pattern and represents the radius around the central pixel. It is usually set to 1.
- **Neighbors**: the number of sample points to build the circular local binary pattern. Keep in mind: the more sample points you include, the higher the computational cost. It is usually set to 8.
- **Grid X**: the number of cells in the horizontal direction. The more cells, the finer the grid, the higher the dimensionality of the resulting feature vector. It is usually set to 8.
- **Grid Y**: the number of cells in the vertical direction. The more cells, the finer the grid, the higher the dimensionality of the resulting feature vector. It is usually set to 8.

Don't worry about the parameters right now, you will understand them after reading the next steps.

2. Training the Algorithm: First, we need to train the algorithm. To do so, we need to use a dataset with the facial images of the people we want to recognize. We need to also set an ID (it may be a number or the name of the person) for each image, so the algorithm will use this information to recognize an input image and give you an output. Images of the same person must have the same ID. With the training set already constructed, let's see the LBPH computational steps.

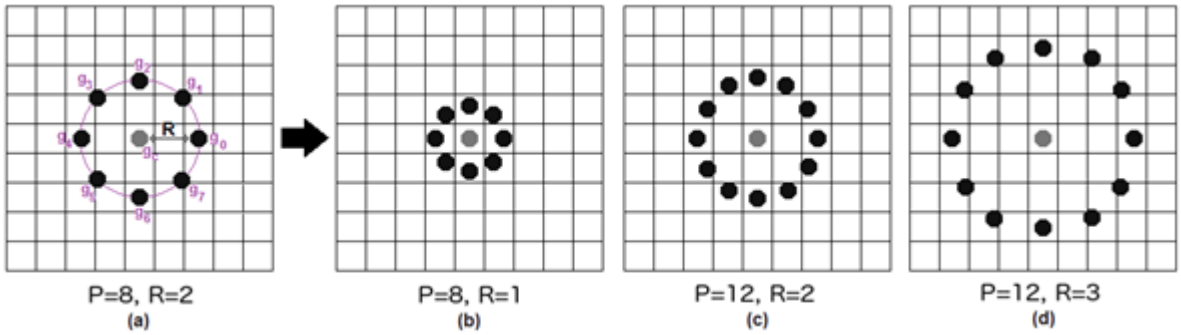
3. Applying the LBP operation: The first computational step of the LBPH is to create an intermediate image that describes the original image in a better way, by highlighting the facial characteristics. To do so, the algorithm uses a concept of a sliding window, based on the parameters **radius** and **neighbors**.

The image below shows this procedure:



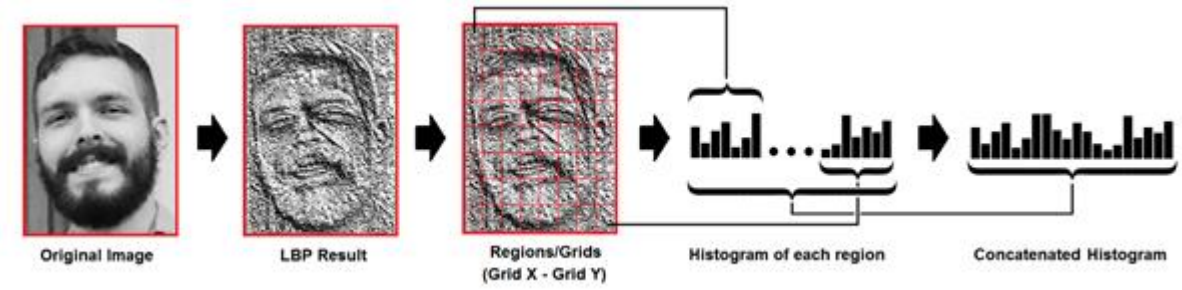
Based on the image above, let's break it into several small steps so we can understand it easily:

- Suppose we have a facial image in grayscale.
- We can get part of this image as a window of 3x3 pixels.
- It can also be represented as a 3x3 matrix containing the intensity of each pixel (0~255).
- Then, we need to take the central value of the matrix to be used as the threshold.
- This value will be used to define the new values from the 8 neighbors.
- For each neighbor of the central value (threshold), we set a new binary value. We set 1 for values equal or higher than the threshold and 0 for values lower than the threshold.
- Now, the matrix will contain only binary values (ignoring the central value). We need to concatenate each binary value from each position from the matrix line by line into a new binary value (e.g. 10001101). Note: some authors use other approaches to concatenate the binary values (e.g. clockwise direction), but the final result will be the same.
- Then, we convert this binary value to a decimal value and set it to the central value of the matrix, which is actually a pixel from the original image.
- At the end of this procedure (LBP procedure), we have a new image which represents better the characteristics of the original image.
- **Note**: The LBP procedure was expanded to use a different number of radius and neighbors, it is called Circular LBP.



It can be done by using **bilinear interpolation**. If some data point is between the pixels, it uses the values from the 4 nearest pixels (2x2) to estimate the value of the new data point.

4. Extracting the Histograms: Now, using the image generated in the last step, we can use the **Grid X** and **Grid Y** parameters to divide the image into multiple grids, as can be seen in the following image:



Based on the image above, we can extract the histogram of each region as follows:

- As we have an image in grayscale, each histogram (from each grid) will contain only 256 positions (0~255) representing the occurrences of each pixel intensity.
- Then, we need to concatenate each histogram to create a new and bigger histogram. Supposing we have 8x8 grids, we will have 8x8x256=16.384 positions in the final histogram. The final histogram represents the characteristics of the image original image.

The LBPH algorithm is pretty much it.

5. Performing the face recognition: In this step, the algorithm is already trained. Each histogram created is used to represent each image from the training dataset. So, given an input image, we perform the steps again for this new image and creates a histogram which represents the image.

- So to find the image that matches the input image we just need to compare two histograms and return the image with the closest histogram.
- We can use various approaches to compare the histograms (calculate the distance between two histograms), for example: **euclidean distance**, **chi-square**, **absolute value**, etc. In this example, we can use the Euclidean distance (which is quite known) based on the following formula:

$$D = \sqrt{\sum_{i=1}^{n}(hist1_i - hist2_i)^2}$$

- So the algorithm output is the ID from the image with the closest histogram. The algorithm should also return the calculated distance, which can be used as a '**confidence**' measurement. **Note**: don't be fooled about the 'confidence' name, as lower confidences are better because it means the distance between the two histograms is closer.
- We can then use a threshold and the 'confidence' to automatically estimate if the algorithm has correctly recognized the image. We can assume that the algorithm has successfully recognized if the confidence is lower than the threshold defined.

