# FACE RECOGNITION BASED ON LOCAL BINARY PATTERN USING PYTHON

**A PROJECT REPORT**

***Submitted by***

| | | |
|---|---|---|
| **BALA SUBRAMANIYAN K** | - | **952319104003** |
| **SANJEEV KUMAR C** | - | **952319104036** |
| **VIGNESH S** | - | **952319104052** |
| **VISWA M** | - | **952319104053** |

***In partial fulfilment for the award of the degree***

*Of*

**BACHELOR OF ENGINEERING**

**IN**

**COMPUTER SCIENCE AND ENGINEERING**

**PSN ENGINEERING COLLEGE,**

**TIRUNELVELI.**

**ANNA UNIVERSITY: CHENNAI 600 025**

**MAY 2023**

# FACE RECOGNITION BASED ON LOCAL BINARY PATTERN USING PYTHON

**A PROJECT REPORT**

***Submitted by***

| | | |
|---|---|---|
| **BALA SUBRAMANIYAN K** | - | **952319104003** |
| **SANJEEV KUMAR C** | - | **952319104036** |
| **VIGNESH S** | - | **952319104052** |
| **VISWA M** | - | **952319104053** |

***In partial fulfilment for the award of the degree***

*Of*

**BACHELOR OF ENGINEERING**

**IN**

**COMPUTER SCIENCE AND ENGINEERING**

**PSN ENGINEERING COLLEGE,**

**TIRUNELVELI.**

**ANNA UNIVERSITY: CHENNAI 600 025**

**MAY 2023**

**ANNA UNIVERSITY: CHENNAI 600 025**

**BONAFIDE CERTIFICATE**

Certified that this project report **“FACE RECOGNITION BASED ON LOCAL BINARY PATTERN USING PYTHON”** is the bonafide work of **BALA SUBRAMANIYAN K (952319104003), SANJEEV KUMAR G (952319104036),VIGNESH S (952319104052) and VISWA M (952319104053)** who carried out the project work under my supervision.

| | |
|---|---|
| SIGNATURE | SIGNATURE |
| Ms.M.ANITHA BE.,M.Tech., | Mrs.A.THARIK NAZEEM |
| | B.Tech.,M.E., |
| HEAD OF THE DEPARTMENT | SUPERVIOSR |
| | Assistant professor |
| Department Of Computer Science | Department Of Computer Science |
| and Engineering | and Engineering |
| PSN Engineering College | PSN Engineering College |
| Tirunelveli-627 358 | Tirunelveli-627 358 |

**Submitted to Anna University Viva-voce held on ……………………..**

**INTERNAL EXAMINER** **EXTERNAL EXAMINER**

# ABSTRACT

This project proposes a face recognition system based on a Local Binary Pattern. Humans are able to share multiple emotions and feelings through their facial gestures and body language. In this project, in order to detect the emotions from the human face gesture, we use the complex process of explicit feature extraction in traditional face expression recognition and a face expression recognition method based on image edge detection. The facial expression image is normalized, and the edge of each layer of the image is extracted in the convolution process. With the help of a local binary pattern and a convolution neural network, we will be able to automatically detect the face recognition of human emotions. The classifier module is comprised of a global average pooling and softmax layer to calculate the probability of each class. The overall design optimizes the network parameters and maintains classification accuracy. This project is implemented using Python.

# ACKNOWLEDGEMENT

I express my sincere and heart full thanks to our Honorable chairman **Dr.P.SUYAMBU, Ph.D**. PSN Group of Institutions. I am very much indebted to principal, **Dr. M.S.RAVIKUMAR, Ph.D**. PSN Engineering college, Melathediyoor , Tirunelveli**.**

I express my extreme gratefulness to our Director**, Dr. N.PRABHU,M.E,Ph.D,M.I.S.T.E,** PSN Engineering college, Melathediyoor,Tirunelveli for providing me kindly advice.

I record my sincere thanks to **Ms. M. ANITHA, B.E.,M.Tech.,** Head of the department, Department of Computer Science and Engineering , PSN Engineering college, Melathediyoor, Tirunelveli, for allowing me to use the facilities in the college.

I express my sincere thanks to my guide **Mrs.A.THARIK NAZEEM, B. Tech., M.E., Assistant professor,** Department of Computer Science PSN Engineering college, Melathediyoor, Tirunelveli, for her encouragement and valuable guidance.

I extend my heartful thanks to all my staff members and friends those who are contributed moral support and encouragement to do this work.

I also thanks all those who have directly or indirectly helped for the successful completion of this project work.

# TABLE OF CONTENT

## LIST OF FIGURES

## LIST OF ABBREVIATIONS

| S.NO | ABBREVIATIONS | |
|---|---|---|
| 1 | CNN | Convolution Neural Network |
| 2 | ANN | Artificial Neural Network |
| 3 | SIANN | Space Invariant Artificial Neural Network |
| 4 | MLP | Multilayer Perception Neural Network |
| 5 | LTrP | Local Tetra pattern |
| 6 | LVP | Local Vector Pattern |
| 7 | LCP | Local Clustering Pattern |
| 8 | LBP | Local Binary Pattern |
| 9 | LDA | Linear Discriminate Analysis |
| 10 | LMEBP | Local Maximum Edge Binary Pattern |
| 11 | CCP | Convex -Concave Partition |
| 12 | MGP | Modified Gradient Pattern |
| 13 | FCCP | Fuzzy Convex- Concave Partition |
| 14 | ZMPDCNN | Z-Normalization along with the moore Penrose-centric Deep Convolution Neural Network |

# CHAPTER 1

# INTRODUCTION

## 1.1 INTRODUCTION

In deep learning, a convolutional neural network (CNN, or ConvNet) is a class of artificial neural network (ANN), most commonly applied to analyze visual imagery. CNNs are also known as Shift Invariant or Space Invariant Artificial Neural Networks (SIANN), based on the shared-weight architecture of the convolution kernels or filters that slide along input features and provide translation-equivariant responses known as feature maps. Counter-intuitively, most convolutional neural networks are not invariant to translation, due to the down sampling operation they apply to the input. They have applications in image and video recognition, recommender systems, image classification, image segmentation, medical image analysis, natural language processing, brain–computer interfaces, and financial time series. CNNs are regularized versions of multilayer perceptrons. Multilayer perceptrons usually mean fully connected networks, that is, each neuron in one layer is connected to all neurons in the next layer. The "full connectivity" of these networks make them prone to over fitting data. Typical ways of regularization, or preventing over fitting, include: penalizing parameters during training (such as weight decay) or trimming connectivity (skipped connections, dropout, etc.) CNNs take a different approach towards regularization: they take advantage of the hierarchical pattern in data and assemble patterns of increasing complexity using smaller and simpler patterns embossed in their filters. Therefore, on a scale of connectivity and complexity, CNNs are on the lower extreme. Convolutional networks were inspired by biological processes in that the connectivity pattern between neurons resembles the organization of the animal visual cortex. Individual cortical neurons respond to stimuli only in a restricted region of the visual field known as the receptive field. The receptive fields of different neurons partially overlap such that they cover the entire visual field. CNNs use relatively little pre-processing compared to other image classification algorithms. This means that the network learns to optimize the filters (or kernels) through automated learning, whereas in traditional algorithms these filters are hand-engineered. This independence from prior knowledge and human intervention in feature extraction is a major advantage. Convolutional neural networks are a specialized type of artificial neural networks that use a mathematical operation called convolution in place of general matrix multiplication in at least one of their layers. They are specifically designed to process pixel data and are used in image recognition and processing.

## 1.2 ARCHITECTURE OF CNN

A convolutional neural network consists of an input layer, hidden layers and an output layer. In any feed-forward neural network, any middle layers are called hidden because their inputs and outputs are masked by the activation function and final convolution. In a convolutional neural network, the hidden layers include layers that perform convolutions. Typically this includes a layer that performs a dot product of the convolution kernel with the layer's input matrix. This product is usually the Frobenius inner product, and its activation function is commonly ReLU. As the convolution kernel slides along the input matrix for the layer, the convolution operation generates a feature map, which in turn contributes to the input of the next layer. This is followed by other layers such as pooling layers, fully connected layers, and normalization layers.

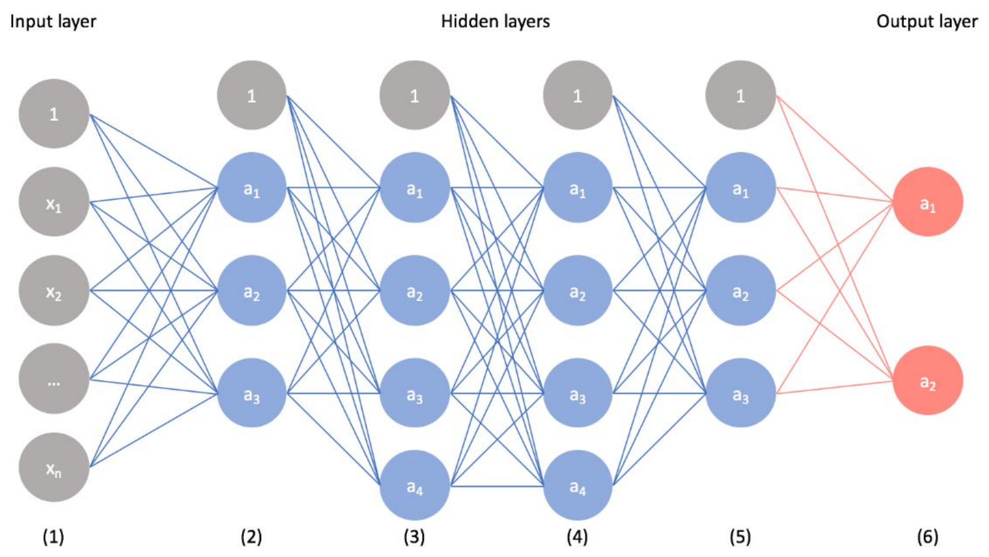


FIG. 1.2.1 **ARCHITECTURE OF CNN**

## 1.3 CONVOLUTION LAYER

In a CNN, the input is a tensor with shape: (number of inputs) × (input height) × (input width) × (input channels). After passing through a convolutional layer, the image becomes abstracted to a feature map, also called an activation map, with shape: (number of inputs) × (feature map height) × (feature map width) × (feature map channels). Convolutional layers convolve the input and pass its result to the next layer. This is similar to the response of a neuron in the visual cortex to a specific stimulus. Each convolutional neuron processes data only for its receptive field. Although fully connected feed forward neural networks can be used to learn features and classify data, this architecture is generally impractical for larger inputs (e.g., high-resolution images), which would require massive numbers of neurons because each pixel is a relevant input feature. A fully connected layer for an image of size 100 × 100 has 10,000 weights for each neuron in the second layer. Convolution reduces the number of free parameters, allowing the network to be deeper. For example, using a 5 × 5 tiling region, each with the same shared weights, requires only 25 neurons. Using regularized weights over fewer parameters avoids the vanishing gradients and exploding gradients problems seen during back propagation in earlier neural networks. To speed processing, standard convolutional layers can be replaced by depth wise separable convolutional layers, which are based on a depth wise convolution followed by a point wise convolution. The depth wise convolution is a spatial convolution applied independently over each channel of the input tensor, while the point wise convolution is a standard convolution restricted to the use of  kernels.

