

**R.V. COLLEGE OF ENGINEERING
BENGALURU – 560059**
(Autonomous Institution Affiliated to VTU, Belagavi)



**“Personalized Emoji Based on Facial Expression Using
Deep Learning and Computer Graphics - MyEmoji”**

MINOR PROJECT REPORT
Submitted by

**Ashutosh Utpal Gandhi
K.S. Amogh Vardhan
Kashish Sharma**

**1RV15CS039
1RV15CS067
1RV15CS069**

Under the Guidance of

Dr. Soumya A

Associate Professor

Department of CSE, R.V.C.E.,

Bengaluru - 560059

*In partial fulfilment for the award of degree
Of*

Bachelor of Engineering

In

COMPUTER SCIENCE AND ENGINEERING

2018-2019

R.V. COLLEGE OF ENGINEERING, BANGALORE - 560059
(Autonomous Institution Affiliated to VTU, Belagavi)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING



CERTIFICATE

Certified that the minor project work titled “**Personalized Emoji Based on Facial Expression Using Deep Learning and Computer Graphics - MyEmoji**” has been carried out by **Ashutosh Utpal Gandhi (1RV15CS039), K.S. Amogh Vardhan (1RV15CS067) and Kashish Sharma (1RV15CS09)** are the bonafide students of R.V. College of Engineering, Bengaluru in partial fulfilment for the award of degree of **Bachelor of Engineering in Computer Science and Engineering** of the **Visvesvaraya Technological University, Belagavi** during the year **2018-2019**. It is certified that all corrections/suggestions indicated for the internal assessment have been incorporated in the report deposited in the departmental library. The minor project report has been approved as it satisfies the academic requirements in respect of project work prescribed by the institution for the said degree.

Dr. Soumya A

Associate Professor,

Department of CSE,

R.V.C.E., Bengaluru –59

Dr. Ramakanth Kumar P

Prof. & Head of Department,

Department of CSE,

External Viva

Name of the Examiners

1. _____

2. _____

Signature with Date

R.V. COLLEGE OF ENGINEERING, BENGALURU - 560059
(Autonomous Institution Affiliated to VTU)

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

DECLARATION

We, **Ashutosh Utpal Gandhi (1RV15CS039), K.S.Amogh Vardhan (1RV15CS067) and Kashish Sharma (1RV15CS09)**, students of Seventh Semester B.E., Computer Science and Engineering, R.V. College of Engineering, Bengaluru hereby declare that the minor project titled **“Personalized Emoji Based on Facial Expression Using Deep Learning and Computer Graphics - MyEmoji”** has been carried out by us and submitted in partial fulfilment for the award of degree of **Bachelor of Engineering in Computer Science and Engineering** of **Visvesvaraya Technological University, Belagavi** during the academic year **2018 -2019**. We declare that matter embodied in this Minor Project report has not been submitted to any other university or institution for the award of any other degree or diploma.

Place: Bengaluru
Date:

Signature
Ashutosh Utpal Gandhi
K.S. Amogh Vardhan
Kashish Sharma

Acknowledgement

Any achievement, be it scholastic or otherwise does not depend solely on the individual efforts but on the guidance, encouragement and cooperation of intellectuals, elders and friends. A number of personalities, in their own capacities have helped us in carrying out this project work. We would like to take this opportunity to thank them all.

First and foremost we would like to thank **Dr. Subramanya. K. N**, Principal, R.V.C.E, Bengaluru, for his moral support towards completing my project work.

We deeply express my sincere gratitude to my guide **Dr. Soumya A, Associate Professor**, Department of CSE, R.V.C.E, Bengaluru, for her able guidance, regular source of encouragement and assistance throughout this project

We would like to thank **Dr. Ramakanth kumar. P**, Prof. & Head of Department, Computer Science & Engineering, R.V.C.E, Bengaluru, for his valuable suggestions and expert advice.

We thank our Parents, and all the Faculty members of Department of Computer Science & Engineering for their constant support and encouragement.

Last, but not the least, we would like to thank our peers and friends who provided me with valuable suggestions to improve my project.

Abstract

The increase in usage of social media has led to an increase in communication over the internet. People are shifting from the use of text to communicate to use of emoji for communication. The emoji being used by the people is in accordance with the statement read, which indirectly or directly effects their mood which will be depicted through an emoji. But with a vast collection of emoji(s) to choose from it becomes confusing as what to pick and most of the time many emoji(s) available remain unused. The proposed system deals with the said issue. With change in emotion, the emoji(s) being depicted on the screen changes giving a user a wide range of emoji which the user may not use ordinarily.

The proposed system consist of two main modules, first being that of the emotion recognition and the second being the emoji making. For the first module the concept of Convolution Neural Networks is used to recognise the emotions based on the facial expressions. A given input image is passed through a pre-trained model from which the emotion is determined. The emotion evaluated from the first module is then sent to the second module. In the second module twenty different emoji(s) are made using Computer Graphics libraries such as GLEW, GLFW3 and GLM. There are a minimum of two emoji(s) made with respect to each of the seven emotions identified by the Deep Learning model. The emotion from the first module is then mapped to emoji(s) made in the second module and the corresponding emoji(s) are displayed.

The proposed system analyses two different standard architectures of CNN namely AlexNet and LeNet. The LeNet architecture gave an accuracy of 58% for an epoch value of 8, whereas the AlexNet Architecture gave an accuracy of 57% for an epoch value of 14. Based on the understanding of these two architectures a custom architecture was built which gave an accuracy of 59% for an epoch value of 7.