DBSCAN Algorithm:

The main concept of DBSCAN algorithm is to locate regions of high density that are separated from one another by regions of low density. Steps to measure density of a region:-

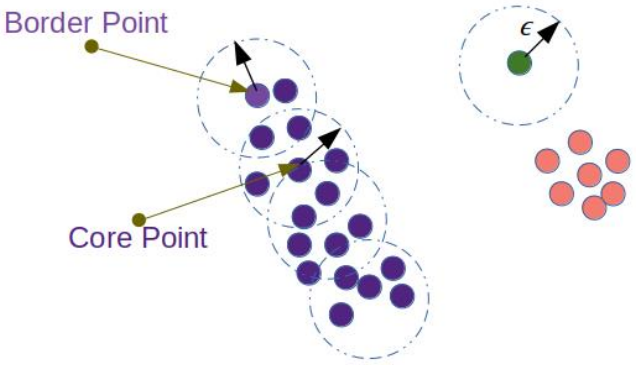
- Density at a point P: Number of points within a circle of Radius Eps (ϵ) from point P.
- Dense Region: For each point in the cluster, the circle with radius ϵ contains at least minimum number of points (MinPts).

The Epsilon neighborhood of a point P in the database D is defined as (following the definition from Ester et.al.)

$$N(p) = \{q \in D \mid dist(p, q) \leq \epsilon\} \cdots (1)$$

Following the definition of dense region, a point can be classified as a **Core Point** if $|N(p)| \geq$ MinPts. The Core Points, as the name suggests, lie usually within the interior of a cluster. A **Border Point** has fewer than MinPts within its ϵ-neighborhood (N), but it lies in

the neighborhood of another core point. **Noise** is any data point that is neither core nor border point. See the picture below for better understanding.



$$N_{Eps}(p) = \{q \in D \text{ such that } dist(p,q) \leq \epsilon\}$$

ϵ = 1 unit, MinPts = 7

Core and Border Points in a Database D. Green data point is Noise. (Source: Re-created by Author, Original Reference [2])

One problem of this approach could be — ϵ-neighborhood of the border points contain significantly less number of points than the core points. Since MinPts is a parameter in the algorithm, setting it to a low value to include the border points in the cluster can cause problem to eliminate the noise. Here comes the concept of density-reachable and density-connected points.

Directly Density Reachable: Data-point a is directly density reachable from a point b if —

1. |N (b)| ≥ MinPts; i.e. b is a core point.
2. a ∈ N(b) i.e. a is in the epsilon neighborhood of b.

Considering a border point and a core point, we can understand that notion of directly density reachable is not symmetric, because even though the core point falls in the epsilon neighborhood of border point, the border point doesn't have enough MinPts, and thus fail to satisfy both conditions.

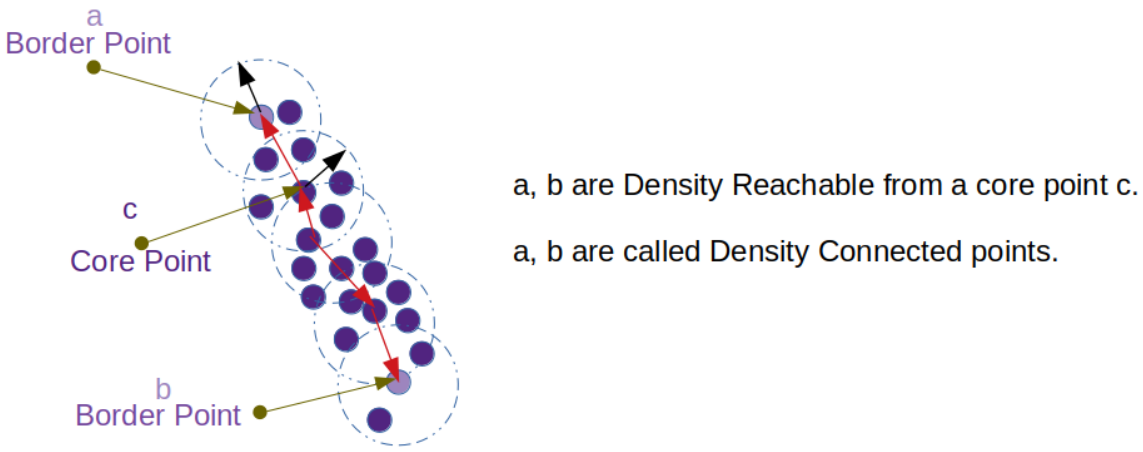
Density Reachable: Point a is density reachable from a point b with respect to ϵ and MinPts, if —

> For a chain of points b_1, $b_2 ... b_n$, where $b_1 = b$, $b_n = a$, such that b_{i+1} is directly density reachable from b_i.

Density reachable is transitive in nature but, just like direct density reachable, it is not symmetric.

Density Connected: There can be cases when 2 border points will belong to the same cluster but they don't share a specific core point, then we say that they are density connected if, there exists a common core point, from which these borders points are density reachable. As you can understand that density connectivity is symmetric. Definition from the Ester et.al. paper is given below —

"A point **a** is density connected to a point **b** with respect to ϵ and MinPts, if there is a point **c** such that, both **a** and **b** are density reachable from **c** w.r.t. to ϵ and MinPts."



Two border points a, b are density connected through the core point c. Source: Created by Author

Steps of DBSCAN Algorithm:

1. The algorithm starts with an arbitrary point which has not been visited and its neighborhood information is retrieved from the ϵ parameter.

2. If this point contains MinPts within ϵ neighborhood, cluster formation starts. Otherwise the point is labeled as noise. This point can be later found within the ϵ neighborhood of a different point and, thus can be made a part of the cluster. Concept of density reachable and density connected points are important here.
3. If a point is found to be a core point then the points within the ϵ neighborhood is also part of the cluster. So all the points found within ϵ neighborhood are added, along with their own ϵ neighborhood, if they are also core points.
4. The above process continues until the density-connected cluster is completely found.
5. The process restarts with a new point which can be a part of a new cluster or labeled as noise.

Drawbacks of the DBSCAN algorithm.

- If the database has data points that form clusters of varying density, then DBSCAN fails to cluster the data points well, since the clustering depends on ϵ and MinPts parameter, they cannot be chosen separately for all clusters.
- If the data and features are not so well understood by a domain expert then, setting up ϵ and MinPts could be tricky and, may need comparisons for several iterations with different values of ϵ and MinPts.