## 1.4 POOLING LAYER

Convolutional networks may include local and/or global pooling layers along with traditional convolutional layers. Pooling layers reduce the dimensions of data by combining the outputs of neuron clusters at one layer into a single neuron in the next layer. Local pooling combines small clusters, tiling sizes such as 2 × 2 are commonly used. Global pooling acts on all the neurons of the feature map.[19][20] There are two common types of pooling in popular use: max and average. Max pooling uses the maximum value of each local cluster of neurons in the feature map,[21][22] while average pooling takes the average value.

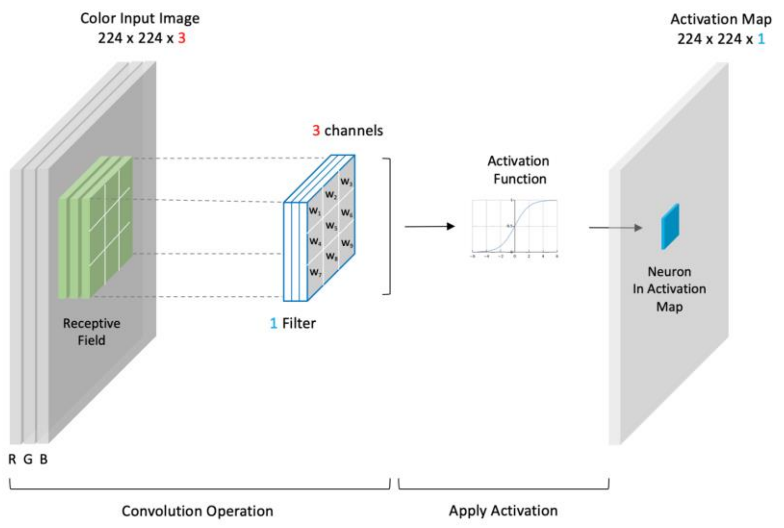


Fig.1.5 Convolutional layer

## 1.5 Fully connected layer

Fully connected layers connect every neuron in one layer to every neuron in another layer. It is the same as a traditional multilayer perceptron neural network (MLP). The flattened matrix goes through a fully connected layer to classify the images.

## 1.6 Local tetra pattern

Local tetra pattern (LTrP) adopts the concepts of LBP and LDP which extends the spatial relationship from one-dimensional to two-dimensional. LTrP uses two high-order derivative directions with four distinct values to encode the micropattern for extract more discriminative information. Due to the intelligence security monitoring is more popular in recent years, the automatically recognizing face is needed for various visual surveillance systems, for example the accessing control system for personal or company to verify the legal/illegal people, policing system for identifying the thief and the robber who presents the illegal behavior in public or private space. To construct an efficient face recognition system, the facial descriptor with discriminated characteristic is required.

The facial descriptor refers to the process of extracting the discriminative features to represent a given face image. Numerous methodologies are proposed to recognize face and those can be classified as global and local facial descriptors. The global facial descriptor describes the facial characteristics with the whole face image, such as principal component analysis (PCA) and linear discriminant analysis (LDA). PCA converts the global facial descriptor from high dimension to low dimension by using the linear transform methodology to reduce the computational cost. Linear discriminant analysis (LDA) also called the Fishers Linear Discriminant is similar to PCA, while it is a supervised methodology. Although the global facial descriptor can extract the principal component from the facial images, reduces the computational cost, and maintains the variance of the facial image, the performance is sensitive to the change of the environment, such as the change of light.The flexibilities of the local facial descriptors are better than the global facial descriptors because they successfully and effectively represent the spatial structure information of an input image. A well local facial descriptor generates discriminative and robust features to achieve good recognition results with computational simplicity. In this chapter, we represent a number of approaches in the local facial descriptor including the local binary pattern (LBP), local derivation pattern (LDP), local tetra pattern (LTrP), local vector pattern (LVP) and local clustering pattern (LCP).

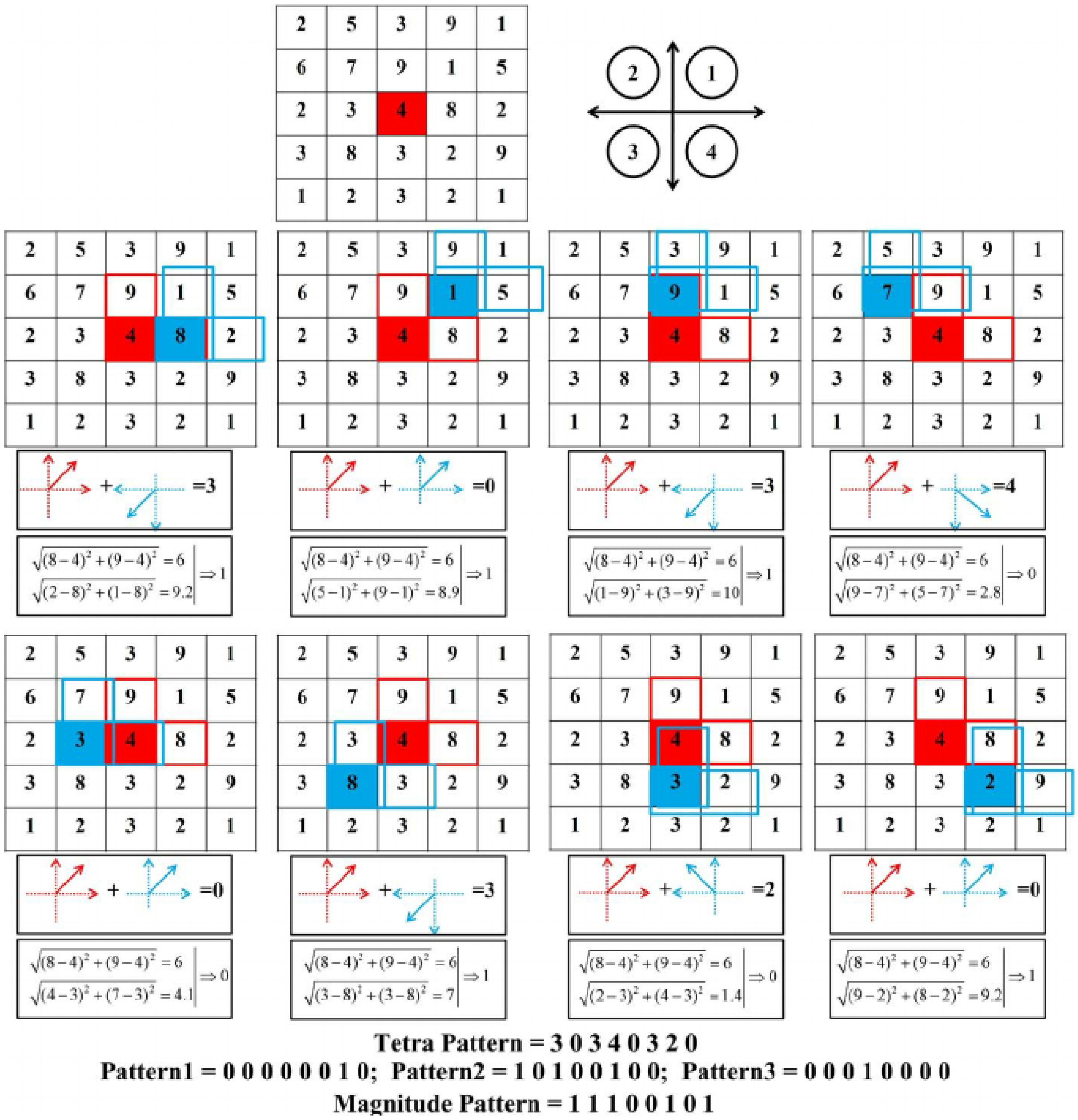

Tetra Pattern = 3 0 3 4 0 3 2 0
Pattern1 = 0 0 0 0 0 0 1 0; Pattern2 = 1 0 1 0 0 1 0 0; Pattern3 = 0 0 0 1 0 0 0 0
Magnitude Pattern = 1 1 1 0 0 1 0 1

**Fig.1.6 Tetra pattern**

## 1.7 Local pattern descriptor

A local pattern considers the variations of subregion in an image, which is also called a micropattern. In this section, we introduce the basic and several popular techniques of local pattern descriptor for facial recognition.

## 1.8 Local binary pattern

Local binary pattern (LBP) is designed to describe the texture in a local neighborhood is an invariant texture measure and has been various comparative studies, such as fingerprint recognition, face recognition, and license plate recognition. The main characteristics of LBP are: (1) highly discriminative capability (2) and computational efficiency.

### 1.9 Local derivative pattern

LBP is a nondirectional first-order local derivative pattern of images and fails to extract more detailed information, such as the directions between neighborhoods and referenced pixel, and the high-order gradient information. Local derivative pattern (LDP) can be considered as an extension of LBP with directional high-order local derivative pattern. To encode the nth�th -order LDP, the (n−1)th�−1th -order local derivative variations with various distinctive spatial relationships along 0∘0∘, 45∘45∘ , 90∘90∘ , and 135∘135∘ directions are used. The first-order derivatives of the referenced pixel Xc�� along 0∘0∘, 45∘45∘, 90∘90∘, and 135∘135∘ directions

### 1.10 Local vector pattern

Local vector pattern (LVP) is inspired by local binary pattern (LBP) which is sample and intuitive. To compare with LBP and LDP, LVP further considers the neighbourhood relationship with various distances from different directions and the relationship between various derivative directions.

### 1.11 Local clustering pattern

Local clustering pattern (LCP) is designed to solve the problems in face recognition: (1) to reduce feature length with low computational cost and (2) to enhance the accuracy for face recognition. To generate the local clustering pattern, four phases have to be considered: (1) to generate the local derivative variations with various directions; (2) to project the local derivative variations with various directions on the pairwise combinatorial directions in the rectangular coordinate system; (3) to transform the coordinate from the rectangular coordinate system into the polar coordinate system; and (4) encoding the facial descriptor which is local clustering pattern, as a micro pattern for each pixel by applying the clustering algorithm. The details are described in the following subsections: local clustering pattern (LCP) and coding scheme.

# CHAPTER 2

# LITERATURE SURVEY

## 2.1 INTRODUCTION

The main objective of the local pattern is to describe the image with high discriminative features so that the local pattern descriptors are more suitable for face recognition. The word "local" represents the measured image with the sub region and is the key in this chapter. Regardless of the techniques proposed, the local pattern is one of the most interesting areas in face recognition. The local facial descriptor is a local pattern that generates the descriptor by considering the sub region of an image. Techniques based on various combination methods from the local facial descriptors are not unusual. This chapter is concerned primarily to help the reader to develop a basic understanding of the local pattern descriptors and how they apply to face recognition. We begin to describe the outline of the local pattern in face recognition and its relative facial descriptors. Next, we give an introduction to the popular local patterns and establish examples to demonstrate the process of each method. To the end of this chapter, we conclude those methods with a discussion of issues related to the properties of the local patterns.

