Major Project (CS802PC)

on

EVOLUTIONARY ALGORITHM BASED FEATURE SELECTION AND FUSION FOR FACIAL EXPRESSION RECOGNITION

Submitted

in partial fulfillment of the requirements for

the award of the degree of

Bachelor of Technology

in

Computer Science and Engineering

by

Ms. R. MEGHNA (18261A05A6)

Ms. S. LAXMI PRANITHA (18261A05A8)

Under the Guidance of

Mr. P. SATYA SHEKAR VARMA

(Assistant Professor)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

MAHATMA GANDHI INSTITUTE OF TECHNOLOGY

(Affiliated to Jawaharlal Nehru Technological University Hyderabad)

Kokapet(V), Gandipet(M), Hyderabad

Telangana - 500 075. (India)

2021-2022

MAHATMA GANDHI INSTITUTE OF TECHNOLOGY

(Affiliated to Jawaharlal Nehru Technological University Hyderabad)
Gandipet, Hyderabad - 500075, Telangana.

Certificate

This is to certify that the project entitled **Evolutionary Algorithm Based Feature Selection And Fusion For Facial Expression Recognition** being submitted by **Ms. R. Meghna** bearing roll no:**18261A05A6 and Ms. S. Laxmi Pranitha** bearing roll no:**18261A05A8** in partial fulfilment for the award of **Bachelor of Technology** in **Computer Science and Engineering** to **Jawaharlal Nehru Technological University Hyderabad** is a record of bonafide work carried out by them under our guidance and supervision.

Guide	Head of the Department
Mr. P. Satya Shekar Varma	**Dr. C.R.K. Reddy**
Assistant Professor	Professor

External Examiner

Declaration

This is to certify that the work reported in this project titled "**Evolutionary Algorithm Based Feature Selection And Fusion For Facial Expression Recognition"** is a record of work done by us in the Department of Computer Science and Engineering, Mahatma Gandhi Institute of Technology, Hyderabad.

No part of the work is copied from books/journals/internet and wherever the portion is taken, the same has been duly referred in the text. This report is based on the work done entirely by us and not copied from any other source.

The results embodied in this project have not been submitted to any other university or Institute for the award of any degree or diploma.

R. MEGHNA
H.T. No:18261A05A6
S. LAXMI PRANITHA
H.T. No: 18261A05A8

Acknowledgement

The satisfaction that accompanies the successful completion of any task would be incomplete without the mention of people who made it possible because success is the abstract of hard work and perseverance, but steadfast of all is encouraging guidance. So, we acknowledge all those whose guidance and encouragement served as a beacon light and crowned our efforts with success.

We would like to express my sincere thanks to our guide **Mr. P. Satya Shekar Varma**, **Assistant Professor**, Department of CSE, MGIT, for his constant guidance, encouragement and moral support throughout the project.

We are extremely thankful to **Dr. A. Nagesh, Professor, Dr. V. Subba Ramaiah, Sr. Assistant Professor and A. Ratnaraju, Assistant Professor**, Department of CSE, Project work Coordinators, MGIT, for their encouragement and support throughout the project.

We convey our heartfelt thanks to **Dr. C.R.K Reddy**, **Professor & HOD**, Department of Computer Science and Engineering, MGIT, for all the timely support and valuable suggestions during the period of project.

We are also thankful to **Dr. K. Jaya Sankar, Principal MGIT**, for providing the work facilities in the college.

Finally, we would like to thank all the faculty and staff of CSE Department who helped us directly or indirectly, for completing this project.

R.MEGHNA
HT. No:18261A05A6
S.LAXMI PRANITHA
H.T. No:18261A05A8

Table of contents

List of figures

List of tables

Abstract

Emotion recognition has become one of the most active research areas in pattern recognition due to the emergence of human–machine interaction systems. Describing facial expression is a very challenging problem since it relies on the quality of the face representation. Commonly, two types of features are used to discriminate facial emotions: geometric and appearance features. But one major drawback with them is the difficulty to generalize across different persons. To solve this pitfall a genetic programming (GP) based framework is proposed for posed and spontaneous facial expression recognition allowing to combine hybrid facial features, then tested on the fusion of geometric and appearance features. The expected results, on four posed and spontaneous facial expression datasets (DISFA, DISFA+, CK+ and MUG), with the proposed facial expression recognition method to outperform or achieve a comparable performance to the state-of-the-art methods.

1. Introduction

Recognition of human emotions has long been the subject of active research area. A wide range of human interaction applications have to decipher the facial emotional state. Unlike other non-verbal gesture, the emotional state of the face can be relied to several expressions. Most research has focused on posed facial expressions and reached high level of efficiency recognizing human emotions.

However, some posed expressions are still very hard to discriminate such as disgust and sadness emotions. Quite fewer works have done advances interpreting spontaneous facial emotions. There are several factors affecting the precision of facial expression recognition (FER) systems on spontaneous or posed expressions, including prominent facial feature selection, feature fusion and classifier design. Since, FER applications have to deal with natural emotions, our goal in this work is to develop a system that can achieve accurate recognition rates on posed as well as on spontaneous facial expressions. Extracting efficient facial features is crucial towards facial emotion recognition.

Commonly, two types of features are used to discriminate facial emotions:

- Geometric features
- Appearance features.

Geometric features give clues about shape and position of face components. **Appearance features** contain information about the furrows, bulges, wrinkles, etc. Appearance features contain micro-patterns which provide important information about the facial expressions. But one major drawback with them is the difficulty to generalize across different persons. Although geometric features are noise sensitive and difficult to extract, they proved to be sufficient to give accurate facial expression recognition results. There is a misclassification among different expressions like disgust and sadness using geometric based features and correctly recognized using local binary patterns (LBP). Therefore, facial geometry distortions, given by geometrical features, are complementary with textural information captured by appearance ones. In other words, considering geometrical and appearance feature fusion can be an interesting way to design more discriminative features to deal with FER challenges.

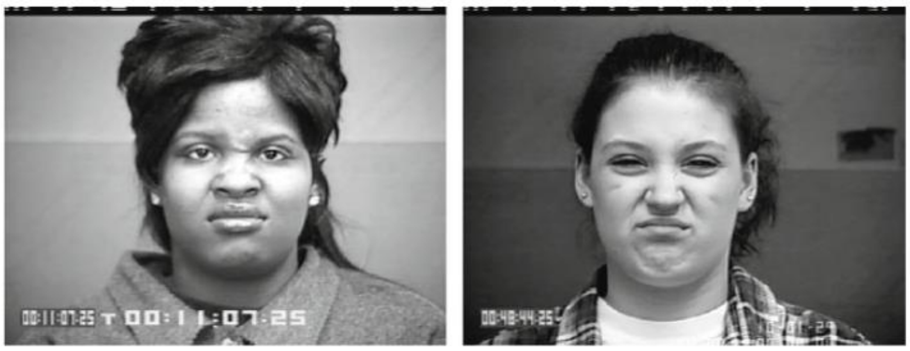


Figure 1.1 Misclassification of images

Feature fusion based methods face the problem of high dimensionality which can affect the quality of the facial emotion recognition. Indeed, dealing with large number of features can increase the computational time and overwhelm classifiers with unnecessary or redundant information. In this case, a rigorous feature selection step is necessary. To carry out selection of a good feature subset, several factors must be considered. First, feature selection cannot be performed in the same way for spontaneous and posed expressions. Indeed, spontaneous facial muscle movements have been proven significantly different from deliberate ones. Performing a static selection method and choosing relatively the best features subset to implement the prediction for all the expressions may not be efficient. In fact, choosing the average does not always mean choosing the best. Although the selected subset has shown good results in most expressions, it may perform poorly in others. Thus, for better training and emotion detection, choosing the right and effective features is crucial as irrelevant and noisy features may mislead and negatively affect the recognition system.

So here a genetic programming (GP) based framework is proposed for posed and spontaneous facial expression recognition allowing to combine hybrid facial features, then we test it on the fusion of geometric and appearance features. The feature selection and fusion, in this work, are performed in a binary way: the most prominent features are selected and fused differently for each pair of expressions.

1.1 Problem Definition

Describing facial expression is a very challenging problem since it relies on the quality of the face representation. A multitude of features have been proposed in the literature to describe facial expression. None of these features is universal for accurately capturing all the emotions since facial expressions vary according to the person, gender and type of emotion (posed or spontaneous).To develop an enhanced system for Facial Emotion Recognition using Evolutionary Algorithm that can perform better with both natural and posed emotions reducing the misclassifications compared with the previous system of FER.

1.2 Existing System

There are many methods by which Facial Expression can be Recognized. Using Support Vector Machine (SVM) applies facial feature point displacements and local texture differences between the neutral and apes expressions. It achieves an accuracy of 95% using CK+ dataset with support vector machine (SVM) classifier. Uisng Gabor filters accuracy of 90% is achieved for appearance features and some distance based methods are used for geometric features. Like this many methods are used like active shapes model (ASM) algorithm, active appearance model (AMM) empirical normalized distances approach and correlation features selection (CFS) approach. Facial Expression Recognition (FER) is traditionally implemented using geometric features or texture features independently. But in some examples emotions having similar facial alignment are being misclassified. To overcome these pitfalls we have proposed another system.

1.3 Proposed System

The proposed system is implemented using python. In this work, a genetic programming framework for feature selection and fusion for facial expression recognition is proposed, which we called GP − FER. The main component of this framework is a tree-based genetic program with three functional layers (feature selection, feature fusion and classification). The proposed genetic program is a binary classifier that performs discriminative feature selection and fusion differently for each pair of expression classes. The predicted expression for input image is captured by performing unique tournament elimination between all the classes using learned binaryprograms.

1.4 Requirements Specification

1.4.1 Software Requirements

The following details are the software requirements for Medical Image Enhancement using Evolutionary Algorithm.

Operating System	: Windows 10, Linux, Mac OS
Programming Language	: Python
Packages	: Numpy, Opencv, Matplotlib, Skimage, Genetic Algorithm and dlib
IDE	: Pycharm

1.4.2 Hardware Requirements

The following details are the hardware requirements for Medical Image Enhancement using Evolutionary Algorithm.

RAM	: 4GB
Processor	: Intel i3 or higher
Hard Disk	: 20GB.

1.5 Software Tools Used

1.5.1 Python

Python is an interpreted, object-oriented, high-level programming language with dynamic semantics. Python is simple and easy to learn. Python supports modules and packages, which encourages program modularity and code reuse. The Python interpreter and the extensive standard library are available in source or binary form without charge for all major platforms, and can be freely distributed. Often, programmers fall in love with Python because of the increased productivity it provides. Since there is no compilation step, the edit-test-debug cycle is incredibly fast. Debugging Python programs is easy and a bug or bad input will never cause a segmentation fault. Instead, when the interpreter discovers an error, it raises an exception. When the program doesn't catch the exception, the interpreter prints a stack trace. A source level debugger allows inspection of local and global variables, evaluation of arbitrary expressions, setting breakpoints, stepping through the code a line at a time, and so on. The debugger is written in Python itself, testifying to Python's introspective power. The Proposed System works on python 3.5 and above.

1.5.2 Google Colab

Google is quite aggressive in AI research. Over many years, Google developed AI framework called TensorFlow and a development tool called Colaboratory. Today TensorFlow is open-sourced and since 2017, Google made Colaboratory free for public use. Colaboratory is now known as Google Colab or simply Colab. Another attractive feature that Google offers to the developers is the use of GPU. Colab supports GPU and it is totally free. The reasons for making it free for public could be to make its software a standard in the academics for teaching machine learning and data science. It may also have a long term perspective of building a customer base for Google Cloud APIs which are sold per-use basis. Irrespective of the reasons, the introduction of Colab has eased the learning and development of machine learning applications

2. Literature Survey

“A genetic programming-based feature selection and fusion for facial expression recognition” is developed by Haythem Ghazouani *et.al*. It was published in 2021. They have used Genetic programming on images. It has drawbacks like Finding Optimal Classifier is difficult, consumes more time. [1].