VI. EXPERIMENTS RESULTS

Experiments are executed to evaluate the performance of the proposed technique using LBPH, Haar-cascade algorithm,CNN,clustering (DBSCAN)to cluster the dataset.

At first we use DbScan to cluster the images and then we use one of the clusters for face recognition.

Face clustering:

❑ **dataset**:Contains 107 pictures of our two people. Notice that there is no identifying information in the filenames or another file that identifies who is in each image. It would be impossible to know which person is in which image based on filenames alone. We devised a face clustering algorithm to identify the similar and unique faces in the dataset.

❑ **encode_faces.py**: This is our first script — it computes face embeddings for all faces in the dataset and outputs a serialized encodings file.Before we can cluster a set of faces we first quantify them.

This process of quantifying the face will be accomplished using a deep neural network responsible for:

- Accepting an input image
- And outputting a 128-d feature vector that quantifies the face

Arguments:

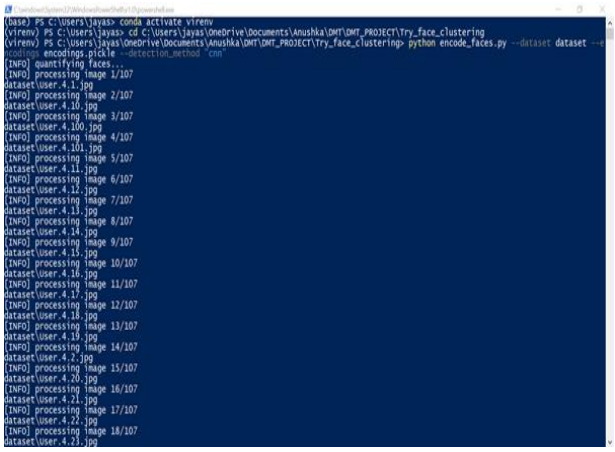
- -i --dataset : The path to the input directory of faces and images.
- -e --encodings : The path to our output serialized pickle file containing the facial encodings.
- -d --detection_method : Face detection method to be used. Can be "hog" or "cnn" (Default: cnn)

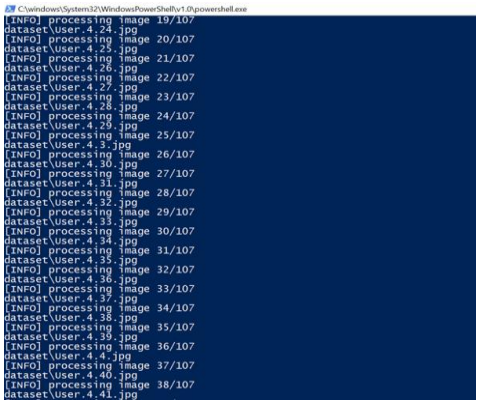
We have used the CNN face detector for higher accuracy, but it will take a significantly longer time to run if you are using a CPU rather than a GPU.

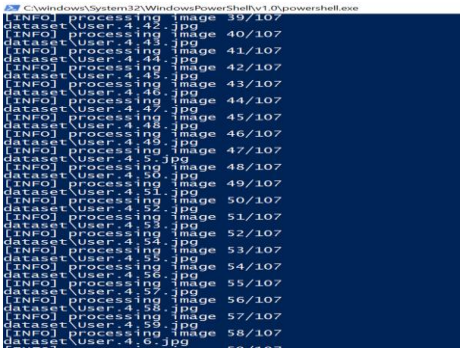
What it does

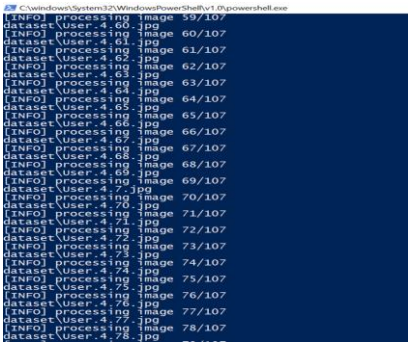
- We create a list of all imagePaths in our dataset using the dataset path provided in our command line argument.
- we compute the 128-d face encodings for each detected face in the rgb image
- For each of the detected faces + encodings, we build a dictionary that includes:
  - The path to the input image
  - The location of the face in the image (i.e., the bounding box)
  - The 128-d encoding itself
- Can be reused. write the data list to disk as a serialized encodings.pickle file

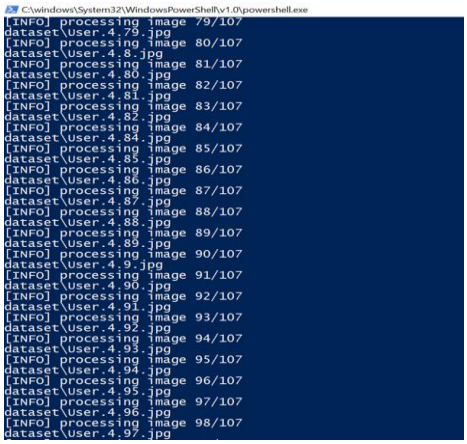
To run $python encode_faces.py --dataset dataset --encodings encodings.pickle --detection_method "cnn"

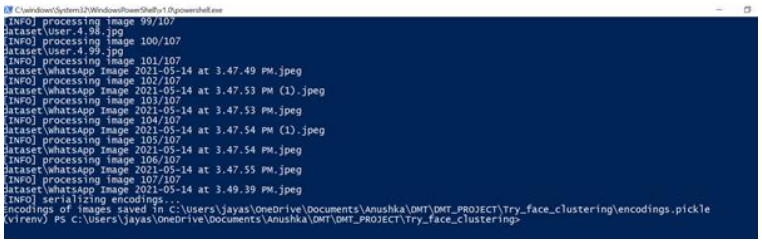












- ❏ **encodings.pickle**: Our face embeddings serialized pickle file.
- ❏ **cluster_faces.py:** we'll cluster similar faces and ideally find the outliers.

We have quantified and encoded all faces in our dataset as 128-d vectors, the next step is to cluster them into groups. Our hope is that each unique individual person will have their own separate cluster

For this task we need a clustering algorithm, many clustering algorithms such as k-means and Hierarchical Agglomerative Clustering, require us to specify the number of clusters we seek ahead of time. Therefore, we need to use a density-based or graph-based clustering algorithm Density-based spatial clustering of applications with noise (DBSCAN).It also naturally handles outliers, marking them as such if they fall in low-density regions where their "nearest neighbors" are far away.

