*Jyothi Hills, Panjal Road,*
*Vettikattiri PO, Cheruthuruthy , Thrissur,*
*Kerala 679531*

Jyothi Engineering College
NAAC Accredited College with NBA Accredited Programmes*
Approved by AICTE & affiliated to APJ Abdul Kalam Technological University
A CENTRE OF EXCELLENCE IN SCIENCE & TECHNOLOGY BY THE CATHOLIC ARCHDIOCESE OF TRICHUR
JYOTHI HILLS, VETTIKATTIRI P.O, CHERUTHURUTHY, THRISSUR. PIN-679531 PH : +91- 4884-259000, 274423 FAX : 04884-274777
NBA accredited B.Tech Programmes in Computer Science & Engineering, Electronics & Communication Engineering, Electrical & Electronics Engineering and Mechanical Engineering valid for the academic years 2016-2022. NBA accredited B.Tech Programme in Civil Engineering valid for the academic years 2019-2022.

# Bhavas Classification Using Deep Learning

## MAIN PROJECT REPORT

**NOVA DILEEP (JEC17CS078)**

**SHINOZ MOHAMMED P P (JEC17CS094)**

**SREELAKSHMI C (JEC17CS098)**

*in partial fulfillment for the award of the degree of*

**BACHELOR OF TECHNOLOGY (B.Tech)**

in

**COMPUTER SCIENCE & ENGINEERING**

of

**A P J ABDUL KALAM TECHNOLOGICAL UNIVERSITY**

Under the guidance of

**Ms. RESHMA K V**

CREATING TECHNOLOGY LEADERS OF TOMORROW

JUNE 2021

**Department of Computer Science & Engineering**

*Jyothi Hills, Panjal Road,*
*Vettikattiri PO, Cheruthuruthy , Thrissur,*
*Kerala 679531*

Jyothi Engineering College
NAAC Accredited College with NBA Accredited Programmes*
Approved by AICTE & affiliated to APJ Abdul Kalam Technological University
A CENTRE OF EXCELLENCE IN SCIENCE & TECHNOLOGY BY THE CATHOLIC ARCHDIOCESE OF TRICHUR
JYOTHI HILLS, VETTIKATTIRI P.O, CHERUTHURUTHY, THRISSUR. PIN-679531 PH : +91- 4884-259000, 274423 FAX : 04884-274777
NBA accredited B.Tech Programmes in Computer Science & Engineering, Electronics & Communication Engineering, Electrical & Electronics Engineering and Mechanical Engineering valid for the academic years 2016-2022. NBA accredited B.Tech Programme in Civil Engineering valid for the academic years 2019-2022.

# Bhavas Classification Using Deep Learning

## MAIN PROJECT REPORT

**NOVA DILEEP (JEC17CS078)**

**SHINOZ MOHAMMED P P (JEC17CS094)**

**SREELAKSHMI C (JEC17CS098)**

*in partial fulfillment for the award of the degree of*

**BACHELOR OF TECHNOLOGY (B.Tech)**

in

**COMPUTER SCIENCE & ENGINEERING**

of

**A P J ABDUL KALAM TECHNOLOGICAL UNIVERSITY**

Under the guidance of

**Ms.RESHMA K V**

CREATING TECHNOLOGY LEADERS OF TOMORROW

JUNE 2021

**Department of Computer Science & Engineering**

**Department of Computer Science and Engineering**

**JYOTHI ENGINEERING COLLEGE, CHERUTHURUTHY**

**THRISSUR 679 531**

JUNE 2021

## BONAFIDE CERTIFICATE

This is to certify that the main project report entitled **Bhavas Classification Using Deep Learning** submitted by **NOVA DILEEP (JEC17CS078), SHINOZ MOHAMMED P P (JEC17CS094) and SREELAKSHMI C(JEC17CS098)** in partial fulfillment of the requirements for the award of **Bachelor of Technology** degree in **Computer Science and Engineering** of **A P J Abdul Kalam Technological University** is the bonafide work carried out by them under our supervision and guidance.

| **Ms. Reshma K V** | **Mr. Shaiju Paul** | **Dr. Saju P John** |
|---|---|---|
| Project Guide | Project Coordinator | Head of The Dept |
| Assistant Professor | Assistant Professor | Professor |
| Dept. of CSE | Dept. of CSE | Dept. of CSE |

# DEPARTMENT OF

# COMPUTER SCIENCE & ENGINEERING

## COLLEGE VISION

Creating eminent and ethical leaders through quality professional education with emphasis on holistic excellence.

## COLLEGE MISSION

- To emerge as an institution par excellence of global standards by imparting quality engineering and other professional programmes with state-of- the-art facilities.
- To equip the students with appropriate skills for a meaningful career in the global scenario.
- To inculcate ethical values among students and ignite their passion for holistic excellence through social initiatives.
- To participate in the development of society through technology incubation, entrepreneurship and industry interaction.

# DEPARTMENT OF

# COMPUTER SCIENCE & ENGINEERING

## DEPARTMENT VISION

Creating eminent and ethical leaders in the domain of computational sciences through quality professional education with a focus on holistic learning and excellence.

## DEPARTMENT MISSION

- To create technically competent and ethically conscious graduates in the field of Computer Science & Engineering by encouraging holistic learning and excellence.
- To prepare students for careers in Industry, Academia and the Government.
- To instill Entrepreneurial Orientation and research motivation among the students of the department.
- To emerge as a leader in education in the region by encouraging teaching, learning, industry and societal connect

# PROGRAMME OUTCOMES (POs)

1. **Engineering Knowledge**: Apply the knowledge of mathematics, science, engineering fundamentals, and an engineering specialization to the solution of complex engineering problems.
2. **Problem Analysis**: Identify, formulate, review research literature, and analyze complex engineering problems reaching substantiated conclusions using first principles of mathematics, natural sciences, and engineering sciences.
3. **Design/Development of Solutions**: Design solutions for complex engineering problems and design system components or processes that meet the specified needs with appropriate consideration for the public health and safety, and the cultural, societal, and environmental considerations.
4. **Conduct Investigations of Complex Problems**: Use research-based knowledge and research methods including design of experiments, analysis and interpretation of data, and synthesis of the information to provide valid conclusions.
5. **Modern Tool Usage**: Create, select, and apply appropriate techniques, resources, and modern engineering and IT tools including prediction and modeling to complex engineering activities with an understanding of the limitations.
6. **The Engineer and Society**: Apply reasoning informed by the contextual knowledge to assess societal, health, safety, legal and cultural issues and the consequent responsibilities relevant to the professional engineering practice.
7. **Environment and Sustainability**: Understand the impact of the professional engineering solutions in societal and environmental contexts, and demonstrate the knowledge of, and need for sustainable development.
8. **Ethics**: Apply ethical principles and commit to professional ethics and responsibilities and norms of the engineering practice.
9. **Individual and Team Work**: Function effectively as an individual, and as a member or leader in diverse teams, and in multidisciplinary settings.
10. **Communication**: Communicate effectively on complex engineering activities with the engineering community and with society at large, such as, being able to comprehend and write effective reports and design documentation, make effective presentations, and give and receive clear instructions.
11. **Project Management and Finance**: Demonstrate knowledge and understanding of the engineering and management principles and apply these to one's own work, as a member and leader in a team, to manage projects and in multidisciplinary environments.
12. **Life-Long Learning**: Recognize the need for, and have the preparation and ability to engage in independent and life-long learning in the broadest context of technological change.

## PROGRAMME EDUCATIONAL OBJECTIVES (PEOs)

1. The graduates shall have sound knowledge of Mathematics, Science, Engineering and Management to be able to offer practical software and hardware solutions for the problems of industry and society at large.
2. The graduates shall be able to establish themselves as practising professionals, researchers or Entrepreneurs in computer science or allied areas and shall also be able to pursue higher education in reputed institutes.
3. The graduates shall be able to communicate effectively and work in multidisciplinary teams with team spirit demonstrating value driven and ethical leadership.

## PROGRAMME SPECIFIC OUTCOMES (PSOs)

1. An ability to apply knowledge of data structures and algorithms appropriate to computational problems.
2. An ability to apply knowledge of operating systems, programming languages, data management, or networking principles to computational assignments.
3. An ability to apply design, development, maintenance or evaluation of software engineering principles in the construction of computer and software systems of varying complexity and quality.
4. An ability to understand concepts involved in modeling and design of computer science applications in a way that demonstrates comprehension of the fundamentals and trade-offs involved in design choices.

# COURSE OUTCOMES (COs)

C410.1 The students will be able to analyse a current topic of professional interest and present it before an audience.

C410.2 Students will be able to identify an engineering problem, analyse it and propose a work plan to solve it.

C410.3 Students will have gained thorough knowledge in design, implementations and execution of Computer science related projects.

C410.4 Students will have attained the practical knowledge of what they learned in theory subjects.

C410.5 Students will become familiar with usage of modern tools.

C410.6 Students will have ability to plan and work in a team.

## PO - CO Mapping

| | POs | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **COs** | PO1 | PO2 | PO3 | PO4 | PO5 | PO6 | PO7 | PO8 | PO9 | PO10 | PO11 | PO12 |
| C410.1 | 3 | 2 | 3 | 3 | 3 | 3 | 2 | 3 | 3 | 2 | 3 | 2 |
| C410.2 | 2 | 3 | 3 | 3 | 3 | 3 | 2 | 3 | 2 | 3 | 3 | 3 |
| C410.3 | 3 | 2 | 2 | 3 | 3 | 3 | 3 | 3 | 2 | 3 | 3 | 3 |
| C410.4 | 3 | 3 | 2 | 3 | 3 | 3 | 3 | 2 | 3 | 3 | 3 | 3 |
| C410.5 | 2 | 3 | 3 | 3 | 2 | 3 | 3 | 2 | 2 | 3 | 2 | 2 |
| C410.6 | 3 | 2 | 3 | 2 | 2 | 3 | 2 | 3 | 2 | 3 | 2 | 2 |
| **Average** | 2.67 | 2.5 | 2.67 | 2.83 | 2.67 | 3 | 2.5 | 2.67 | 2.33 | 2.83 | 2.67 | 2.67 |

## PSO - CO Mapping

| | PSOs | | | |
|---|---|---|---|---|
| **COs** | PSO1 | PSO2 | PSO3 | PSO4 |
| C410.1 | 3 | 3 | 1 | 3 |
| C410.2 | 2 | 3 | 2 | 3 |
| C410.3 | 3 | 1 | 3 | 2 |
| C410.4 | 2 | 1 | 3 | 2 |
| C410.5 | 3 | 1 | 2 | 2 |
| C410.6 | 3 | 2 | 3 | 3 |
| **Average** | 2.67 | 1.833 | 2.33 | 2.5 |

# ACKNOWLEDGEMENT

We take this opportunity to express our heartfelt gratitude to all respected personalities who had guided, inspired and helped us in the successful completion of this project. First and foremost, we express our thanks to **The Lord Almighty** for guiding us in this endeavour and making it a success.

We take immense pleasure in thanking the **Management** of Jyothi Engineering College and **Dr. Sunny Joseph Kalayathankal**, Principal, Jyothi Engineering College for having permitted us to carry out this project. Our sincere thanks to **Prof. Dr. Saju P John**, Head of the Department of Computer Science and Engineering for permitting us to make use of the facilities available in the department to carry out the interim project successfully.

We express our sincere gratitude to **Mr. Shaiju Paul & Mrs. Swapna B Sasi**, Project Coordinators for their invaluable supervision and timely suggestions. We are very happy to express our deepest gratitude to our mentor **Ms. Reshma K V**, Assistant Professor, Department of Computer Science and Engineering, Jyothi Engineering College for her able guidance and continuous encouragement.

Last but not least, we extend our gratefulness to all teaching and non-teaching staff who directly or indirectly involved in the successful completion of this interim project work and to all friends who have patiently extended all sorts of help for accomplishing this undertaking.

# ABSTRACT

'Bhavas Classification' is used to classify the different 'bhavas' of classical dance such as 'Karunam'(Compassion), 'Veeram'(Valor), 'Bhayanakam'(Fear), 'Bhibhatsam'(Disgust), and 'Others'('Sringaram'(Love),'Hasyam'(Comic),'Raudram' (Anger), 'Adbhutam' (Wonder) and 'Shantam' (Tranquility)). These five classes (four'bhavas'and one 'others') are something which conveys the meaning intended ,through facial expressions. Normally, these 'bhavas' in classical dance are difficult for the non-trained dancers to understand. This system tries to identify the 'bhavas' from facial expressions and classify them to the respective classes. There by all people can easily understand the meaning of each 'bhavas' and enjoy the dance in its full extent. This system uses technology of deep learning to classify the images of facial expressions. The project aims at facial expression detection and emotional classification using deep learning technique of Convolution Neural Network (CNN).The main goal of this project is to validate the usage of this methodology of facial expression detection and classification.

*Keywords: Facial Expression Recognition; Deep Learning; CNN Architecture;Emotion Classification.*

# CONTENTS

# List of Figures

# List of Abbreviations

**CNN** : Convolutional Neural Network
**JAFFE** : Rectified Linear Units
**ReLu** : Rectified Linear activation function
**SFEW** : Static Facial Expressions in the Wild
**RPNs** : Region Proposal Net-works
**MLP** : Multilayer Perceptron
**ICML**: International Conference on Machine Learning

# CHAPTER 1
# INTRODUCTION

## 1.1 Overview

There are different ways for communication. Both verbal and non verbal communication plays an important role in communicating with each other in day to day life. Among the non verbal communication, facial expressions are the most significant as it is universal. The facial expressions for happiness, sadness, anger, surprise, fear, and disgust are the same across cultures. It can display personal emotions and indicate an individual's intentions within a social situation. It is said that the emotional expression through face can convey 55% of information. Generally, facial expressions are classified into six types such as happiness, sadness, anger, surprise, fear, and disgust. But in classical dance, there are nine different types of facial expressions such as 'Sringara'(Love), 'Hasya'(Comic), 'Karuna'(Compassion), 'Raudra'(Anger), 'Veera'(Valor), 'Bhayankara'(Fear), 'Bhibhatsa'(Disgust), 'Adbhuta'(Wonder) and 'Shanta' (Tranquility). It is difficult for a non trained dancer to identify the facial expression as some of the facial expressions look similar like raudra and veera.

This project aims at classifying the facial expression of the given image to the 'bhava' that it denotes using deep learning technology. CNN algorithm is used for extracting features like eye, lip and eyebrows and also for classification. As a first step, the input image is preprocessed. Then its features are extracted and it is classified to its 'bhava' using CNN. Due to the use of CNN algorithm, the process of complex artificial feature extraction in traditional facial expression recognition is avoided, and feature extraction and expression classification are performed simultaneously.

## 1.2 Objectives

The main objective of this project is to classify the facial expression of the image to bhavas using CNN algorithm of deep learning technology. Thereby all people can understand the intended meaning of each bhava regardless of whether the person is trained or not.

## 1.3 Data Description

The dataset for this project has been created by ourselves by downloading various images from internet and making a collection of different bhavas. The data-set as a whole is generally divided into three categories: training, test and validation. The complete dataset in the drive is divided into two files train and test. In train set, images that are trained for different bhavas are stored. As is the case with a usual deep learning problem, we would be training the model using train dataset and evaluating the performance with the test dataset.