Table of Contents

| | |
|---|------------|
| Acknowledgement | i |
| Abstract | ii |
| Table of Contents | iii |
| List of Figures | vii |
| List of Tables | ix |
| Glossary | x |
| | |
| Chapter – 1 | 1 |
| Introduction | 1 |
| 1. 1 State of the Art Developments | 2 |
| 1.2 Motivation | 4 |
| 1.3 Problem Statement | 4 |
| 1.4 Objectives | 4 |
| 1.5 Methodology | 5 |
| 1.6 Organization of Report | 5 |
| | |
| Chapter - 2 | 7 |
| Overview of the Domain | 7 |
| 2.1 Pattern Recognition and Image Analysis | 7 |
| 2.1.1 Brief Description of Deep Neural Network | 8 |
| 2.2 Computer Graphics using OpenGL | 10 |
| | |
| Chapter - 3 | 12 |
| Software Requirements Specification of MyEmoji | 12 |
| 3.1 Overall Description | 12 |
| 3.1.1 Product Perspective | 12 |

| | |
|---|---------------|
| 3.1.2 Product Functions | 12 |
| 3.1.3 User Characteristics | 12 |
| 3.1.4 Constraints and Dependencies | 13 |
| 3.2 Specific Requirements | 13 |
| 3.2.1 Functional Requirements | 13 |
| 3.2.2 Performance Requirements | 13 |
| 3.2.3 Supportability | 13 |
| 3.2.4 Software Requirements | 14 |
| 3.2.5 Hardware Requirements | 14 |
| 3.2.6 Design Constraints | 15 |
| 3.2.7 Interface | 15 |
| 3.2.8 Non-Functional Requirements | 15 |
| Chapter – 4 | 17 |
| High Level Design of MyEmoji | 17 |
| 4.1 Design Considerations | 17 |
| 4.1.1 Basic Constraints | 17 |
| 4.1.2 Building Process | 17 |
| 4.2 High Level Design Strategies | 18 |
| 4.2.1 Programing Language | 18 |
| 4.2.2 Bugs, Bugs Fixes and Recover Mode | 18 |
| 4.2.3 Graphical User Interface | 18 |
| 4.2.4 Data Store Handling | 18 |
| 4.3 System Modelling | 19 |
| 4.3.1 Data Flow Diagram Level-0 | 19 |
| 4.3.2 Data Flow Diagram Level-1 | 20 |
| 4.3.3 Data Flow Diagram Level-2 | 21 |
| Chapter – 5 | 23 |
| Detailed Design of MyEmoji | 23 |
| 5.1 UML Diagram | 23 |
| 5.1.1 Facial Image Analysis using Deep Learning | 24 |
| 5.1.2 Emoji Builder | 25 |
| 5.2 Structural Chart | 26 |

| | |
|--|-----------|
| 5.3 Functional Description of Modules | 26 |
| 5.3.1 Face Detection Module | 27 |
| 5.3.2 Emotion Recognition Module | 27 |
| 5.3.3 Inter-Module Communication | 28 |
| 5.3.4 Emoji Builder Module | 29 |
| Chapter – 6 | 31 |
| Implementation of MyEmoji | 31 |
| 6.1 Programming language selection | 31 |
| 6.2 Platform selection | 31 |
| 6.3 Code Conventions | 31 |
| 6.3.1 Naming convention | 31 |
| 6.3.2 File organization | 32 |
| 6.3.3 Properties declaration | 33 |
| 6.3.4 Class declaration | 33 |
| 6.3.5 Comments | 33 |
| 6.4 Difficulties Encountered and Strategies Used to Tackle | 33 |
| 6.4.1 Integration of components | 34 |
| 6.4.2 Synchronization between services | 34 |
| Chapter – 7 | 35 |
| Software Testing of MyEmoji | 35 |
| 7.1 Test Environment | 35 |
| 7.1.2 Unit testing of Deep learning modules | 35 |
| 7.2.3 Unit Testing of the Emotion | 36 |
| 7.3 Integration testing of the modules | 40 |
| 7.4 System testing | 41 |
| Chapter – 8 | 43 |
| Experimental Analysis and Results of MyEmoji | 43 |
| 8.1 Evaluation Metric | 43 |
| 8.2 Experimental Dataset | 44 |
| 8.3 Performance Analysis | 44 |

| | |
|--------------------------------|-----------|
| 8.4 Inference from the Result | 45 |
| Chapter – 9 | 47 |
| Conclusion | 47 |
| 9.1 Limitations of the project | 47 |
| 9.2 Future Enhancement | 47 |
| References | 49 |
| Appendix | 51 |

List of Figures

| Figure No. | Figure Name | Page No. |
|-------------------|---|-----------------|
| Figure 2-1 | Generic Deep learning Architecture | 8 |
| Figure 2-2 | AlexNet architecture | 9 |
| Figure 2-3 | LeNet Architecture | 10 |
| Figure 2-4 | OpenGL Programming Pipeline | 11 |
| Figure 4-1 | DFD Level-0 of MyEmoji | 19 |
| Figure 4-2 | DFD Level-1 of MyEmoji | 20 |
| Figure 4-3 | DFD Level-2 of MyEmoji | 22 |
| Figure 5-1 | UML Class Diagram of MyEmoji | 24 |
| Figure 5-2 | Emoji(s) of different Emotions | 25 |
| Figure 5-3 | Structure Chart of MyEmoji | 26 |
| Figure 5-4 | Modular Description of Face Detection | 27 |
| Figure 5-5 | Modular Description of Emotion Recognition | 28 |
| Figure 5-6 | Modular Description of Inter-Module Communication | 29 |
| Figure 5-7 | Modular Description of Emoji Builder | 30 |
| Figure 8-1 | Analysis of the LeNet Architecture | 45 |
| Figure 8-2 | Analysis of the AlexNet Architecture | 46 |
| Figure 8-3 | Analysis of the Custom Architecture | 46 |
| Figure A-1 | Emotion detection using LeNet | 51 |
| Figure A-2 | Emotion detection using AlexNet | 52 |
| Figure A-3 | Emotion detection using custom model | 52 |
| Figure A-4 | Sad Emoji(s) | 52 |