“GA-SVM based Facial Emotion Recognition using Facial Geometric Features” is developed by Xiao Liu, Xiangyi Cheng, Kiju Lee *et.al*. It was published in 2020. They have used Genetic algorithm , SVM on images. It has drawbacks like SVM is not suitable for large datasets and GA is an expensive approach. [2].

“Person-Independent Facial Expression Recognition Based on Improved Local Binary Pattern and Higher-Order Singular Value Decomposition” is developed by Ying He, Shuxin Chen *et.al*. It was published in 2020. They have used LBP,
HOSVD (Higher-Order Singular Value Decomposition) on images. It has drawbacks like produce rather long histograms, which slow down the recognition speed especially on large-scale face database [3].

“Deep-Emotion: Facial Expression Recognition Using Attentional Convolutional Network” is developed by Shervin Minaee,Amirali Abdolrashidi *et.al*. It was published in 2019. They have used CNN on images. It has drawbacks like Overfitting , class imbalance are the major challenges while training the model using CNN [4].

“Facial expression recognition using distance and texture signature relevant features” is developed by Asit Barman, Paramartha Dutta *et.al*. It was published in 2019. They have used CNN, MLP (Multi-layer perceptron) on images. It has drawbacks like time-consuming [5].

“Facial Emotion Classification Using Concatenated Geometric And Textural Features” is developed by Debashis Sen, Samyak Datta, R.Balasubramanian *et.al*. It was published in 2018. They have used SVM on images. It has drawbacks like Overlapping [6].

“Facial Expression Recognition Using Geometric Normalization and Appearance Representation” is developed by Hamid Sadeghi, Abolghasem-A, Mohammad-Reza Mohammadi *et.al*. It was published in 2013. They have used LBP on images. It has drawbacks like binary data produced by LBP are sensitive to noise. [7].

“Histograms of Oriented Gradients for Human Detection” is developed by N. Dalal,
B. Triggs *et.al*. It was published in 2005. They have used HOG on images. It has drawbacks like increase of computational complexity [8].

Table 2.1 shows the literature survey of Evolutionary Algorithm Based Feature Selection and Fusion for Facial Expression Recognition. It contains name of year, author, title of proposed work, method, drawbacks.

Table 2.1: Literature survey of Evolutionary Algorithm Based Feature Selection and Fusion for Facial Expression Recognition

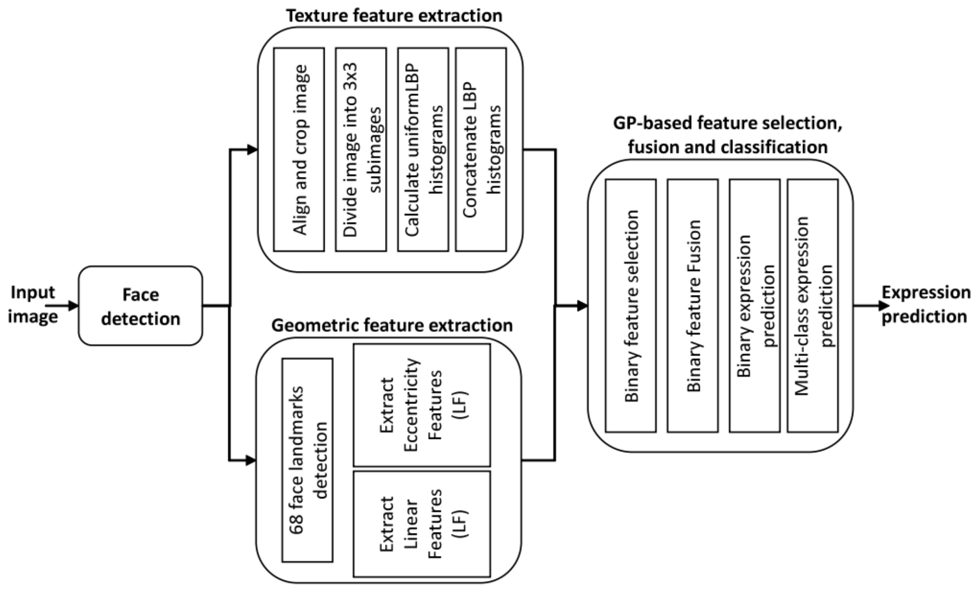
S.No	**Year**	**Author**	**Title**	**Method**	**Drawbacks**
1.	2021	Haythem Ghazouani	A genetic programming-based feature selection and fusion for facial expression recognition	Genetic Programming	Finding Optimal Classifier is difficult. It consumes more time.
2.	2020	Xiao Liu, Xiangyi Cheng, Kiju Lee	GA-SVM based Facial Emotion Recognition using Facial Geometric Features	Genetic algorithm , SVM	SVM is not suitable for large datasets and doesn’t perform well when target classes are overlapping. GA is an expensive approach.
3.	2020	Ying He, Shuxin Chen	Person-Independent Facial Expression Recognition Based on Improved Local Binary Pattern and Higher-Order Singular Value Decomposition	LBP, HOSVD (Higher-Order Singular Value Decomposition).	LBP produce rather long histograms, which slow down the recognition speed especially on large-scale face database and miss the local structure as they don't consider the effect of the center pixel.
4.	2019	Shervin Minaee, Amirali Abdolrashidi	Deep-Emotion: Facial Expression Recognition Using Attentional Convolutional Network	CNN	Overfitting, exploding gradient, and class imbalance are the major challenges while training the model using CNN.

5.	2019	Asit Barman, Paramartha Dutta	Facial expression recognition using distance and texture signature relevant features	CNN, MLP	computations are difficult and time-consuming
6.	2018	Debashis Sen, Samyak Datta, R.Balasubramanian	Facial Emotion Classification Using Concatenated Geometric And Textural Features	Support Vector Machine (SVM)	SVM doesn't perform well when target classes are overlapping
7.	2013	Hamid Sadeghi, Abolghasem-A, Mohammad-Reza Mohammadi	Facial Expression Recognition Using Geometric Normalization and Appearance Representation	Local Binary Patterns(LBP)	The binary data produced by LBP are sensitive to noise.
8.	2005	N. Dalal, B. Triggs	Histograms Of Oriented Gradients For Human Detection	HOG	The size of the features increases exponentially with the number of neighbours which leads to an increase of computational complexity in terms of time and space

3. Methodology

3.1 System Architecture

In proposed system, image is taken as input. Firstly, it detects face in the image. After that Texture features and Geometric features are extracted. 68 Facial landmarks are detected in Geometric features traction and there are 2 types of sub features in Geometric features they are linear features and Eccentric features both are computed. In Texture feature extraction, Texture features are extracted using LBP i.e. Local Binary Patterns. LBP uses histograms. After extraction of geometric and texture features some subset of essential features are selected using genetic programming in binary form and then these features and fused and expression is predicted at last.



Texture feature extraction
Align and crop image
Divide image into 3x3 subimages
Calculate uniformLBP histograms
Concatenate LBP histograms
Input image
Face detection
Geometric feature extraction
68 face landmarks detection
Extract Eccentricity Features (LF)
Extract Linear Features (LF)
GP-based feature selection, fusion and classification
Binary feature selection
Binary feature Fusion
Binary expression prediction
Multi-class expression prediction
Expression prediction

Figure 3.1.1 System Architecture of proposed method

Steps in proposed method

Step-1: Face detection is done in input image using HOG.

Step-2: 68 facial landmarks are detected using shape predictor dat file.

Step-3: Geometric features are extracted using LBP (Local Binary Patterns).Geometric features contains Linear and Eccentric Features.

Step-4: Linear features are extracted using Euclidean distance between all pairs of landmarks.

$$d(X, Y) = sqrt((x1 - x2)^2 + (y1 - y2)^2)$$

Step-4: Eccentric features are extracted by calculating eccentricity between landmarks.

Step-5: Texture features are extracted using LBP (Local Binary Patterns). The LBP histogram contains information about the distribution of the local micro-patterns, so it can be used to statistically describe facial characteristics.

Step-6: Evolutionary algorithm based feature selection and fusion

Step-7: Finally facial expression present in image is predicted.

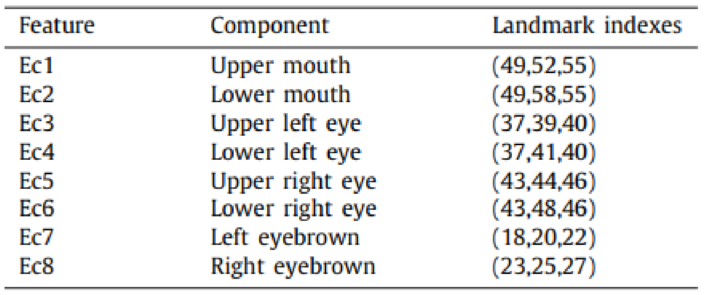
Geometric feature extraction:

For the linear features (LF), Euclidean distance between all pairs of landmarks, are commonly used to capture the facial activities during the emotion deliberation where

(x1, y1) - coordinates of the first landmark.

(x2, y2) - coordinates of the second landmark.

$d(X, Y) = sqrt((x1 - x2) 2 + (y1 - y2))$



Feature	Component	Landmark indexes
Ec1	Upper mouth	(49,52,55)
Ec2	Lower mouth	(49,58,55)
Ec3	Upper left eye	(37,39,40)
Ec4	Lower left eye	(37,41,40)
Ec5	Upper right eye	(43,44,46)
Ec6	Lower right eye	(43,48,46)
Ec7	Left eyebrown	(18,20,22)
Ec8	Right eyebrown	(23,25,27)

Figure 3.1.2 Ellipses on face to find eccentricity features

Since 68 keypoints are extracted in this work, 2278 (68(68 − 1)/2) different linear features can be calculated. The Euclidean distance between the right corner of the left eye (L43) and the left corner of the right eye (L40) was chosen to calculate Ncoef due to its stability during the different facial deformations.

Nd(Xi, Yi) = d(Xi, Yi) / Ncoef

Ncoef = d(L43, L40)

The eccentricity is a geometric term defining the ovalness level of an ellipse. In a more formal way, the eccentricity (e) is the ratio of the ellipse foci (c) to its semi-major axe (a). In fact, if the eccentricity is close to zero then the ellipse has a circular form. However, if it is close to one, then the ovalness of ellipse is high.

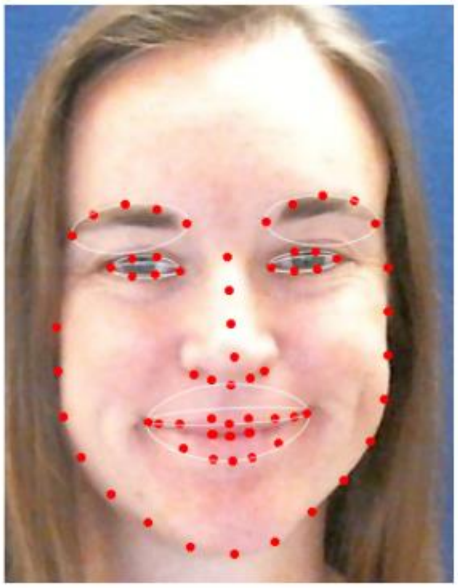


Figure 3.1.3 Representation of the eight facial ellipses

Taking into consideration the elliptical shape of these facial components, the eccentricity features may provide important information about their geometrical shape modification throughout the emotional state. For instance, during the display of surprise, the mouth is generally wide open presenting more of a circular form. However, the mouth is more long and skinny when the person is smiling displaying an ellipse-like shape. The same difference can be denoted for the eyes during the surprise and the disgust or sadness reaction.