**Junding Sun et al [2020]** proposed a novel method, called local gradient number pattern (LGNP), is firstly presented in the paper for face description. For LGNP, the Sobel operator is adopted to extract the local gradient information, and the position of the gray transitions in the local neighborhood is used to form the LGNP code based on the LDP-based methods. Then, the concept of fuzzy convex-concave partition (FCCP) is introduced to fuse the global and regional information based on convex-concave partition (CCP). By the combination of LGNP and FCCP, the proposed descriptor is denoted as FCCP_LGNP. To evaluate the performance of FCCP_LGNP comprehensively, a series of experiments were carried out on four different face databases ORL, CALTECH, GEORGIA, and FACE94, and the results show that FCCP_LGNP is superior to the recent state-of-the-art methods based on hand-crafted features. Even compared with the deep learning methods, VGG16 and ResNet101, the proposed descriptor still shows good performance.

Kishore Kumar Kamarajugadda **et al [2019]** presented a Feed-Forward Neural Network based face recognition in order to elevate the effectiveness of the face recognition systems in this paper. The convolution neural network provides successively larger features in a hierarchical set of layers. Pre-processing is performed to filter the images prior to feature extraction. The feature extraction is performed via Linear Discriminative Analysis (LDA) and Local Tetra Pattern (LTP) which captures both frequency and location information. Further, feature reduction is performed via (LDA) for image enhancement which enables better image classification. Then, the classification process is performed via M-FNN technique. It estimates the primary component of every class and perceives the classifier based on data projection on subspaces traversed by the principal components. Finally, the performance of the proposed technique is analyzed via parameters such as Accuracy, Sensitivity, Specificity, Precision, and Recall. It is apparent from the results of the performed experiments that the proposed technique offers better results compared to the existing techniques.

**Santhosh S. et al [2022]** proposed a conventional methodologies, the Deep Learning (DL)-centric Efficient Face Recognition System (FRS) using ZNormalization along with the Moore Penrose-centric Deep Convolutional Neural Network (ZMPDCNN) algorithm is presented in this paper. The input image is enhanced by using the BIG-Weber technique to enhance the contrast. Cost Function-grounded You Only Look Once version 3 (CFYOLOv3) algorithm is used to detect the face as of the enhanced image. Then, to localize as well as to represent the face's salient regions, the facial points are extracted using the Multi-Gated Supervised Descent (MGSD) approach. Next, the facial parts are segmented as of the detected face using the Angle rotation Adaptive Viola-Jones Algorithm (A2VJA). The significant features are selected as of the extracted features using the Newtonian Constant of Gravitation-based Grasshopper Optimization Algorithm (NCGOA). To recognize the face, the selected features were given as input to the ZMP-DCNN framework. A similar process of input images was done for the query images. The query image is recognized using trained outcome of the input image. Finally, performance of the proposed technique is compared with the existing frameworks. Thus, in contrast to the other mechanisms, the proposed face recognition system achieves superior performance.

**Chaorong Li et al [2019]** made a novel method for texture image recognition is proposed in this paper. The aim of the proposed method is to represent texture by combining the Gabor wavelet transform and deep learning which are efficient techniques for image analysis. We developed the cumulative distribution function (cdf) space covariance model of Gabor wavelet (CSCM-GW), which can jointly model multivariate data in cdf space, in the Gabor wavelet domain to represent texture. The images having different sizes will be transformed by CSCM-GW into same size covariance matrices. Because CSCM-GW is based on the covariance matrix which belongs to Riemannian space, it has the high computational cost in the recognition phase. Therefore, we proposed a novel method of texture recognition called CSCM-GWF-CNN which uses CNN to project the fused covariance of CSCM-GW into low-dimensional vector space for reducing the computational cost and improving the recognition performance. The experiments on Brodatz (111) and KTH-TIPS2-b texture databases show that the proposed method is efficient for texture representation and outperforms most of the state-of-the-art recognition methods.

**Wei Jia et al [2020]** made a study in order to investigate the problem of deep learning based 2D and 3D palmprint recognition and palm vein recognition in-depth, we conduct performance evaluation of seventeen representative and classic convolutional neural networks (CNNs) on one 3D palmprint database, five 2D palmprint databases and two palm vein databases. A lot of experiments have been carried out in the conditions of different network structures, different learning rates, and different numbers of network layers. We have also conducted experiments on both separate data mode and mixed data mode. Experimental results show that these classic CNNs can achieve promising recognition results, and the recognition performance of recently proposed CNNs is better. Particularly, among classic CNNs, one of the recently proposed classic CNNs, i.e., EfficientNet achieves the best recognition accuracy. However, the recognition performance of classic CNNs is still slightly worse than that of some traditional recognition methods.

**Chaorong Li et al [2018]** made a novel method for texture image recognition is proposed in this paper. The aim of the proposed method is to represent texture by combining Gabor wavelet transform and deep learning which are efficient techniques for image analysis. We developed CDF (Cumulative Distribution Function) Space Covariance Model of Gabor wavelet (CSCM-GW), which can jointly model multivariate data in CDF space, in the Gabor wavelet domain to represent texture. The images having different size will be transformed by CSCM-GW into same size covariance matrices. Because CSCM-GW is based on the covariance matrix which belongs to Riemannian space, it has the high computational cost in the recognition phase. Therefore, we proposed a novel method of texture recognition called CSCM-GWF-CNN which uses Convolutional Neural Network (CNN) to project the fused covariance of CSCM-GW into low dimension vector space for reducing the computational cost and improving the recognition performance. Experiments on Brodatz (111) and KTH-TIPS2-b texture databases show the proposed method is efficient for texture representation and outperforms most of the state-of-the-art recognition methods.

**Siti Nurulain Mohd Rum et al [2021]** focuses on the development of a prototype system named FishDeTec to the detect the freshwater fish species found in Malaysia through the image processing approach. In this study, the proposed predictive model of the FishDeTec is developed using the VGG16, is a deep Convolutional Neural Network (CNN) model for a large-scale image classification processing. The experimental study indicates that our proposed model is a promising result.

**D. Bhattacharjee et al [2019]** presented a novel local image descriptor called Pattern of Local Gravitational Force (PLGF) in this paper. It is inspired by Law of Universal Gravitation. PLGF is a hybrid descriptor, which is a combination of two feature components: one is the Pattern of Local Gravitational Force Magnitude (PLGFM), and another is Pattern of Local Gravitational Force Angle (PLGFA). PLGFM encodes the local gravitational force magnitude, and PLGFA encodes the local gravitational force angle that the center pixel exerts on all other pixels within a local neighborhood. We propose a novel noise resistance and the edge-preserving binary pattern called neighbors to center difference binary pattern (NCDBP) for gravitational force magnitude encoding. Finally, the histograms of the two components are concatenated to construct the PLGF descriptor. Experimental results on the existing face recognition databases, texture database, and biomedical image database show that PLGF is an effective image descriptor, and it outperforms other widely used existing descriptors. Even if in complicated variations like noise, and illumination with smaller databases, a combination of PLGF and convolutional neural network (CNN) performs consistently better than other state-of-the-art techniques.

**Turker Tuncer et al [2019]** discussed in this study about a novel chess based local image descriptor is presented for textural image recognition. The proposed descriptor is inspired by chess game and the main objective of it is to extract distinctive textural features using chess game rules. Patterns of the proposed method are created by using the movements of the knight, rook and bishop chessmen and six feature images are constructed using the proposed chess-based textural image descriptor. Therefore, this method is called as chess pattern (chess-pat) consisting of *4* phases. These four phases are block division, binary features calculation using chess patterns, histogram extraction, feature reduction with maximum pooling and classification. In the first phase, the image is divided into *5* x*5* overlapping blocks. To extract the features, the proposed chess patterns are used. In the proposed chess-pat, *6* varied patterns are used based on chessmen moves and 6 feature images are created based on these patterns.

**Swalpa Kumar Roy et al [2021]** made a study on the convolution operation on real world images by using the trainable correlative Gaussian kernel adds correlations to the output images, which hinder the recognition process due to the blurring effects introduced by the convolution kernel application in this paper. As a result the pixel-wise and channel-wise correlations or redundancies could appear in both single and multiple feature maps obtained by a hidden layer. In this sense, convolution-based models fail to generalize the feature representation because of both the strong correlations presence in neighboring pixels and the channel-wise high redundancies between different channels of the feature maps, which hamper the effective training.

Deconvolution operation helps to overcome the shortcomings that limit the conventional SiConvNet performance, learning successfully correlation-free features representation. In this paper, a simple but efficient Siamese convolution deconvolution feature fusion network (SiCoDeF 2 Net) is proposed.

# CHAPTER 3

# EXISTING SYSTEM

## 3.1 INTRODUCTION

In recent years, face recognition plays an important role in intensive research. With the current discerned world security situation, governments as well as private require reliable methods to accurately identify individuals, without overly contravene on rights to privacy or requiring significant compliance on the part of the individual being recognized. Face recognition extends an acceptable solution to this problem. A number of techniques have been applied to face recognition and they can be divided into two categories 1) Geometric feature matching and 2) Template matching. Geometric feature matching involves segmenting the different features of the face: eyes, nose, mouth, etc. and extracting illustrative information about them such as their widths and heights. Values of these measures can then be stored for each person and it can be compared with those of the known individuals. Template matching is a non-segmentation approach to recognize the face. Each face is treated as a two dimensional array of intensity values, which is then compared with other face's intensity arrays. Earliest methods treated faces as points in very high dimensional space and then the Euclidean distance between them is calculated. Dimensional reduction techniques including Principal Component Analysis (PCA) have now been successfully applied to the problem, hence reducing complexity of the recognition process without negatively infringing on accuracy. Nowadays different patterns are used for feature extraction. The Local Binary Pattern (LBP) is designed to encode the relationship between the referenced pixel and its surrounding pixels. LBP is applied successfully to all facial expression analysis. Its performance is much better than Eigen face. LBP produces micro patterns. The center pixel is subtracted from the eight neighbouring pixels. Assign 0 for negative values and assign 1 for the positive values. These micro patterns are constructed by combining eight neighbouring bits clockwise. Due to the simplicity and robustness, it has been widely used in face recognition. However this LBP can fail in some situations. In order to avoid such situation, the Local Tetra Pattern (LTP) was introduced to capture more detailed information than LBP. LTP is an extension of LBP. LTP is less sensitive to noise than LBP as well as small pixel difference is encoded into a separate state. To reduce its dimensionality, the ternary code constructed by LTP is split into two binary codes: a positive LBP and a negative LBP. A threshold value is added with the center pixel (u) and is subtracted from the center pixel(l) and generate a boundary. Assign -1, if the neighbouring pixel is lesser than l, assign 1 if the neighbouring pixel is greater than u and assign 0 if it is lies between l and u . This ternary code is split into two. Assigns 0 to -1's to construct higher bit pattern. Assign 1 to -1's to construct lower bit pattern and construct higher and higher bit pattern. LTP only encodes the texture features of an image depending on the grey level difference between center pixel and its neighbours, which are coded using two directions. Local tetra pattern (LTrP) is an extension of LBP and LTP. It is able to extract more detailed information than LBP and LTP. The LTrP encodes the relationship between the center pixel and its neighbours by using the first-order derivatives in vertical and horizontal directions not like LTP. Local tetra pattern (LTrP) extracts information based on the distribution of edges which are coded using four directions (1,2,3,4). LTrP encodes the relationship depends on the direction of the center pixel and its neighbours, which are calculated by combining (n-1)th - order derivatives of the 0 degree and 90 degree directions. Local Derivative Pattern was proposed by Baochang Zhang for face recognition with high order local pattern descriptor. It encodes directional bit pattern based on local derivative variations. It can capture more detailed information including different angles than the first order LBP. LDP is a micro pattern representation which can also be modeled by histogram. Histogram is used to preserve the information about the distribution of the LDP micro patterns. LBP is always considered first-order local pattern operator, because LBP encodes all-direction first-order derivative binary result whereas LDP encodes directional higher-order derivative information. So it extracts more distinctive features than LBP. This paper proposes the new method Local derivative tetra pattern (LDTrP). Which combines the feature directional and neighbouring patterns from both LDP and LTrP. LBP can only be considered as a non-directional 8-neighbour pattern. LTP is also a non-directional pattern. The turning point in the LTrp and LDP is the direction and n-order derivations. The proposed LDTrP is a micro pattern which can also be modeled by histogram. The rest of this paper is organized as follows. Section 2 introduces and discusses various feature extraction methodologies. In section 3 explains the proposed method in detail. Section 4 discusses the histogram intersection. In section 5 an extensive experiments over JAFFE and ORL databases are conducted to evaluate the performance of the proposed method on face recognition. Finally, conclusion and future work is drawn in section 6 with some discussions.