| | | |
|-------------|-------------------|----|
| Figure A-5 | Angry Emoji(s) | 53 |
| Figure A-6 | Disgust Emoji(s) | 53 |
| Figure A-7 | Fear Emoji(s) | 53 |
| Figure A-8 | Happy Emoji(s) | 54 |
| Figure A-9 | Surprise Emoji(s) | 54 |
| Figure A-10 | Neutral Emoji(s) | 54 |

List of Tables

| Table No. | Table Name | Page No. |
|------------------|--|-----------------|
| Table 7-1 | Testing AlexNet Architecture | 35 |
| Table 7-2 | Testing LeNet Architecture | 36 |
| Table 7-3 | Testing of Emotion – Happy | 37 |
| Table 7-4 | Testing of Emotion – Neutral | 37 |
| Table 7-5 | Testing of Emotion – Surprise | 38 |
| Table 7-6 | Testing of Emotion – Disgust | 38 |
| Table 7-7 | Testing of Emotion – Sad | 39 |
| Table 7-8 | Testing of Emotion – Angry | 39 |
| Table 7-9 | Testing of Emotion – Fear | 40 |
| Table 7-10 | System Testing - Happy | 41 |
| Table 7-11 | System Testing - Sad | 41 |
| Table 7-12 | System Testing - Happy | 42 |
| Table 7-13 | System Testing - Fear | 42 |
| Table 8-1 | Summary of the different architectures | 44 |

Glossary

| | | |
|---------|---|---|
| CNN | : | Convolution Neural Network |
| CVPR | : | Computer Vision and Pattern Recognition |
| ConvNET | : | Convolution Neural Network |
| FERC | : | Facial Expression Recognition Challenge |
| GLEW | : | OpenGL Extension Wrangler Library// |
| GLFW | : | Graphics Library Framework |
| GLM | : | OpenGL Mathematics |
| GPU | : | Graphical Processing Unit |
| iOS | : | iPhone Operating System |
| MCDNN | : | Multi-column Deep Neural Network |
| MNSIT | : | Modified National Institute of Standards and Technology |
| OpenGL | : | Open Graphics Library |
| ResNet | : | Residual Network |
| SVM | : | Support Vector Machine |
| TE | : | Training Error |
| TPU | : | Tensor Processor Unit |
| UM | : | Unified Modelling Language |
| VE | : | Validation Error |

Chapter – 1

Introduction

The increase in usage of social media has led to an increase in communication over internet. People are shifting from the use of text to communicate to the use of emoji(s) for communication. Companies are already using sentiment analysis to gauge consumer mood towards their product or brand. By mining tweets, reviews, and other sources, companies can easily derive sentiment from natural language. Customer representatives in stores can see when a customer is frustrated or angry, but they can't be everywhere at once. However, companies have been using in-store cameras to monitor customer behaviour for years. Companies use this data to track hot-spots in the store, the paths customers take, and even what they're specifically looking at.

The primary issue is that it's not easy to analyse the vivid emotions a person depicts and accurately detecting emotion is the key. For human to analyse other person's emotion is easy because the humans have been observing such emotions since the day they were born and labelling each emotion accordingly but for a computer to analyse this emotion is a difficult task, as for a computer it is all about numbers. The system is able to look at an image of a person's face and easily differentiate between a frown and a smile, but for a ML model, it isn't easy.

Ever since computers were developed, scientists and engineers thought of artificially intelligent systems that are mentally and/or physically equivalent to humans. In the past decades, the increase of generally available computational power provided a helping hand for developing fast learning machines, whereas the internet supplied an enormous amount of data for training. These two developments boosted the research on smart self-learning systems, with neural networks among the most promising techniques.

Accuracy of any facial expression recognition system depends mostly on the extraction of appropriate feature that can represent certain facial expression. An extracted feature of a face can be considered a proper representation if it can satisfy three conditions:

- It should reduce variations of expressions within the class while maximizes variations between classes.
- It should conveniently get needed details from raw face image.

1. 1 State of the Art Developments

In this section, various existing techniques and their corresponding challenges in the field of Facial emotion recognition have been discussed further.

T. Ahsan, T. Jabid, and U.-P. Chong ^[1] explain the use of Gabor Filters for edge detection. Gabor Filters have been found to be particularly appropriate for texture representation and discrimination. Gabor filter of 8 orientations are applied and the resultant images are all superimposed. The superimposed image is then passed to the SVM for the further classification. The 7 classification categories consisted of Anger, disgust, Fear, Joy, Sadness, Surprise and Neutral. The result is an error rate of approx. 11%.

D. Ciresan, U Meier, and J Schmidhuber ^[2] elaborate the MCDNN architecture. They basically explored the architecture for the purpose of handwritten digits recognition and traffic sign recognition. The architecture used in this paper has more number of maps on each layer because of which the training time increases. The MCDNN architecture used in this paper consists of 2 convolution layers and 2 Max-Pooling layers, the first convolution layer has 20 4×4 filters and the second convolution layer has 800 5×5 filters whereas the fully connected layer consists of 150 neurons. The Architecture used in this model gave an error rate of 0.23% on the MNSIT dataset (Handwritten digits dataset).