## 1.4 Organization of the project

The report is organised as follow:

- **Chapter 1:Introduction** Gives an introduction to "Bahavs Classification Using Deep Learning".
- **Chapter 2:Literature Survey** Summarizes the various exiting techniques that helps in achieving the desired result.
- **Chapter 3: Problem Statement** Discuses about the need for the proposed system.
- **Chapter 4:Project Management** Contains the effective project management model to be used for the project.
- **Chapter 5:Methodology** Describes the various steps involved to produce this project.
- **Chapter 6:Results** Describes the various technologies needed for implementation.
- **Chapter 7:Conclusion & Future Scope** Concludes with the future scope of implementation.
- **Chapter 8:Appendix** Screenshots of our project code.
- **References** Includes the references for the project.

# CHAPTER 2
# LITERATURE SURVEY

## 2.1 Human facial expression recognition using deep learning technique [5]

Recognize emotions from facial expressions by using static images. It is a type of signal processing, which is used in various fields, similar to human-computer interaction. Some resources for automatic emotion recognition using machine learning methods are proposed. Deep learning technology solves many real-time problems. Defined a Convolutional Neural Network (CNN) to recognize the 6 basic emotions of the technology and have been implemented in MATLAB. Facial expression recognition has become a source of progress in applications such as neuro marketing. Facial expression is an important factor in human communication that helps us understand the intentions of others. Even the image of the same person can have many variations. In spite of frequent research, it is rarely avoided in training and testing to show fairly evaluated works. Facial expression is the main important feature of human emotion recognition. The face is always the most sensitive and communicative part of human beings. Feature extraction and classification have been successfully determined in the convolutional neural network (CNN) architecture.

### 2.1.1 Face Recognition System

The method of recognizing the emotion of the person based on the characteristics of the person is called the face recognition system. The common method of facial emotion recognition includes three steps: face detection, feature extraction and classification. Face detection processes facial images, and can identify facial positions from input images without manual intervention. After placing the face, the facial expression will be extracted. The final stage is to recognize facial expressions. Facial changes can be recognized as emotions or actions. We know that humans are full of emotions, but basically we assume basic emotions. Facial muscle movement helps to recognize human emotions. These functions are the key parameters that can recognize emotions. Facial parameters include eyebrows, mouth, nose, eyes and cheeks.

**Figure 2.1: Six basic emotions**

| Sr. No. | EMOTION CLASS | Description of Facial expression |
|---|---|---|
| 1 | Happy | The eyebrows are relaxed. The mouth is open and the mouth corners upturned. |
| 2 | Sad | The inner eyebrows are bent upward. The eyes are slightly closed. The mouth is usually relaxed. |
| 3 | Fear | The eyebrows are raised and pulled together. The inner eyebrows are bent upward. The eyes are open and tense. |
| 4 | Anger | The inner eyebrows are pulled downward and together. The eyes are wide open. The lips are tightly closed or opened to expose the teeth. |
| 5 | Surprise | The eyebrows are raised. The upper eyelids and the eyes are wide open. The mouth is opened. |
| 6 | Disgust | The eyebrows and eyelids are relaxed. The upper lip is raised and curled, often asymmetrically. |

**Figure 2.2: Description of facial expression**

### 2.1.2 Facial Expression

Have expressed their emotions. It is to display messages and signs. Like it, it may cause the eyebrows to relax, the mouth opens and the corners of the mouth turn up. A process must be able to recognize several emotions.

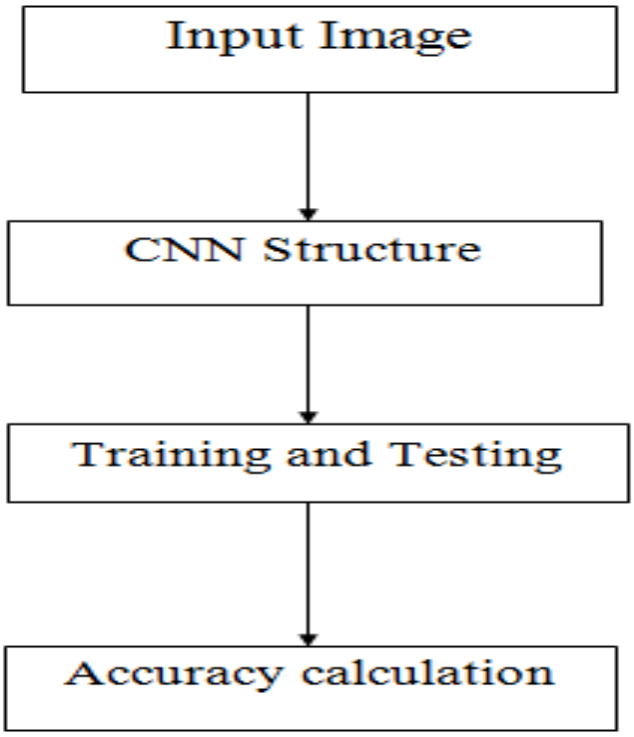



**Figure 2.3: Block diagram**

### 2.1.3 Methods and Materials

1. **Dataset**

   The process is finished by utilizing JAFFE data sets which has 6 feelings in it. The total images present in this are 213 images with the dimensions of 256x256.

2. **Convolutional Neural Network method(CNN)**

   A CNN takes an image as an input and tries to combine the given image and target data in matlab. In CNN, weights connect a small part of the input to each different neuron in the first layer. These different sets of weights are called "kernels". Convolution is like a mathematical operation. The kernel is placed in the upper left corner of the input image. If we already know the kernel to use, then convolution will be very powerful in finding the features of the input image. Kernel design is an art form that has been improved over the past few decades and can do some amazing things to the image. Each convolution result is placed in the next layer of hidden nodes. Each function of the convolutional image is a node in the hidden layer.

| CATEGORY | JAFFE |
|---|---|
| Labels | 6 Emotions |
| The number | 213 |
| Participants | 10(Female) |
| Resolution | 256x256 |
| Format | .tiff |

**Figure 2.4: Experimental setups**

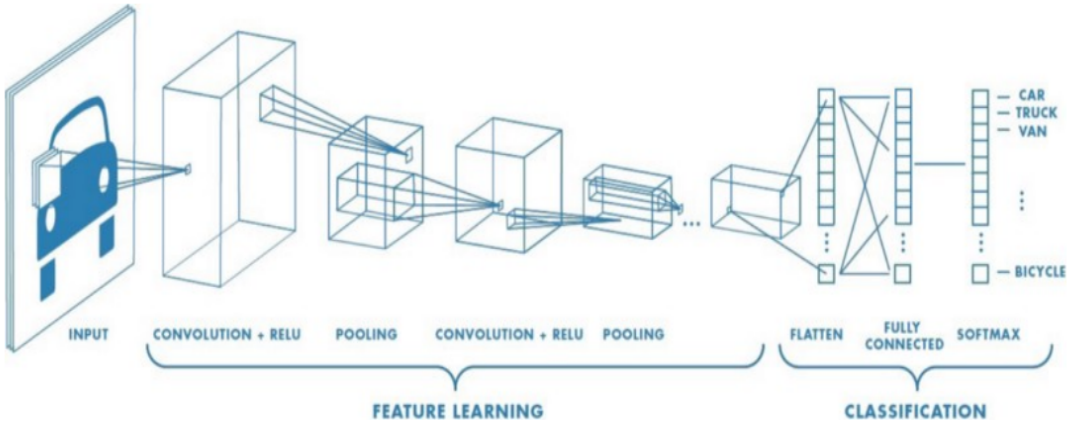



**Figure 2.5: CNN model**

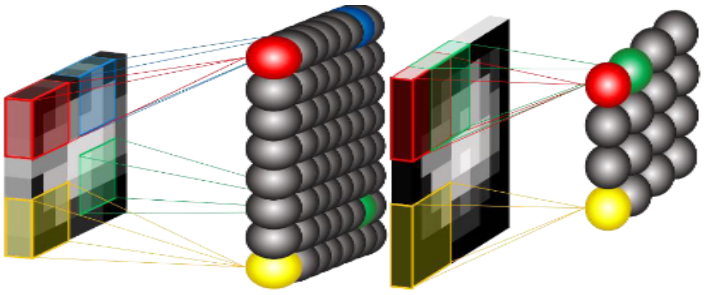


**Figure 2.6: Hidden layer nodes in NN**

### 2.1 **Input layer**

In this layer, the given input image is 2D grayscale. The input image is converted into a pattern. The kernel is a set of weights, which is multiplied by the input image pattern. This dot product method starts from the corner edge of the input image pattern. For grayscale images, the size of the captured input image is 256 x 256 x 1. For some sizes, CNN has the best input effect.

### 2.2 **Convolution layer**

It uses stride and zero padding to get the image, and if the kernel is ready for use, it provides output. The order of input image size is mainly considering the realization of CNN in matlab.The Convolutional Layer makes use of a set of learnable filters. A filter is used to detect the presence of specific features or patterns present in the original image (input). It is usually expressed as a matrix (MxMx3), with a smaller dimension but the same depth as the input file. This filter is convolved (slided) across the width and height of the input file, and a dot product is computed to give an activation map.

### 2.3 **Activation function**

It maps the resulting values in between 0 to 1 or -1 to 1 depending upon the function. ReLU function is the most widely used activation function in neural networks today. One of the greatest advantage ReLU has over other activation functions is that it does not activate all neurons at the same time. From the image for ReLU function above, we'll notice that it converts all negative inputs to zero and the neuron does not get activated. This makes it very computational efficient as few neurons are activated per time. It does not saturate at the positive region. In practice, ReLU converges six times faster tanh tang and sigmoid activation functions. Some disadvantage ReLU presents is that it is saturated at the negative region, meaning that the gradient at that region is zero. With the gradient equal to zero, during back propagation all the weights will not be updated, to fix this, we use Leaky ReLU. Also, ReLU functions are not zero-centered. This means that for it to get to its optimal point, it will have to use a zig-zag path which may be longer.

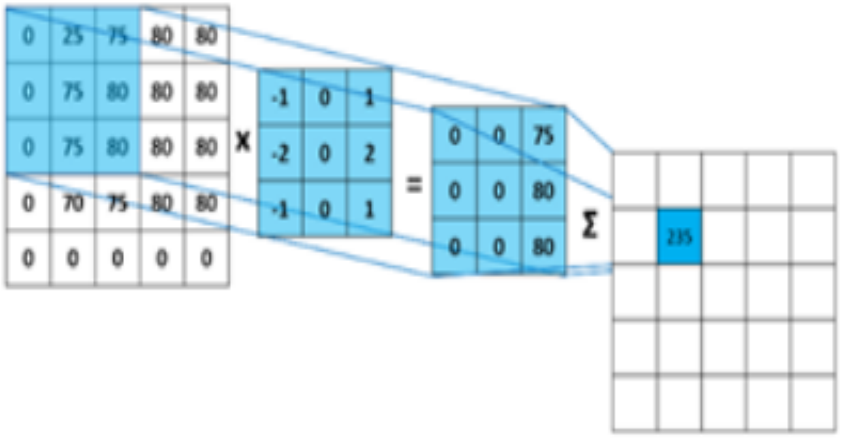



**Figure 2.7: Hidden layer nodes in NN**

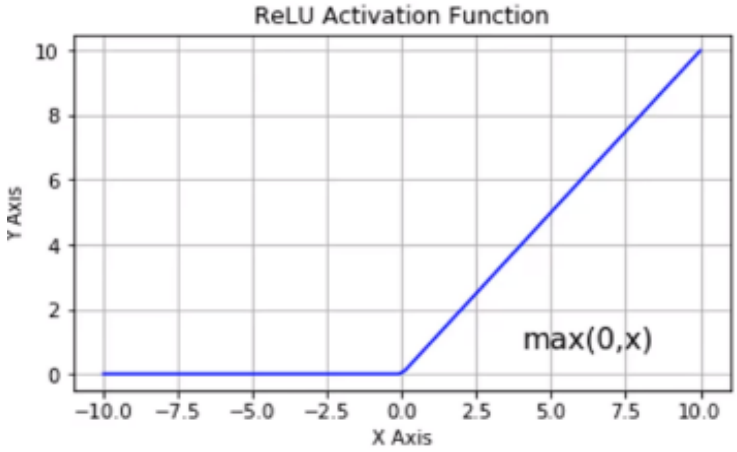



**Figure 2.8: ReLU Activation Function**

## 2.4 Pooling layer

The layer after convolutional layer is mostly pooling layer in CNN architecture. It divides the input image into a set of rectangles and, for each such sub-region, outputs a value. The Pooling layer can be seen between Convolution layers in a CNN architecture. This layer basically reduces the number of parameters and computation in the network, controlling overfitting by progressively reducing the spatial size of the network. There are two operations in this layer; Average pooling and Maximum pooling. Only Max-pooling will be discussed in this post.

Max-pooling, like the name states; will take out only the maximum from a pool. This is actually done with the use of filters sliding through the input; and at every stride, the maximum parameter is taken out and the rest is dropped. This actually down-samples the network.

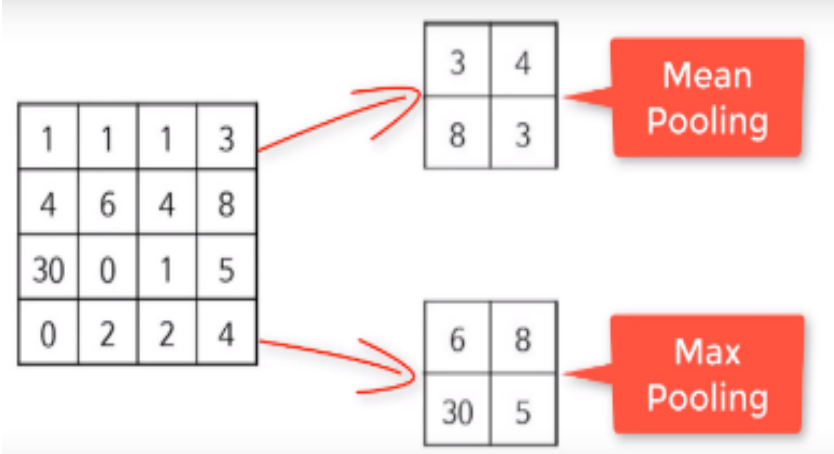



**Figure 2.9: Pooling Layer**

| LAYERS | No of layers used |
|---|---|
| Convolution layer | 2 layers |
| Pooling layer | 2 layers |
| Activation function | Happens in all layers |
| Fully connected layer | 2 layers |

**Figure 2.10: Layers used**

Average pooling involves calculating the average for each patch of the feature map. This means that each 2×2 square of the feature map is down sampled to the average value in the square. For example, the output of the line detector convolutional filter in the previous section was a 6×6 feature map. We can look at applying the average pooling operation to the first line of that feature map manually.

## 2.5 Fully Connected layer

Fully Connected layers are 1D vectors and each neurons in this layer will be connected to the previous volume. In this layer, the neurons have a complete connection to all the activations from the previous layers. Their activations can hence be computed with a matrix multiplication followed by a bias offset. This is the last phase for a CNN network. The Convolutional Neural Network is actually made up of hidden layers and fully-connected layer(s).

The Accuracy is proposed by number of correct prediction by total number of test images with multiples of 100.
Accuracy (Acc) = no of correct prediction / total no of test images x 100
In this work a CNN method for emotion recognition from static images of facial expression were explored.An accuracy of 91.6 % has been achieved by the test result.Now a days computers are widely used in various fields of human computer interactions in order to recognize human expressions. As the facial expression recognition systems are becoming vigorou sand effective in communications, many other innovative applications and uses are yet to be seen.