Texture feature extraction:

For computational and simplicity reasons, the local binary pattern (LBP) histograms are selected to be used as textural features. The LBP operator, is one of the most widely used descriptors for feature detection and extraction. It locates keypoints within an image and generates a histogram that corresponds to their distribution. The LBP operator scans the pixels using a sliding window and generates a binary code based on the differences between the central pixel and its equidistant circular neighbors. The distance is defined by the radius parameter r and the number of neighbors is denoted by the pixel parameter p. The representation of the LBP operator is defined as follows:

$$LBP_{p,r} = \sum_{i=0}^{p-1} \mathbb{1}_{R^+}(I(x_i,\ y_i) - I(x_c,\ y_c)).2^i,$$

where, 1S denotes the characteristic function of a subset S, xc and yc are the coordinates of the central pixel, xi and yi(4) are the coordinates of its ith neighbor within the input image I.

yi = yc − r.sin(2πi/p) xi = xc + r.cos(2πi/p)

The LBP histogram contains information about the distribution of the local micro-patterns, so it can be used to statistically describe facial characteristics. To represent texture in different face locations, input face images have been aligned and cropped to a uniform spatial resolution of 120 × 120 pixels and then divided into 9 equal-sized sub-images of 40 × 40 pixels each.

Learning binary programs by genetic programming:

The goal is to evolve a program that can predict the most likely emotion between A and B for a given input face image. First, the GP process randomly generates a population of tree-based programs (i.e. binary classifiers). Then, this population is evolved using classical genetic operators (crossover, mutation. The generated classifiers are evaluated with a fitness function based on the training data involving instances from both classes A and B. Finally, the genetically elected classifier is used on unseen faces to predict the most likely facial expression between A and B. Each tree is made up of a root node (upper layer), a number of internal nodes (mid-layer), and leaf nodes (lower layer).

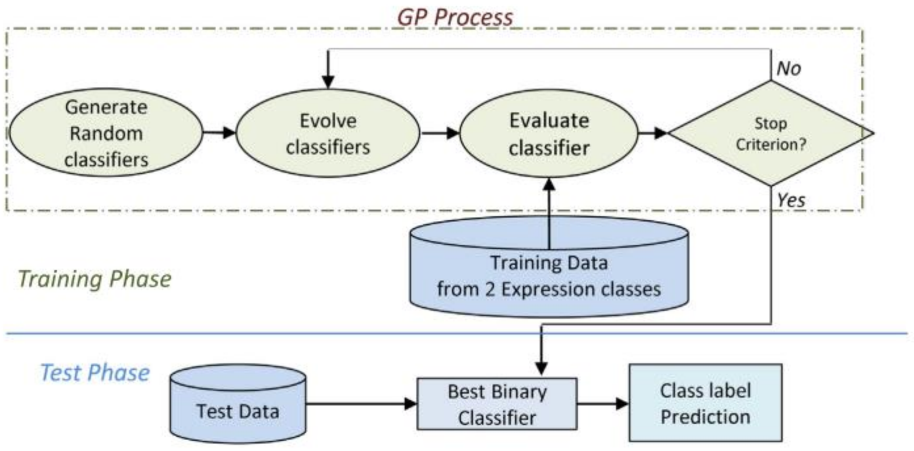


Figure 3.1.4 Overview of the proposed process for evolving a binary program

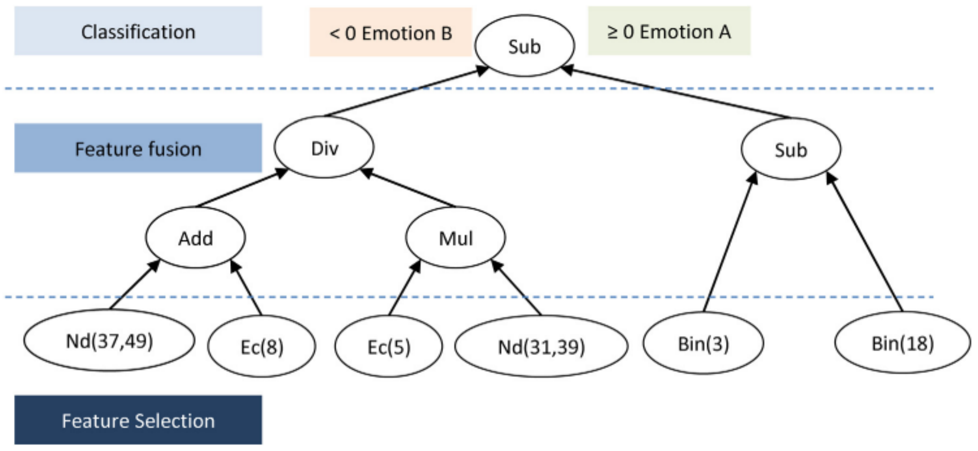


Figure 3.1.5 Simplified tree representation of a binary program

3.2 Histogram of Oriented Gradients

A feature descriptor is a representation of an image or an image patch that simplifies the images by extracting useful information and throwing away extraneous information. Histogram of Oriented Gradients (HOG) is a feature descriptor used to detect objects in computer vision and image processing. The HOG descriptor technique counts occurrences of gradient orientation of gradient orientation in localized portions of an image - detection window or region of interest (ROI).The HOG descriptor focuses on the structure or the shape of an object. HOG is able to provide the edge direction as well. This is done by extracting the gradient and orientation of the edges. Additionally, these orientations are calculated in 'localized' portions. This means that the complete image is broken down into smaller regions and for each region, the gradients and orientation are calculated. Finally the HOG would generate a Histogram for each of these regions separately. The histograms are created using the gradients and orientations of the pixel values, hence the name 'Histogram of Oriented Gradients'

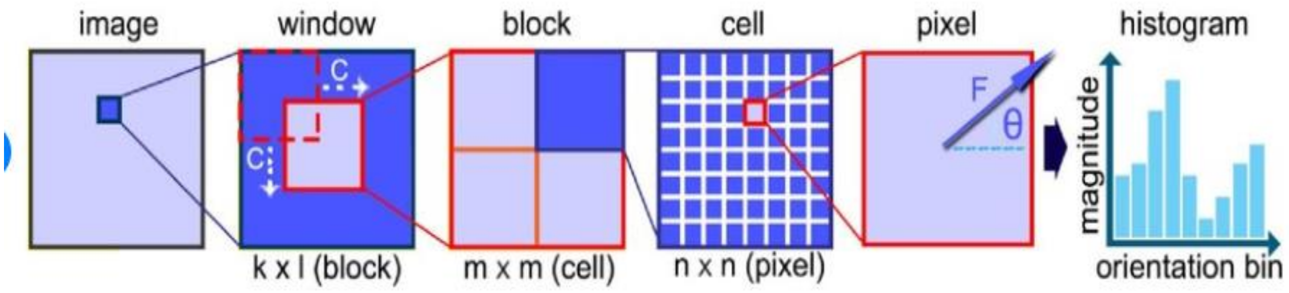


Figure 3.2.1 Computation of Histogram of Oriented Gradients

Steps to calculate HOG:

1) Preprocessing

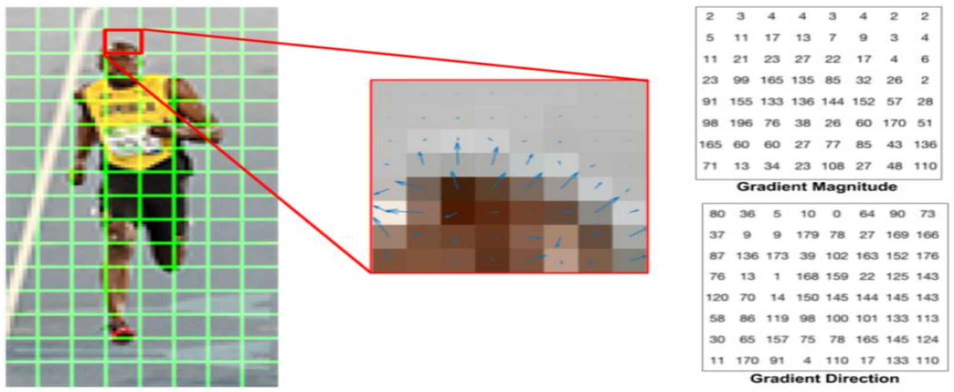
   Cropping and resizing the image into 64 x 128 size.

2) Calculate Gradient Images

   To calculate a HOG descriptor, we need to first calculate the horizontal and vertical gradients; after all, we want to calculate the histogram of gradients.

3) Calculate Histogram of Gradients in 8 x 8 cells.

The image is divided into 8×8 cells and a histogram of gradients is calculated for each 8×8 cells.



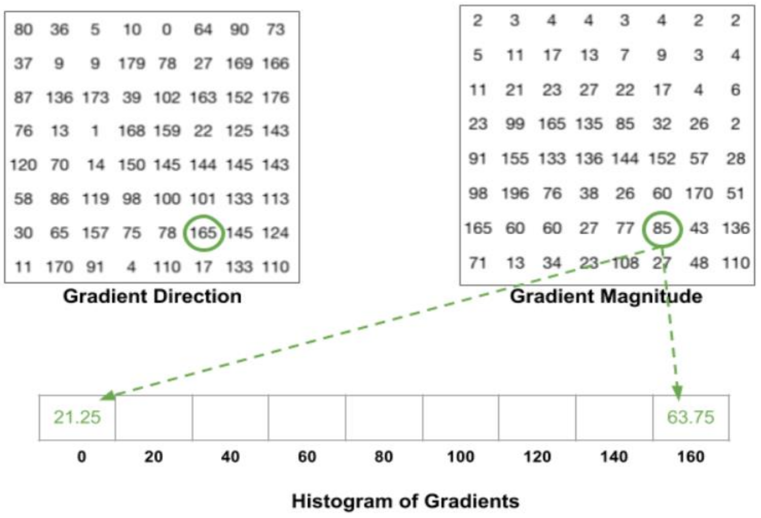
2	3	4	4	3	4	2	2
5	11	17	13	7	9	3	4
11	21	23	27	22	17	4	6
23	99	165	135	85	32	26	2
91	155	133	136	144	152	57	28
98	196	76	38	26	60	170	51
165	60	60	27	77	85	43	136
71	13	34	23	108	27	48	110

Gradient Magnitude

80	36	5	10	0	64	90	73
37	9	9	179	78	27	169	166
87	136	173	39	102	163	152	176
76	13	1	168	159	22	125	143
120	70	14	150	145	144	145	143
58	86	119	98	100	101	133	113
30	65	157	75	78	165	145	124
11	170	91	4	110	17	133	110

Gradient Direction

Figure 3.2.2 Image with the respective gradient magnitudes and directions



80	36	5	10	0	64	90	73
37	9	9	179	78	27	169	166
87	136	173	39	102	163	152	176
76	13	1	168	159	22	125	143
120	70	14	150	145	144	145	143
58	86	119	98	100	101	133	113
30	65	157	75	78	165	145	124
11	170	91	4	110	17	133	110

Gradient Direction

2	3	4	4	3	4	2	2
5	11	17	13	7	9	3	4
11	21	23	27	22	17	4	6
23	99	165	135	85	32	26	2
91	155	133	136	144	152	57	28
98	196	76	38	26	60	170	51
165	60	60	27	77	85	43	136
71	13	34	23	108	27	48	110

Gradient Magnitude

0	20	40	60	80	100	120	140	160
21.25								63.75

Histogram of Gradients

Figure 3.2.3 Placing the pixel in the bins depending on their direction and magnitude

4) Block Normalization.