A. Krizhevsky, I, Sutskever, and G and E. Hinton ^[3] discuss the effects of over-fitting for a large dataset. To counter the over-fitting of the model a method of regularization called dropout is used. Dropout is a technique where randomly selected neurons are ignored during training. They are “dropped-out” randomly. This means that their contribution to the activation of downstream neurons is temporally removed on the forward pass and any weight updates are not applied to the neuron on the backward pass. The Model used in this paper consists of 5 convolution layers, most of them followed by a max-Pooling layer, 3 fully connected layer ending with a 1000-way softmax. At the end of each convolution layer an activation function called as ReLU is introduced. The input images are pre-processed to a size of 256×256 pixels. The training of the model is carried out on multiple GPUs do to the large size of the dataset. The results are an error rate of 37.5% in top-1 classifier and 17% in top-2 classifiers.

Enrique Correa, Arnoud Jonker, Michael Ozo, and Rob Stolk ^[4] explore the standard Deep Learning architectures for the Facial Expression Recognition Challenge. For this Challenge the dataset used was FEREC-2013 having grayscale images of 48×48 pixels and close to 28K sample

images. The model used in this paper consists of 3 convolution layers, 2 max-pooling layers, one fully connected layer with 3072 neurons and an output softmax layer for the 7 different classes. The 7 classification categories consisted of Anger, disgust, Fear, Joy, Sadness, Surprise and Neutral. The dataset had a few shadowed images and so in a few images the face where no detectable and so need to use a pre-processing method such as HaarCascade method for the face detection. The model gave an accuracy of 63% with 60 epochs.

Dilbag Singh ^[5] discusses the application of feature extraction of facial expressions with combination of neural network for the recognition of different facial emotions namely angry, sad, fear, surprise, neutral, disgust, and Happy. The images are firstly pre-processed and resized to enhance the input image and to remove any type of noises present in the image. The resizing is done to consider only the human face using an eye selection method. The neural method use provides an accuracy of 97%.

Denis Sokolov and Mikhail Patkin ^[6] explore a system for a real-time emotion recognition for mobile devices. Here the features from a real-time video are extracted and passes to a modified architecture of ResNet. OpenCV is used for the pre-processing of the image. The training is performed on a NVIDIA Titan X GPU. The architecture provided a validation accuracy of 64.89%.

Yiliang Xie, Hongyuan Jin, Eric C.C. Tsang ^[7] LeNet architecture is chosen as the baseline model. Different techniques are used to optimize the present architecture. Few of the techniques are adding activation layers right after each convolution layer and applying batch normalisation. Top-8 accuracy of 96% accuracy is achieved.

Lisha Xiao, Qin Yan ^[8] AlexNet architecture is chosen as the baseline architecture. An improved AlexNet model is proposed based on the design principles of convolution neural networks. The large convolution kernel is decomposed into a structure cascaded by two small convolution kernels with reduced stride. Another convolutional layer is added after the first one to enhance the integration process of the low-level features or the spatial information. The asymmetric convolution kernel is applied in the last three convolutional layers. An accuracy of 84% was achieved for classification of places.

Matthew Zeiler and Rob Fergus ^[9] ZFNet architecture is chosen as the baseline architecture. ZF Net used filters of size 7x7 and a decreased stride value. The reasoning behind this modification is that a smaller filter size in the first conv layer helps retain a lot of original pixel information in the input volume. A filtering of size 11x11 proved to be skipping a lot of relevant

information, especially as this is the first conv layer. Used ReLUs for their activation functions, cross-entropy loss for the error function, and trained using batch stochastic gradient descent. Developed a visualization technique named Deconvolutional Network, which helps to examine different feature activations and their relation to the input space.

Kaiming He, Xiangyu Zhang, Shaoqing Ren and Jian Sun ^[10] ResNet architecture is chosen as the baseline architecture. The model had 152 layers. The input images were compressed to 56x56 pixels. The model is also called as “ultra-deep”. The training happened over the course of three weeks on eight separate GPUs.

1.2 Motivation

Emoji(s) are ideograms and smileys used in electronic messages and web pages. Emoji are generally used in scenarios to lighten the mood through humour or sarcasm, expressing oneself, when words fail us. Generally messaging applications consist of a large number of emoji to choose from and finding the appropriate emoji takes up a lot of time. People generally use only a limited set of emoji(s) in their day-to-day messaging. Using this application helps them expand the horizon by using multiple expressions of emoji which otherwise they would not have used.

The number of new emoji(s) being made is growing exponentially over the last 5 years because of this growth the user has a tedious process of going through a list of emoji(s) while texting. This application can save this time by recommending a list of emoji(s) based on user's current emotion.

1.3 Problem Statement

An application that recognizes the expression of a person, and based on that expression suggests a certain number of emoji(s) from which the user can select one.

1.4 Objectives

The objectives of the project are set in the direction to complete the implementation with expected results. Initial task is to create fun customizable emoji(s) that mirror the facial expressions. This is done using Computer Graphics libraries such as GLEW and GLFW3. The next objective is to identify the emotion of the user based on his facial expression, this is done using Deep Learning models such as a Convolution Neural Network implemented through a

machine learning framework such as Keras. Further the emoji is produced as the output by mapping the results of the two objective tasks.