## 3.2 METHODOLOGIES

The popular methods designed for feature extraction are Local Binary Patterns, Local Ternary Patterns, Local Derivative Pattern and Local Tetra Pattern. Texture is an image feature that describes about the structural arrangement of the surface. It defines the surface properties of an object and their relationship to the neighbouring environment. Although several algorithms are available, LBP follows a simple algorithm whereas LTP is more resistant to noise and small pixel variations. However, LDP extracts higher order information by encoding various spatial relationships contained in a given local region. The LTrP check the relationship between the reference pixel and its neighbours in both vertical and horizontal directions. These methods are summarized in the following section. 2.1.1 Local Binary Pattern (LBP): The standard local binary pattern (LBP) searches the relationship between the referenced pixel and its surrounding 8- neighbours and then calculates the gray-level difference. LBP is a grayscale invariant texture measure and is a useful tool to model texture images. In LBP, it labels the pixels of an image by thresholding the 3×3 neighbourhood of each pixel with the value of central pixel and concatenating the bits from top left. The thresholding function for f (·,·) for the basic LBP.

| | | |
|---|---|---|
| $N_1$ | $N_2$ | $N_3$ |
| $N_8$ | $N_0$ | $N_4$ |
| $N_7$ | $N_6$ | $N_5$ |

**Figure 3.1.** Eight neighbourhoods around N0

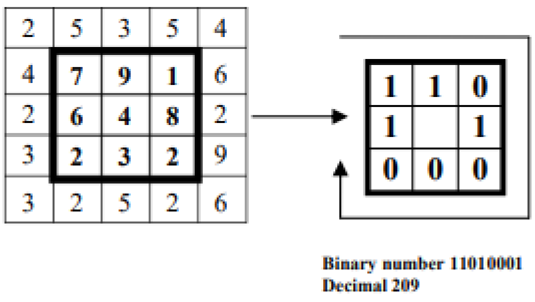


**Figure 3.2.** Micro pattern obtained from the black square

### 3.3 Local Ternary Pattern (LTP):

Local Ternary Pattern is an advanced version of LBP. It has three-valued code with relation to grey values of its neighbours. Unlike LBP, it does not threshold the pixels into 0 and 1, but it uses a threshold constant to threshold pixels into three values 1, 0 and -1. As shown in Fig.3 considering t as the threshold constant, N0 as the value of the center pixel and the value of the neighbouring pixels Ni, i = 1,2,..8. The result of the thresholding function for f (·,·) for the basic LTP.

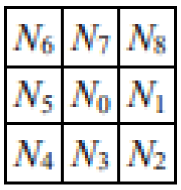


**Figure 3.3** Eight – Neighbourhood around N0

Assign 0 in a cell when the pixel value is between N0-t and N0 + t, where N0 is the center intensity of the pixel. Therefore, because the intensity is 45 in the centre of this window, the range is between , where value of t is 5. Any cells that are above 50 get assigned 1 and any cells that are below 40 get assigned -1. Once construct the ternary code , and then split up the code into higher and lower patterns. Basically, any values that get assigned a -1 get assigned 0 for higher patterns and any values that get assigned a -1 get assigned 1 for lower patterns. The Figure 3.4 shows local ternary pattern, when the threshold is set to 5. Also, for the lower pattern, any value of the cell that is 1 in the original window gets mapped to 0. The final bit pattern starts from the second row third column, then going around anti-clockwise. Therefore, when this modification is made the gives both lower patterns and higher patterns as output to the given image.

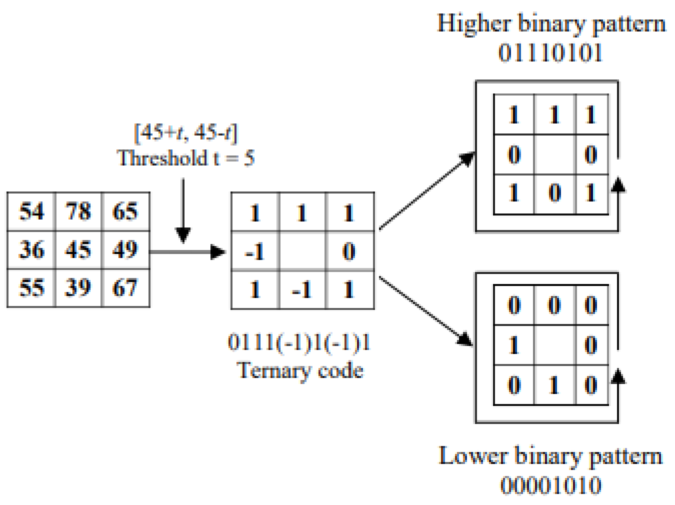


**Figure 3.4** Steps to obtain ternary pattern

## 3.4 METHODOLOGY

The proposed system combines the directional features of both LDP and LTrP. The LDTrp consider 900 direction from LDP and horizontal and vertical directional feature from the LTrP.

### 3.4.1 LOCAL DERIVATIVE TETRA PATTERN (LDTRP):

This section provides a brief review of the second-order local derivative pattern (LDP) to calculate the first-order derivative direction variation and the LTrP extract the texture features of an image based on the distribution of edges. After that, the definition and feasibility of the second-order LDP ( ) ( ) 1 1 90 90 0 , i I Z I Z, where, LTrP are presented and discussed. Finally, the spatial histogram is described for modeling the distribution of LDP of a face. LBP is incapable of describing more detailed information. LDP is a high-order local pattern for face representation. LBP is a no directional first-order local pattern operator, but LDP encodes the higher-order derivative information which contains more detailed features than the LBP. Local Derivative Pattern representation for face recognition, provide a strong discriminative capability in describing detailed texture information. The higher-order LDP can capture more detailed information. LDP encodes directive pattern features from local derivative variation. The main process of this method is that an LDP captures local features in four directions: $0^{o}$ , $45^{o}$ , $90^{o}$ and $135^{o}$ . The goal of the proposed method is to match the most relevant images from the databases. In this paper, the LDP includes LTrP, and the Pattern generated which is used to retrieve feature from the images. This proposed method considers only the $90^{o}$, because human face is vertically symmetrical. Given an image I(N), first order derivatives along $90^{o}$ direction is denoted as ( ) 1 90 I N . Let N0 be a point in I(N) and N0, i = 1,2,...8 be the neighbouring point around N0. The first order derivatives at N = N0.

The proposed LDP operator labels the pixels of an image by comparing two derivative directions at neighbouring pixels. Reflecting various distinctive spatial relationships in a local region.

The existing system combines the directional features of both LDP and LTrP. The LDTrp consider 900 direction from LDP and horizontal and vertical directional feature from the LTrP. The local tetra pattern (LTrP) encodes the relationship between the reference pixel and its neighbours by using the first-order derivatives in vertical and horizontal directions. LTrP extracts values which are based on the distribution of edges which are coded using four directions. The LDTrP combines the higher order directional feature from both LDP and LTrP. Experimental results on ORL and JAFFE database show that the performance of LDTrP is consistently better than LBP, LTP and LDP for face identification under various conditions. The performance of the proposed method is measured in terms of recognition rate.

## 3.6 DISADVANTAGES

- It can also be improved to reduce space.
- Time consumption is more.

# CHAPTER 4

# PROPOSED SYSTEM

## 4.1 INTRODUCTION

Biological features, including human faces, have attracted the attention of many scholars due to their high reliability. Compared with other biometric features, human face contains a lot of detailed information, which is uniqueness for human beings , such as eyes, eyebrows and mouths. In addition, because of the low cost of collection and high performance, the face recognition has been commonly applied to surveillance, biometrics, security and other fields in recent years. The schemes for face recognition can generally be divided into two categories: geometric-based and appearance-based methods. The former uses geometric information to describe the facial features, which incorporates the shape and position of the face into a feature vector. The latter analyses the face information by means of descriptors, and extracts the whole or local features to represent the changes of the facial appearance. Although face analysis and recognition has made great progress in the past decades, those challenges are still the big problem in face recognition fields, such as different expressions, backgrounds, and illumination. For face recognition, the most important is to find an effective and robust descriptor, which can effectively extract the resolvable face features and remove the influence of light, noise and expression. Many approaches for face analysis and recognition have been mentioned in the literature, such as, collaborative preserving fisher discriminant analysis (CPFDA), discriminative multi-scale sparse coding (DMSC), two-dimensional quaternion principal component analysis (2D-QPCA), and real-time face recognition using deep learning and visual tracking (RFR-DLVT). These methods use the global facial information for recognition and they are sensitive to the changes of illumination, partial occlusions, expression and poses. Therefore, local descriptors have been deeply studied in the past decades. As a typical representative, local binary pattern (LBP) was presented, which is simple in principle and low in computational complexity. In addition, it combines structural as well as the statistical features. Because of the advantages of LBP, numerous improved operators have been introduced in recent years. In order to achieve multi-resolution analysis, Ojala *et al.* extended LBP from 8-