## 2.2 Facial Expression Recognition using Deep Learning [8]

In all ages, facial expressions have become a common way of nonverbal communication. The ability to recognize facial expressions will pave the way for many novel applications. Regardless of how conventional methodologies are implemented in a controlled environment, these methodologies will fail on nuanced data sets that contain partial faces. In this work, it is planned to create a deep learning model that can detect people's emotions even under natural conditions. For the FER-2013 data set. The main purpose of this work is to create a system that can detect people's emotions under natural conditions and is more accurate than traditional methods and human-level performance.Human emotions play a vital role in interpersonal nonverbal communication. However, emotions are abstract and cannot be seen directly with the naked eye. You can still easily discover your emotional state by simply observing facial expressions, speech and even text. Among them, facial expressions have become one of the most popular expressions due to such reasons. They are visible and easier to collect facial expression data than some other functions. In recent years, due to developments in fields such as human-computer interaction, security and testing, people's awareness of human emotions has increased. Although it is known that facial expressions are successful under controlled conditions, emotions can still be found in natural situations due to occlusion, changes in posture and changes in lighting. In the past ten years, with the rise of neural networks and other deep learning technologies, they have reached an excellent benchmark for facial expression recognition under normal conditions, even exceeding the accuracy of humans. This paved the way for many novel applications in various fields such as medicine, security and robotics, which were previously thought to be impossible. In this work, this main goal is to better understand and improve the performance of emotion recognition models in the process. In also adopted some methods from recent publications, including transfer learning and integration, to improve the accuracy of the model. At the same time, apart from the "data set", did not use any other auxiliary data to train this model.

### 2.2.1 Related Works

In one of the most influential works in the field of emotion recognition, Paul Ekman Wallace identified six main emotions, namely happiness, sadness, anger, surprise, fear and disgust. Later, "neutral" was added to the list as the main emotion, resulting in seven basic emotions. Therefore, most of the current data sets follow the initial trend set earlier and consist of seven emotions.

Earlier work mainly involved a two-step method. The first step was to extract some key features from the image, and the second step was to use a classifier (such as a neural network or SVM) to detect emotions. This involves hand-made functions such as histogram of gradients (HOG), Gabor wavelet and Haar functions. This method is elegant for simple data sets, but not good enough for data sets with larger intra-class differences.

Improvements in deep learning, especially in the field of computer vision and classification of convolutional neural networks, have prompted the development of models to solve the FER problem. As part of the ICML 2013 Representation Learning Seminar, such a competition was held in Kaggle. The competition introduced the FER2013 data set, and most traditional methods cannot achieve reasonable accuracy.

The top three teams in the competition all use CNN in combination with image conversion. The champion of the competition, Y. Tang, used the original formula of SVM as the loss function for training, and used the L2-SVM loss function. This achieved good results at the time, reaching an accuracy of 71.2 %, ahead of the competition.

A deep learning model based on the end-to-end principle is proposed to distinguish emotions, usually by increasing the number of layers in the convolutional network to improve accuracy. Nevertheless, due to the moderate number of sensory categories, they proposed a convolutional network with less than ten layers to achieve promising results (with an accuracy of 70.2%).

### 2.2.2 Dataset

FER is a well-studied field with various datasets available. In this work, we used FER-2013 as our Dataset for both training and testing with no usage of any auxiliary data.FER-2013:This Dataset was first introduced in the ICML 2013 Challenges in Representation learning. Ever since then, this has been used in several other competitions and research papers. This is one of the challenging datasets with human-level accuracy of 65±5 %, and the highest performing published works, achieving 75±2 %. This Dataset contains 35,887 images of 48x48


**Figure 2.11: Images from each class of the FER2013 Dataset**

resolution, most of which are taken under an uncontrolled setting. FER2013 contains seven facial expressions, with distributions of anger (4953), Disgust (547), Fear (5121), Happy (8989), Sad (6077), Surprise (4002), Neutral (6198).

### 2.2.3 Experimental Setup

The dataset is split into training set (28709 images), validation set (3589 images), and testing set (3589 images). All the accuracies we present in the result section is by working on the testing set. We ran all our models in Google Colabratory storing the dataset and other necessary files in Google drive and mounting it on our colabratory with the basic GPU support.

### 2.2.4 Proposed Model

1. **Baseline Model**

   The first model built consisted of two 3x3x32 same padding convolution layers, two 3x3x64 convolution layers with ReLu activations, three 2x2 Maxpooling layers, and completed with an FC layer and a softmax layer. Also, batch normalization and 20% dropout layers were included to achieve an accuracy of 61%.

2. **Five Layer Model**

   This is also model which consists of four stages of two convolution layers in combination with a max-pooling layer, dropout (50%) layers have been added to each convolutional layers block., Fc layer of size 512 and a softmax output layer. In the first convolution layer, L2 (0.01) regularization is used, and in the remaining convolution layers, batch normalization is utilized. All convolution layers apply 32 filters of size 3x3. The max-

pooling layers used kernels of size 2x2 and stride 2. “ReLU” has been used as the activation function for all convolutional layers. The categorical cross-entropy function was used as our loss function. Made the loss function to eliminate any "plateaus" by reducing the learning rate parameter of the optimization function with a specific value (0.9 in this case) if the loss function doesn’t get better with a particular epoch(3 consecutive epochs). Stop the training process if accuracy does not change for a specific epoch(after eight epochs). With this model, we were able to get an accuracy of 66.2%

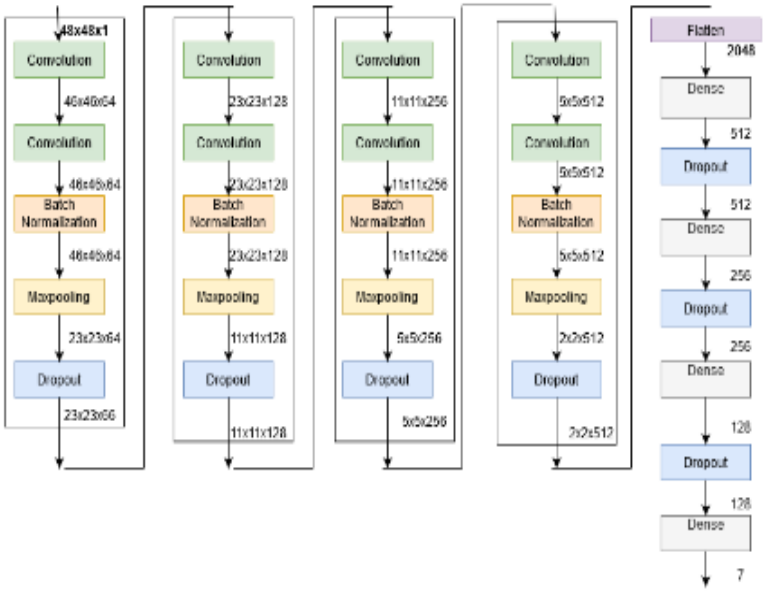



**Figure 2.12: Example of a figure caption. (figure caption)**

3. **Transfer Learning**

   Since FER2013 is a relatively small and unbalanced dataset, transfer learning was beneficial for boosting our model’s accuracy. Primarily ResNet50 as pre-trained model from the VGG-face library by freezing some first layers and training on the last few layers to make a prediction. Also, the ResNet50 model had different input requirements than our actual input dimensions. All the images have to be resized from 48x48 to 197x197 aspects and should be colored from grayscale to RGB.

4. **ResNet50 Model**

   This model is a residual network residual net with 50 layers. The Keras implementation of ResNet50 consists of 175 layers similar to the works Brechet . It replaced the output layer withtwo FC layers (FC 4096 and FC 1024) and a softmax layer that provides a probability for each of the seven emotions. It achieved 73% accuracy on the test set.Freezing the entire pre-trained network came to be unsuccessful since the model failed to fit on to

the training set in the initial stages.

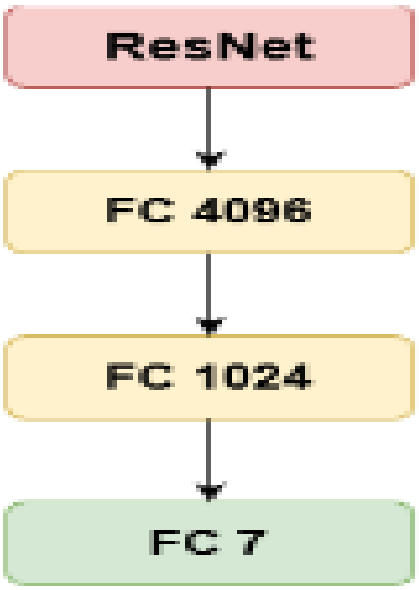



**Figure 2.13: ResNet50 Module with Transfer Learning**

5. **EDNET–EMOTION DETECTION NETWORK**

   This combination of the two models ResNet50 and the five-layer model the novel aspect in our ResNet50 model was that we supplanted the original output layer with FC layers of size1024 and4096 and a softmax layer this has given us an accuracy of 73% . Then we further ensemble this with the Five-Layer Model to for our ED NET, the assembling has improved the accuracy significantly from 73% to 73.92% whose accuracy is comparable to even the state-of-the-art models.

### 2.2.5 Results

Accuracies of some of the models achieved on the FER2013 test data. Results from the Yichuan tang model , our baseline, five-layer model, ResNet50 model, and the ensemble of the five-layer model and ResNet50 are presented The state-of-the-art model was an ensemble of six other pre-trained models; most of them from the ImageNet project were fine-tuned and trained using multiple datasets. Cannot recreate such a model due to this limited computing power.

| Model | Depth | Accuracy |
|---|---|---|
| Human-level | - | 65±5% |
| Yichuan Tan [1] | 4 | 71.2% |
| Shervin Minaee[5] | 9 | 70.02% |
| Baseline | 5 | 61% |
| Five-layer model | 5 | 66.2% |
| ResNet50 | 50 | 73% |
| ED NET | - | 73.9% |
| State-of-the-art[] | - | 76.8% |

**Figure 2.14: Result for FER 2013 private test set**

The main objective of this paper, in the beginning, was to create a good CNN model to outperform the traditional approaches and human-level accuracy. It wasable to achieve this initial objectives with a simple five-layer model. From there on went through recent publications on using pre-trained models of the ImageNetproject for other activities by freezing and manipulating a few layers of the models and found a ResNet50 model and fine-tuned it according to our requirements and ensembles it with our five-layer model to get results that were comparable to even the state-of-the-art models without using any auxiliary data.

## 2.3 Facial Expression Recognition via Deep Learning [4]

The recognition of facial expressions is difficult problem for machine learning techniques, since people can vary significantly in the way they show their expressions. An automatic facial expression recognition system is an important component in human machine interaction. It consists of evaluating the possibility of emotions' recognition. However, this is not an easy task. Deep learning is a new area of research within machine learning method which can classify images of human faces into emotion categories using Deep Neural Networks (DNN). Convolutional neural networks (CNN) have been widely used to overcome the difficulties in facial expression classification. Deep learning methods have been successfully applied to extract features and classification, in particular Convolutional Neural Networks (CNN) architectures

have biologically inspired from the multi-stage one that learned automatically the hierarchies of invariant features.
The ConvNets consist of a multistage processing of an input image to extract hierarchical and high-level feature representations. This paper uses an effective approach system based on ConvNets for facial expression recognition. It proposes a new architecture which the input of the system is an image. Then, we use CNN to predict the facial expression label which should be one these labels: anger,happiness, sadness, disgust, surprise and neutral.

### 2.3.1 Proposed method

It consists of two steps. In first step, the CNN architecture is fine tuned by VGG model to get the first model. In second step, inorder to improve classification the CNN architecture is fine tuned by first model and final model is obtained.

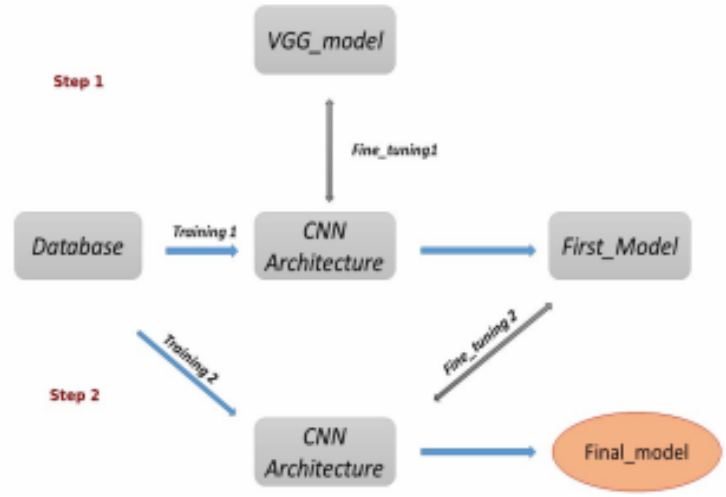



**Figure 2.15: Proposed approach**

### 2.3.2 Dataset

CK+, MUG, and RAFD are the datasets used. To train CNN architecture, the size of all images are standardised to 224 × 242 pixels. The CK+ database consists of 327 expression sequences. From each image sequence, only the last four frames are selected to save the most clear expression. Meanwhile the first frames from each sequence are collected for neutral expression.

KDEF database is a set of totally 4900 pictures of human facial expressions of emotion. This database proposed 7 facial expressions in different positions. But only 3 positions of the

images of this database are used in the training step. The Radboud Faces Database (RaFD) is a set of pictures of 67 models (including Caucasian males and females, Caucasian children, both boys and girls, and Moroccan Dutch males) displaying 8 emotional expressions. MUG database consists of image sequences of 86 subjects performing facial expressions. In the database participated 35 women and 51 men all of Caucasian origin between 20 and 35 years of age. Each image was saved with a jpg format, 896896 pixels. Thus the experimental dataset consists of a total of 37000 images. 330000 images are used in training step, and the rest of images are used as test images.[7]

### 2.3.3 Network architecture

The proposed deep ConvNet contain four convolutional layers with three max-pooling layers to extract features hierarchically, followed by the fully connected layer and the softmax output layer indicating 6 expression classes. The input of the network is 165×165 × k for all patches, where k = 3 for color patches and k = 1 for gray patches. The output is one of the 6 expression classes.

| Layer type | Size/Stride | Output<br>Dropout Probability |
|---|---|---|
| Data | 165×165 | - |
| Convolution_1 | 4×4 / 1 | 20 |
| Max_Pooling_1 | 2×2 / 2 | - |
| Convolution_2 | 3×3 / 2 | 40 |
| Max_Pooling_2 | 2×2 / 2 | - |
| Convolution_3 | 3×3 / 2 | 60 |
| Max_Pooling_3 | 2×2 / 2 | - |
| Convolution_4 | 2×2 / 2 | 80 |
| Fully Connected | - | 160<br>Dropout=0.5 |
| Fully Connected | - | 7 |

**Figure 2.16: Network Configuration**

### 2.3.4 Fine Tuning

VGG is fine tuned. Caffe framework is used to get the model weights. VGG model was trained on the large CASIA WebFace dataset and the Static Facial Expressions in the Wild (SFEW) dataset, which is a smaller database of labeled facial emotions and it has been developed by selecting frames from Acted Facial Expressions In The Wild.

### 2.3.5 Performance Analysis

To verify the effectiveness of the proposed approach, a validation is performed on standard databases MUG, RaFD and CK+. The proposed method achieves the best performance on only 5 database classes (Disgust, Happy,Neutral, Sad and Surprise) with recognition rate 100%. The recognition rate of Angry emotion is still difficult (96%) because of the database characteristics. The model excelled with classifying the CK+ dataset with recognition rate of 99.33%. Also the proposed method achieved a recognition rate of only 87.65% on MUG database. From RaFD database, an accuracy rate of 93.33% is achieved. In this database there are some incorrect classification for Sad and Neutral emotions and specially the Sad expression due to the unclarity features in this class.

The Deep learning method achieved the best performance on CK+ database. The proposed method achieved the last best performance of 71.04% accuracy on CK+.