1.5 Methodology

The section provides the brief overview of the methodology followed during the course of development of the proposed system. The main steps followed are listed below:

- Pre-processing image for expression analysis.
- Mapping user's expression with the generated emoji using Deep Neural Network
 - Tweaking hyper parameters based on Train Error and Validation Error.
 - Given an input an appropriate Emotion is classified.
- Generate Emoji using the libraries GLEW and GLFW3 in Computer Graphics.
 - The emotion thus classified is represented in graphical form.
 - Graphical representation is generated using OpenGL.
- Customizing Emoji(s) extent of expression based on user's preference.

1.6 Organization of Report

The section gives a brief description of modules in this report. The report contains 7 chapters. The brief description of every chapter is provided in the following.

- **Chapter 2** gives a brief overview of the fundamental concepts of Computer Graphics and Deep Learning. This chapter explains basics of OpenGL and elaborates of the two standard architectures of Deep Learning namely LeNet and AlexNet.
- **Chapter 3** provides the Software Specification Requirements of the project. It mentions the functional and non-functional requirements along with the constraints of the system on the hardware and software. It also elaborates on the design and performance constraints on the system.
- **Chapter 4** describes the high-level design of the project with sub-headings related to design considerations, architecture strategies, and data flow diagrams. The data flow diagrams are represented through 3 different levels viz. level 0, level 1 and level 2.
- **Chapter 5** describes the low-level design or internal design of the project with sub-headings related to functional descriptions of the model.
- **Chapter 6** details the different techniques used in the Implementation of the project/model. It also provides details about programming language and platform

selecting along with conventions for naming, properties, features and class declarations/definitions.

- **Chapter 7** deals on software testing of the model with sub-headings in both unit and integration testing of the two main modules of the model namely the emotion detection and Emoji generation. The test environment along with test cases, their executions and the corresponding results are outlined in this section.
- **Chapter 8** deals with the experimental analysis of the results thus obtained from the model. Along with this the performance analysis of the system is also done as a part. The inferences collected through the results are also mentioned.
- **Chapter 9** is in regard to conclusion outlining the summary of the project and also the limitations of the system along with the future work.

Chapter - 2

Overview of the Domain

The chapter elaborates the two main areas used in the development of the proposed system namely Deep Neural Network and Computer Graphics. The Deep Neural Network concepts are used in the first module where the emotion is to be recognised based on the facial expression. The Computer Graphics concepts are used for the Emoji Builder module, here the main libraries used include OpenGL, GLEW, GLM, and GLFW3. The first section in this chapter discusses the fundamentals of DLL and the preceding chapter the fundamentals of OpenGL.

2.1 Pattern Recognition and Image Analysis

Pattern Recognition provides the solution to various problems from speech recognition, face recognition to classification of handwritten characters and medical diagnosis. Three processes take place in pattern recognition task.

First step is data acquisition. Data acquisition is the process of converting data from one form into another form which should be acceptable to the computing device for further processing. Data acquisition is generally performed by sensors, digitizing machine and scanners. Second step is data analysis. After data acquisition the task of analysis begins. During data analysis step the learning about the data takes place and information is collected about the different events and pattern classes available in the data. This information or knowledge about the data is used for further processing. Third step used for pattern recognition is classification. Its purpose is to decide the category of new data on the basis of knowledge received from data analysis process.

There are several classifiers used for pattern recognition applications, a few of them are:

Support Vector Machine (SVM) is a supervised machine learning algorithm which can be used for both classification and regression challenges. It is mostly used in classification problems. In this algorithm, each data item is plotted as a point in n-dimensional space with the value of each feature being the value of a particular coordinate.

The k-nearest neighbours (k-NN) algorithm is a non-parametric method used for classification

and regression. In both cases, the input consists of the k closest training examples in the feature space. The output depends on whether k -NN is used for classification or regression.

Artificial Neural Network (ANN) involves a large number of processors operating in parallel and arranged in tiers. The first tier receives the raw input information. Each successive tier receives the output from the tier preceding it, rather than from the raw input - in the same way neurons further from the optic nerve receive signals from those closer to it. The last tier produces the output of the system.

A Deep Neural Network (DNN) is a neural network with a certain level of complexity. It is a neural network with more than two layers. Deep neural networks use sophisticated mathematical modelling to process data in complex ways. A brief description of it is provided in the following section.

2.1.1 Brief Description of Deep Neural Network

Deep Neural Network is a subset of machine learning. It finds its importance when it comes to working on images. The word “deep” comes from the fact that there are numerous numbers of layers in the CNN architecture. A large amount of data along with a well- designed Deep Learning model gives best results. Deep Learning has a huge success in solving various problems as its conceptualized based on how a human brain works. There are many Deep Learning frameworks and APIs which have made programming very easy. Few of the famous one’s include TensorFlow and Keras.

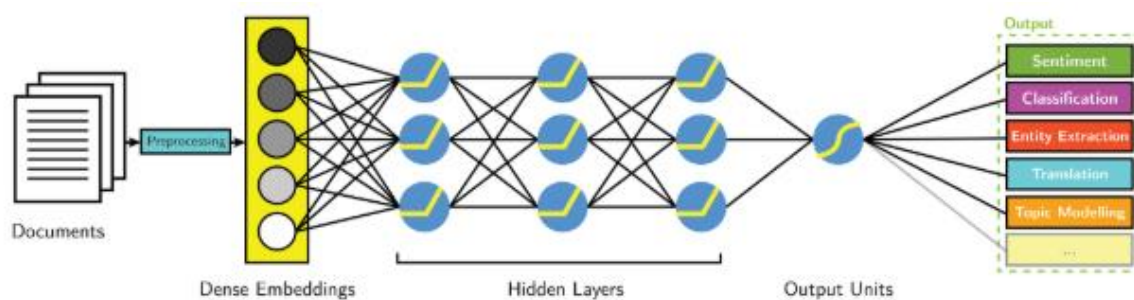


Figure 2-1 Generic Deep learning Architecture
(Courtesy of www.completegate.com)

A few of the architectures of DNN:

- AlexNet

The developer of this architecture achieved a top5 error of 15 percent. This is a very good result in comparison to many other models. AlexNet is best used for image classification.