A 16×16 block has 4 histograms which can be concatenated to form a 36 x 1 element vector and it can be normalized just the way a 3×1 vector is normalized. The window is then moved by 8 pixels and a normalized 36×1 vector is calculated over this window and the process is repeated.

5) Form HOG feature vector.

To calculate the final feature vector for the entire image patch, the 36×1 vectors are concatenated into one giant vector.

3.3 Support Vector Machine

A Support Vector Machine (SVM) is a discriminative classifier which intakes training data, the algorithm outputs an optimal hyperplane which categorizes new examples. It is a supervised machine learning algorithm which can be used for both classification and regression. Goal of the SVM algorithm is to create the best line or decision boundary that can segregate n-dimensional space into classes so that we can easily put the new data point in the correct category in the future. This best decision boundary is called a hyperplane. SVM chooses the extreme points/vectors that help in creating the hyperplane. These extreme cases are called as support vectors, and hence algorithm is termed as Support Vector Machine.

SVM can be of two types:

- **Linear SVM:** Linear SVM is used for linearly separable data, which means if a dataset can be classified into two classes by using a single straight line, then such data is termed as linearly separable data, and classifier is used called as Linear SVM classifier.
- **Non-linear SVM:** Non-Linear SVM is used for non-linearly separated data, which means if a dataset cannot be classified by using a straight line, then such data is termed as non-linear data and classifier used is called as Non-linear SVM classifier.

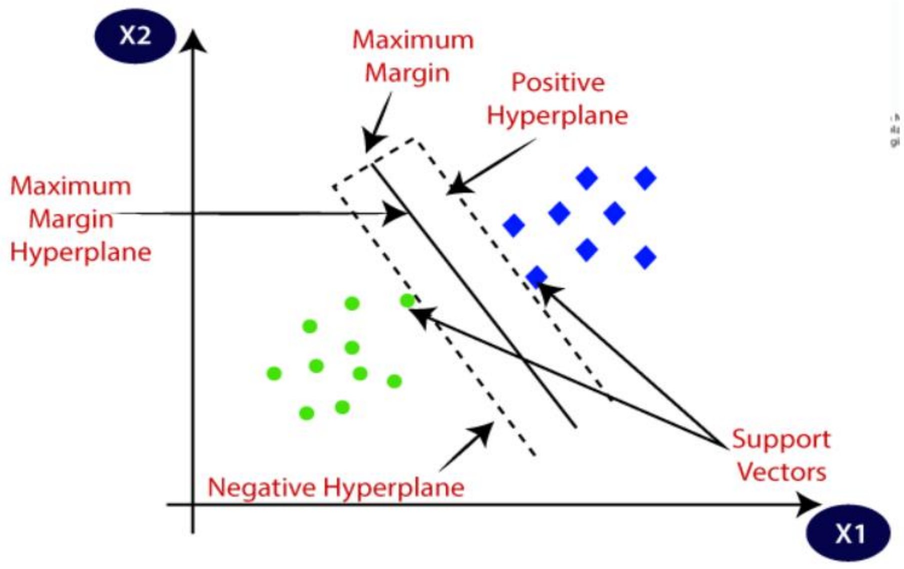


Figure 3.3.1 Different hyperplanes using support vectors

3.4 Local Binary Patterns

Local Binary Pattern (LBP) is a texture operator which labels the pixels of an image by thresholding the neighbourhood of each pixel and considers the result as a binary number. LBPs compute a local representation of texture. This local representation is constructed by comparing each pixel with its surrounding neighbourhood 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 neighbourhood 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. We need to calculate the LBP value for the center pixel. The results of this binary test are stored in an 8-bit array, which we then convert to decimal. This process of thresholding, accumulating binary strings, and storing the output decimal value in the LBP array is then repeated for each pixel in the input image. The last step is to compute a histogram over the output LBP array. Since a 3 x 3 neighbourhood has 2 ^ 8 = 256 possible patterns, our LBP 2D array thus has a minimum value of 0 and a maximum value of 255, allowing us to construct a 256-bin histogram of LBP codes as our final feature vector.

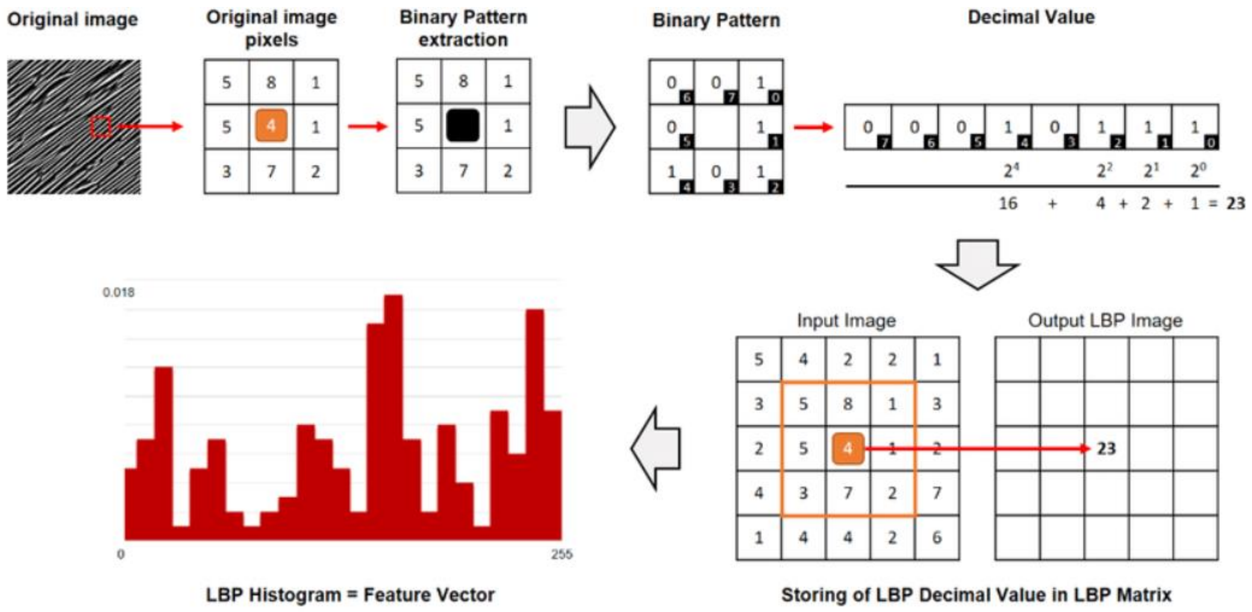


Figure 3.4.1 Calculation process in LBP

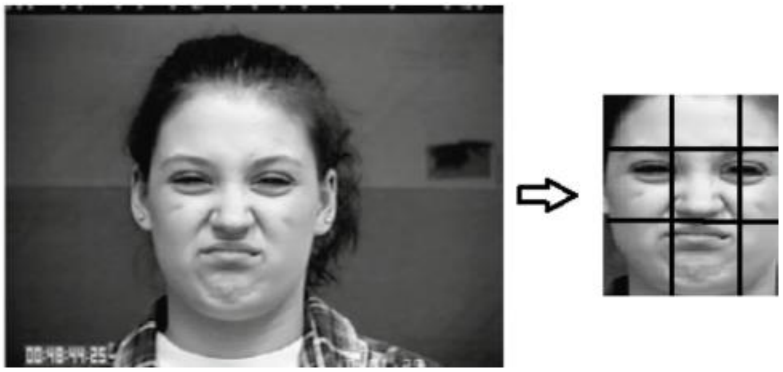


Figure 3.4.2 Original face image and cropped image divided into 9 sub-images

3.5 Genetic Programming

Genetic programming is a technique to create algorithms that can program themselves by simulating biological breeding and Darwinian evolution. Instead of programming a model that can solve a particular problem, genetic programming only provides a general objective and lets the model figure out the details itself. The basic approach is to let the machine automatically test various simple evolutionary algorithms and then “breed” the most successful programs in new generations. While applying the same natural selection, crossover, mutations and other reproduction approaches as evolutionary and genetic algorithms, gene programming takes the process a step further by automatically creating new models and letting the system select its own goals. The entire process is still an area of active research.

One of the biggest obstacles to widespread adoption of this genetic machine learning approach is quantifying the fitness function, i.e to what degree each new program is contributing to reaching the desired goal. Genetic programming is a specialization of genetic algorithms where each individual is a computer program.

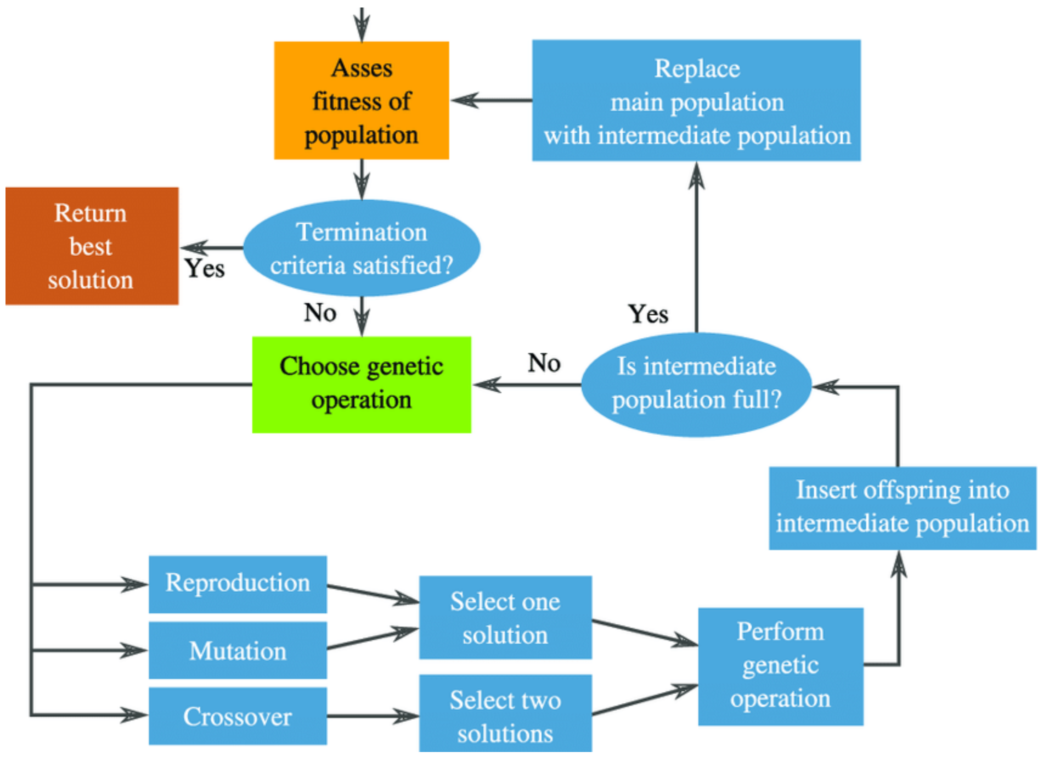

Asses fitness of population
Replace main population with intermediate population
Return best solution
Yes
Termination criteria satisfied?
No
Yes
Choose genetic operation
No
Is intermediate population full?
Insert offspring into intermediate population
Reproduction
Mutation
Crossover
Select one solution
Select two solutions
Perform genetic operation

Figure 3.5.1 Flowchart of Genetic programming

The main difference between genetic programming and genetic algorithms is the representation of the solution. The output of the genetic algorithm is a quantity, while the output of the genetic programming is another computer program. It starts with an initial set of programs composed of functions and terminals that may be handpicked or randomly generated. The functions may be standard arithmetic operations, programming operations, mathematical functions, or logical functions.