Arguments:

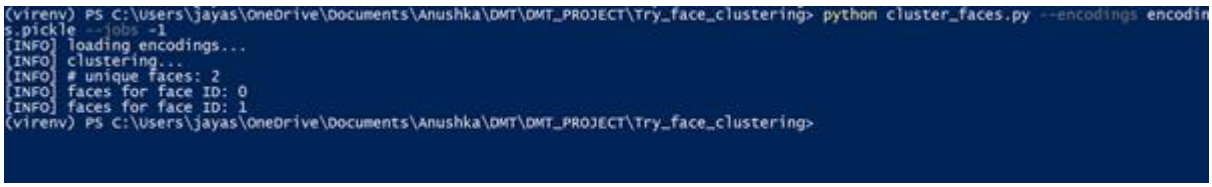
- -i --encodings : The path to the encodings pickle file that we generated in our previous script.
- -d --jobs : DBSCAN is multithreaded and a parameter can be passed to the constructor containing the number of parallel jobs to run. A value of -1 will use all CPUs available (default).

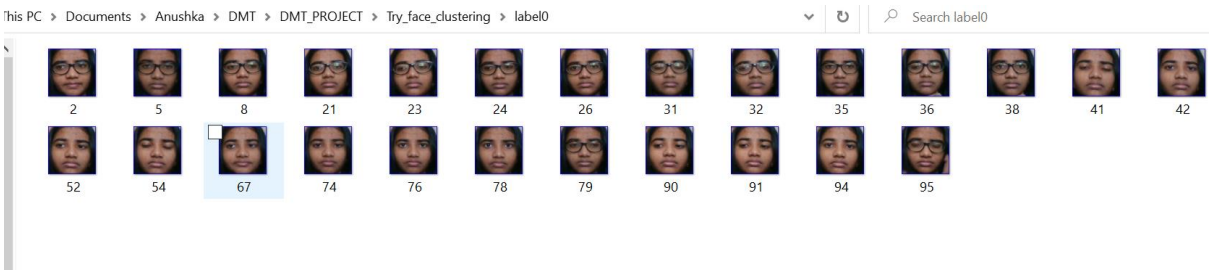
What it does

- Loaded the facial encodings data from disk, Organized the data as a NumPy array, Extracted the 128-d encodings from the data , placing them in a list
- create a DBSCAN object and then fit the model on the encodings
- loop to populate all the images in the database, and check the cluster and create a directory for the cluster.
- We employ the build_montages function of imutils to generate a single image montage containing a 5×5 grid of faces

To run $python cluster_faces.py --encodings encodings.pickle --jobs -1

We also import the build_montages module from imutils.We used this function to build a "montage of faces" for each cluster.

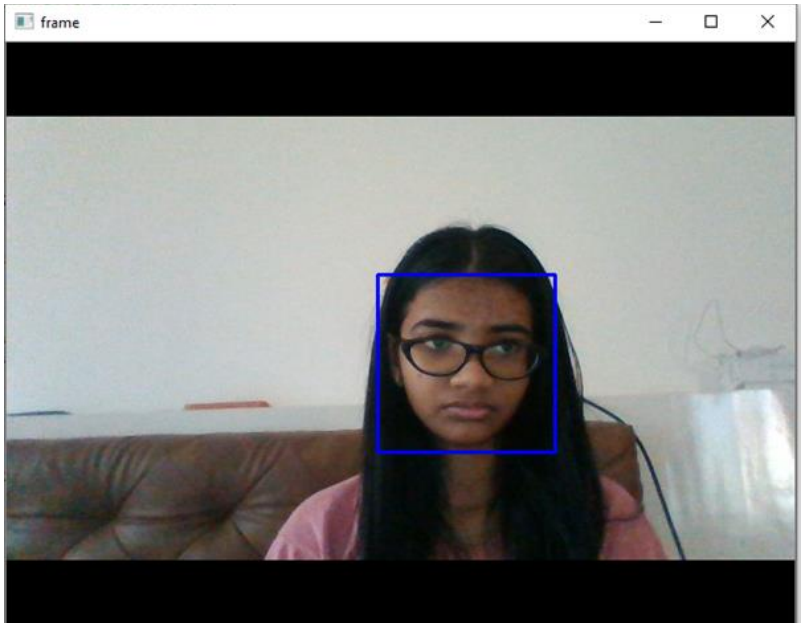




Montages

Face recognition:

❑ **face_image_storer.py:**Images from are stored in the folder “dataSet”



❑ **face_trainer.py:**We took all user data from our dataset and “trainer” the OpenCV Recognizer. This is done directly by a specific OpenCV function. The result will be a .yml file that will be saved in a “trainer/” directory.

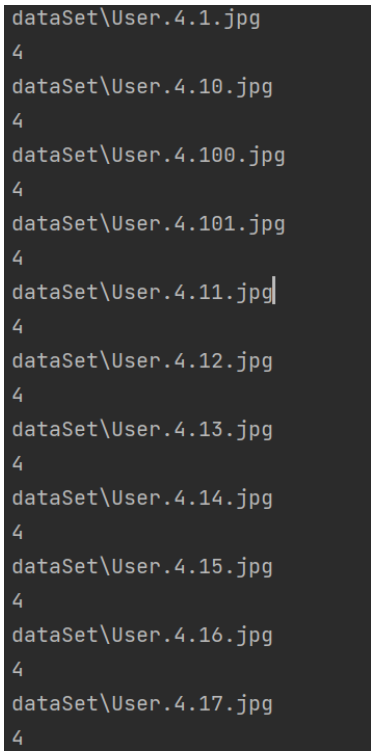
We have used a recognizer, the LBPH (LOCAL BINARY PATTERNS HISTOGRAMS) Face Recognizer, included on the OpenCV package.