neighborhood to arbitrary circular neighborhoods (R,P) , where P and R represent the number of sampling points and the radius respectively. For noise-resistance, Tan and Triggs proposed local ternary pattern (LTP), which extends the binary discriminant to the ternary discriminant, and the modified gradient pattern (MGP) defines a new threshold for binarization on the basis of the modified census transform (MCT).It is generally known that the edge and direction information is important for image analysis. These features were also incorporated into the extended methods of LBP, such as local maximum edge binary patterns (LMEBP), local edge binary pattern (LEBP) , and local tetra patterns (LTrP) . Further, Jabid *et al.* proposed the local directional pattern (LDP) to reduce the influence of the random noise and monotonic illumination changes, which is coded by comparing the edge response values of different directions. However, LDP still produces unreliable codes for the uniform and smooth neighborhoods. Based on LDP, enhanced local directional pattern (ELDP) , local direction number (LDN), local direction texture pattern (LDTP), and local edge direction pattern (LEDP) were put forward to overcome the shortcomings. Besides the methods based on local features, the deep learning approaches concerning Convolutional Neural Network (CNN) have emerged in recent years. Those methods can learn high-quality information in face images by training models on a large amount of data. Although CNN-based methods have attracted considerable attention, the research on local features is still continuing. For most of the LDP-based methods, they are actually coded in intensity space, i.e., the size of the edge response value. In contrast, the local gradient number pattern (LGNP), presented in the paper, is coded in gradient space. In other words, the distribution of gray values is considered in the new coding scheme. Comparing with the intensity space, the gradient space can preserve the intrinsic relationship between the pixels in the neighborhood. These intrinsic relationships essentially reveal the underlying structural information of the local neighborhood, which tends to be more discriminative. For the traditional LDP-based variants, they calculate the edge response values by Kirsch template. In difference, LGNP considers the horizontal and vertical directions of the local neighborhoods by Sobel operators to reduce the computational complexity. Further, the fuzzy convex-concave partition (FCCP) is introduced in the paper based on the convex-concave partition to fuse the spatial features of the face images. By the combination of LGNP and FCCP, the novel code, called FCCP_LGNP, is extracted for face recognition.

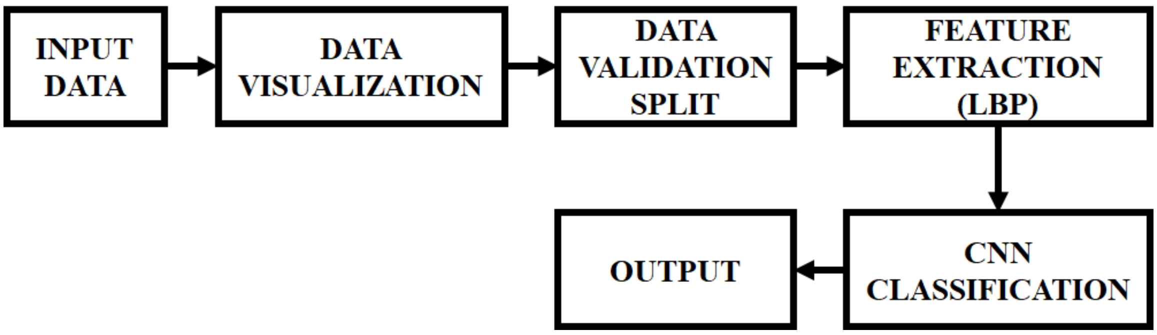


**Figure 4.1** Block diagram of Proposed System

In this project a face recognition system based on a local tetra pattern and a convolutional neural network is proposed. In this project, in order to detect the emotions from the human face gesture, we use the complex process of explicit feature extraction in traditional face expression recognition and a face expression recognition method based on image edge detection. The facial expression image is normalized, and the edge of each layer of the image is extracted in the convolution process. With the help of a local tetra pattern and a convolution neural network, we will be able to automatically detect the face recognition of human emotions. The classifier module is comprised of a global average pooling and softmax layer to calculate the probability of each class. The overall design optimizes the network parameters and maintains classification accuracy.

## 4.2 DATA VISUALIZATION

Data visualization is the representation of data through use of common graphics, such as charts, plots, infographics, and even animations. These visual displays of information communicate complex data relationships and data-driven insights in a way that is easy to understand. Data visualization can be utilized for a variety of purposes, and it’s important to note that is not only reserved for use by data teams. Data visualization is commonly used to spur idea generation across teams. They are frequently leveraged during brainstorming or Design Thinking sessions at the start of a project by supporting the collection of different perspectives and highlighting the common concerns of the collective. While these visualizations are usually unpolished and unrefined, they help set the foundation within the project to ensure that the team is aligned on the problem that they're looking to address for key stakeholders. Data visualization is a critical step in the data science process, helping teams and individuals convey data more effectively to colleagues and decision makers. Teams that manage reporting systems typically leverage defined template views to monitor performance. However, data visualization isn't limited to performance dashboards. For example, while text mining an analyst may use a word cloud to capture key concepts, trends, and hidden relationships within this unstructured data. Alternatively, they may utilize a graph structure to illustrate relationships between entities in a knowledge graph. There are a number of ways to represent different types of data, and it's important to remember that it is a skillset that should extend beyond your core analytics team.

## 4.3 Local Binary Patterns

Local Binary Patterns, or LBPs for short, are a texture descriptor, Multiresolution Grayscale and Rotation Invariant Texture Classification with Local Binary Patterns .Unlike Haralick texture features that compute a global representation of texture based on the Gray Level Co-occurrence Matrix, LBPs instead compute a local representation of texture. This local representation is constructed by comparing each pixel with its surrounding neighborhood of pixels. The first step in constructing the LBP texture descriptor is to convert the image to grayscale. For each pixel in the grayscale image, we select a neighborhood of size r surrounding the center pixel. A LBP value is then calculated for this center pixel and stored in the output 2D array with the same width and height as the input image.

## 4.4 CNN CLASSIFICATION

CNNs are a powerful tool for segmentation and classification, but they're not the only way to do these tasks. CNN architectures have two primary types: segmentations CNNs that identify regions in an image from one or more classes of semantically interpretable objects, and classification CNNs that classify each pixel into one or more classes given a set of real-world object categories. In this article we covered some important points about CNNs including how they work, what their typical architecture looks like, and which applications use them most frequently.

Image segmentation has two levels of granularity. They are as following:

- Semantic segmentation: Semantic segmentation classifies image pixels into one or more classes which are semantically interpretable, rather, real-world objects. Categorizing the pixel values into distinct groups using CNN is known as region proposal and annotation. Region proposals are also called candidate objects patches (COMPs), which can be thought of as small groups of pixels that are likely to belong to the same object.
- Instance segmentation: In case of instance segmentation, each instance of each object is identified. The difference between instance segmentation and semantic segmentation is that semantic segmentation doesn't categorize every pixel. If there are three objects of same type (say, bicycle) in an image, each individual bicycle is identified in instance segmentation while semantic segmentation classifies all the bicycles as one instance.

CNN architecture for segmentation makes use of encoder and decoder models. The encoders are used to encode the input into a representation that can be sent through the network, and then decoders are used to decode the representation back. Encoders can be convolutional neural networks and the decoders can be based on the de-convolutional or transposed neural networks with the purpose of creating a segmentation map. The picture below represents the basic CNN architecture for image segmentation:

# CHAPTER 5

# SYSTEM REQUIREMENTS

## HARDWARE REQUIREMENTS:

- System: Pentium IV 2.4 GHz.
- Hard Disk: 40 GB.
- Monitor: 15 VGA Colour.
- Mouse :
- Ram: 512 Mb.

## SOFTWARE REQUIREMENTS:

- Operating system: Windows XP/7.
- Coding Language : Anaconda Jupyter
- Tool : Python language

# CHAPTER 6
# RESULTS AND DISCUSIION

The performance of the proposed system is computed by using the test images in the database. This experiment conducts comparative performance evaluations on all the forty. All the faces were cropped into 92x112 pixels. It can also conduct comparative performance evaluations on all the 10 subsets with expression, lighting and aging variations. The goal of the proposed system is to detect the most relevant images from the database. In this paper, LBP includes the features of LBP and LBP. This is used to retrieve features from the images. The query image and images in the database are compared by using histogram intersection for obtaining the same measurement and best matched images are retrieved from the database of images in response to query image. The database was retrieved from URLs as a ZIP file of same sizes. For the implementation, they should be converted into tiff files.

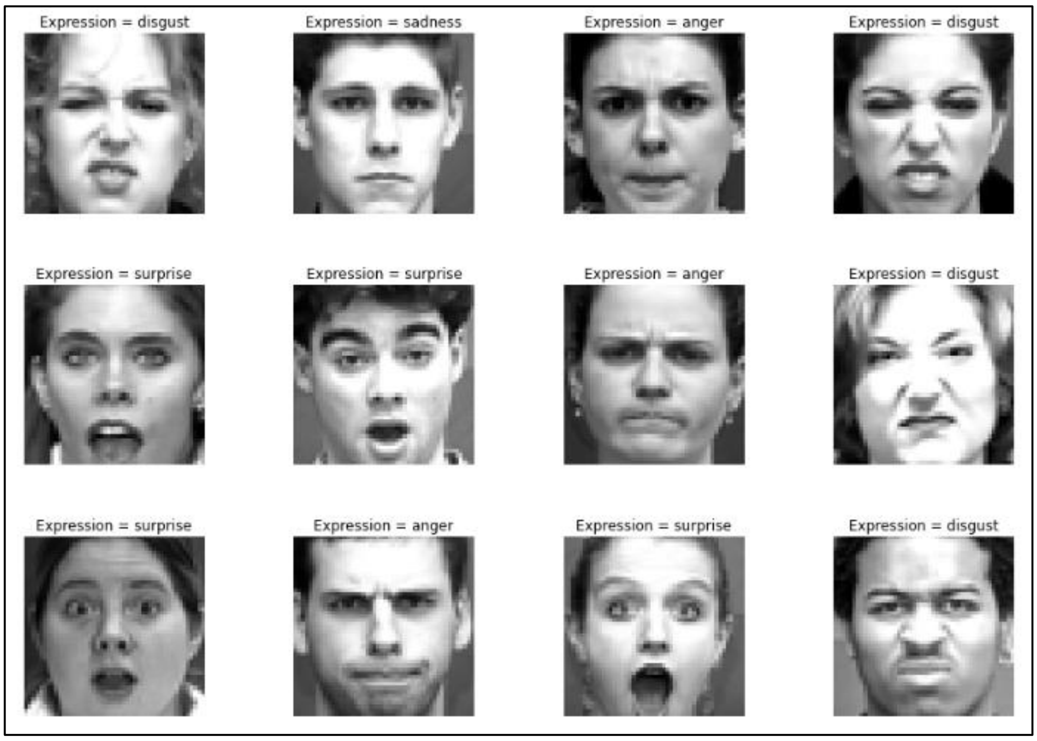
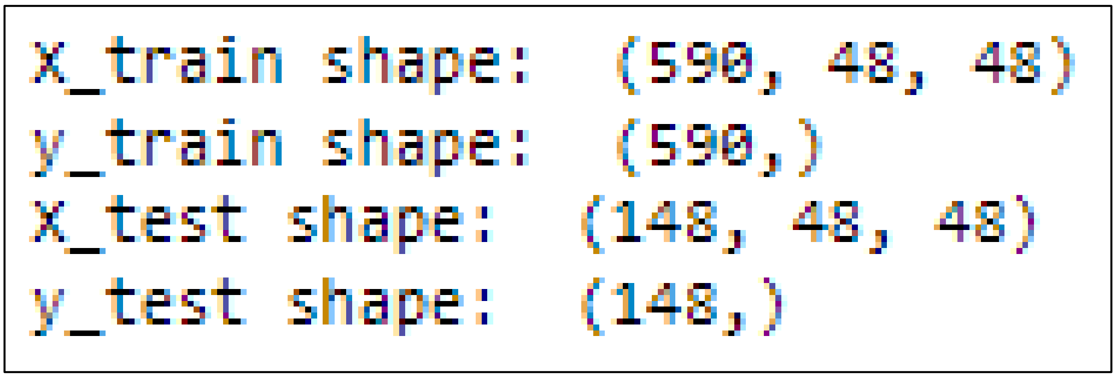