The programs compete with each other over given input and expected output data. Each computer program in the population is measured in terms of how well it performs in a particular problem environment. This measure is called a fitness measure. Top-performing programs are picked, mutation and breeding are performed on them to generate the next generation. Next-generation competes with each other, the process goes on until the perfect program is evolved.

Programs in genetic programming are expressed as syntax trees rather than as lines of code. Trees can be easily evaluated recursively. The tree includes nodes (functions) and links (terminals).

Steps for GP:

- Determining the set of terminals
- Determining the set of functions
- Determining the fitness measure
- Determining the parameters for the run
- Determining the method for designating a result and the criterion for terminating a run.

Crossover

For each pair of parents to be mated, a crossover point is chosen at random from within the genes. Offspring are created by exchanging the genes of parents among themselves until the crossover point is reached. The new offspring are added to the population.

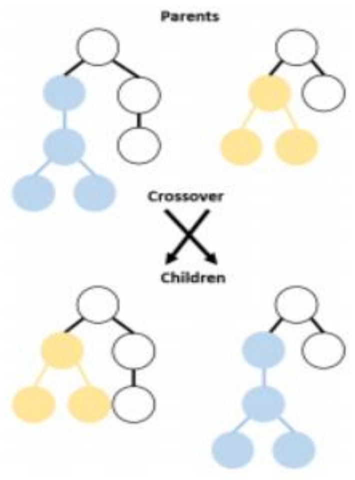


Figure 3.5.2 Crossover Operator for Genetic Programming

Mutation

In certain new offspring formed, some of their genes can be subjected to a mutation with a low random probability. This implies that some of the bits in the bit string can be flipped.

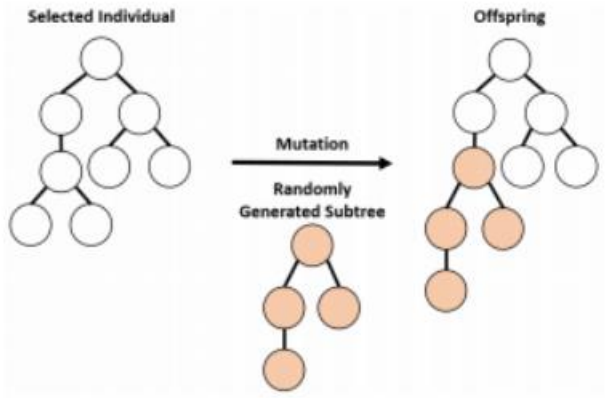


Figure 3.5.3 Mutation Operator for Genetic Programming

3.6 UML Diagrams

UML is an acronym that stands for Unified Modeling Language. Simply put, UML is a modern approach to modeling and documenting software. In fact, it's one of the most popular business process modeling techniques. It is based on diagrammatic representations of software components. As the old proverb says: "a picture is worth a thousand words". By using visual representations, we are able to better understand possible flaws or errors in software or business processes**.**

3.6.1 Use Case Diagram

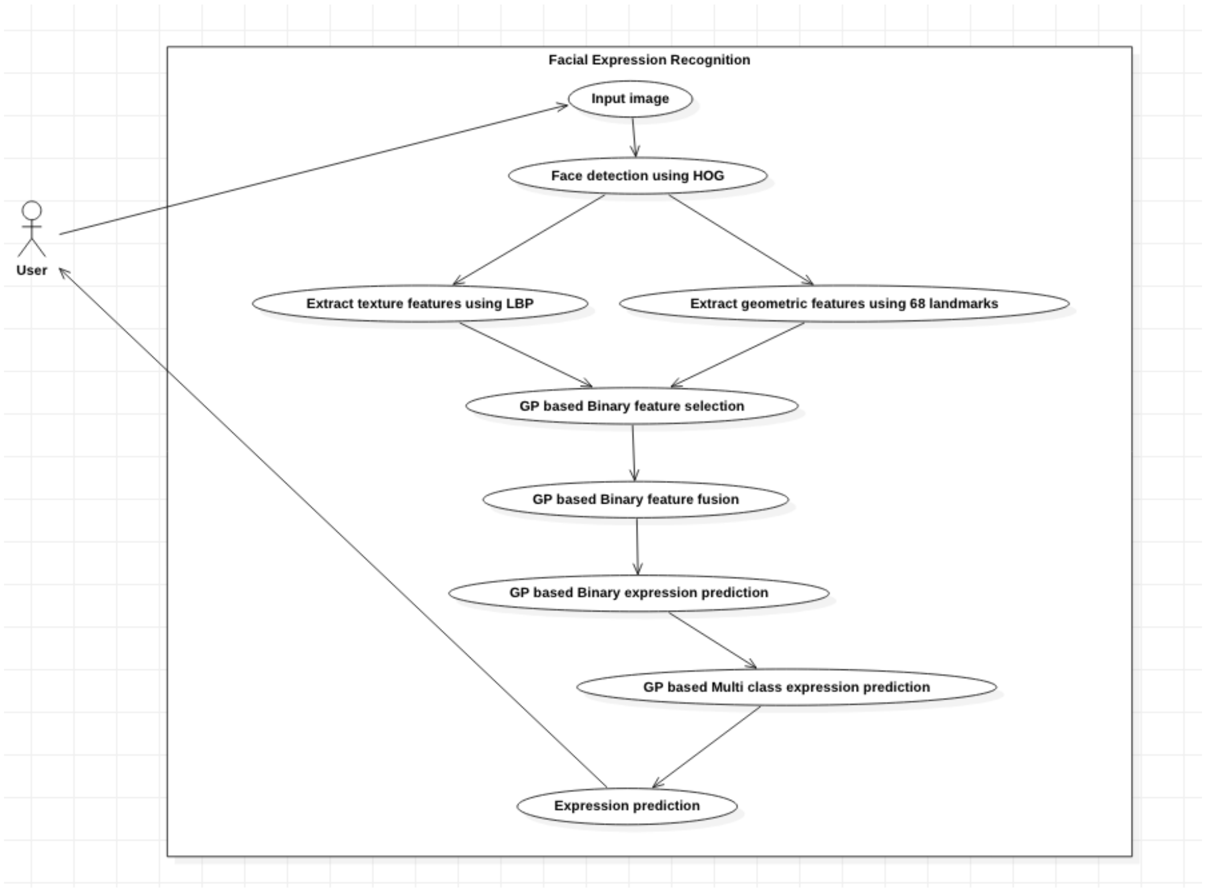

Facial Expression Recognition
Input image
Face detection using HOG
User
Extract texture features using LBP
Extract geometric features using 68 landmarks
GP based Binary feature selection
GP based Binary feature fusion
GP based Binary expression prediction
GP based Multi class expression prediction
Expression prediction

Figure 3.6.1 Use case Diagram for proposed method

User provides input images and face is detected using HOG in image and linear and geometric features are extracted and Evolutionary algorithm based feature selection, fusion are done and expression is predicted.

3.6.2 Sequence Diagram:

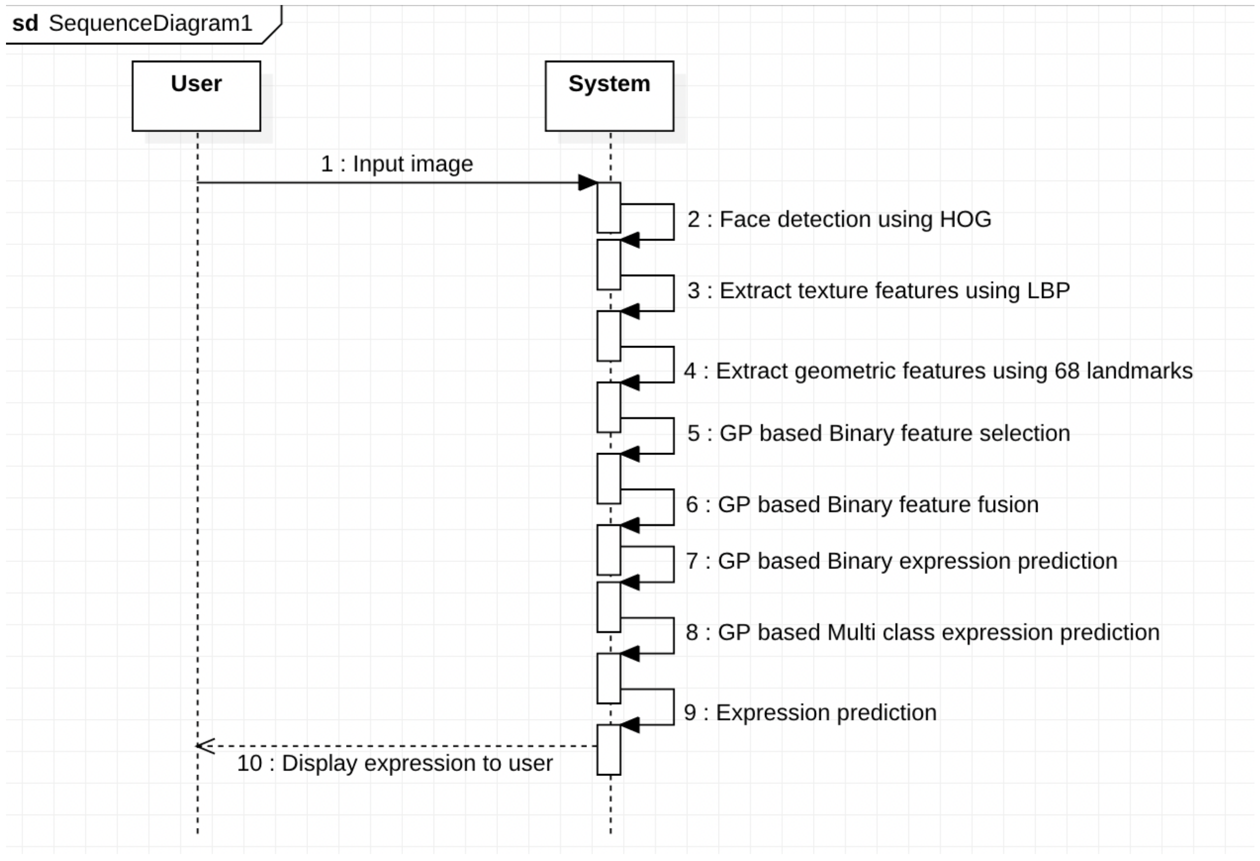

sd SequenceDiagram1
User
System
1 : Input image
2 : Face detection using HOG
3 : Extract texture features using LBP
4 : Extract geometric features using 68 landmarks
5 : GP based Binary feature selection
6 : GP based Binary feature fusion
7 : GP based Binary expression prediction
8 : GP based Multi class expression prediction
9 : Expression prediction
10 : Display expression to user

Figure 3.6.2 Sequence Diagram for proposed method

In this system user provides the input image to the system. Then system detects face in image using HOG and linear and geometric features are extracted and Evolutionary algorithm based feature selection, fusion are done and expression is predicted. Expression is displayed to user

4. Implementation and Results

Google is quite aggressive in AI research. Over many years, Google developed AI framework called TensorFlow and a development tool called Colaboratory. Today TensorFlow is open-sourced and since 2017, Google made Colaboratory free for public use. Colaboratory is now known as Google Colab or simply Colab.