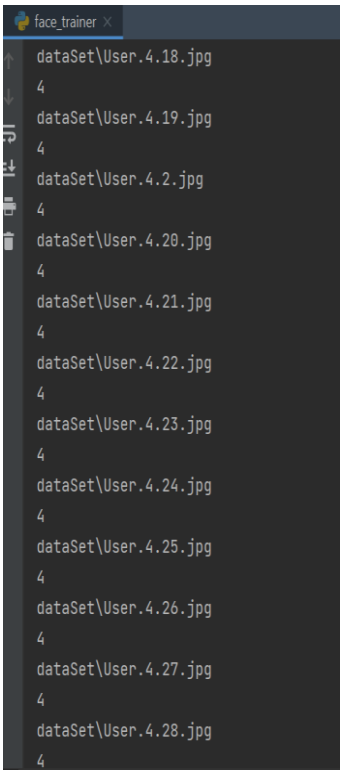
```
recognizer = cv2.face.LBPHFaceRecognizer_create()
```

The function “getImagesAndLabels (path)”, will take all photos in the directory: “dataset/”, returning 2 arrays: “Ids” and “faces”. With those arrays as input, we will “train our recognizer”:

```
recognizer.train(faces,Ids)
```

As a result, a file named “trainingData.yml” is saved in the trainer directory that was previously created by us.

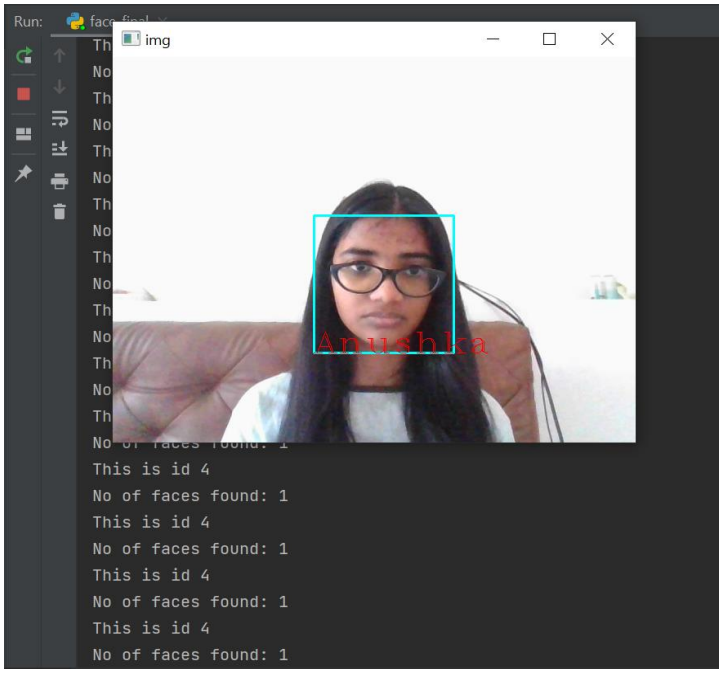


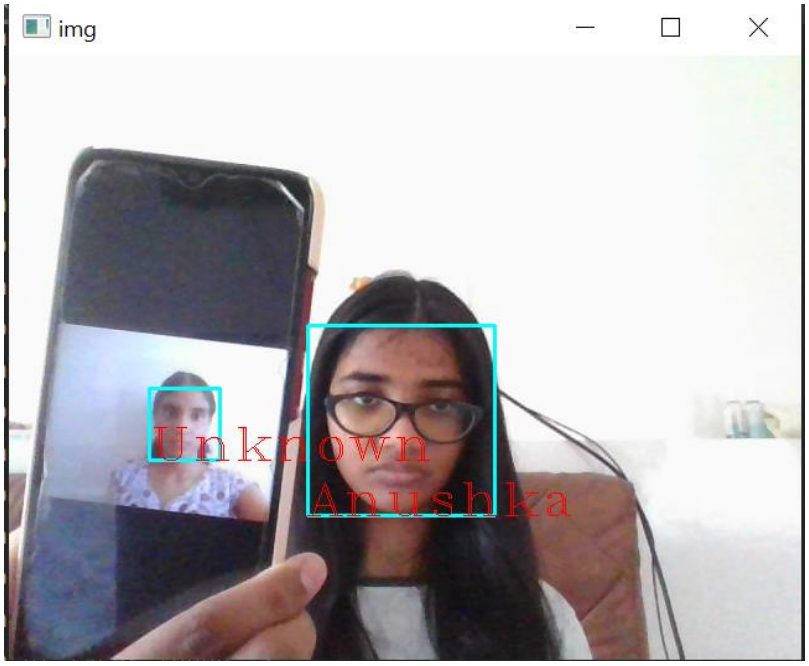


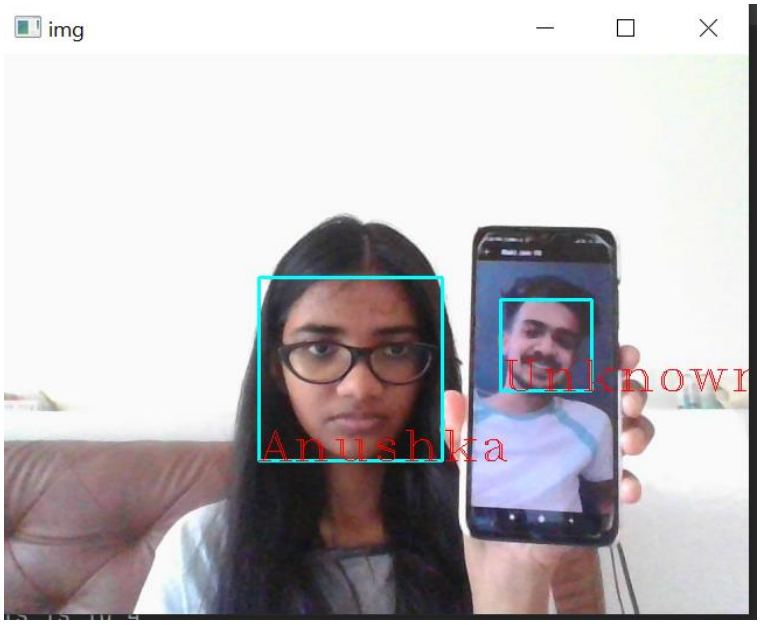
❑ **face_final.py:** We have captured a fresh face on our camera and if this person had his face captured and trained before, our recognizer will make a “prediction” returning its id and an index, showing how confident the recognizer is with this match.

The rec.predict (), will take as a parameter a captured portion of the face to be analyzed and will return its probable owner, indicating its id.

If the recognizer could predict a face, we put a text over the image with that person’s name . If not, an “unknown” label is put on the face.