**Figure 6.1** Input Database

Input database is illustrated in figure 5.1. Various expressions are given to the data, thus gives the correct face recognition in the output.



```
x_train shape:  (590, 48, 48)
y_train shape:  (590,)
x_test shape:  (148, 48, 48)
y_test shape:  (148,)
```

**Figure 6.2** Validation split

```
(-0.5, 47.5, 47.5, -0.5)
```

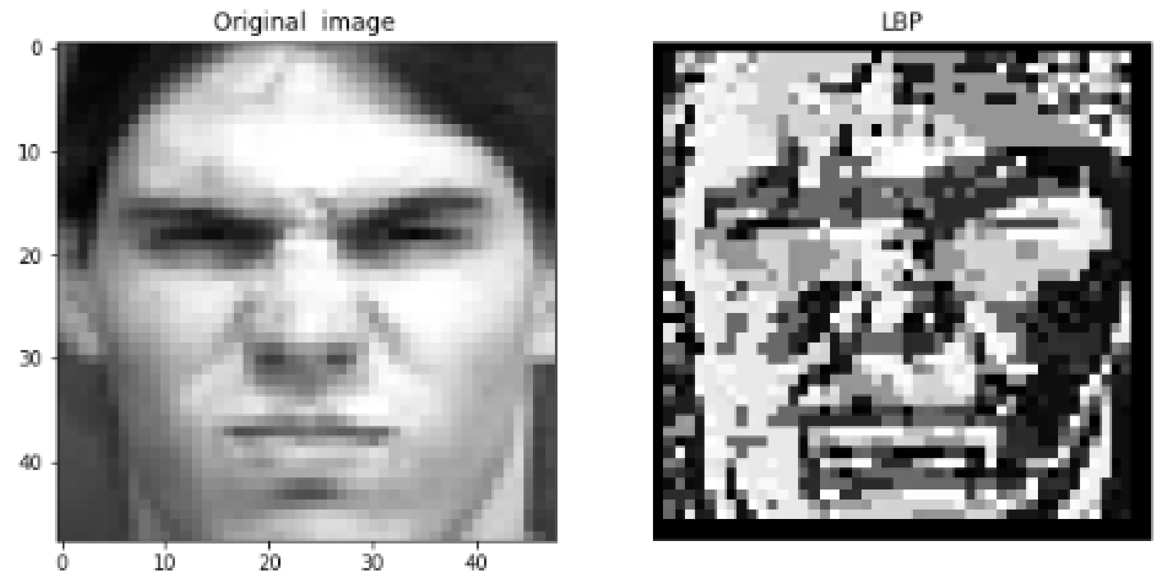


**Figure 6.3** LBP

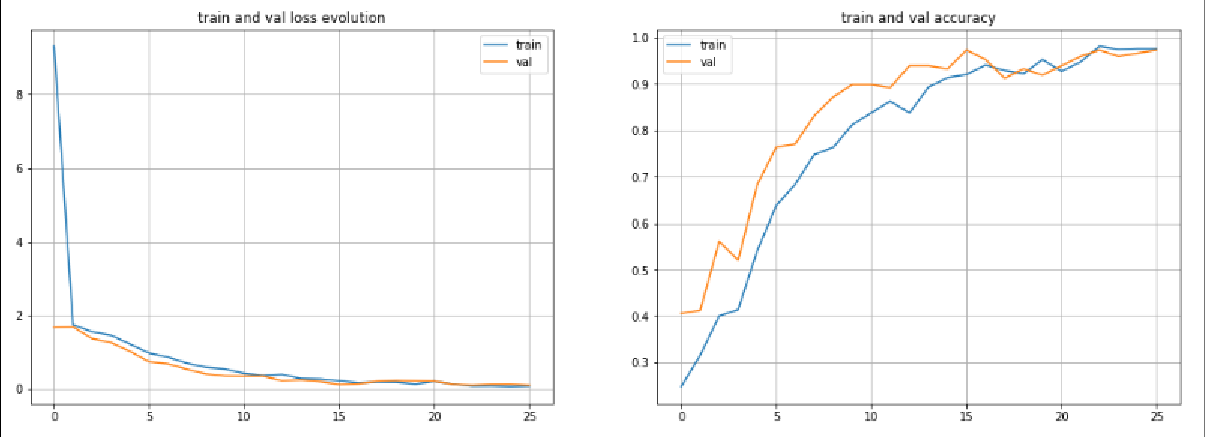
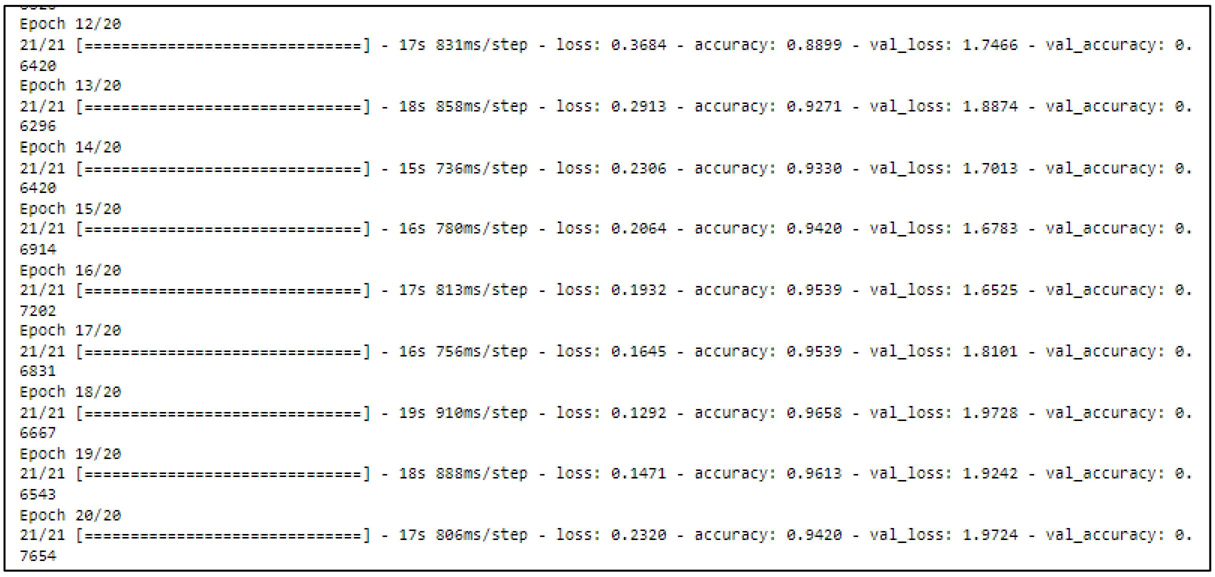


**Figure 6.4** Accuracy graph



```
Epoch 12/20
21/21 [==============================] - 17s 831ms/step - loss: 0.3684 - accuracy: 0.8899 - val_loss: 1.7466 - val_accuracy: 0.
6420
Epoch 13/20
21/21 [==============================] - 18s 858ms/step - loss: 0.2913 - accuracy: 0.9271 - val_loss: 1.8874 - val_accuracy: 0.
6296
Epoch 14/20
21/21 [==============================] - 15s 736ms/step - loss: 0.2306 - accuracy: 0.9330 - val_loss: 1.7013 - val_accuracy: 0.
6420
Epoch 15/20
21/21 [==============================] - 16s 780ms/step - loss: 0.2064 - accuracy: 0.9420 - val_loss: 1.6783 - val_accuracy: 0.
6914
Epoch 16/20
21/21 [==============================] - 17s 813ms/step - loss: 0.1932 - accuracy: 0.9539 - val_loss: 1.6525 - val_accuracy: 0.
7202
Epoch 17/20
21/21 [==============================] - 16s 756ms/step - loss: 0.1645 - accuracy: 0.9539 - val_loss: 1.8101 - val_accuracy: 0.
6831
Epoch 18/20
21/21 [==============================] - 19s 910ms/step - loss: 0.1292 - accuracy: 0.9658 - val_loss: 1.9728 - val_accuracy: 0.
6667
Epoch 19/20
21/21 [==============================] - 18s 888ms/step - loss: 0.1471 - accuracy: 0.9613 - val_loss: 1.9242 - val_accuracy: 0.
6543
Epoch 20/20
21/21 [==============================] - 17s 806ms/step - loss: 0.2320 - accuracy: 0.9420 - val_loss: 1.9724 - val_accuracy: 0.
7654
```