4.1 How to run project

1) Go to https://colab.research.google.com/?utm_source=scs-indexdownload.
2) Select the Edition based on your OS.
3) Open Google colab after downloading it.
4) Create a folder and name it accordingly.
5) Upload the input image in google drive.
6) Mount the google drive to the colab.
7) Import the required packages.
8) Run the modules with the input image

Code

The source code with the dataset developed for "Evolutionary Algorithm Based Feature Selection And Fusion For Facial Expression Recognition" is available at

https://colab.research.google.com/drive/1JKIz4VssYxQcXLolFYiZ1KHONhOiuRlo#scrollTo=Zjdqq_DLZHA9

4.2 Results

4.2.1 Output for First Image:

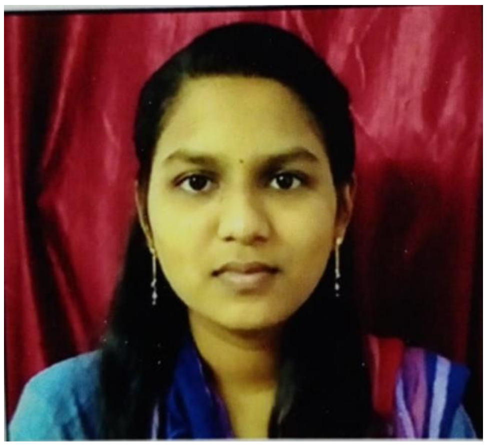


Figure 4.1.1 Input Image

Image is taken as input. Initially image is uploaded into Google drive and then the drive is mounted.

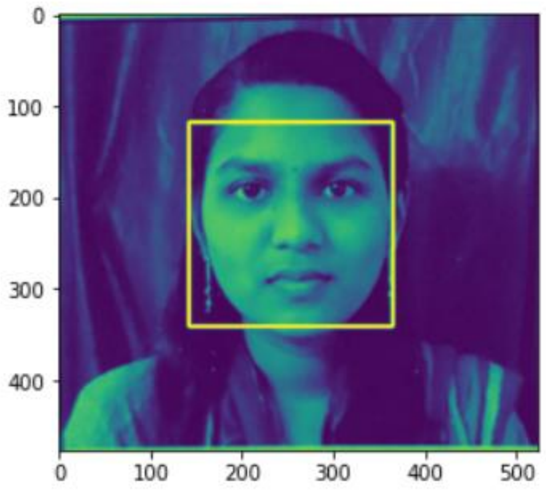


Figure 4.1.2 Face detection with Bounding Boxes

Converting input image into gray scale image and performing face detection thereby drawing Bounding Boxes around the image.

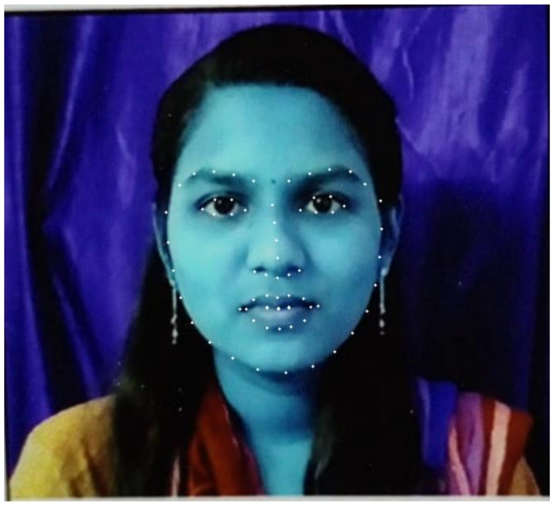
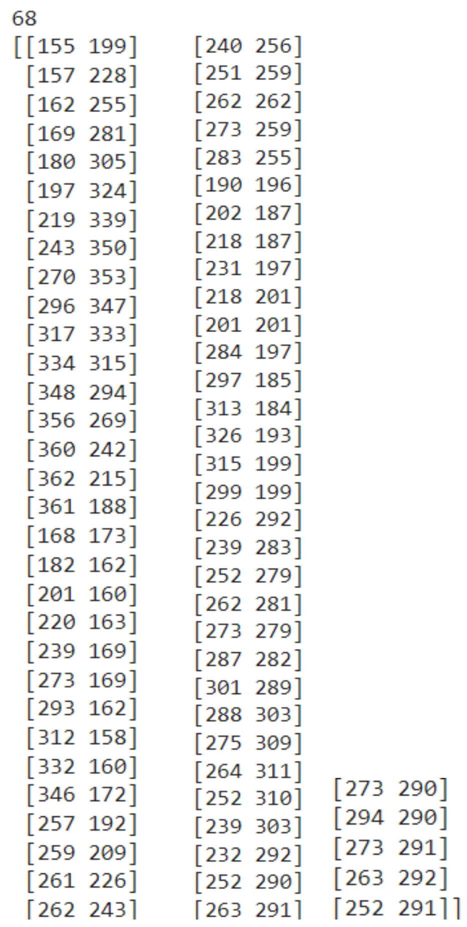


Figure 4.1.3 Plotting 68 facial landmarks using shape_predictor.dat file

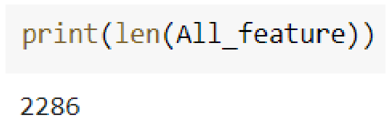
Here shape_predictor.dat file is used for extracting the 68 landmarks in the input image.



```
68
[[155 199]     [240 256]
 [157 228]     [251 259]
 [162 255]     [262 262]
 [169 281]     [273 259]
 [180 305]     [283 255]
 [197 324]     [190 196]
 [219 339]     [202 187]
 [243 350]     [218 187]
 [270 353]     [231 197]
 [296 347]     [218 201]
 [317 333]     [201 201]
 [334 315]     [284 197]
 [348 294]     [297 185]
 [356 269]     [313 184]
 [360 242]     [326 193]
 [362 215]     [315 199]
 [361 188]     [299 199]
 [168 173]     [226 292]
 [182 162]     [239 283]
 [201 160]     [252 279]
 [220 163]     [262 281]
 [239 169]     [273 279]
 [273 169]     [287 282]
 [293 162]     [301 289]
 [312 158]     [288 303]
 [332 160]     [275 309]
 [346 172]     [264 311]   [273 290]
 [257 192]     [252 310]   [294 290]
 [259 209]     [239 303]   [273 291]
 [261 226]     [232 292]   [263 292]
 [262 243]     [252 290]   [252 291]]
               [263 291]
```

Figure 4.1.4 Extraction of 68 landmark coordinates from Bounding Boxes

Storing the 68 landmarks in a numpy file named as shape_np and displaying them as (x,y) coordinates.

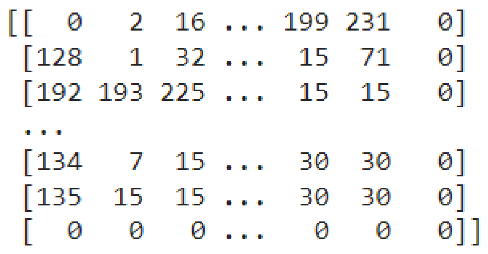


```
print(len(All_feature))

2286
```

Figure 4.1.5 All_features list containing set of Geometric features

There are 2278 linear features and 8 eccentric features together they will form 2286 geometric features



```
[[  0   2  16 ... 199 231   0]
 [128   1  32 ...  15  71   0]
 [192 193 225 ...  15  15   0]
 ...
 [134   7  15 ...  30  30   0]
 [135  15  15 ...  30  30   0]
 [  0   0   0 ...   0   0   0]]
```

Figure 4.1.6 Numpy array consisting set of Texture feature vectors

Extracting the Texture features from the input image using Local Binary Patterns (LBP). Texture based features contain information about the furrows, bulges, wrinkles, etc. Appearance features contain micro-patterns which provide important information about the facial expressions.

4.2.2 Output for second Image:

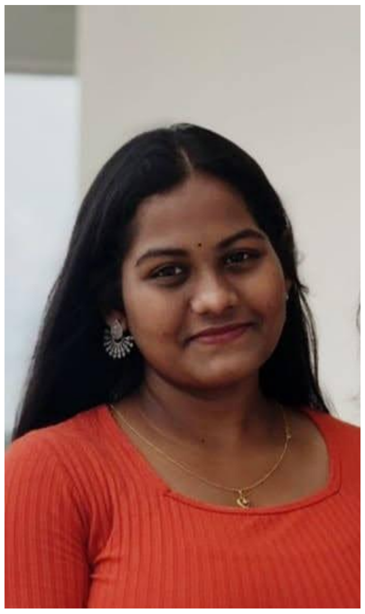


Figure 4.2.1 Input Image

Image is taken as input. Initially image is uploaded into Google drive and then the drive is mounted.

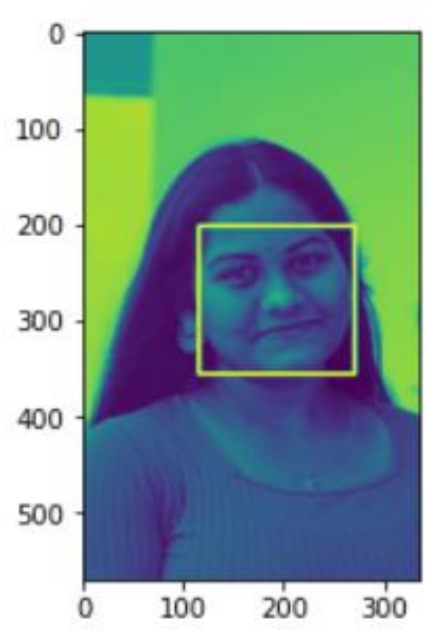


Figure 4.2.2 Face detection with Bounding Boxes

Converting input image into gray scale image and performing face detection thereby drawing Bounding Boxes around the image.

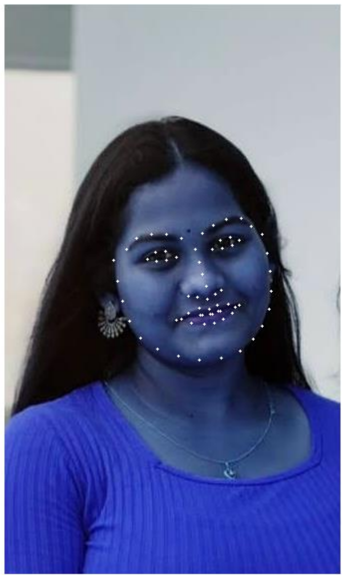
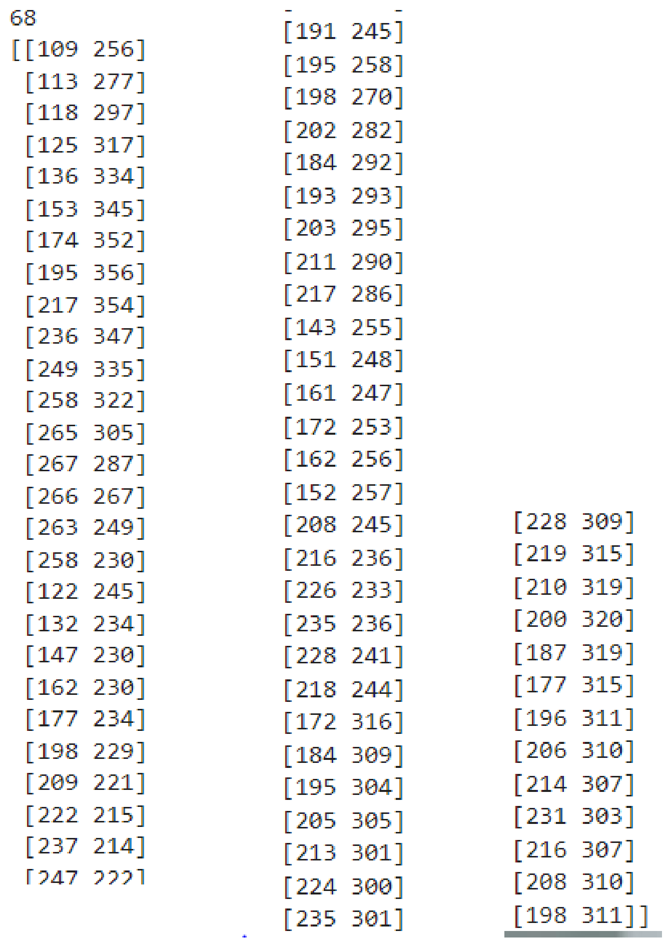


Figure 4.2.3 Plotting 68 facial landmarks using shape_predictor.dat file

Here shape_predictor.dat file is used for extracting the 68 landmarks in the input image.