Run:
img
Anushka
This is id 4
No of faces found: 1
This is id 4
No of faces found: 1
This is id 4
No of faces found: 1
This is id 4
No of faces found: 1
This is id 4
No of faces found: 1

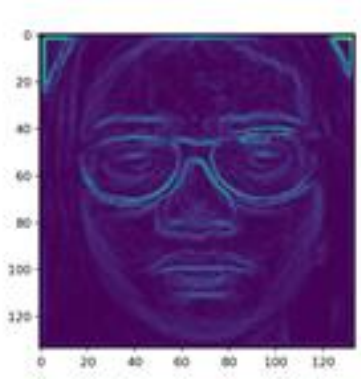

img
Unknown
Anushka


img
Unknown
Anushka

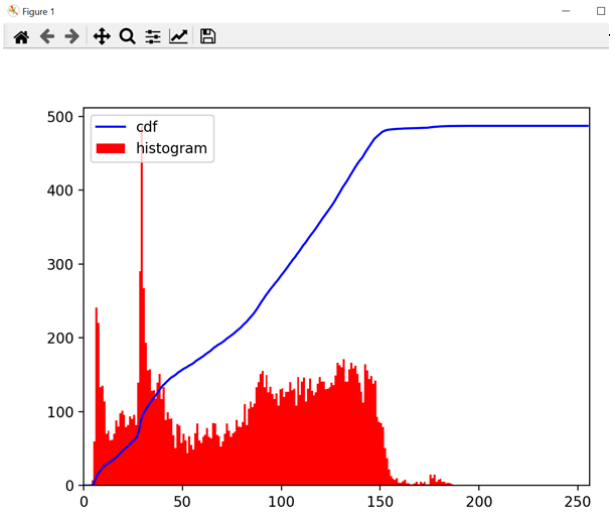
VII. COMPARATIVE STUDY / RESULTS AND DISCUSSION

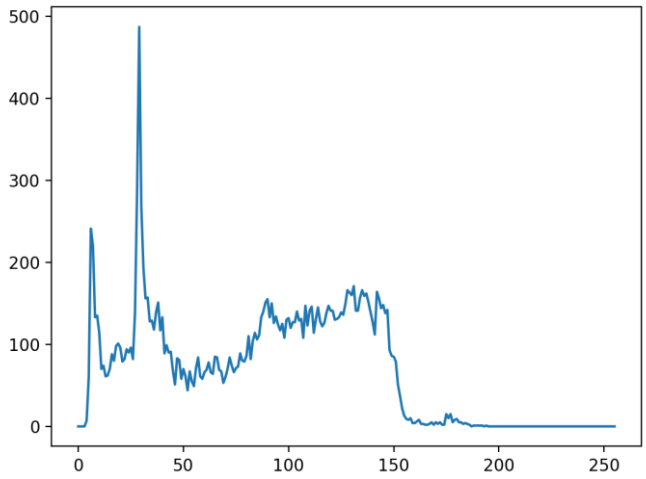
We evaluated the proposed algorithm on the real-time video frame (using LBPH and CNN). The accuracy of the proposed method is almost around 98.6%.

Calculation of Gradient:Gradients are typically large around edges and corners and allow us to detect those regions.The gradient of an image typically removes non-essential information.



Histogram and Cdf: The output shows that each number of pixels of an image lie upon a range of 0 to 200.





This efficiency cannot be generalized as it is performed on a smaller number of tests of images and conditions under which they are tested may be changed on other times.

The system successfully recognized the person and worked better in different face expressions.

VIII. CONCLUSION AND FUTURE WORK

This new proposed system provides an improved framework for face recognition using LBPH descriptor and Convolution Neural Network. The LBPH descriptor and CNN are helpful to provide a training data set with distinction patterns based on the correspondence between the original training images. The dataset helps the CNN to converge faster with greater accuracy and takes less computation time. Moreover, Precision, Recall, F Measure and Accuracy of the system are calculated. In future, this proposed work is extended to identify different facial expressions like smile, disgust etc, among the recognized faces. LBPH is one of the easiest face recognition algorithms. It can represent local features in the images. It is possible to get great results (mainly in a controlled environment). It is robust against monotonic gray scale transformations. It is provided by the OpenCV library (Open Source Computer Vision Library) Face recognition technology has come a long way in the last twenty years. These applications usually work in controlled environments and recognition algorithms can take advantage of the environmental constraints to obtain high recognition accuracy.

IX. REFERENCES

1. Mas idayu md sabri, Review of existing algorithms for face detection and recognition, December 2009.

2. Kyungnam kim ,Face recognition using principle component analysis ,2010.

3. Abhishek Singh, Saurabh Kumar, Face recognition using pca and eigen face approach, May 2012.

4. Elizabeth B. Varghese, M. Wilscy,Face Recognition Based On Vector Quantization Using Fuzzy Neuro Clustering, 2013

5. Sang-Woon Kim and Robert P.W. Duin, On Using a Pre-clustering Technique to Optimize LDA-Based Classifiers for Appearance-Based Face Recognition,2013

6. K .H. Wanjale, Amit Bhoomkar, Ajay Kulkarni, Somnath Gosavi, Use Of Haar Cascade Classifier For Face Tracking System In Real Time Video,,2013

7. Ashoka S.B ,Face Detection and Recognition Using K-means and Neural Network Methods,2014

8. Sayali Ghadge, Sana Khan, Sonam Vadsaria, Face recognition system , 2014

9. Pavan Pratap Chauhan, Vishal Kumar lath, Mr. Praveen rai, Comparison of different face recognition algorithms ,April 2016

10. Waldemar Wójcik, Konrad Gromaszek, Muhtar Junisbekov, Face Recognition: Issues, Methods And Alternative Applications, July 2016

11. Cuixia Li & Yingjun Tan& Dingbiao Wang & Peijie Ma, Research on 3D face recognition method in cloud environment based on semi supervised clustering algorithm ,
2016

12. Chih-Wei Lin1(B) and Kuan-Yin Lu, Local Clustering Patterns in Polar Coordinate for Face Recognition,2016

13. Abhjeet Sekhon, Dr. Pankaj Agarwal,Face Recognition Using K-Means and RBFN, IJCSMC, Vol. 6, Issue. 2, February 2017

14. Atul Dhingra ,Face Identification and Clustering, 2017

15. M.Vineetha Sai G.Varalakshmi, G.Bala kumar, J.Prasad, Face recognition system with face detection , 2017

16. Omkar M. Parkhi, Andrea vedaldi, Andrew zisserman, Deep face recognition , 2017

17. Pengcheng Wei1, Zhen Zhou , Li Li and Jiao Jiang, Research on face feature extraction based on K-mean algorithm ,2018