## 2.4 Facial Expression Recognition with Faster R-CNN [6]

In order to avoid the complexity of explicit feature extraction process and the low level data operation involved in traditional facial expression recognition, Faster R-CNN is proposed for facial expression recognition in this paper. Firstly, the facial expression image is normalized and the features are extracted by using the trainable convolution kernel. Then, the maximum pooling is used to reduce the dimensions of the extracted features. The Region Proposal Networks (RPNs) are used for predicting accurate region proposals. The proposed method uses just one convolutional neural network (CNN) for all purposes and classifies facial expressions to worried, angry, disgust, surprise, anxious, happy, sad and neutral. So, the region proposal is nearly cost free by sharing convolutional features of the down flow detection network. It also improved the regional proposal quality and the accuracy of the overall target detection.

Region proposal network (RPN) and Fast R-CNN require an original feature extraction network. The ImageNet Classification Library is used to train the initial parameter W0 of the network. Then the network is fine-tuned by the specified dataset. The proposed method provides three methods: 1) train RPN to extract the anchors on the training set from W0 . 2) train Fast R-CNN by using the anchors from W0 , and the parameter is denoted as W1 3) train RPN from W1.

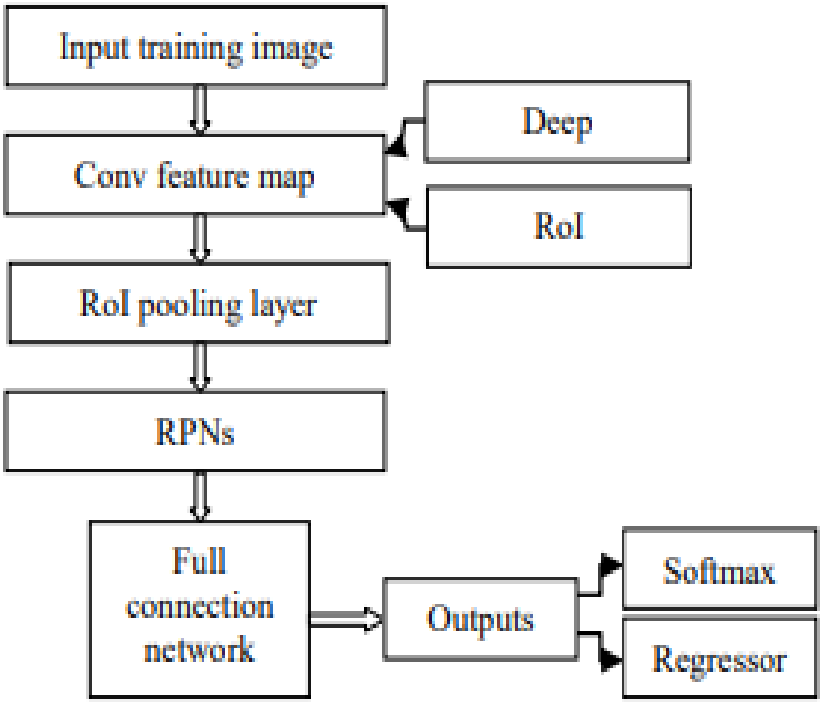



**Figure 2.17: Facial expression recognition method**

### 2.4.1 Dataset

The dataset is provided by Chinese Linguistic Data Consortium(CLDC), which is composed of multimodal emotional audio and video data. Total eight expression are collected from TV series or movies. The data set consists of 66486 pictures with eight categories. Among them are 6174 worried pictures, 10862 angry pictures, 1687 disgust pictures, 2574 surprise pictures, 12019 anxious pictures, 9867 happy pictures , 18326 sad pictures and 4977 neural pictures. The background was considered as one class. So, total 9 categories were used

### 2.4.2 Data labelling

Since Faster R-CNN is used, the region of interest must be marked first. For that, a software is used to get the coordinates of RoI and then it is converted into XML format.

### 2.4.3 Model

The three pascal voc model provided by the py-faster-rcnn such as VGG CNN M 1024, ZF and VGG16, which are called the faster rcnn alt opt network to fine-turn imagenet model

### 2.4.4 Training and Testing

The threshold is set as 0.8 and those proposals that are above this threshold are only considered as positive and those below this threshold are considered as negative and are ignored. The loss function is defined as:

$$L(p_i,t_i) = L_{cls}(p_i,p_i^*) + \lambda p_i^* L_{reg}(t_i,t_i^*)$$

Here pi is the anchor i being an object of the probability of prediction. If the anchor label is positive, pi* is 1. If it is negative, it is 0. ti=tx, ty, tw, thi represents the predicted bounding box of four parametrized coordinates and ti*=tx*, ty*, tw*, th*i indicates that the ground-truth box is associated with a positive anchor

### 2.4.5 Result

From the table, it can be known that the neutral type has not much recognition rate compared to other expressions. It is because neutral expression is very hard to identify and is easy to be confused with other expressions. In this paper, facial expressions like worry, angry, disgust, neutral, surprise, anxious, happy, sad are detected using Faster R-CNN. The Region Proposal Networks (RPNs) was used to generate an efficient and an accurate region proposal. In each image, the proposed method locate the face region and recognize the expression directly. The experimental results show that the proposed method achieved better recognition performance.

| pascal_voc.model | worried | angry | disgust | surprise | anxious | happy | sad | neural | mAP |
|---|---|---|---|---|---|---|---|---|---|
| VGG_CNN_M_1024 | 0.7922 | 0.8998 | 0.9798 | 0.8879 | 0.8806 | 0.8513 | 0.9078 | 0.3604 | 0.8200 |
| ZF | 0.8015 | 0.8998 | 0.9739 | 0.8979 | 0.8799 | 0.8666 | 0.9083 | 0.3344 | 0.8203 |
| VGG16 | 0.8315 | 0.9016 | 0.9782 | 0.8973 | 0.8857 | 0.8812 | 0.9091 | 0.3646 | 0.8312 |

**Figure 2.18: Training result of three kinds of networks**

## 2.5 Facial Expression Recognition with Convolutional Neural Networks [9]

Facial expressions are essential to human social communication, as this communication is both verbal and non-verbal. Facial expressions are one aspect of non-verbal communication, as the face expresses prominent signals of communication, which includes eye contact. Other aspects of non-verbal communication are gestures and body language. It is easy for humans

to notice and understand faces and facial expressions. In this paper, they demonstrate the classification of FER based on static images, using CNNs, without requiring any pre-processing or feature extraction tasks. It also illustrates techniques to improve future accuracy in this area by using pre-processing, which includes face detection and illumination correction. Feature extraction is used to extract the most prominent parts of the face, including the jaw, mouth, eyes, nose, and eyebrows. Paper also discuss about CNN architecture, and the challenges of using max-pooling and dropout, which will help in better performance.

### 2.5.1 Dataset

The FER2013 dataset is used in this system. FER2013 is a large dataset, which is publicly accessible on Kaggle's FER Challenge. The FER2013 dataset contains 35,887 face crops, including training, validation and testing images, with 28,709, 3,589 and 3,589, respectively. It include the images of seven different emotions with their corresponding labels (0= anger, 1= disgust, 2= fear, 3=happiness, 4= sadness, 5= surprise and 6= neutral). All images are of 48x48 pixel resolutions and on grayscale. A 30-ms analysis window is outlined, and the speech signal is divided into totally different time frames by shifting the window. Since audio signal sample is 1s each, there will (1000-30)/10+1=98 time frames.


**Figure 2.19: Example images from the FER2013 dataset with labels**

### 2.5.2 Technical works

The work is split into three different components :

1. **Pre-processing**

Pre-processing can be used to enhance FER system performance and can be done previous to the feature extraction process. It includes various processes, such as the detection and alignment of faces, correction of illumination, pose, occlusion, and data augmentation. In the selected dataset FER2013, the faces are registered automatically, so that they have similar space requirements and are more or less centered in the images. When images are captured in various types of light, expression features are sometimes inaccurately detected, and therefore, the expression recognition rate can be low and make feature extraction more difficult. In-order to prevent that the illumination correction is done by histogram equalization.[3]

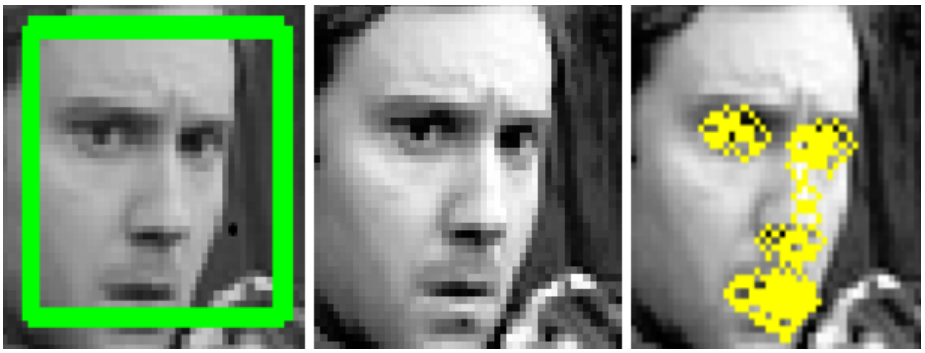


**Figure 2.20: a) Illustration of face detection (green square), b) illumination correction and c) features extraction of the right eye, left eye, nose and mouth (yellow color)**

2. **Feature Extraction**

   The extraction of facial features requires translating the input data into a set of features. By using feature extraction we can reduce an immense amount of data down to a relatively small set, which allows for faster computation. Here, they applied dlib facial landmark detector pre-trained on iBUG 300-W dataset for feature extraction, and extracted the eight most prominent parts of a face, including both eyebrows, both eyes, the nose, the inner and outer outlines of the mouth, and the jaw.

3. **CNN Architecture**

   CNNs have been widely used in a variety of computer vision applications, including FER. Early in the 21st century, several studies of FER literature determined that CNNs work well on changes of face location, as well as variations in scale. They were also found to work better than multilayer perceptron (MLP) when looking at face pose variations not seen previously. Researchers used CNN to help solve various facial expression recognition problems, such as translation, rotation, subject independence, and scale in-

variance. Fig. 4 shows that finding an optimized architecture or network structure is a very challenging task. Our model was trained using the following characteristics:

- Six convolutional layers using "RELU" as an activation function
- Three max-pooling: out of which the first two using pool size (3,3) and stride (2,2), and the third using pool size (2,2) and stride (2,2); every max pooling is followed by every two convolutional layers
- Two drop out with value 0.2
- One flattened layer and two dense layers: one dense layer with "RELU," and the other with "Softmax" as an activation function

| Layer (type) | Output Shape | Param # |
|---|---|---|
| conv2d_1 (Conv2D) | (None, 46, 46, 128) | 1280 |
| conv2d_2 (Conv2D) | (None, 44, 44, 128) | 147584 |
| max_pooling2d_1 (MaxPooling2 | (None, 21, 21, 128) | 0 |
| conv2d_3 (Conv2D) | (None, 19, 19, 128) | 147584 |
| conv2d_4 (Conv2D) | (None, 17, 17, 128) | 147584 |
| max_pooling2d_2 (MaxPooling2 | (None, 8, 8, 128) | 0 |
| dropout_1 (Dropout) | (None, 8, 8, 128) | 0 |
| conv2d_5 (Conv2D) | (None, 6, 6, 128) | 147584 |
| conv2d_6 (Conv2D) | (None, 4, 4, 128) | 147584 |
| max_pooling2d_3 (MaxPooling2 | (None, 2, 2, 128) | 0 |
| flatten_1 (Flatten) | (None, 512) | 0 |
| dense_1 (Dense) | (None, 1024) | 525312 |
| dropout_2 (Dropout) | (None, 1024) | 0 |
| dense_2 (Dense) | (None, 7) | 7175 |

Total params: 1,271,687
Trainable params: 1,271,687
Non-trainable params: 0

**Figure 2.21: Illustration of CNN Architecture**

### 2.5.3 Result

six CNNs layer architecture performed competitively and achieved a FER2013 test accuracy of 61.7%, without involving any pre-processing or feature extraction techniques. The state-of-the-art test accuracy for the seven emotion categories using the ensemble of CNNs was 75.2%.

## 2.6 Face Expression Recognition Based on Convolutional Neural Network [10]

Facial expression indicates human emotion in communication. Facial expression recognition, as the key technology of the emotional computing system, is not only used to human-computer interaction, but also extended to the field of interactive game platforms, safe driving, smart recommendation and auxiliary medical care, etc. It is showing a good application prospect in various fields. In this paper, a novel method is proposed based on convolutional neural network (CNN) in order to reduce the complexity for extracting artificial features from the face image in facial expression recognition (FER).

The FER usually includes three steps: the face detection, the feature extraction and the expression classification. Among them, the feature extraction of facial expression plays a key role in the expression recognition system, which affects the recognition accuracy.In the field of machine learning, data-driven feature learning algorithm has been proposed that is first to use deep learning to extract features. Deep learning can automatically extract facial features without human participation. The essential features can be characterized autonomously from the sample data by the multilayered deep neural network.CNN is a deep neural network containing input layer, convolutional layers, pooling layers, fully connected layer and output layer. The CNN contains basic convolution operations and pooling operations. In CNN, the subblocks (local receptive areas) are the input with respect to the lowest layer of the hierarchical structure in the image, and the information is transmitted to different layers through layer by layer. The most significant feature of the observed data is obtained by a digital filter in every layer.

This paper presents a facial expression recognition method based on CNN. Firstly, face detection and preprocessing are performed on the facial expression image. After that, the features of the facial expression are extracted using some trainable convolution kernels. Then, the extracted face expression feature is reduced by the maximum pooling method. Finally, the Softmax classifier is used to categorize facial expression images, and facial expressions are classified into seven types of facial expression: happiness, surprise, sadness, anger, fear, disgust and neutrality.

### 2.6.1 CNN model Structure Design

CNN is a feed-forward neural network that extracts features from a two-dimension image and uses a back-propagation algorithm to optimize the network parameters. The mode consists of a 7-layer network: 3 convolutional layers (C1, C2 and C3), 2 layered pooling layers (P1

and P2), 1 full-connected layer, and 1 Softmax layer. The input is a 48x48 face pixel matrix. The convolutional layer and the pooling layer have several feature maps. Each feature map is connected to the previous layer with a partially connected manner. Convolution layers C1, C2, and C3 operate with 64, 64, and 128 convolution kernels, respectively, where the convolution kernels of C1 and C2 have a size of 5x5, and the convolution kernel of C3 has a size of 4x4. The pooling layers P1 and P2 use a window size of 2x2. There are 4608 neurons in the fully connected layer, which is fully connected to the pooling layer P2. The Softmax layer contains seven neurons and classifies the features of the fully connected layer, and the facial expressions are classified into seven types of facial expressions: happiness, surprise, sadness, anger, fear, disgust, and neutrality.