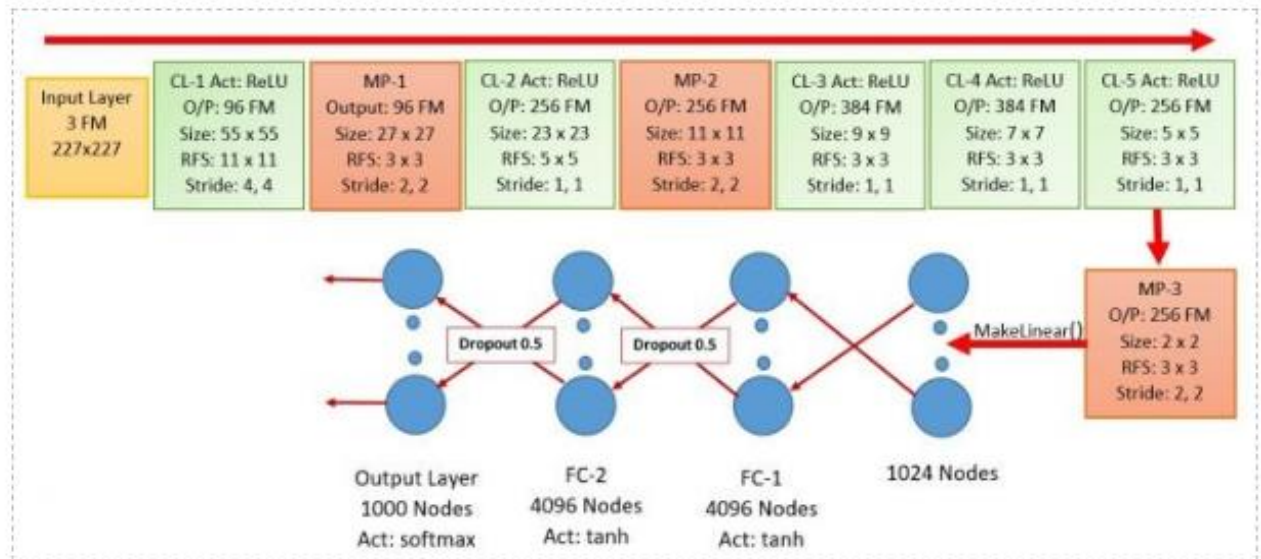


Figure 2-2 AlexNet architecture
(Courtesy of www.completegate.com)

- LeNet

LeNet architecture is quite old compared to AlexNet. It is specially designed to work for optical character recognition. The architecture can also be used to do image classification but it is expected to give lesser accuracy compared to AlexNet.

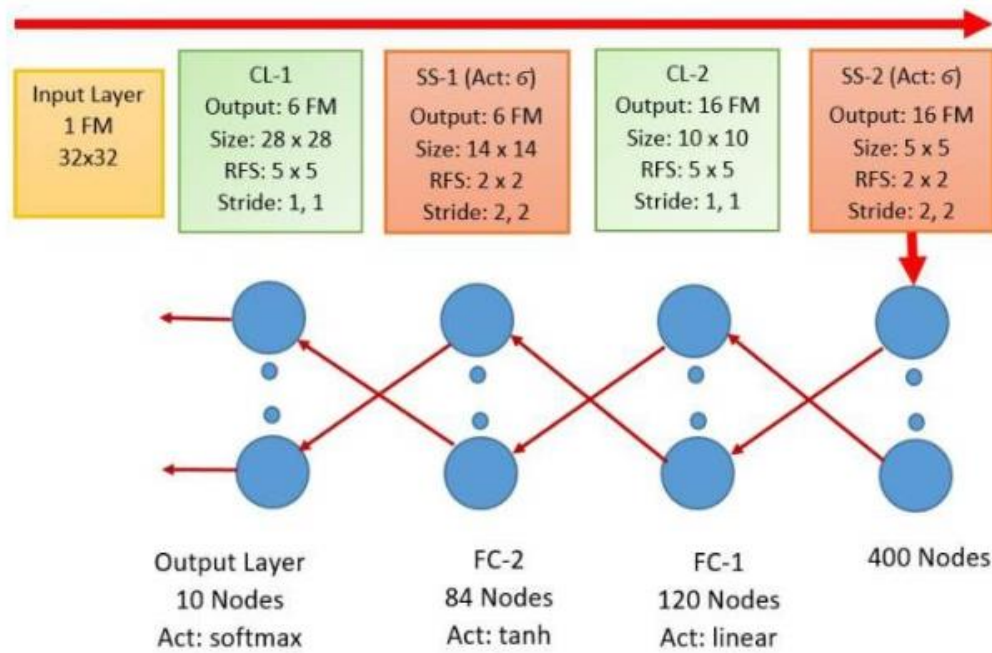


Figure 2-3 LeNet Architecture
(Courtesy of www.completegate.com)

2.2 Computer Graphics using OpenGL

OpenGL provides application program interface (API) to render 2D and 3D graphics application. It provides fast rendering of graphics in addition to features like texture mapping, camera control, pitch and yaw control, lighting effects (point, directional or ambient lighting) and other powerful visualizations.