18.Dr. Hanumanthappa M, Dr. Ashoka S B, Face detection and recognition using kmeans and enhanced k-means algorithm with ann , June 2018.
19. Facial expression recognition using data mining algorithm,November 2018
20. Vinodpuri Rampuri Gosavi, Anil Kishanrao Deshmane, Ganesh Shahuba Sabl, Enhanced Face recognition system: Integration of Collaborative Representation based Classification (CRC) _KNN , International Journal of Electronics, Communications, and Measurement Engineering, January 2019.
21. Bingrong Xu, Qingshan Liu and Tingwen Huang, A Discrete-Time Projection Neural Network for Sparse Signal Reconstruction with Application to Face Recognition, IEEE Transactions on Neural networks and learning systems, January 2019.
22. Muhammad zeeshan khan, Saad harous, Saleet ul Hassan, Muhammad Usman Ghani Khan, Razi Iqbal and Shahid Mumtaz , Deep Unified Model for Face Recognition Based on Convolution Neural Network and Edge Computing ,IEEE Access, May 2019.
23.Face Recognition using Deep Neural Network across Variations in Pose and illumination. Author: S.Meenakshi, M. Siva Jothi, D. Murugan, International Journal of Recent Technology and Engineering (IJRTE), June 2019.
24.Deep face Recognition for Biometric Authentication Author: Maheen Zulfiqar, Fatima Syed,Muhammad Jaleed Khan ,International conference on Electrical, communication and computer Engineering, July 2019.
25. Xian-Feng Xu, Li Zhang, Chen-Dong Duan and Yong Lu Journal: IEEE Access, Research on Inception Module Incorporated Siamese Convolutional Neural Networks to Realize Face Recognition .December 2019.
26. Bendjillali Ridha Ilyas, Khaled Merit, Mohammed Bellingham, Abdelmalik talebahmed , Illumination-robust face recognition based on deep convolution al neural networks architectures .
27. Yassin Kortli, Maher Jridi 1, Ayman Al Falou 1 and Mohamed Atri , Face Recognition Systems: A Survey , 7 January 2020
28. Guangxin Lou, Hongzhen Shi, Face Image Recognition Based on Convolutional Neural Network , China Communications, February 2020.
29. Suresh Madhvan & Nitin Kumar,Incremental methods in face recognition ,12 August 2020.
30.Aruni RoyChowdhury, Xiang Yu, Kihyuk Sohn, Erik Learned-Miller, and Manmohan Chandraker ,Improving Face Recognition by Clustering Unlabeled Faces in the Wild, 2020.

Appendix

Code for face recognition:

face_image_storer.py

```
import cv2
cam = cv2.VideoCapture(0)
detector=cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
import os
import numpy as np
from PIL import Image


Id=input('enter your id')
sampleNum=0
while(True):
    ret, img = cam.read()
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    faces = detector.detectMultiScale(gray, 1.3, 5)
    for (x,y,w,h) in faces:
        cv2.rectangle(img,(x,y),(x+w,y+h),(255,0,0),2)

        #incrementing sample number
        sampleNum=sampleNum+1
        #saving the captured face in the dataset folder
        cv2.imwrite("dataSet/User."+Id   +'.'+   str(sampleNum)   +   ".jpg",
img[y:y+h,x:x+w])

        cv2.imshow('frame',img)
    #wait for 100 miliseconds
    if cv2.waitKey(1000) & 0xFF == ord('q'):
        break
    # break if the sample number is morethan 20
    elif sampleNum>100:
        break
cam.release()
cv2.destroyAllWindows()
```

face_trainer,py

```
import cv2
import os
import numpy as np
from PIL import Image

recognizer = cv2.face.LBPHFaceRecognizer_create()
path='dataSet'
def getImagesWithID(path):
    imagePaths=[os.path.join(path,f) for f in os.listdir(path)]
    faces=[]
    IDs=[]

    for imagepath in imagePaths:
        faceImg=Image.open(imagepath).convert('L')
        faceNp=np.array(faceImg,'uint8')
        print(imagepath)
        ID=int(os.path.split(imagepath)[-1].split(".")[1])
        print(os.path.split(imagepath)[-1].split(".")[1])
        faces.append(faceNp)
        IDs.append(ID)
        cv2.imshow("training",faceNp)
        cv2.waitKey(10)
    return np.array(IDs),faces

Ids,faces=getImagesWithID(path)
recognizer.train(faces,Ids)
recognizer.save('trainingData.yml')
cv2.destroyAllWindows()
```

face_final.py:

```
import cv2
import pyautogui,time
pyautogui.FAILSAFE = True



face_cascade = cv2.CascadeClassifier("haarcascade_frontalface_default.xml")
rec = cv2.face.LBPHFaceRecognizer_create()
cap = cv2.VideoCapture(0)
```

```
rec.read("trainingData.yml")

# loop runs if capturing has been initialized.
while (True):

    # reads frames from a camera
    ret, img = cap.read()

    # convert to gray scale of each frames
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)

    # Detects faces of different sizes in the input image
    faces = face_cascade.detectMultiScale(gray, 1.3, 5)
    print("No of faces found:",len(faces))

    for (x,y,w,h) in faces:
        # To draw a rectangle in a face
                        cv2.rectangle(img,(x,y),(x+w,y+h),(255,255,0),2)
                        Id,conf=rec.predict(gray[y:y+h,x:x+w])
                        print("This is id",Id)
                        if(conf<70):
                            if Id==4:
                               Id="Anushka"
                               conf = "  {0}%".format(round(100 - conf))
                            #elif(Id==7):
                                #Id="Anushka 2"
                        else:
                            Id="Unknown"
                            conf = "  {0}%".format(round(100 - conf))
                        cv2.putText(img,          str(Id),          (x,y+h),
cv2.FONT_HERSHEY_COMPLEX_SMALL, 2.0, (0, 0, 255))
                        cv2.putText(img,    str(conf),    (x,    y    +    h),
cv2.FONT_HERSHEY_COMPLEX_SMALL, 2.0, (0, 0, 255))


    # Display an image in a window
    cv2.imshow('img',img)

    # Wait for Esc key to stop
    k = cv2.waitKey(5)
```

```
   if k == 27:
        break

# Close the window
cap.release()

# De-allocate any associated memory usage
cv2.destroyAllWindows()
```