**Figure 6.5** Classification

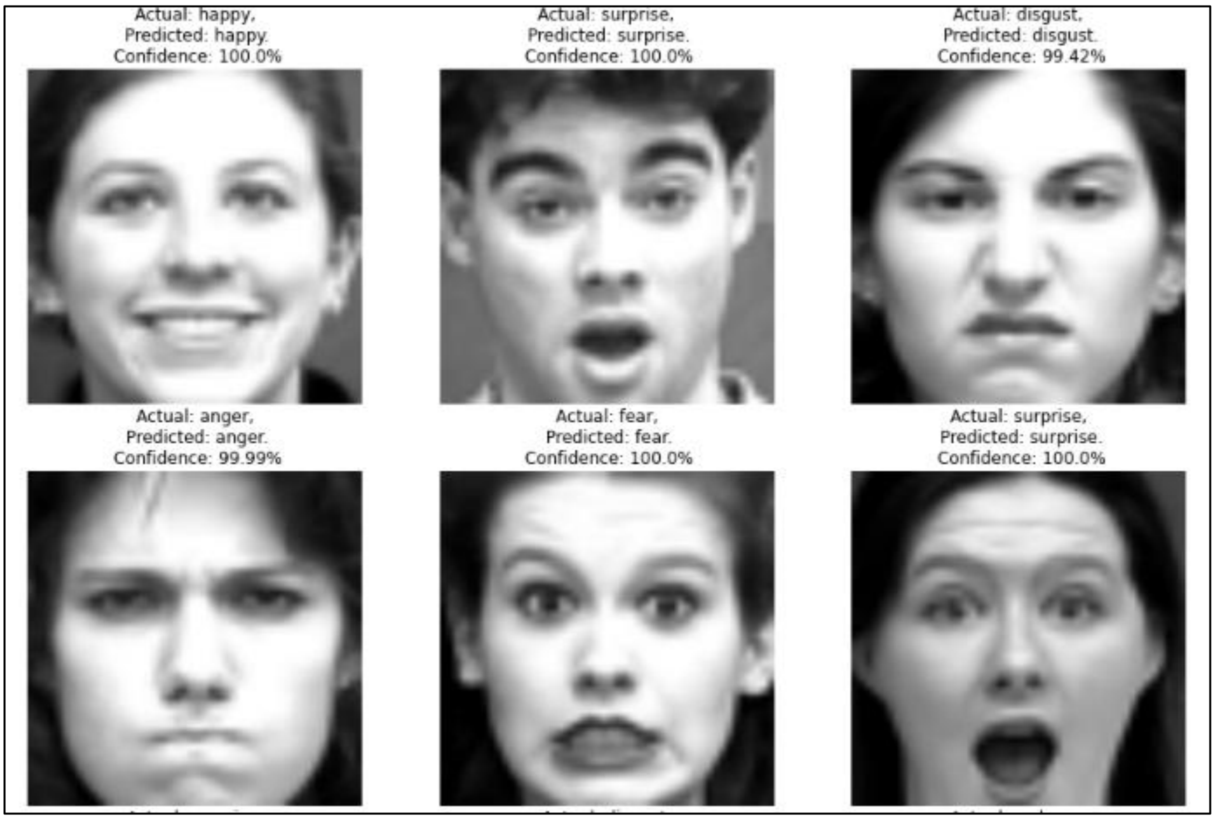
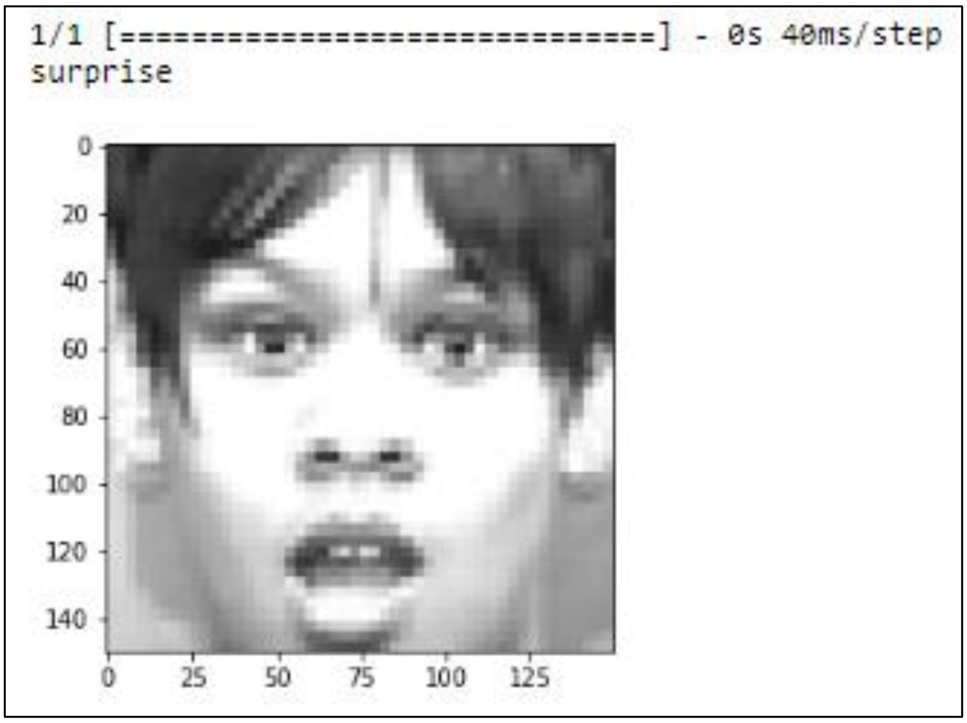
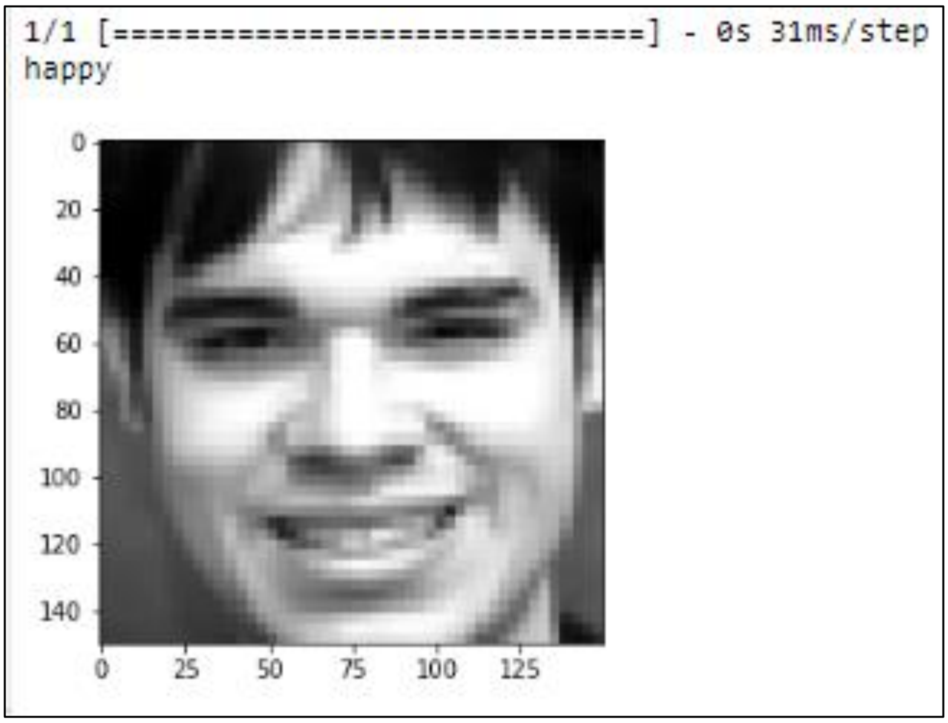


**Figure 6.6** Predicted Output

Predicted output is shown in figure 6.6. The output shows the desired result and achieves 99.9% in accuracy, Happy, Surprise, Disgust, Anger, Fear and all everything shows perfectly in output.



```
1/1 [==============================] - 0s 40ms/step
surprise
```



```
1/1 [==============================] - 0s 31ms/step
happy
```

**Figure 6.7** Output

# CHAPTER 7
# CONCLUSION

Humans are able to share multiple emotions and feelings through their facial gestures and body language. In this paper, we proposed a face recognition system based on a local tetra pattern and a convolutional neural network. In order to detect the emotions from the human face gesture, we use the complex process of explicit feature extraction in traditional face expression recognition and a face expression recognition method based on image edge detection. The facial expression image is normalized, and the edge of each layer of the image is extracted in the convolution process. With the help of a local tetra pattern and a convolution neural network, automatically detect the face recognition of human emotions is achieved. The classifier module is comprised of a global average pooling and softmax layer to calculate the probability of each class. The overall design optimizes the network parameters and maintains classification accuracy. The project is implemented using python Jupyter software and gives a good result in the final output.

## REFERENCE

[1] J. Sun, Y. Lv, C. Tang, H. Sima and X. Wu, "Face Recognition Based on Local Gradient Number Pattern and Fuzzy Convex-Concave Partition," in IEEE Access, vol. 8, pp. 35777-35791, 2020.

[2] K. K. Kamarajugadda and P. Movva, "A Novel Multi-Angular LTP and MLDA Based Face Recognition Using Modified Feed Forward Neural Network," 2019 IEEE 10th Annual Information Technology, Electronics and Mobile Communication Conference (IEMCON), Vancouver, BC, Canada, 2019, pp. 0647-0553.

[3] Santhosh, S and Dr SV Rajashekararadhya. "Efficient Face Recognition System Using Z-Normalization and Moore Penrose-Based Deep Convolutional Neural Network." Advanced Engineering Science 54, no. 02 (2022): 3177-3202.

[4] C. Li, X. Guan, P. Yang, Y. Huang, W. Huang and H. Chen, "CDF Space Covariance Matrix of Gabor Wavelet With Convolutional Neural Network for Texture Recognition," in IEEE Access, vol. 7, pp. 30693-30701, 2019.

[5] Jia, Wei, Jian Gao, Wei Xia, Yang Zhao, Hai Min, and Jing-Ting Lu. "A performance evaluation of classic convolutional neural networks for 2D and 3D palmprint and palm vein recognition." International Journal of Automation and Computing 18, no. 1 (2021): 18-44.

[6] HUANG, WEI, and HUAFU CHEN. "CDF Space Covariance Matrix of Gabor Wavelet with Convolutional Neural Network for Texture Recognition."

[7] Rum, Siti Nurulain Mohd, and Fariz Az Zuhri Nawawi. "FishDeTec: a fish identification application using image recognition approach." International Journal of Advanced Computer Science and Applications 12, no. 3 (2021).

[8] D. Bhattacharjee and H. Roy, "Pattern of Local Gravitational Force (PLGF): A Novel Local Image Descriptor," in IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 43, no. 2, pp. 595-607, 1 Feb. 2021.

[9] Tuncer, Turker, Sengul Dogan, and Volkan Ataman. "A novel and accurate chess pattern for automated texture classification." Physica A: Statistical Mechanics and its Applications 536 (2019): 122584.

[10] S. K. Roy, P. Kar, M. E. Paoletti, J. M. Haut, R. Pastor-Vargas and A. Robles-Gómez, "SiCoDeF$^2$ Net: Siamese Convolution Deconvolution Feature Fusion Network for One-Shot Classification," in IEEE Access, vol. 9, pp. 118419-118434, 2021.