It is known for its stability, reliability and portability as well as works on image data and geometric primitives which makes it a handy graphical application program interface.

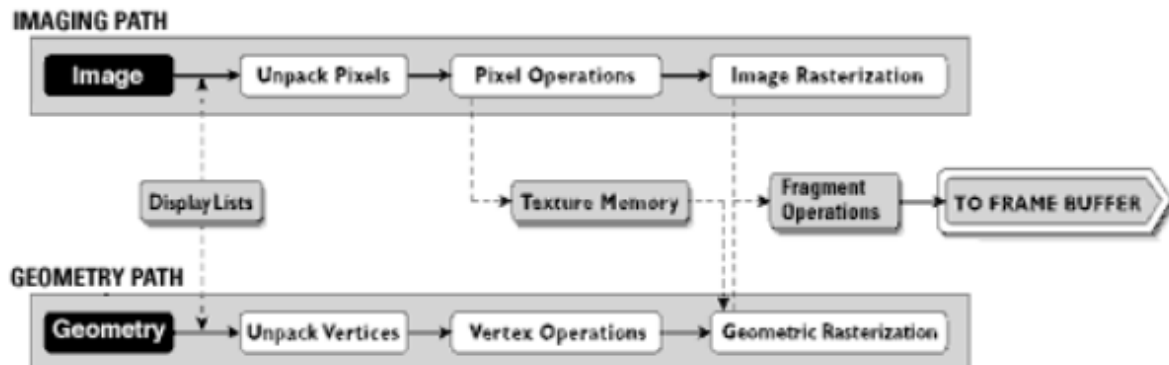


Figure 2-4 OpenGL Programming Pipeline
(Courtesy of www.opengl.org)

OpenGL provides event handlers which let the user interact with the system in one way or the other, which can be either using keyboard or mouse or using some other input device.

The different libraries used for rendering and object are:

- **GLEW:** The OpenGL Extension Wrangler Library is a cross-platform C/C++ library that helps in querying and loading OpenGL extensions
- **GLFW3:** GLFW is an Open Source, multi-platform library for OpenGL, OpenGL ES and Vulkan development on the desktop. It provides a simple API for creating windows, contexts and surfaces, receiving input and events.

Chapter - 3

Software Requirements Specification of MyEmoji

SRS document specifies the functional and non-functional requirements of the system. It also mentions the hardware and the software requirements for the proposed system to run. In this chapter the complete details of the requirements of the proposed system are mentioned.

3.1 Overall Description

The section points out the various requirements towards the implementation of the system. It provides the product perspective, performance constraints, dependencies, assumptions and the design constraints.

3.1.1 Product Perspective

The proposed system uses the Microsoft provided IDE – Visual Studio, this significantly eases the process of using the Computer Graphics libraries on the windows platform.

3.1.2 Product Functions

The major functions of the proposed system is to identify the emotions based on the facial expression and to map this emotion to an output display emoji. For the recognition of the emotion, a user interface is provided from which he can test a new image against a pre-trained model. Once the emotion is recognised it is passed on the next module which displays the corresponding emoji(s) to the received emotion.

3.1.3 User Characteristics

The product design is made in such a way that the user can easily navigate his/her way around the system. The design is in such a way that the user can select the input image and also the model architecture to run the image against. For the tester an option is provided to run the pre-existing model on a different dataset by making a few minor adjustment to the code.

3.1.4 Constraints and Dependencies

The proposed system as such as no stringent constraints. The dependencies of the proposed system include the functional libraries such as the python environment and a few libraries such as GLEW, GLFW3, GLUT and GLM preinstalled.

3.2 Specific Requirements

The section mentions the requirements of the system which must be adhered to during the implementation phase of the project. This section discusses the functional, non-functional requirements as well as the performance and design constraints.

3.2.1 Functional Requirements

The list of functional requirements are mentioned below:

- The system should be able to take in a new image for the processing of the image. This input should then be provided to first module to determine the emotion.
- The system should be able to process the input image and test it on a number of pre-trained model and provide the appropriate accuracy of each model.
- The emotion recognised is then passed on to the next module through the help of an interface.
- The second module then processes the emotion and generates relevant emoji(s) for it.
- Once the relevant emoji is generated, the output emoji is provided in two forms, one the pictorial form and the other through a keyboard emoji.

3.2.2 Performance Requirements

Performance forms an important aspect of the software development lifecycle. In the proposed system the performance specifications is to provide a provision of pre-trained model which reduces the processing time of the emotion recognition.

3.2.3 Supportability

The proposed system should support both the windows as well as the Ubuntu platforms. It should support both the x64 and x86 architectures of the windows platform and all the different versions of windows after windows 7.

3.2.4 Software Requirements

The list of software requirements is provided below:

- **Keras:** It is an open source neural network library written in Python. It enables fast experimentation with deep neural networks.
- **GLEW:** The OpenGL Extension Wrangler Library is a cross-platform C/C++ library that helps in querying and loading OpenGL extensions
- **GLFW3:** GLFW is an Open Source, multi-platform library for OpenGL, OpenGL ES and Vulkan development on the desktop. It provides a simple API for creating windows, contexts and surfaces, receiving input and events.
- **Editors:**
 - **Atom:** Atom is a free and open-source text and source code editor for macOS, Linux, and Microsoft Windows with support for plug-ins written in Node.js, and embedded Git Control, developed by GitHub.
 - **Xcode:** Xcode is an integrated development environment (IDE) for macOS containing a suite of software development tools developed by Apple for developing software for macOS, iOS, watchOS, and tvOS.