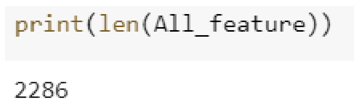
```
68
[[109 256]
 [113 277]
 [118 297]
 [125 317]
 [136 334]
 [153 345]
 [174 352]
 [195 356]
 [217 354]
 [236 347]
 [249 335]
 [258 322]
 [265 305]
 [267 287]
 [266 267]
 [263 249]
 [258 230]
 [122 245]
 [132 234]
 [147 230]
 [162 230]
 [177 234]
 [198 229]
 [209 221]
 [222 215]
 [237 214]
 [247 222]
```

```
[191 245]
[195 258]
[198 270]
[202 282]
[184 292]
[193 293]
[203 295]
[211 290]
[217 286]
[143 255]
[151 248]
[161 247]
[172 253]
[162 256]
[152 257]
[208 245]
[216 236]
[226 233]
[235 236]
[228 241]
[218 244]
[172 316]
[184 309]
[195 304]
[205 305]
[213 301]
[224 300]
[235 301]
```

```
[228 309]
[219 315]
[210 319]
[200 320]
[187 319]
[177 315]
[196 311]
[206 310]
[214 307]
[231 303]
[216 307]
[208 310]
[198 311]]
```

Figure 4.2.4 Extraction of 68 landmark coordinates from Bounding Boxes

Storing the 68 landmarks in a numpy file named as shape_np and displaying them as (x,y) coordinates.

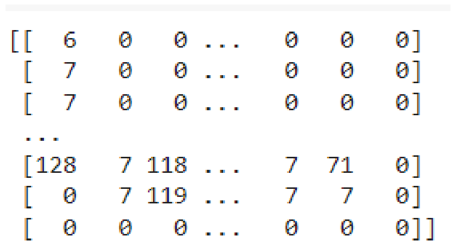


```
print(len(All_feature))

2286
```

Figure 4.2.5 All_features list containing set of Geometric features

There are 2278 linear features and 8 eccentric features together they will form 2286 geometric features



```
[[  6   0   0 ...   0   0   0]
 [  7   0   0 ...   0   0   0]
 [  7   0   0 ...   0   0   0]
 ...
 [128   7 118 ...   7  71   0]
 [  0   7 119 ...   7   7   0]
 [  0   0   0 ...   0   0   0]]
```

Figure 4.2.6 Numpy array consisting set of Texture feature vectors

Extracting the Texture features from the input image using Local Binary Patterns (LBP). Texture based features contain information about the furrows, bulges, wrinkles, etc. Appearance features contain micro-patterns which provide important information about the facial expressions.

4.2.3 Output for Third Image:

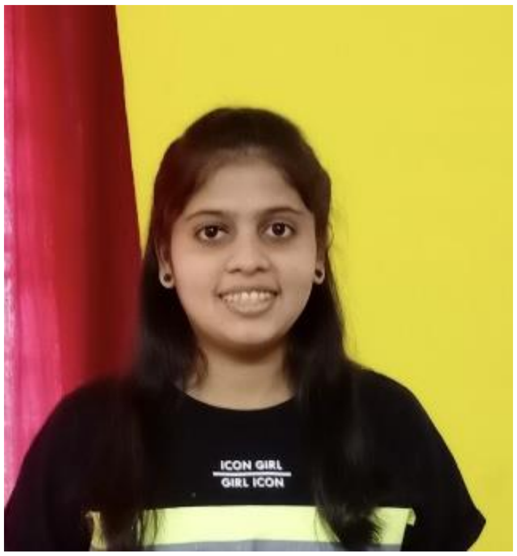


Figure 4.3.1 Input Image

Image is taken as input. Initially image is uploaded into Google drive and then the drive is mounted.

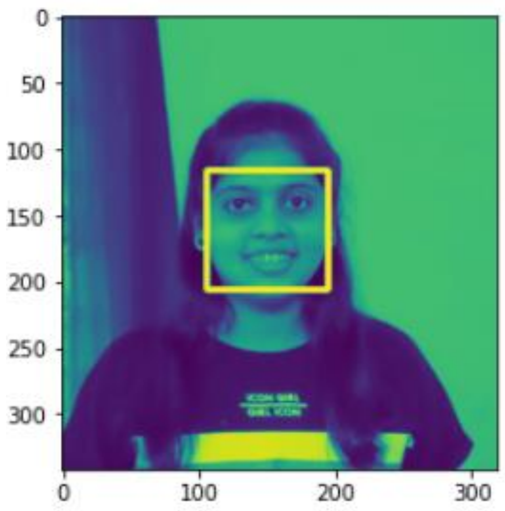


Figure 4.3.2 Face detection with Bounding Boxes

Converting input image into gray scale image and performing face detection thereby drawing Bounding Boxes around the image.

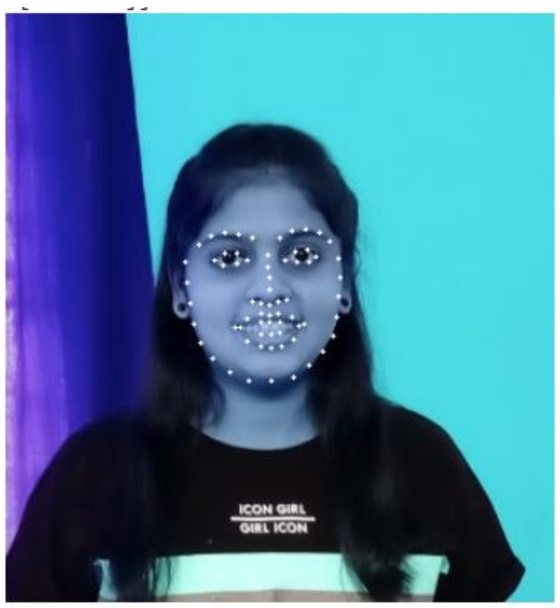
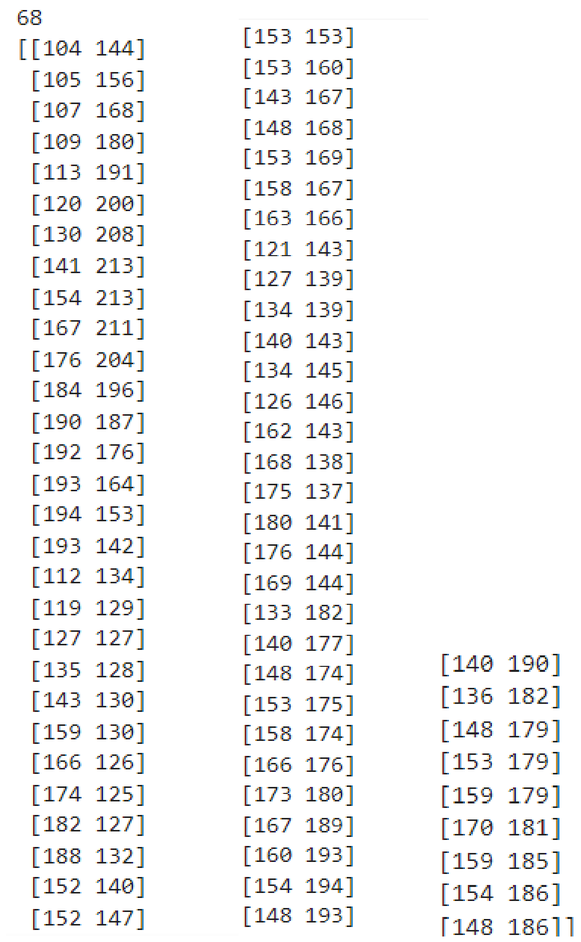


Figure 4.3.3 Plotting 68 facial landmarks using shape_predictor.dat file

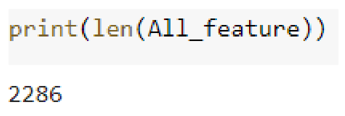
Here shape_predictor.dat file is used for extracting the 68 landmarks in the input image.



```
68
[[104 144]
 [105 156]
 [107 168]
 [109 180]
 [113 191]
 [120 200]
 [130 208]
 [141 213]
 [154 213]
 [167 211]
 [176 204]
 [184 196]
 [190 187]
 [192 176]
 [193 164]
 [194 153]
 [193 142]
 [112 134]
 [119 129]
 [127 127]
 [135 128]
 [143 130]
 [159 130]
 [166 126]
 [174 125]
 [182 127]
 [188 132]
 [152 140]
 [152 147]
 [153 153]
 [153 160]
 [143 167]
 [148 168]
 [153 169]
 [158 167]
 [163 166]
 [121 143]
 [127 139]
 [134 139]
 [140 143]
 [134 145]
 [126 146]
 [162 143]
 [168 138]
 [175 137]
 [180 141]
 [176 144]
 [169 144]
 [133 182]
 [140 177]
 [148 174]
 [153 175]
 [158 174]
 [166 176]
 [173 180]
 [167 189]
 [160 193]
 [154 194]
 [148 193]
 [140 190]
 [136 182]
 [148 179]
 [153 179]
 [159 179]
 [170 181]
 [159 185]
 [154 186]
 [148 186]]
```

Figure 4.3.4 Extraction of 68 landmark coordinates from Bounding Boxes

Storing the 68 landmarks in a numpy file named as shape_np and displaying them as (x,y) coordinates.

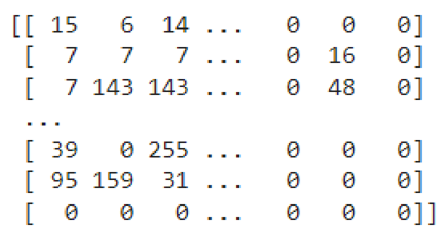


```
print(len(All_feature))

2286
```

Figure 4.3.5 All_features list containing set of Geometric features

There are 2278 linear features and 8 eccentric features together they will form 2286 geometric features



```
[[ 15   6  14 ...   0   0   0]
 [  7   7   7 ...   0  16   0]
 [  7 143 143 ...   0  48   0]
 ...
 [ 39   0 255 ...   0   0   0]
 [ 95 159  31 ...   0   0   0]
 [  0   0   0 ...   0   0   0]]
```

Figure 4.3.6 Numpy array consisting set of Texture feature vectors

Extracting the Texture features from the input image using Local Binary Patterns (LBP). Texture based features contain information about the furrows, bulges, wrinkles, etc. Appearance features contain micro-patterns which provide important information about the facial expressions.

5. Conclusion and Future Scope

5.1 Conclusion

In this work, a robust method for automatic facial expression recognition, is proposed. Genetic programming-based binary programs, which incorporate feature selection and fusion in the learning process, are proposed to discriminate between pairs of expression classes. The overall expression recognition is performed using a unique tournament elimination between the learned binary classifiers. The suggested method selects and combines differently linear, eccentricity and LBP features for each pair of expressions. This allows to choose the most discriminating subset of features separating each pair of classes and to fuse them while avoiding information redundancy, thanks to GP optimization. Unlike deep learning-based methods, which need data augmentation to perform training phase, the suggested method works well with limited training instances.