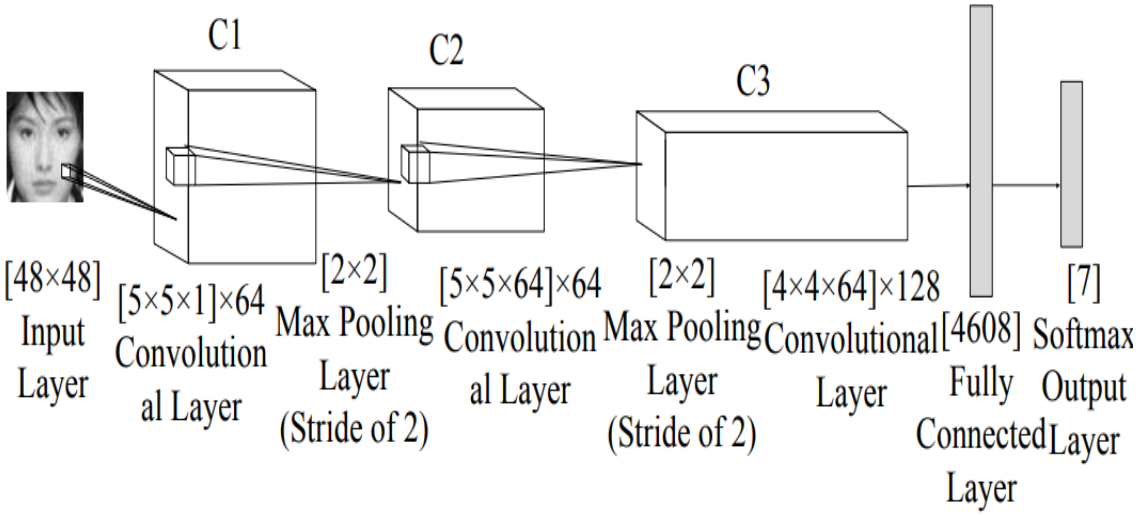



**Figure 2.22: CNN structure for face expression recognition**

1. **Convolution Layer**

   The statistical characteristics of different subblocks in the natural image are usually consistent, which means that the features learned from a certain subblock of the image can be used as a detector. All the subblocks of the full image are traversed to obtain the activation value with the same feature. Different trainable convolution kernels are used to perform convolutional summation over all the feature maps with respect to the previous layer and add offsets. Then the neuron in the current layer is output by the activation function.The convolutional layer is a feature extraction layer. The input of each neuron is connected with the local receptive field in the previous layer, and the local features are extracted from the previous layer.

2. **Pooling Layer**

   If all the features are used to train the Softmax classifier, it will cause enormous dimensions. In order to avoid this problem, a pooled layer is usually used to reduce the feature dimension. The pooling layer plays a role of down-sampling. The pooling layer does not change the number of feature maps, but shrink the feature maps. That means the pooling layer can robust some operation such as translation, scaling, and rotation.

3. **Fully Connected Layer**

   The input of the fully connected layer is a one-dimensional array. The feature map in convolutional layer C3 is a two-dimensional array that is output by the pooled layer P2. The neuron in feature map is corresponding to a point in fully connected layer. Then 128 one-dimensional arrays are concatenated into a-dimensional feature vector in the fully connected layer.

4. **Softmax Layer**

   The Softmax classifier is used in the last layer of CNN, and it is a multi-output competitive classifier. Inputting a given sample, each neuron will output a rate between 0 and 1. The facial expression can be categorized according to the rate.

### 2.6.2 CNN parameter training

The training of CNNs is divided into two stages:

1. **Forward training :** A sample x is extracted from the training sample set, and its corresponding class label is y , x is input to the CNN network, and the output of the upper layer is the input of the current layer. The output of the current layer is calculated through the activation function and passed downed layer by layer. Finally, the output of the Softmax layer is y with a 7-dimensional vector, which represents the probability to classify x into each type of facial expressions.

2. **Back propagation :** The error between the output y of the Softmax layer and the class label vector y is calculated. It is propagated back. Only the element corresponding to the category label y is 1, and the other elements are 0 in y which is a 7- dimensional vector. The weight parameter is adjusted by minimizing the mean square error cost function.

# CHAPTER 3

# PROBLEM STATEMENT

The project "bhavas classification using deep learning" aims at helping the non trained dancers to easily understand the bhavas of classical dance which are ‘Sringara’(Love), ‘Hasya’ (Comic), ‘Karuna’(Compassion), ‘Raudra’(Anger), ‘Veera’(Valor), ‘Bhayankara’(Fear), ‘Bhibhatsa’(Disgust), ‘Adbhuta’(Wonder), ‘Shanta’(Tranquility). The normal human facial expressions like happy, angry, fear, sad, disgust and surprise are comparatively easier to understand. But the nine bhavas are difficult to understand by a normal non trained dancer. Because some emotions like ’raudra’ and ’bhayankara’ may appear same for a normal person who watches these facial expressions. But their meanings are entirely different. Without knowing the intended meaning of the nine bhavas, a person cannot enjoy the dance in its full extent. There are no systems existing to classify human emotions in terms of ’navarasams’ in classical dance. So the proposed system is very useful for a person to understand how each bhava will look like and will help the person to learn the bhavas of classical dance.

# CHAPTER 4
# PROJECT MANAGEMENT

## 4.1 Introduction

Project management is the discipline of planning, organizing, securing, managing, leading, and controlling resources to achieve specific goals. A project is a temporary endeavor with a defined beginning and end (usually time-constrained, and often constrained by funding or deliverables), undertaken to meet unique goals and objectives, typically to bring about beneficial change or added value. The temporary nature of projects stands in contrast with business as usual (or operations), which are repetitive, permanent, or semi-permanent functional activities to produce products or services. In practice, the management of these two systems is often quite different, and as such requires the development of distinct technical skills and management strategies.

In our project we are following the typical development phases of an engineering project

1. Initiation
2. Planning and Design
3. Execution and Construction
4. Monitoring and Controlling Systems
5. Completion

### 4.1.1 Initiation

The initiating processes determine the nature and scope of the project. The initiating stage should include a plan that encompasses the following areas :

1. Analysing the business needs/requirements in measurable goals
2. Reviewing of the current operations
3. Financial analysis of the costs and benefits including a budget
4. Stakeholder analysis, including users, and support personal for the project

5. Project charter including costs, tasks, deliverables, and schedule

### 4.1.2 Planing and design

After the initiation stage, the project is planned to an appropriate level of detail (see example of a flow-chart). The main purpose is to plan time, cost and resources adequately to estimate the work needed and to effectively manage risk during project execution. As with the initiation process, a failure to adequately plan greatly reduces the project's chances of successfully accomplishing its goals.

- Determining how to plan
- Developing the scope statement
- Selecting the planning team
- Identifying deliverables and creating the work breakdown structure
- Identifying the activities needed to complete those deliverables
- Developing the schedule
- Risk planning

### 4.1.3 Execution

Executing consists of the processes used to complete the work defined in the project plan to accomplish the project's requirements. The execution process involves coordinating people and resources, as well as integrating and performing the activities of the project in accordance with the project management plan. The deliverables are produced as outputs from the processes performed as defined in the project management plan and other frameworks that might be applicable to the type of project at hand.

### 4.1.4 Monitoring & controlling

Monitoring and controlling consists of those processes performed to observe project execution so that potential problems can be identified in a timely manner and corrective action can be taken, when necessary, to control the execution of the project. The key benefit is that project performance is observed and measured regularly to identify variances from the project management plan.

## 4.2 System Development Life Cycle

The Systems development life cycle (SDLC), or Software development process in systems engineering, information systems, and software engineering, is a process of creating or altering information systems, and the models and methodologies that people use to develop these systems. In software engineering, the SDLC concept underpins many kinds of software development methodologies. These methodologies form the framework for planning and controlling the creation of an information system.

The SDLC phases serve as a programmatic guide to project activity and provide a flexible but consistent way to conduct projects to a depth matching the scope of the project. Each of the SDLC phase objectives is described in this section with key deliverables, a description of recommended tasks, and a summary of related control objectives for effective management. The project manager must establish and monitor control objectives during each SDLC phase while executing projects. Control objectives help to provide a clear statement of the desired result or purpose and should be used throughout the entire SDLC process.

### 4.2.1 Spiral Model

We have used the Spiral model in our project. The Spiral model incorporates the best characteristics of both- waterfall and prototyping model. In addition, the Spiral model also contains a new component called Risk Analysis, which is not there in the waterfall and prototype model. In the Spiral model, the basic structure of the software product is developed first. After the basic structure is developed, new features such as user interface and data administration are added to the existing software product. This functionality of the Spiral model is similar to a spiral where the circles of the spiral increase in diameter. Each circle represents a more complete version of the software product. The spiral is a risk-reduction oriented model that breaks a software project up into main projects, each addressing one or major risks. After major risks have been addressed the spiral model terminates as a waterfall model. Spiral iteration involves six steps:

1. Determine objectives, alternatives and constraints.
2. Identify and resolve risks.
3. Evaluate alternatives.
4. Develop the deliverables for the iteration and verify that they are correct.
5. Plan the next iteration.

6. Commit to an approach for the next iteration.

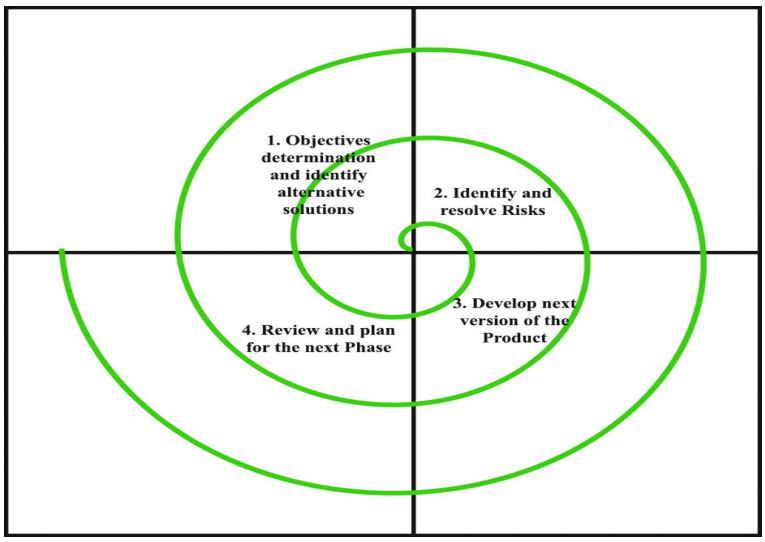



**Figure 4.1: Spiral Model**

# CHAPTER 5
# METHODOLOGY

## 5.1 System Requirements & Specifications

### 5.1.1 Windows 10

Windows 10 is a series of personal computer operating systems produced by Microsoft as part of its Windows NT family of operating systems. It is the successor to Windows 8.1 and was released to manufacturing on July 15, 2015, and to retail on July 29, 2015. Windows 10 receives new builds on an ongoing basis, which are available at no additional cost to users. Mainstream builds of Windows 10 are labeled version YYMM with YY representing the year and MM representing the month of release. For example, the latest mainstream build of Windows 10 is Version 1809. There are additional test builds of Windows 10 available to Windows Insiders. Devices in enterprise environments can receive these updates at a slower pace, or use long-term support milestones that only receive critical updates, such as security patches, over their ten-year lifespan of extended support.

### 5.1.2 Pandas

In computer programming, pandas is a software library written for the Python programming language for data manipulation and analysis. In particular, it offers data structures and operations for manipulating numerical tables and time series. The name is derived from the term "panel data", in econometrics term for data sets that include observations over multiple periods for the same individuals. Pandas is an open-source, BSD-licensed Python library providing high-performance, easy-to-use data structures and data analysis tools for the Python programming language. Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, statistics, analytics, etc

### 5.1.3 Google Colaboratory

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.

### 5.1.4 Keras

Keras is an open-source software library that provides a Python interface for artificial neural networks. Keras acts as an interface for the TensorFlow library. Up until version 2.3 Keras supported multiple backends, including TensorFlow, Microsoft Cognitive Toolkit, Theano, and PlaidML. As of version 2.4, only TensorFlow is supported. Designed to enable fast experimentation with deep neural networks, it focuses on being user-friendly, modular, and extensible. It was developed as part of the research effort of project ONEIROS (Open-ended Neuro-Electronic Intelligent Robot Operating System), and its primary author and maintainer is François Chollet, a Google engineer. Chollet is also the author of the XCeption deep neural network model.

### 5.1.5 TensorFlow

TensorFlow is an end-to-end open source platform for machine learning. It has a comprehensive, flexible ecosystem of tools, libraries, and community resources that lets researchers push the state-of-the-art in ML and developers easily build and deploy ML-powered applications. TensorFlow was originally developed by researchers and engineers working on the Google Brain team within Google's Machine Intelligence Research organization to conduct machine learning and deep neural networks research. The system is general enough to be applicable in a wide variety of other domains, as well. TensorFlow provides stable Python and C++ APIs, as well as non-guaranteed backward compatible API for other languages. Keep up-to-date with release announcements and security updates by subscribing to announce@tensorflow.org.

## 5.2 Proposed System

The proposed system consists of CNN networks. The CNN for segmenting features of face such as eye, lip and eyebrows. Then classifying the facial expression into bhavas. So the proposed system is divided into three modules which are dataset creation, data preprocessing, and classification.

### 5.2.1 Dataset Creation

The first step in this project is to create a dataset. The dataset consists of 5 subfolders which are 'veeram', 'karunam', 'bhayanakam', 'bheebhalsam' and 'others' which represents the 5 classes. The 'veeram' folder consists of 778 augmented images of 'veeram'. The 'karunam' folder has 580 augmented images of 'karunam'. The 'bhayanakam' folder consists of 720 augmented images of 'bhayanakam'. The 'bheebhalsam' folder consists of 620 augmented images of 'bheeebhalsam'. The 'others' folder has 1966 augmented images of 'others'. There are a total of 4664 images in dataset.

### 5.2.2 Data Preprocessing Module

Data preprocessing is an important task. This step transforms the raw sourced data into a format that enables successful model training. Data preprocessing involves various steps but not limited to data reformatting, data cleaning, augmentation and data normalisation. There are 5 steps used in the proposed system they are image reformatting, feature Extraction, normalisation and augmentation.

In image reformatting, all the images are re-scaled into a 150x150 fixed size. The extraction of facial features requires translating the input data into a set of features the features are eye, lip and eyebrows. The segmented image of eye will be obtained which will be added to an empty matrix of size having same as that of component segmented image. This process is repeated for lip and eyebrows. Thus intermediate mask is obtained. Thus final mask is obtained for all segments separately. The component segmented image and final mask image gives final iconized image. Data normalisation involves two steps which are normalizing data per image and then normalising data per pixel. In normalising data per image, from each image the mean value over that image is subtracted and the result is divided by an appropriate standard deviation. The mean image over the training set is computed and for each training image, its mean value is subtracted from each image. The result thus obtained is divided by an appropriate standard deviation. In data augmentation step, the images are flipped and rotated. It will help

to increase the number of images and also plays a role in regularisation. The training masks are generate by detecting and localizing facial landmarks on a face image. In augmentation , images are flipped,rotate,zoomed and shared. Then each images are duplicates in 20 images, which helps to increase the number of images. Normalization step is involves the normalising the data per pixel.

### 5.2.3 Classification Module

Classified of the facial expression into the corresponding bhavas. Then pre trained EfficientNetB5 model is used, 5 Dense layer and one softmax layer will be used for classification into bhavas.