Code for clustering:

encode_faces.py:

```
# USAGE
  # python encode_faces.py --dataset dataset --encodings encodings.pickle
#extract a 128-d feature vector representation for each face.
# import the necessary packages
from imutils import paths
# face_recognition library by @ageitgey
import face_recognition
# argument parser
import argparse
# pickle to save the encodings
import pickle
# openCV
import cv2
# operating system
import os
from constants import ENCODINGS_PATH
```

```
# construct the argument parser and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-i", "--dataset", required=True,
  help="path to input directory of faces + images")
ap.add_argument("-e", "--encodings", required=True,
  help="path to serialized database of facial encodings")
ap.add_argument("-d", "--detection_method", type=str, default="cnn",
  help="face detection model to use: either `hog` or `cnn`")
args = vars(ap.parse_args())

# grab the paths to the input images in our dataset, then initialize
# out data list (which we'll soon populate)
print("[INFO] quantifying faces...")
imagePaths = list(paths.list_images(args["dataset"]))
data = []

# loop over the image paths
for (i, imagePath) in enumerate(imagePaths):
  # load the input image and convert it from RGB (OpenCV ordering)
  # to dlib ordering (RGB)
  print("[INFO] processing image {}/{}".format(i + 1,
     len(imagePaths)))
  print(imagePath)

  # loading image to BGR
  image = cv2.imread(imagePath)
```

```
    # converting image to RGB format
    image = cv2.cvtColor(image, cv2.COLOR_BGR2RGB)

    # detect the (x, y)-coordinates of the bounding boxes
    # corresponding to each face in the input image
    boxes = face_recognition.face_locations(image,
        model=args["detection_method"])

    # compute the facial embedding for the face
    encodings = face_recognition.face_encodings(image, boxes)

    # build a dictionary of the image path, bounding box location,
    # and facial encodings for the current image
    d = [{"imagePath": imagePath, "loc": box, "encoding": enc}
        for (box, enc) in zip(boxes, encodings)]
    data.extend(d)

# dump the facial encodings data to disk
print("[INFO] serializing encodings...")
f = open(args["encodings"], "wb")
f.write(pickle.dumps(data))
f.close()
print("Encodings of images saved in {}".format(ENCODINGS_PATH))
```

constants.py:

```
import os
```

```
# face_data (directory) represents the path component to be joined.


FACE_DATA_PATH = os.path.join(os.getcwd(),'face_cluster')

ENCODINGS_PATH = os.path.join(os.getcwd(),'encodings.pickle')

CLUSTERING_RESULT_PATH = os.getcwd()
```

cluster_faces.py:

```
# USAGE

# python cluster_faces.py --encodings encodings.pickle


# import the necessary packages

# DBSCAN model for clustering similar encodings

from sklearn.cluster import DBSCAN

from imutils import build_montages

import numpy as np

import argparse

import pickle

import cv2

import shutil

import os

from constants import FACE_DATA_PATH, ENCODINGS_PATH, CLUSTERING_RESULT_PATH


# add constants file in the code (clustering_result)


def move_image(image,id,labelID):
```

```
  path = CLUSTERING_RESULT_PATH+'/label'+str(labelID)
  # os.path.exists() method in Python is used to check whether the specified path exists or not.
  # os.mkdir() method in Python is used to create a directory named path with the specified numeric mode.
  if os.path.exists(path) == False:
     os.mkdir(path)

  filename = str(id) +'.jpg'
  # Using cv2.imwrite() method
  # Saving the image

  cv2.imwrite(os.path.join(path , filename), image)

  return



# construct the argument parser and parse the arguments
ap = argparse.ArgumentParser()
ap.add_argument("-e", "--encodings", required=True,
  help="path to serialized db of facial encodings")
ap.add_argument("-j", "--jobs", type=int, default=-1,
  help="# of parallel jobs to run (-1 will use all CPUs)")
args = vars(ap.parse_args())

# load the serialized face encodings + bounding box locations from
# disk/encodings pickle file, then extract the set of encodings to so we can cluster on them
```

```
print("[INFO] loading encodings...")

data = pickle.loads(open(args["encodings"], "rb").read())

data = np.array(data)

encodings = [d["encoding"] for d in data]


# cluster the embeddings

print("[INFO] clustering...")


# creating DBSCAN object for clustering the encodings with the metric "euclidean"

clt = DBSCAN(metric="euclidean", n_jobs=args["jobs"])

clt.fit(encodings)


# determine the total number of unique faces found in the dataset

# clt.labels_ contains the label ID for all faces in our dataset (i.e., which cluster each face belongs to).

# To find the unique faces/unique label IDs, used NumPy’s unique function.

# The result is a list of unique labelIDs

labelIDs = np.unique(clt.labels_)


# we count the numUniqueFaces . There could potentially be a value of -1 in labelIDs — this value corresponds

# to the “outlier” class where a 128-d embedding was too far away from any other clusters to be added to it.

# “outliers” could either be worth examining or simply discarding based on the application of face clustering.

numUniqueFaces = len(np.where(labelIDs > -1)[0])

print("[INFO] # unique faces: {}".format(numUniqueFaces))
```

```
# loop over the unique face integers
for labelID in labelIDs:
  # find all indexes into the `data` array that belong to the
  # current label ID, then randomly sample a maximum of 25 indexes
  # from the set
  print("[INFO] faces for face ID: {}".format(labelID))
  idxs = np.where(clt.labels_ == labelID)[0]
  idxs = np.random.choice(idxs, size=min(25, len(idxs)),
      replace=False)

  # initialize the list of faces to include in the montage
  faces = []

  # loop over the sampled indexes
  for i in idxs:
      # load the input image and extract the face ROI
      image = cv2.imread(data[i]["imagePath"])
      (top, right, bottom, left) = data[i]["loc"]
      face = image[top:bottom, left:right]

      # puting the image in the clustered folder
      move_image(image,i,labelID)

      # force resize the face ROI to 96mx96 and then add it to the
      # faces montage list
      face = cv2.resize(face, (96, 96))
```

```
    faces.append(face)

# create a montage using 96x96 "tiles" with 5 rows and 5 columns
montage = build_montages(faces, (96, 96), (5, 5))[0]

# show the output montage
title = "Face ID #{}".format(labelID)
title = "Unknown Faces" if labelID == -1 else title
"""
cv2.imshow(title, montage)
cv2.waitKey(0)
"""
cv2.imwrite(os.path.join(CLUSTERING_RESULT_PATH, title+'.jpg'), montage)
```