# APPENDIX

```
#!/usr/bin/env python
# coding: utf-8
# In[1]:
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import cv2
import os
import random
from tqdm import tqdm
from sklearn.model_selection import train_test_split
import tensorflow as tf
from tensorflow.keras.preprocessing.image import ImageDataGenerator
from tensorflow.keras.models import Sequential
from tensorflow.keras.callbacks import Callback, EarlyStopping,
ReduceLROnPlateau
from tensorflow.keras.layers import Conv2D, BatchNormalization, MaxPool2D,
Dropout, Flatten, Dense
from tensorflow.keras.optimizers import Adam,RMSprop,SGD,Adamax
from tensorflow.keras import regularizers
# In[2]:
pip install keras
# In[3]:
pip install opencv_python
# In[4]:
pip install tensorflow
# In[5]:
np.random.seed(42)
# In[6]:
os.listdir('F:\\project\\ABT-639\\data')
# In[20]:
train_dir = 'F:\\project\\ABT-639\\data\\train\\'
test_dir = 'F:\\project\\ABT-639\\data\\test\\'
# In[21]:
data = r'F:\\project\\ABT-639\\data\\train'
# In[22]:
batch_size=32
dataset = tf.keras.preprocessing.image_dataset_from_directory(
    train_dir,
    seed=123,
    shuffle=True,
    image_size=(150,150),
    batch_size=batch_size
)
# In[23]:
val_ds = tf.keras.preprocessing.image_dataset_from_directory(
    test_dir,
    seed=123,
    shuffle=True,
    image_size=(150,150),
    batch_size=batch_size
)
# In[24]:
catagories = os.listdir(data)
catagories
# In[25]:
class_names=dataset.class_names
# In[26]:
def load_data():
    datadir= r'F:\\project\\ABT-639\\data\\train'
    data = []
    for category in catagories:
        path = os.path.join(datadir, category)
        class_num = catagories.index(category)
        for img in tqdm(os.listdir(path)):
            img_array = cv2.imread(os.path.join(path, img), 0)
            data.append([img_array, class_num])
    return data
# In[27]:
data = load_data()
# In[28]:
len(data)
# In[29]:
L = 4
W = 4
fig, axes = plt.subplots(L, W, figsize = (15,15))
axes = axes.ravel()
for i in range(0, L * W):
    sample = random.choice(data)
    axes[i].set_title("Expression = "+str(catagories[sample[1]]))
    axes[i].imshow(sample[0], cmap='gray')
    axes[i].axis('off')
plt.subplots_adjust(wspace=0.5)
```

```
# In[30]:
X = np.array([ x[0] for x in data])
y = np.array([Y[1] for Y in data])
# In[31]:
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42, shuffle = True)
# In[32]:
print("X_train shape: ", X_train.shape)
print("y_train shape: ", y_train.shape)
print("X_test shape: ", X_test.shape)
print("y_test shape: ", y_test.shape)
# In[33]:
# reshaping y_train and y_test
y_train = np.reshape(y_train, (len(y_train),1))
y_test  = np.reshape(y_test , (len(y_test ),1))
print("y_train shape: ", y_train.shape)
print("y_test shape: ", y_test.shape)
# In[34]:


X_train_Gabor  = X_train
X_test_Gabor = X_test
# In[35]:
X_train = np.expand_dims(X_train, axis=3)
X_test = np.expand_dims(X_test, axis=3)
print("X_train shape: ", X_train.shape)
print("X_test shape: ", X_test.shape)
# In[36]:
X_train = X_train / 255.0
X_test = X_test / 255.0
# In[37]:
y_train[0]
# In[38]:
y_train_SVM = y_train
y_test_SVM = y_test
y_train = tf.keras.utils.to_categorical(y_train)
y_test = tf.keras.utils.to_categorical(y_test)
# In[39]:
y_train[0]
# In[40]:
y_train.shape, y_test.shape
# In[41]:
def Binarypattern(im):
    img= np.zeros_like(im)
    n=3
    for i in range(0,im.shape[0]-n):
        for j in range(0,im.shape[1]-n):
            x  = im[i:i+n,j:j+n]
            center       = x[1,1]
            img1        = (x >= center)*1.0
            img1_vector = img1.T.flatten()
            img1_vector = np.delete(img1_vector,4)
            digit = np.where(img1_vector)[0]
            if len(digit) >= 1:
                num = np.sum(2**digit)
            else:
                num = 0
            img[i+1,j+1] = num
    return(img)
# In[42]:
plt.figure(figsize = (10,10))
plt.subplot(1,2,1)
img = random.choice(X_train)
plt.title("Original  image")
plt.imshow(img, cmap='gray')
plt.subplot(1,2,2)
plt.title("LBP")
imgLBP=Binarypattern(img)
plt.imshow(imgLBP, cmap='gray')
plt.axis('off')
# In[43]:
X_train.shape
# In[44]:
def create_LBP_features(data):
    Feature_data = np.zeros(data.shape)
    for i in range(len(data)):
        img = data[i]
        imgLBP=Binarypattern(img)
        Feature_data[i] = imgLBP
        return Feature_data
# In[45]:
Feature_X_train = create_LBP_features(X_train)
# In[46]:
Feature_X_train.shape
# In[47]:
img = random.choice(Feature_X_train)
```

```
plt.imshow(img, cmap='gray')
# In[48]:
Feature_X_test = create_LBP_features(X_test)
Feature_X_test.shape
# In[49]:
img = random.choice(Feature_X_test)
plt.imshow(img, cmap='gray')
# In[50]:
model = Sequential()
model.add(Conv2D(6, (5, 5), input_shape=(48,48,1), padding='same', activation
= 'relu'))
model.add(MaxPool2D(pool_size=(2, 2)))
model.add(Conv2D(16, (5, 5), padding='same', activation = 'relu'))
model.add(MaxPool2D(pool_size=(2, 2)))
model.add(Conv2D(64, (3, 3), activation = 'relu'))
model.add(MaxPool2D(pool_size=(2, 2)))
model.add(Flatten())
model.add(Dense(128, activation = 'relu'))
model.add(Dropout(0.5))
model.add(Dense(7, activation = 'softmax'))
   # In[51]:
es = EarlyStopping(
   monitor='val_accuracy', min_delta=0.0001, patience=10, verbose=2,
   mode='max', baseline=None, restore_best_weights=True
)
lr = ReduceLROnPlateau(
   monitor='val_accuracy', factor=0.1, patience=5, verbose=2,
   mode='max', min_delta=1e-5, cooldown=0, min_lr=0
)
callbacks = [es, lr]
# In[52]:
LBP_model = model
# In[53]:
LBP_model.compile(loss='categorical_crossentropy', metrics=['accuracy'],
optimizer='adam' )
# In[54]:
LBP_history = LBP_model.fit(Feature_X_train, y_train, batch_size=8 ,
epochs=50, validation_data = (Feature_X_test, y_test) ,callbacks = [callbacks])
# In[55]:
plt.figure(figsize=(18, 6))
plt.subplot(1, 2, 1)
plt.plot(LBP_history.history['loss'], label='train')
plt.plot(LBP_history.history['val_loss'], label='val')
plt.legend()
plt.grid()
plt.title('train and val loss evolution')
plt.subplot(1, 2, 2)
plt.plot(LBP_history.history['accuracy'], label='train')
plt.plot(LBP_history.history['val_accuracy'], label='val')
plt.legend()
plt.grid()
plt.title('train and val accuracy')
# In[56]:
acc = []
LBP_acc = LBP_model.evaluate(Feature_X_test, y_test, verbose = 0)[1]
acc.append(LBP_acc)
print("LBP Accuracy :",LBP_model.evaluate(Feature_X_test, y_test, verbose =
0)[1])
# In[57]:
y_pred =LBP_model.predict(Feature_X_test)
from sklearn.metrics import f1_score, precision_score, recall_score,
accuracy_score
acc = accuracy_score(y_test, y_pred.round())
recall = recall_score(y_test, y_pred.round(),average='micro')
precision = precision_score(y_test, y_pred.round(),average='micro')
f1s = f1_score(y_test, y_pred.round(),average='micro')
print("Accuracy: "+ "{:.2%}".format(acc))
print("Recall: "+ "{:.2%}".format(recall))
print("Precision: "+ "{:.2%}".format(precision))
print("F1-Score: "+ "{:.2%}".format(f1s))
# In[58]:
def get_dataset_partitions_tf(ds, train_split=0.9, test_split=0.1, shuffle=True,
shuffle_size=10000):
    assert (train_split + test_split) == 1
        ds_size = len(ds)
   if shuffle:
        ds = ds.shuffle(shuffle_size, seed=12)
    train_size = int(train_split * ds_size)
    test_size = int(test_split * ds_size)
     train_ds = ds.take(train_size)
    test_ds = ds.skip(train_size).take(test_size)
     return train_ds, test_ds
# In[59]:
train_ds, test_ds = get_dataset_partitions_tf(dataset)
# In[60]:
```

```
train_ds =
train_ds.cache().shuffle(1000).prefetch(buffer_size=tf.data.AUTOTUNE)
val_ds =
val_ds.cache().shuffle(1000).prefetch(buffer_size=tf.data.AUTOTUNE)
test_ds =
test_ds.cache().shuffle(1000).prefetch(buffer_size=tf.data.AUTOTUNE)
# In[61]:
from keras.layers import Conv2D, MaxPooling2D ,
AveragePooling2D,GlobalAveragePooling2D
# In[62]:
def model(width , height):
   model = Sequential()
   model.add(Conv2D(30, kernel_size=(3, 3),
                 activation='relu',
                 input_shape=( width, height, 3)))
   model.add(MaxPooling2D(pool_size=(2, 2)))
   model.add(Conv2D(15, (3, 3), activation='relu'))
   model.add(MaxPooling2D(pool_size=(2, 2)))
   model.add(Dropout(0.25))
   model.add(Flatten())
   model.add(Dense(128, activation='relu'))
   model.add(Dropout(0.5))
   model.add(Dense(50, activation='relu'))
    model.add(Dense(25, activation='softmax'))

model.compile(loss=tf.keras.losses.SparseCategoricalCrossentropy(from_logits
=False), optimizer = 'adam', metrics=['accuracy'])
   return model
# In[63]:
model=model(150 , 150)
# In[64]:
model_fit = model.fit(train_ds, epochs=20 ,batch_size = batch_size
,validation_data=val_ds, verbose =1)
# In[65]:
model.save("face.h5")
# In[66]:
for images_batch, labels_batch in test_ds.take(1):
      first_image = images_batch[0].numpy().astype('uint8')
   first_label = labels_batch[0].numpy()
    print("first image to predict")
   plt.imshow(first_image)
   print("actual label:",class_names[first_label])
```

```
    batch_prediction = model.predict(images_batch)
    print("predicted label:",class_names[np.argmax(batch_prediction[0])])
# In[67]:
def predict(model, img):
    img_array = tf.keras.preprocessing.image.img_to_array(images[i].numpy())
    img_array = tf.expand_dims(img_array, 0)
    predictions = model.predict(img_array)
    predicted_class = class_names[np.argmax(predictions[0])]
    confidence = round(100 * (np.max(predictions[0])), 2)
    return predicted_class, confidence
# In[68]:


plt.figure(figsize=(15, 15))
for images, labels in train_ds.take(3):
    for i in range(9):
        ax = plt.subplot(3, 3, i + 1)
        plt.imshow(images[i].numpy().astype("uint8"))
       predicted_class, confidence = predict(model, images[i].numpy())
        actual_class = class_names[labels[i]]
        plt.title(f"Actual: {actual_class},\n Predicted: {predicted_class}.\n
Confidence: {confidence}%")
        plt.axis("off")
# In[69]:
model.summary()
# In[70]:
from keras.models import load_model
model=load_model("face.h5")
# In[71]:
from keras.preprocessing import image
import keras.utils as image
img_path="F:\\project\\ABT-639\\data\\input\\4.png"
img = image.load_img(img_path, target_size=(150, 150))
plt.imshow(img)
img_tensor = image.img_to_array(img)
img_tensor = np.expand_dims(img_tensor, axis=0)
pred=model.predict(img_tensor)
print(class_names[np.argmax(pred)])
```