5.2 Future Scope

The reported performances were among the best results recently found for posed and spontaneous facial expression recognition. Improvements can still be made to the GP-FER method. Indeed, the proposed method was only tested on the combination of three geometric and appearance features. Other features can be tested, and their integration can be easily achieved by simply defining the corresponding terminal functions within the selection layer. In our future work, a subject-dependent facial expression recognition system will be investigated. A dynamic face-based feature selection method will be incorporated in the learning process. The construction of a ground-truth that associates the facial shape of the subject with the appropriate set of features will be studied. The feature selection mechanism will deal with the challenge of subject dependency of the spontaneous facial expressions to enhance the performance of the FER system. The most suitable subset of features will be selected depending on the input face shape and characteristics.

References

[1] Haythem Ghazouani, Evolutionary Algorithm Based Feature Selection And Fusion For Facial Expression Recognition, It is published in 2021.

[2] Xiao Liu, Xiangyi Cheng, Kiju Lee, GA-SVM based Facial Emotion Recognition using Facial Geometric Features, It is published in 2020.

[3] Ying He, Shuxin Chen, Person-Independent Facial Expression Recognition Based on Improved Local Binary Pattern and Higher-Order Singular Value Decomposition, It is published in 2020.

[4] Shervin Minaee, Amirali Abdolrashidi, Deep-Emotion: Facial Expression Recognition Using Attentional Convolutional Network, It is published in 2019.

[5] Asit Barman, Paramartha Dutta, Facial expression recognition using distance and texture signature relevant features, It is published in 2019.

[6] Debashis Sen, Samyak Datta, R.Balasubramanian, Facial Emotion Classification Using Concatenated Geometric And Textural Features, It is published in 2018.

[7] Hamid Sadeghi, Abolghasem-A, Mohammad-Reza Mohammadi, Facial Expression Recognition Using Geometric Normalization and Appearance Representation, It is published in 2013.

[8] N. Dalal, B. Triggs, Histograms Of Oriented Gradients For Human Detection, It is published in 2005.

[9] H. Bejaoui, H. Ghazouani, W. Barhoumi, Fully automated facial expression recognition using 3D morphable model and mesh-local binary pattern, It is published in 2005.

[10] M.F. Valstar, I. Patras, M. Pantic, Facial action unit detection using probabilistic actively learned support vector machines on tracked facial point data, It is published in 2005.

[11] M. He, S. Wang, Z. Liu, X. Chen, Analyses of the differences between posed and spontaneous facial expressions, It is published in 2013.

[12] D.S. Sen, Datta, R. Balasubramanian, Facial emotion classification using concatenated geometric and textural features, It is published in 2019.

[13] S. Namba, S. Makihara, R. Kabir, M. Miyatani, T. Nakao, Spontaneous facial expressions are different from posed facial expressions, It is published in 2016.
[14] J. Chen, D. Chen, Y. Gong, M. Yu, K. Zhang, L. Wang, Facial expression recognition using geometric and appearance features, It is published in 2012.
[15] G. Littlewort, M. Frank, C. Lainscsek, I. Fasel, J. Movellan, Automatic recognition of facial actions in spontaneous expressions, It is published in 2006.

Appendix

Code:

```
from google.colab import drive
drive.mount('/content/drive')
import cv2
import matplotlib.pyplot as plt
import dlib
from imutils import face_utils
import numpy as np
from google.colab.patches import cv2_imshow
frame = cv2.imread('/content/drive/MyDrive/pic c.jpg')
#frame = cv2.imread('/content/random.jpg')

    #print(frame)
frame = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
face_detect=dlib.get_frontal_face_detector()
predictor=dlib.shape_predictor('/content/drive/MyDrive/shape_predictor_68_face_landmarks.dat
')
    #predictor=dlib.shape_predictor('/shape_predictor_68_face_landmarks.dat')
rects=face_detect(gray,1)
for (i,rect) in enumerate(rects):
    (x,y,w,h)=face_utils.rect_to_bb(rect)
    cv2.rectangle(gray, (x, y), (x+w, y+h), (255, 255, 255), 3)
#plt.imshow(frame)
#plt.imshow(rects)
plt.imshow(gray)
plt.show()

shape = predictor(gray, rect)
  # Convert it to the NumPy Array
```

```
shape_np = np.zeros((68, 2), dtype="int")
for i in range(0, 68):
    shape_np[i] = (shape.part(i).x, shape.part(i).y)
shape = shape_np

print(len(shape))
print(shape)

  # Display the landmarks
for i, (x, y) in enumerate(shape):
# Draw the circle to mark the keypoint
    cv2.circle(frame, (x, y), 1, (255, 255, 255), -1)

# Display the image
cv2_imshow(frame)
import math

left_eye =shape[39]
right_eye = shape[42]

norm_dist = math.sqrt(((left_eye[0]-right_eye[0])**2)+((left_eye[1] -right_eye[1])**2))

Dist = []

for i in range(68):
  a,b= shape[i]
  for j in range(i+1,68):
    c,d = shape[j]
    distance =math.sqrt(((a-c)**2)+((b-d)**2))
    Dist.append([i,j,distance/norm_dist])
```

```
print(len(Dist))
print(Dist)
All_feature=[]
for i in Dist:
  All_feature.append(i[2])
print(All_feature)
m1=math.sqrt(((shape[48][0]-shape[54][0])**2)+((shape[48][1]-
shape[54][1]))**2)/2 #semi major axis
x1=(shape[48][0]+shape[54][0])/2
y1=(shape[48][1]+shape[54][1])/2
d1=math.sqrt(((shape[51][0]-x1)**2)+(shape[51][1]-y1)**2) #semi minor axis
print(d1)
print(m1)
c1=math.sqrt(m1**2-d1**2)
ec1=c1/m1
print(ec1)
All_feature.append(ec1)
All_feature[-1]

m2=math.sqrt(((shape[48][0]-shape[54][0])**2)+((shape[48][1]-shape[54][1]))**2)/2
x2=(shape[48][0]+shape[54][0])/2
y2=(shape[48][1]+shape[54][1])/2
d2=math.sqrt(((shape[58][0]-x2)**2)+(shape[58][1]-y2)**2)
print(d2)
print(m2)
c2=math.sqrt(m2**2-d2**2)
ec2=c2/m2
print(ec2)
All_feature.append(ec2)
All_feature[-1]
```

```
m3=math.sqrt(((shape[36][0]-shape[39][0])**2)+((shape[36][1]-shape[39][1]))**2)/2
x3=(shape[36][0]+shape[39][0])/2
y3=(shape[36][1]+shape[39][1])/2
d3=math.sqrt(((shape[38][0]-x3)**2)+(shape[38][1]-y3)**2)
print(d3)
print(m3)
c3=math.sqrt(m3**2-d3**2)
ec3=c3/m3
print(ec3)
All_feature.append(ec3)
All_feature[-1]

m4=math.sqrt(((shape[36][0]-shape[39][0])**2)+((shape[36][1]-shape[39][1]))**2)/2
x4=(shape[36][0]+shape[39][0])/2
y4=(shape[36][1]+shape[39][1])/2
d4=math.sqrt(((shape[40][0]-x4)**2)+(shape[40][1]-y4)**2)
print(d4)
print(m4)
c4=math.sqrt(m4**2-d4**2)
ec4=c4/m4
print(ec4)
All_feature.append(ec4)
All_feature[-1]

m5=math.sqrt(((shape[42][0]-shape[45][0])**2)+((shape[42][1]-shape[45][1]))**2)/2
x5=(shape[42][0]+shape[45][0])/2
y5=(shape[42][1]+shape[45][1])/2
d5=math.sqrt(((shape[43][0]-x5)**2)+(shape[43][1]-y5)**2)
print(d5)
print(m5)
c5=math.sqrt(m5**2-d5**2)
```

```
ec5=c5/m5
print(ec5)
All_feature.append(ec5)
All_feature[-1]

m6=math.sqrt(((shape[42][0]-shape[45][0])**2)+((shape[42][1]-shape[45][1]))**2)/2
x6=(shape[42][0]+shape[45][0])/2
y6=(shape[42][1]+shape[45][1])/2
d6=math.sqrt(((shape[47][0]-x6)**2)+(shape[47][1]-y6)**2)
print(d6)
print(m6)
c6=math.sqrt(m6**2-d6**2)
ec6=c6/m6
print(ec6)
All_feature.append(ec6)
All_feature[-1]

m7=math.sqrt(((shape[17][0]-shape[21][0])**2)+((shape[17][1]-shape[21][1]))**2)/2
x7=(shape[17][0]+shape[21][0])/2
y7=(shape[17][1]+shape[21][1])/2
d7=math.sqrt(((shape[19][0]-x7)**2)+(shape[19][1]-y7)**2)
print(d7)
print(m7)
c7=math.sqrt(m7**2-d7**2)
ec7=c7/m7
print(ec7)
All_feature.append(ec7)
All_feature[-1]

m8=math.sqrt(((shape[22][0]-shape[26][0])**2)+((shape[22][1]-shape[26][1]))**2)/2
x8=(shape[22][0]+shape[26][0])/2
```

```
y8=(shape[22][1]+shape[26][1])/2
d8=math.sqrt(((shape[24][0]-x8)**2)+(shape[24][1]-y8)**2)
print(d8)
print(m8)
c8=math.sqrt(m8**2-d8**2)
ec8=c8/m8
print(ec8)
All_feature.append(ec8)
All_feature[-1]
import matplotlib.pyplot as plt
import cv2 as cv
from skimage import data
#img = data.coffee()
img=gray
def lbp_basic(img):
    basic_array = np.zeros(img.shape,np.uint8)
    for i in range(basic_array.shape[0]-1):
        for j in range(basic_array.shape[1]-1):
            basic_array[i,j] = bin_to_decimal(cal_basic_lbp(img,i,j))
    return basic_array
def cal_basic_lbp(img,i,j):#Points larger than the center pixel are assigned a value of 1, and those
 smaller than the center pixel are assigned a value of 0. The binary sequence is returned
    sum = []
    if img[i - 1, j ] > img[i, j]:
        sum.append(1)
    else:
        sum.append(0)
    if img[i - 1, j+1 ] > img[i, j]:
        sum.append(1)
    else:
        sum.append(0)
    if img[i , j + 1] > img[i, j]:
        sum.append(1)
    else:
        sum.append(0)
    if img[i + 1, j+1 ] > img[i, j]:
        sum.append(1)
    else:
        sum.append(0)
    if img[i + 1, j ] > img[i, j]:
        sum.append(1)
    else:
        sum.append(0)
    if img[i + 1, j - 1] > img[i, j]:
        sum.append(1)
    else:
        sum.append(0)
    if img[i , j - 1] > img[i, j]:
        sum.append(1)
    else:
        sum.append(0)
    if img[i - 1, j - 1] > img[i, j]:
        sum.append(1)
    else:
        sum.append(0)
    return sum
def bin_to_decimal(bin):#Binary to decimal
    res = 0
    bit_num = 0 #Shift left
    for i in bin[::-1]:
        res += i << bit_num   # Shifting n bits to the left is equal to multiplying by 2 to the nth pow
er
```

```
        bit_num += 1
    return res
def show_basic_hist(a): #Draw histogram of original lbp
    hist = cv.calcHist([a],[0],None,[256],[0,256])
    hist = cv.normalize(hist,hist)
    plt.figure(figsize = (8,4))
    plt.plot(hist, color='r')
    plt.xlim([0,256])
    plt.show()
#img1 = cv.cvtColor(img,cv.COLOR_BGR2GRAY)
basic_array = lbp_basic(img)
print(basic_array)
```