**EfficientNetB5:**One of the EfficientNet models designed to perform image classification. This model was pretrained in TensorFlow. All the EfficientNet models have been pretrained on the ImageNet image database.[1]

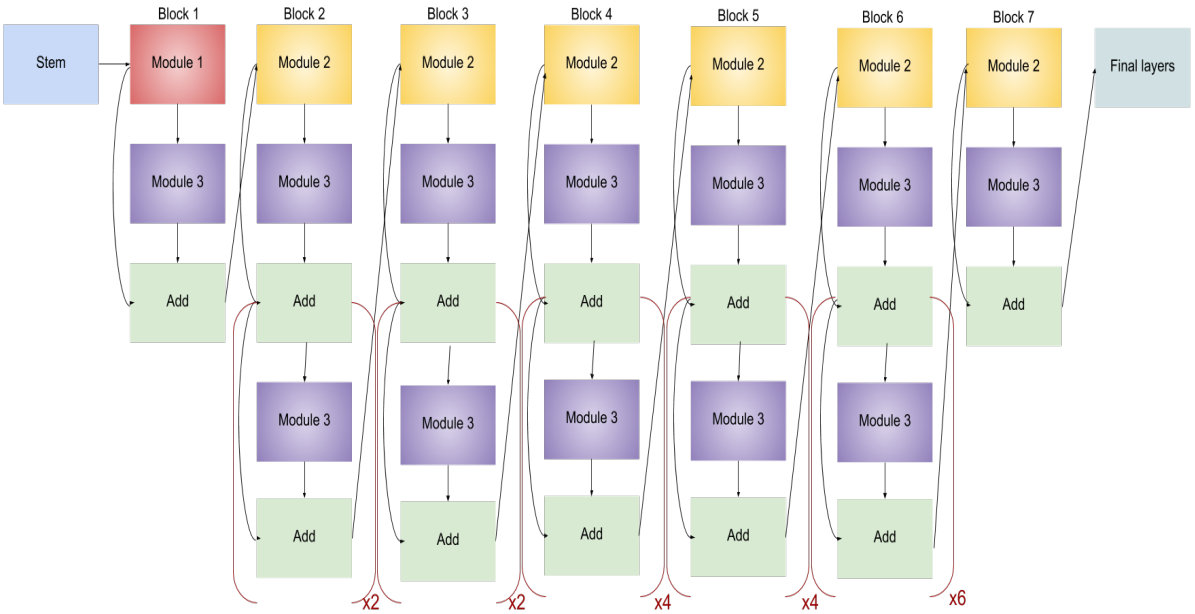



**Figure 5.1: EfficientNetB5 layers**

**Dense Layer:** The dense layer is a neural network layer that is connected deeply, which means each neuron in the dense layer receives input from all neurons of its previous layer. The dense layer is found to be the most commonly used layer in the models. In the background, the dense layer performs a matrix-vector multiplication. The values used in the matrix are actually

parameters that can be trained and updated with the help of back propagation. The output generated by the dense layer is an 'm' dimensional vector. Thus, dense layer is basically used for changing the dimensions of the vector. Dense layers also applies operations like rotation, scaling, translation on the vector.

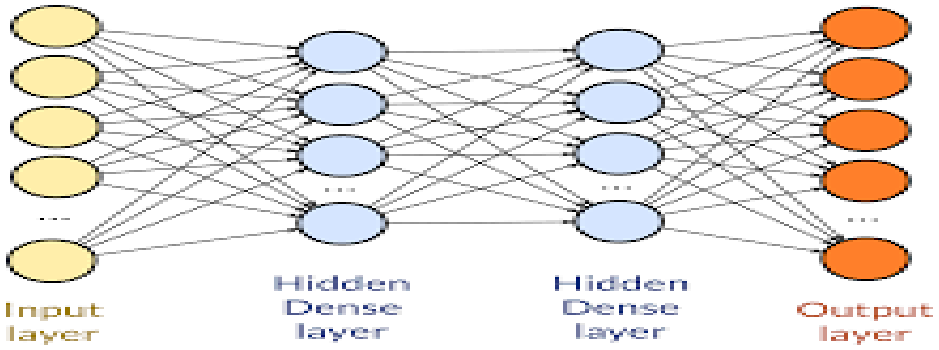



**Figure 5.2: Dense layers**

**Softmax Layer:** The softmax function, also known as softargmax or normalized exponential function,is a generalization of the logistic function to multiple dimensions. It is used in multinomial logistic regression and is often used as the last activation function of a neural network to normalize the output of a network to a probability distribution over predicted output classes, based on Luce's choice axiom.

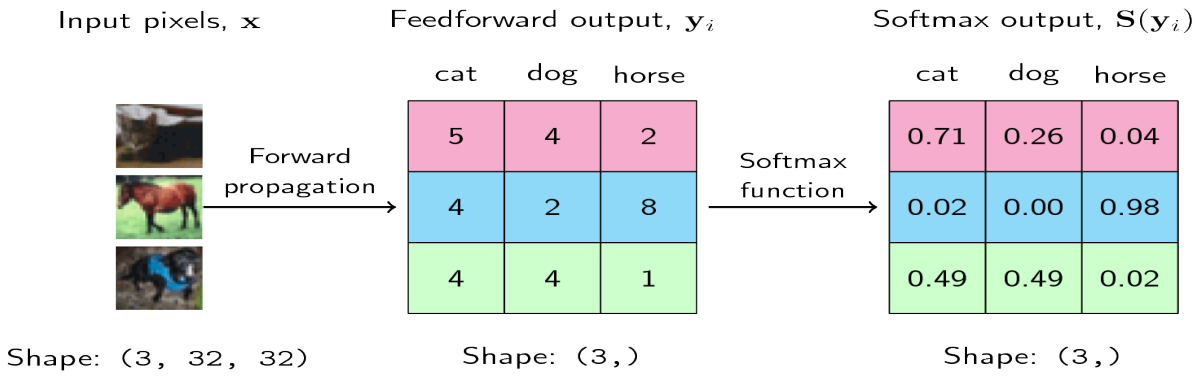



**Figure 5.3: Softmax layer**

## NEURAL NETWORK IDENTIFIED

Neural network identified for classification is Convolutional Neural Network (CNN).

## ALGORITHM USED: CNN

Convolutional Neural Networks or covnets are neural networks that share their parameters. Imagine an image which can be represented as a cuboid having its length, width (dimension of the image) and height (as image generally have red, green, and blue channels).

A covnet is a sequence of layers, and every layer transforms one volume to another through differentiable function.

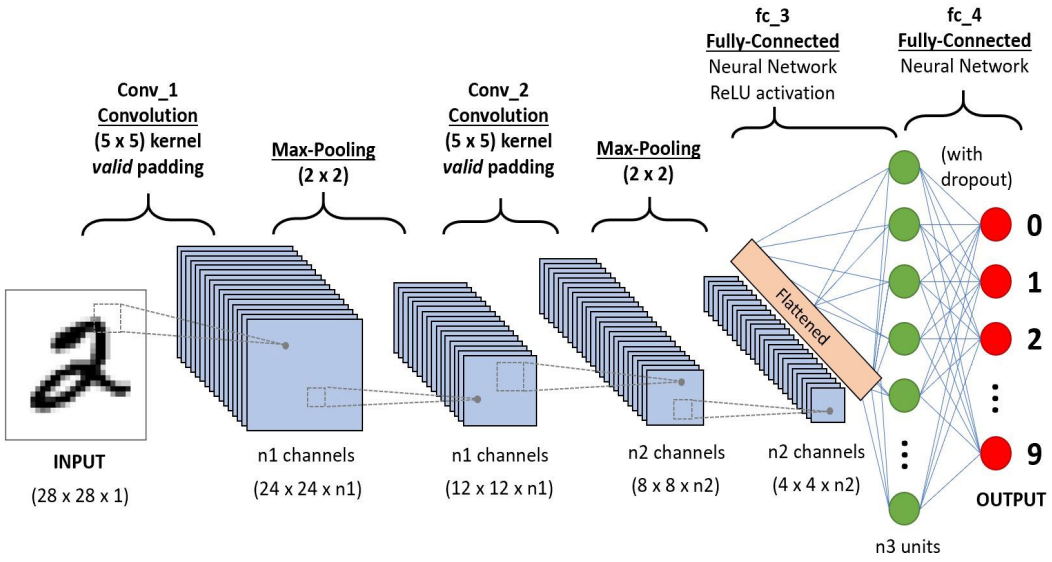



**Figure 5.4: Convolutional Neural Network**

Types of layers in CNN:

Consider a convnet on an image of dimension 32 x 32 x 3.

1. **Input Layer**: This layer holds the raw input of image with width 32, height 32 and depth 3.

2. **Convolution Layer**: This layer computes the output volume by computing dot product between all filters and image patch. Suppose we use total 12 filters for this layer we'll get output volume of dimension 32 x 32 x 12.

3. **Activation Function Layer**: This layer will apply element wise activation function to the output of convolution layer. Some common activation functions are RELU: max(0,

x), Sigmoid: 1/(1+e^-x), Tanh, etc.

4. **Pool Layer**: This layer is periodically inserted in the covnets and its main function is to reduce the size of volume which makes the computation fast reduces memory and also prevents from overfitting. Two common types of pooling layers are max pooling and average pooling. If we use a max pool with 2 x 2 filters and stride 2, the resultant volume will be of dimension 16x16x12.

5. **Fully-Connected Layer**: This layer is regular neural network layer which takes input from the previous layer and computes the class scores and outputs the 1-D array of size equal to the number of classes.

6. **Softmax Layer:** The Softmax Function is a generalization of the Logistic Function, and it makes sure that our prediction add up to 1. Most of the time the Softmax Function is related to the Cross Entropy Function. In CNN, after the application of the Softmax Function, is to test the reliability of the model using as Loss Function the Cross Entropy Function, in order to maximize the performance of our neural network. There are several advantages to using the Cross Entropy Function. One of the best is that if for instance at the start of backpropagation the output value is much smaller than the actual value, the gradient descent will be very slow. Because Cross Entropy uses the logarithm, it helps the network to assess even large errors.[2]

## 5.3 Data Flow Diagrams

### 5.3.1 Data Flow Diagram- Level 0

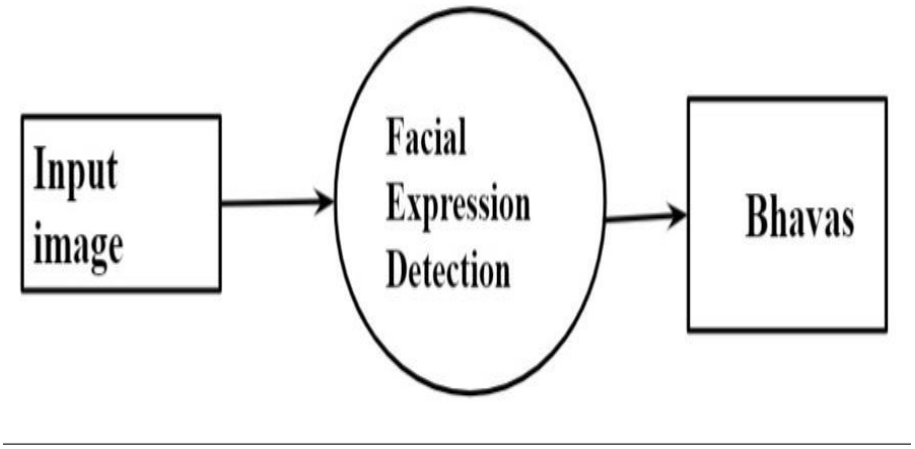



**Figure 5.5: DFD- Level 0**

### 5.3.2 Data Flow Diagram- Level 1

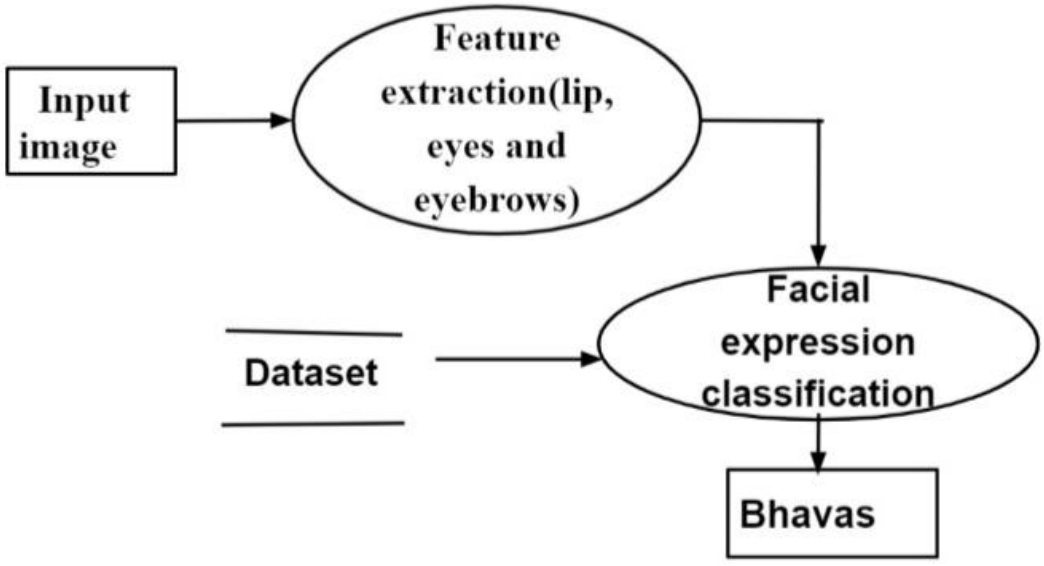



**Figure 5.6: DFD- Level 1**

## 5.4 Architecture

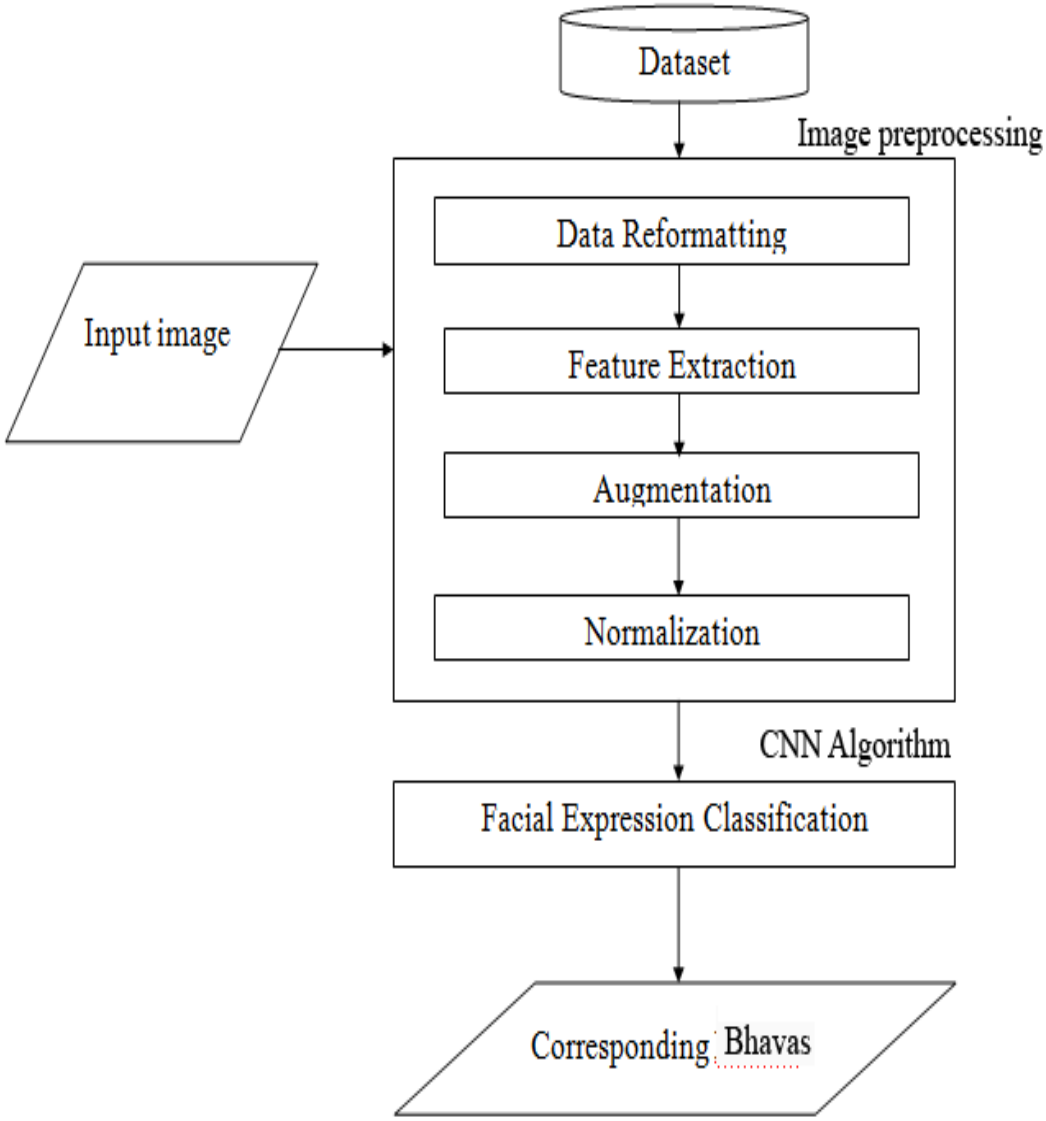



**Figure 5.7: Architecture**

## 5.5 Implementation

### 5.5.1 Libraries

The libraries imported are following:

- numpy
- dlib
- cv2
- matplotlib
- keras
- google.colab.patches
- PIL
- os
- pandas
- skimage
- sklearn
- tensorflow

### 5.5.2 Resizing of images

All the images are resized to 150x150 dimension and to 'png' format.

```
source_folder = '/content/drive/MyDrive/project/new/others/'
destination_folder = '/content/drive/MyDrive/project/new_size/others/'
directory = os.listdir(source_folder)
new_image_width=150
new_image_height=150
for item in directory:
  img = Image.open(source_folder +'/'+ item)
  imgResize = img.resize((new_image_width, new_image_height), Image.ANTIALIAS)
```

```
  imgResize.save(destination_folder + item[:-4] +'.png', quality = 90)
```

### 5.5.3 Downloading landmark detector

```
!wget   http://dlib.net/files/shape_predictor_68_face_landmarks.dat.bz2

!bunzip2 /content/shape_predictor_68_face_landmarks.dat.bz2

datFile =  "/content/shape_predictor_68_face_landmarks.dat"
```

### 5.5.4 Landmark detection, segmentation, and augmentation

In Landmark detection, all the important features like eye, eyebrow, nose, lip, face are detected. In segmentation, only the required features such as eye, eyebrow, lip are segmented from the background. In augmentation, each image in the dataset are flipped, rotated, zoomed etc. Thus from each image, 20 images are obtained. For augmentation, ImageDataGenerator is used.

```
  datagen=ImageDataGenerator(
  rotation_range=30,
  shear_range=0.2,
  zoom_range=0.2,
  width_shift_range=0.2,
  height_shift_range=0.2,
  horizontal_flip=True,
  fill_mode='nearest')


  source_folder = '/content/drive/MyDrive/project/new_size/'
  destination_folder = '/content/drive/MyDrive/project/augmented/'
  for i in l:
    new_s=source_folder+i
    new_d=destination_folder+i+'/'
    directory=os.listdir(new_s)
    print(directory)
    for item in directory:
      img=cv2.imread(new_s+'/'+item)
      grayimg=cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
      predictor=dlib.shape_predictor("shape_predictor_68_face_landmarks.dat")
```

```
detector=dlib.get_frontal_face_detector()
face=detector(grayimg)    #rectangles[[(65, 77) (169, 181)]]

for points in face:
  x1,y1=points.left(),points.top()
  x2,y2=points.right(),points.bottom()
  landmarks=predictor(grayimg,points)
  mypoints=[]
  for n in range(68):
    x=landmarks.part(n).x
    y=landmarks.part(n).y
    mypoints.append([x,y])
    #cv2.circle(grayimg,(x,y),1,(255,255,255),cv2.FILLED)
  mypoints=np.array(mypoints)
    #plt.imshow(grayimg,cmap='gray')
mask=np.zeros_like(grayimg)
lefteye=mypoints[36:42]
lefteyemask=cv2.fillPoly(mask,[lefteye],(255,255,255))
righteye=mypoints[42:48]
righteyemask=cv2.fillPoly(mask,[righteye],(255,255,255))
lefteyebrow=mypoints[17:22]
lefteyebrowmask=cv2.fillPoly(mask,[lefteyebrow],(255,255,255))
righteyebrow=mypoints[22:27]
righteyebrowmask=cv2.fillPoly(mask,[righteyebrow],(255,255,255))
lip=mypoints[48:61]
lipmask=cv2.fillPoly(mask,[lip],(255,255,255))
x=img_to_array(mask)
x = x.reshape((1,) + x.shape)
count = 0
for batch in datagen.flow(
x, batch_size=1,
save_to_dir=new_d,
save_prefix=item,
save_format='png'):
  count=count+1
  if(count==20):
    break
```

### 5.5.5 Creating dataframe for image and class

A Pandas DataFrame is a 2 dimensional data structure, like a 2 dimensional array, or a table with rows and columns. For creating dataframe, two lists of images and labels are created and the images and their class numbers are appended to the respective classes. And then a dataframe is created using the same.

```
images=[]
labels=[]
#appending to list
c=0
source_folder = '/content/drive/MyDrive/project/augmented/'
directory = os.listdir(source_folder)
print(directory)
for i in directory:
  print(i)
  items=os.listdir(source_folder+i+'/')
  for j in items:
    images.append(cv2.imread(source_folder+i+'/'+j))
    labels.append(c)
  c=c+1
table=df()
table['image']=images
table['class']=labels
```

### 5.5.6 Reshaping and converting image to binary image

The dataframe is given column names of image and class. And then the images are reshaped to be suitable for training and are then the entire array of pixels of images are divided by 255 to make them binary images.

```
X=table['image']
Y=table['class']


x=[]
for i in X:
  x.append(i)
x=np.array(x)
```

```
x=x/255
```

### 5.5.7 Splitting of dataset and reshaping of label

The images and their respective labels in the dataset are splitted using train_test_split function. The test_size is given a value of 0.10 which means 10% of the images in dataset will be used for testing and the remaining 90% of the images will be used for training. Then the labels are one-hot encoded using to_categorical function.

```
X_train,X_test,Y_train,Y_test=train_test_split(x,Y,test_size=0.10,
shuffle=True,random_state=1,stratify=Y)

Y_train=to_categorical(Y_train)
Y_test=to_categorical(Y_test)
```

### 5.5.8 Creating Model

The model is created using a pretrained EfficientNetB5 model. On top of it, 5 dense layers are used and finally, a softmax layer is used for classification.

```
!pip install keras_efficientnets
from tensorflow.keras.applications import EfficientNetB5
base_model=EfficientNetB5(include_top=False, weights="imagenet", input_shape=(15
model= Sequential()
model.add(base_model)
model.add(Flatten())
model.add(Dense(1024,activation=('relu'),input_dim=512))

model.add(Dense(512,activation=('relu')))
model.add(Dense(256,activation=('relu')))

model.add(Dense(128,activation=('relu')))

model.add(Dense(5,activation=('softmax')))
```

### 5.5.9 Compiling, training and testing

The model is compiled and foor that 'categorical_crossentropy' is used as the loss function as the system has a multiclass classification. The optimizer used is 'adam' optimizer. And accuracy is used as the metrics. For training, an accuracy of 96.49% is obtained and for testing, and an accuracy of 86.51% is obtained at an epoch of 35.

```
model.compile(loss='categorical_crossentropy',
              optimizer='adam',
              metrics=['accuracy'])
history=model.fit(X_train,Y_train,epochs=35,validation_data=(X_test,Y_test))
loss, acc = model.evaluate(X_test, Y_test, verbose=2)
print("model, accuracy: {:5.2f}%".format(100 * acc))
model.predict(X_test)
```

```
Epoch 28/35
132/132 [==============================] - 73s 552ms/step - loss: 0.1079 - accuracy: 0.9689 - val_loss: 2.1093 - val_accuracy: 0.6531
Epoch 29/35
132/132 [==============================] - 73s 555ms/step - loss: 0.5308 - accuracy: 0.8378 - val_loss: 1.4865 - val_accuracy: 0.4004
Epoch 30/35
132/132 [==============================] - 73s 554ms/step - loss: 0.2386 - accuracy: 0.9253 - val_loss: 438.6175 - val_accuracy: 0.1242
Epoch 31/35
132/132 [==============================] - 73s 554ms/step - loss: 0.0858 - accuracy: 0.9751 - val_loss: 0.7479 - val_accuracy: 0.7944
Epoch 32/35
132/132 [==============================] - 74s 561ms/step - loss: 0.0486 - accuracy: 0.9851 - val_loss: 1.8499 - val_accuracy: 0.5610
Epoch 33/35
132/132 [==============================] - 73s 553ms/step - loss: 0.0857 - accuracy: 0.9778 - val_loss: 2.2539 - val_accuracy: 0.4218
Epoch 34/35
132/132 [==============================] - 73s 554ms/step - loss: 0.0604 - accuracy: 0.9790 - val_loss: 1.3795 - val_accuracy: 0.5289
Epoch 35/35
132/132 [==============================] - 73s 554ms/step - loss: 0.0520 - accuracy: 0.9849 - val_loss: 0.4799 - val_accuracy: 0.8651
```

**Figure 5.8: Training and Testing**

# CHAPTER 6

# RESULTS

## 1. Converting RGB image to grayscale image

**Figure 6.1: Resized color image of Karunam**

**Figure 6.2: Grayscale image of Karunam**

## 2. Detecting features of face

**Figure 6.3: Detecting features**

### 3. Segmenting eyebrow, eye, lip

**Figure 6.4: After segmentation**

### 4. Model prediction

```
img=np.array(img)
```

```
img=img.reshape((1,)+img.shape)
```

```
pred=model.predict(img)
```

```
print(pred[0])
```

```
[0.0000000e+00 1.0000000e+00 1.3168116e-11 0.0000000e+00 0.0000000e+00]
```

```
print(d[np.argmax(pred[0])])
```

```
karunam
```

**Figure 6.5: Model Prediction**

## 5. Accuracy and loss

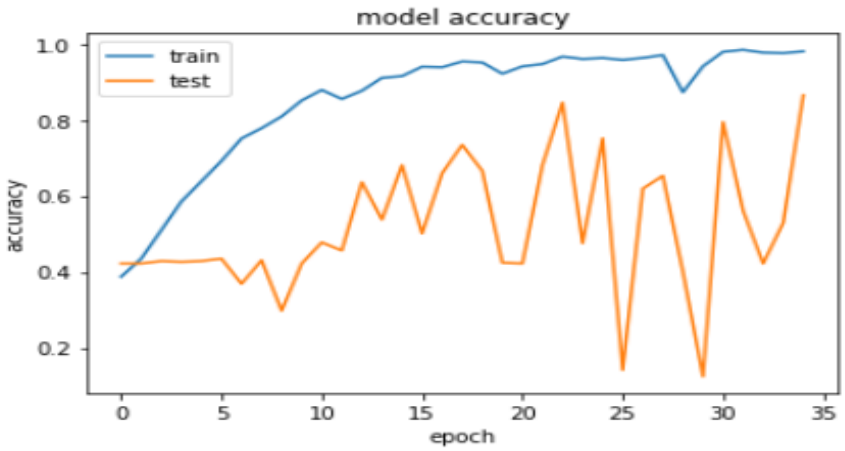



**Figure 6.6: Accuracy**

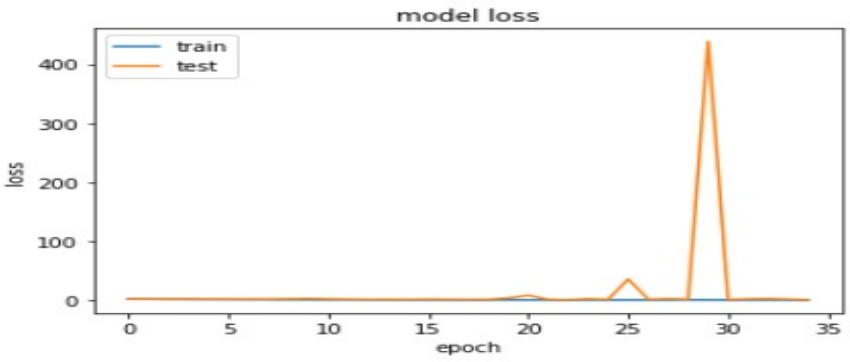



**Figure 6.7: Loss**

## 6. Confusion Matrix

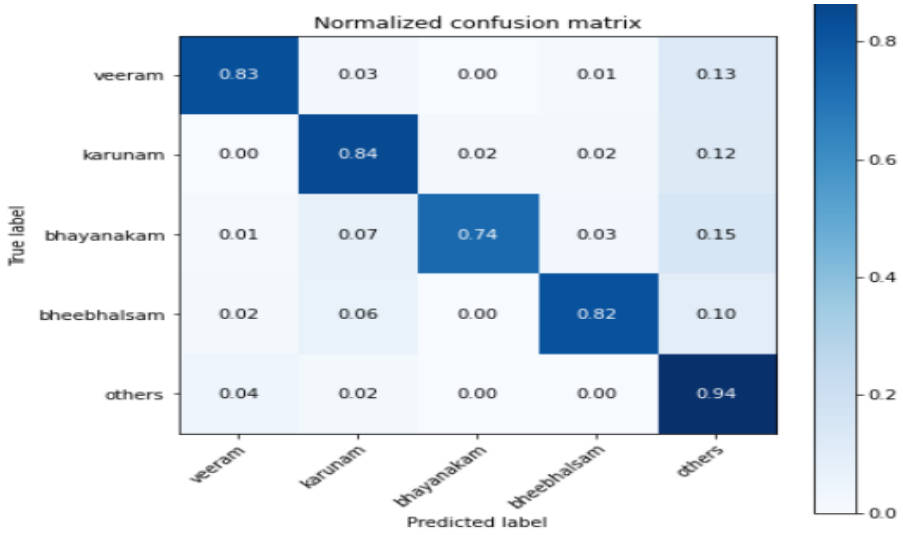



**Figure 6.8: Confusion Matrix**

# CHAPTER 7
# CONCLUSION & FUTURE SCOPE

## 7.1 Conclusion

There are many projects that use facial expression recognition for detecting emotions , neurological disorders etc. Our project aims to classify bhavas according to the image that has been given as input. Until now, there is no other system used for this purpose. So the scope of the system is high. This application allows the normal users (other than dancers) to easily understand the bhavas.

Deep learning techniques have been widely used in many applications such as healthcare, big data analysis, etc. We use the deep learning techniques to classify facial expressions into 5 classes using CNN such as 'Karuna'(Compassion), 'Veera'(Valor), 'Bhayankara'(Fear), 'BhiBhatsa'(Disgust),'Others'('Sringara'(Love),'Hasys'(Comic),'Raudra'(Anger), 'Adbhuta' (Wonder),'Shantam' (Tranquility)). The steps involved in this system are creating dataset, preprocessing of images, extracting features from images, identifying and classifying the bhavas, training and testing ,and finally accuracy is calculated. Our system is helpful for dance beginners and they can easily understand each bhavas. It can be used to detect mental health disorder like depression. It is helpful for measuring customer satisfaction and it can be used for understanding mood of humans.

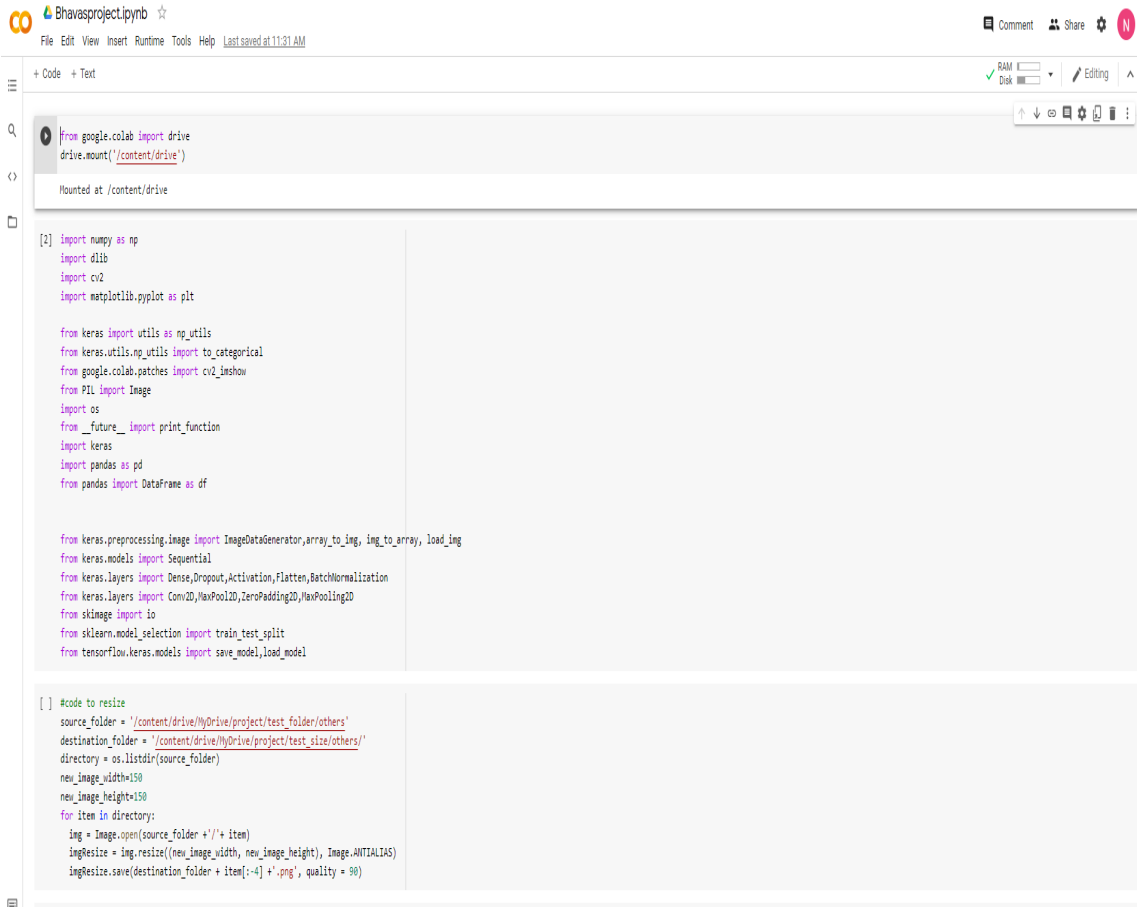
## 7.2 Future Scope

In this system, 4 emotions are separately considered and the rest of the 5 emotions are considered as a single class. In future, we are planning to classify all the nine emotions separately and also to predict bhavas of images from classical dance related videos.

# CHAPTER 8

# APPENDIX

SCREENSHOTS-Google Colaboratory




```
[ ] img=cv2.imread('/content/drive/MyDrive/project/test_size/others/others..png',cv2.IMREAD_COLOR)
    #print(img)
```

```
[ ] print(img)
```

```
[ ] plt.imshow(cv2.cvtColor(img,cv2.COLOR_BGR2RGB))
```

```
[ ] grayimg=cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
    plt.imshow(grayimg,cmap='gray')
    #print(grayimg)
```

```
[3] #landmark predictor download
    !wget   http://dlib.net/files/shape_predictor_68_face_landmarks.dat.bz2 # DOWNLOAD LINK

    !bunzip2 /content/shape_predictor_68_face_landmarks.dat.bz2

    datFile =  "/content/shape_predictor_68_face_landmarks.dat"
```

```
--2021-06-06 04:32:16--  http://dlib.net/files/shape_predictor_68_face_landmarks.dat.bz2
Resolving dlib.net (dlib.net)... 107.180.26.78
Connecting to dlib.net (dlib.net)|107.180.26.78|:80... connected.
HTTP request sent, awaiting response... 200 OK
Length: 64040097 (61M)
Saving to: 'shape_predictor_68_face_landmarks.dat.bz2'

shape_predictor_68_ 100%[===================>]  61.07M  75.9MB/s    in 0.8s

2021-06-06 04:32:17 (75.9 MB/s) - 'shape_predictor_68_face_landmarks.dat.bz2' saved [64040097/64040097]
```

```
[ ] predictor=dlib.shape_predictor("shape_predictor_68_face_landmarks.dat")   #landmark detector
    detector=dlib.get_frontal_face_detector()                                 #face detector
    face=detector(grayimg)    #rectangles[[(65, 77) (169, 181)]]

    for points in face:
      x1,y1=points.left(),points.top()
      x2,y2=points.right(),points.bottom()
      landmarks=predictor(grayimg,points)
      mypoints=[]
      for n in range(68):
        x=landmarks.part(n).x
        y=landmarks.part(n).y
        mypoints.append([x,y])
        cv2.circle(grayimg,(x,y),1,(255,255,255),cv2.FILLED)
      mypoints=np.array(mypoints)
    plt.imshow(grayimg,cmap='gray')
    #print(mypoints)
```

```
[ ] mask=np.zeros_like(grayimg)
    lefteye=mypoints[36:42]
    lefteyemask=cv2.fillPoly(mask,[lefteye],(255,255,255))
    righteye=mypoints[42:48]
    righteyemask=cv2.fillPoly(mask,[righteye],(255,255,255))
    lefteyebrow=mypoints[17:22]
    lefteyebrowmask=cv2.fillPoly(mask,[lefteyebrow],(255,255,255))
    righteyebrow=mypoints[22:27]
    righteyebrowmask=cv2.fillPoly(mask,[righteyebrow],(255,255,255))
    lip=mypoints[48:61]
    lipmask=cv2.fillPoly(mask,[lip],(255,255,255))
    cv2_imshow(mask)
```

```
[ ] l=['veeram','bhayanakam','bheebhalsam','karunam','others']
```

```
[ ] datagen=ImageDataGenerator(
            rotation_range=30,
            shear_range=0.2,
            zoom_range=0.2,
            width_shift_range=0.2,
            height_shift_range=0.2,
            horizontal_flip=True,
            fill_mode='nearest')
```

```
#detection,mask,augmentation
def preprocess():
  source_folder = '/content/drive/MyDrive/project/test_size/'
  destination_folder = '/content/drive/MyDrive/project/aug_test/'
  for i in l:
    new_s=source_folder+i
    new_d=destination_folder+i+'/'
    directory=os.listdir(new_s)
    print(directory)
    for item in directory:
      img=cv2.imread(new_s+'/'+item)
      grayimg=cv2.cvtColor(img,cv2.COLOR_BGR2GRAY)
      predictor=dlib.shape_predictor("shape_predictor_68_face_landmarks.dat")   #landmark detector
      detector=dlib.get_frontal_face_detector()                                 #face detector
      face=detector(grayimg)    #rectangles[[(65, 77) (169, 181)]]

      for points in face:
        x1,y1=points.left(),points.top()
        x2,y2=points.right(),points.bottom()
        landmarks=predictor(grayimg,points)
        mypoints=[]
        for n in range(68):
          x=landmarks.part(n).x
          y=landmarks.part(n).y
          mypoints.append([x,y])
          #cv2.circle(grayimg,(x,y),1,(255,255,255),cv2.FILLED)
        mypoints=np.array(mypoints)
          #plt.imshow(grayimg,cmap='gray')
      mask=np.zeros_like(grayimg)
      lefteye=mypoints[36:42]
      lefteyemask=cv2.fillPoly(mask,[lefteye],(255,255,255))
      righteye=mypoints[42:48]
```

```
[ ] preprocess()
```

```
['veeram..png']
['bhayanakam..png']
['bheebhalsam..png']
['karunam..png']
['others..png']
```

```
[ ] #count images in a folder
source_folder = '/content/drive/MyDrive/project/augmented/'
directory = os.listdir(source_folder)
print(directory)
total=0
for i in directory:
  items=os.listdir(source_folder+i+'/')
  print("number of ",i," images: ",len(items))
  total+=len(items)
print("total number of images in dataset:",total)
```

```
['veeram', 'karunam', 'bhayanakam', 'bheebhalsam', 'others']
number of  veeram  images:  778
number of  karunam  images:  580
number of  bhayanakam  images:  720
number of  bheebhalsam  images:  620
number of  others  images:  1966
total number of images in dataset: 4664
```

```
[ ] #creating list for dataframe
images=[]
labels=[]
```

```
[ ] #appending to list
c=0
source_folder = '/content/drive/MyDrive/project/augmented/'
directory = os.listdir(source_folder)
print(directory)
for i in directory:
  print(i)
  items=os.listdir(source_folder+i+'/')
  for j in items:
    images.append(cv2.imread(source_folder+i+'/'+j))
    labels.append(c)
  c=c+1
```

```
[ ] karunam
    bhayanakam
    bheebhalsam
    others
```

```
[ ] print(labels[779])

    1
```

```
[ ] #creating dataframe
    table=df()
    table['image']=images
    table['class']=labels
```

```
[ ] X=table['image']
    Y=table['class']
```

```
[ ] print(table)
```

```
[ ] X.shape

    (4664,)
```

```
[ ] #to reshape and to convert to binary
    x=[]
    for i in X:
      x.append(i)
    x=np.array(x)
    x=x/255
```

```
[ ] x.shape

    (4664, 150, 150, 3)
```

```
    print(x)
    [[[1]]] in x
```

```
[ ] X_train,X_test,Y_train,Y_test=train_test_split(x,Y,test_size=0.10,shuffle=True,random_state=1,stratify=Y)
```

```
[ ] Y_train.shape

    (4197,)
```

```
[ ] Y_train=to_categorical(Y_train)
    Y_test=to_categorical(Y_test)
```

```
[ ] X_train.shape

    (4893, 150, 150, 3)
```

```
[ ] Y_train.shape

    (4197, 5)
```

```
[ ] len(Y_test)     #number of test images, X_test is also fine
```

```
[ ] from tensorflow.keras.applications import EfficientNetB5
```

```
[ ] base_model=EfficientNetB5(include_top=False, weights="imagenet", input_shape=(150,150,3),classes=5)
```

```
Downloading data from https://storage.googleapis.com/keras-applications/efficientnetb5_notop.h5
115269632/115263384 [==============================] - 2s 0us/step
```

```
[ ] model= Sequential()
    model.add(base_model)
    model.add(Flatten())
    model.add(Dense(1024,activation=('relu'),input_dim=512))

    model.add(Dense(512,activation=('relu')))
    model.add(Dense(256,activation=('relu')))
    #model.add(Dropout(.3))
    model.add(Dense(128,activation=('relu')))
    #model.add(Dropout(.2))
    model.add(Dense(5,activation=('softmax')))
```

```
[ ] model.compile(loss='categorical_crossentropy',
                  optimizer='adam',
                  metrics=['accuracy'])
```

```
[ ]  loss, acc = model.evaluate(X_test, Y_test, verbose=2)
     print("model, accuracy: {:5.2f}%".format(100 * acc))
```

```
print(history.history.keys())
# summarize history for accuracy
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
# summarize history for loss
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```

```
def plot_confusion_matrix(y_true, y_pred, classes,
                          normalize=False,
                          title=None,
                          cmap=plt.cm.Blues):
    if not title:
        if normalize:
            title = 'Normalized confusion matrix'
        else:
            title = 'Confusion matrix, without normalization'

    # Compute confusion matrix
    cm = confusion_matrix(y_true, y_pred)
    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
        print("Normalized confusion matrix")
    else:
        print('Confusion matrix, without normalization')

#Print Confusion matrix
    fig, ax = plt.subplots(figsize=(7,7))
    im = ax.imshow(cm, interpolation='nearest', cmap=cmap)
    ax.figure.colorbar(im, ax=ax)
    # We want to show all ticks...
    ax.set(xticks=np.arange(cm.shape[1]),
           yticks=np.arange(cm.shape[0]),
        xticklabels=classes, yticklabels=classes,
           title=title,
           ylabel='True label',
           xlabel='Predicted label')

   # Rotate the tick labels and set their alignment.
    plt.setp(ax.get_xticklabels(), rotation=45, ha="right",
             rotation_mode="anchor")
    # Loop over data dimensions and create text annotations.
    fmt = '.2f' if normalize else 'd'
    thresh = cm.max() / 2.
    for i in range(cm.shape[0]):
        for j in range(cm.shape[1]):
            ax.text(j, i, format(cm[i, j], fmt),
                    ha="center", va="center",
                    color="white" if cm[i, j] > thresh else "black")
    fig.tight_layout()
    return ax

np.set_printoptions(precision=2)
```

```
#Making prediction
y_pred=model.predict_classes(X_test)
y_true=np.argmax(Y_test,axis=1)

#Plotting the confusion matrix
from sklearn.metrics import confusion_matrix
confusion_mtx=confusion_matrix(y_true,y_pred)

class_names=['veeram', 'karunam', 'bhayanakam', 'bheebhalsam', 'others']

# Plotting non-normalized confusion matrix
plot_confusion_matrix(y_true, y_pred, classes = class_names,title = 'Confusion matrix, without normalization')
```

# REFERENCES

[1] Mingxing Tan 1 Quoc V. Le 1. Efficientnet: Rethinking model scaling for convolutional neural networks. 2020.

[2] Gozde Yolcu Ismail Oztel Serap Kazan1 Cemil Oz Kannappan Palaniappan Teresa E. Lever Filiz Bunyak. Facial expression recognition for monitoring neurological disorders based on convolutional neural network. In *Springer Science+Business Media, LLC, part of Springer Nature 2019*, 2019.

[3] Sien. W. Chew, Patrick Lucey, Simon Lucey, Jason Saragih, Jeffrey F. Cohn, and Sridha Sridharan. Person-independent facial expression detection using constrained local models. In *2011 IEEE International Conference on Automatic Face Gesture Recognition (FG)*, pages 915–920, 2011.

[4] A. Fathallah, L. Abdi, and A. Douik. Facial expression recognition via deep learning. In *2017 IEEE/ACS 14th International Conference on Computer Systems and Applications (AICCSA)*, pages 745–750, 2017.

[5] S. Harshitha, N. Sangeetha, A. P. Shirly, and C. D. Abraham. Human facial expression recognition using deep learning technique. In *2019 2nd International Conference on Signal Processing and Communication (ICSPC)*, pages 339–342, 2019.

[6] Jiaxing Li, Dexiang Zhang, Jingjing Zhang, Jun Zhang, Teng Li, Yi Xia, Qing Yan, and Lina Xun. Facial expression recognition with faster r-cnn. *Procedia Computer Science*, 107:135 – 140, 2017. Advances in Information and Communication Technology: Proceedings of 7th International Congress of Information and Communication Technology (ICICT2017).

[7] Santra B Mukherjee DP Agarwal S. anubhav: recognizing emotions through facial expression. 2018.

[8] Raghu N and Bharathi S. Facial expression recognition using deep learning, 06 2020.

[9] S. Singh and F. Nasoz. Facial expression recognition with convolutional neural networks. In *2020 10th Annual Computing and Communication Workshop and Conference (CCWC)*, pages 0324–0328, 2020.

[10] L. Xu, M. Fei, W. Zhou, and A. Yang. Face expression recognition based on convolutional neural network*. In *2018 Australian New Zealand Control Conference (ANZCC)*, pages 115–118, 2018.