3.2.5 Hardware Requirements

The list of hardware requirements are mentioned below:

- **Processor:** Quad-core processor.
- **Storage:** minimum 10 GB disk space available for caching and temporary files.
- **RAM:** minimum 4 GB.
- **Display:** minimum 1680 x 1050 pixel resolution.
- **Graphics Card** with the latest drivers, 1 GB of graphics memory and OpenGL 3.2* or higher.

3.2.6 Design Constraints

The framework of the proposed system must have a flexible design such that the user interaction is made easier. The proposed system must have two separate modules one for the emotion detection and the other for the graphics component of the emoji generation. The workflow between the two modules must be efficiently designed such that it can be easily understood by the tester.

3.2.7 Interface

The following subsections describe interaction between the modules and interaction of the system with the user.

- **User Interface:** The user is given options to detect emotion as well as to check the performance of the model.
 - Choosing the former require user to input image file name and upload in the system. After processing the output will be displayed.
 - Choosing the later will provide the accuracy of the model

In addition to these features, a choice is given to explicitly choose the architecture to process the image for emotion detection. Different architecture may give different accuracy.

- **Inter Module Interaction:** The two main components of the system viz. Deep Learning Component and OpenGL Component communicate via a socket.
 - Deep learning being the client and OpenGL being the server. The data is sent from client to server which is stored in file. The server reads data from the file containing emotion id and renders the emoji accordingly.
 - The session remains open after processing the request to handle other requests thus there are no overheads in opening and closing connections between server and client for handling multiple requests.

3.2.8 Non-Functional Requirements

The requirements that specify criteria that can be used to judge the operation of the system rather than specific behaviour and are not directly concerned with the specific functions delivered by the system.

- **Expandability:** The framework scripts should be easily expandable to multiple languages.
- **Usability:** The scripts used for testing should be of the form ‘Write once, run on any language’.
- **Performance:** The framework is expected to perform well especially during issue detection. The framework should be optimistic to cover any issue in the UI and at the same time should be reluctant of false alarms.
- **Portability:** The framework is highly portable since it is basically a Visual Studio C++ project. As long as the platform under test in windows, framework is compatible.
- **Efficiency:** The framework should be highly efficient such that it takes much less time to do the issue detection across any language compared to manually processing

Chapter – 4

High Level Design of MyEmoji

Designing phase is the most critical phase of all. It involves creative techniques to provide the needed outcome or even better result or performance. Design acts as a blueprint which guides in developing, integrating and testing the performance, working, usability and much more of the model/products. The required output dictates the design procedure of the model which defines the architecture of the model. This also tells about the shortcomings in the architecture. The following section deals with the various constraints that need to be kept under consideration while designing the model to achieve better efficient and accuracy.

4.1 Design Considerations

While designing various constraints are to be kept in mind. The following section gives an overview of the design considerations that have an effect on the model and the following section gives an overview of the model design.

4.1.1 Basic Constraints

There are some basic constraints of the system:

- System needs understanding by the user
- The handler should understand the functional requirement
- The user has to accustom with the functioning and handling of the system
- Knowledge about the data set and its limitation
- Precisions and accuracy proportionality depends upon the amount of data available in data set.

4.1.2 Building Process

The Building Process involves making data flow diagrams. Data flow diagrams make it easier to understand the architecture of the model. It gives insight about the flow, the functionality and the interaction with the entities. Data flow model gives insight about the flow of data from one component to another in terms of functional units and data store.

4.2 High Level Design Strategies

The system's high level design is provided in this part, providing insights about different approaches and strategies being used or tweaked in the mode.

4.2.1 Programing Language

To choose a programming language from the enormous collection can be confusing. Each language has its advantages and drawbacks. Some languages have large collection of libraries proving coverage in entire domain where as some have limited libraries targeting some particular domains.

The choice between object oriented and procedure oriented is not so confusing. Modelling for real world requires languages that simulate the model as if it was in the real world.

Use of scripting language that provides flexibility, simulate real world and it has libraries which can cover the domain the project involved keeping these things in mind the python language is used.

4.2.2 Bugs, Bugs Fixes and Recover Mode

For any software implementation error handlers are one of the most important elements. To throw exceptions and catch them in order to handle them forms a major part of the implementation. Hence the system deployed handles exception by a number of exception handlers provided to deal with the unexpected bugs.

4.2.3 Graphical User Interface

Graphical user interface plays an important role in any software as it acts as the front for all the background process or the first front of the system. For graphical user interface OpenGL has been used to depict the output based on the result obtained from the background processing.

4.2.4 Data Store Handling

A program becomes more efficient if data processed is being stored efficiently so that on query, the needed result can be obtained quickly without any stalls for smooth functioning of the software thereby provided much needed accuracy and performance.

4.3 System Modelling

System designing is the most important as it helps in identifying and distinguishing different components involved in the model. The usage of data flow diagram helps us achieve that.

With increase in the level of the data flow diagram the clarity in the functioning of the model increases for interaction between functional units to the flow of data from entities to modules then data flow between modules and finally the needed output.

4.3.1 Data Flow Diagram Level-0

DFD Level-0 shown in Figure 4-1 provides high level design of the model in making. DFD level-0 gives an overview of the model. It gives insight about different entities, their interaction with the system, the type of interaction and provide vaguely the flow of data.

A user interacts with the system by providing an image to the system. The system will analyse the image and generates an emotion ID based on which emoji(s) are depicted. The user has the provision to provide more input images and correspondingly the emoji(s) will be depicted.

Administrator test, validates and analyses the accuracy and response of the system and accordingly makes changes.

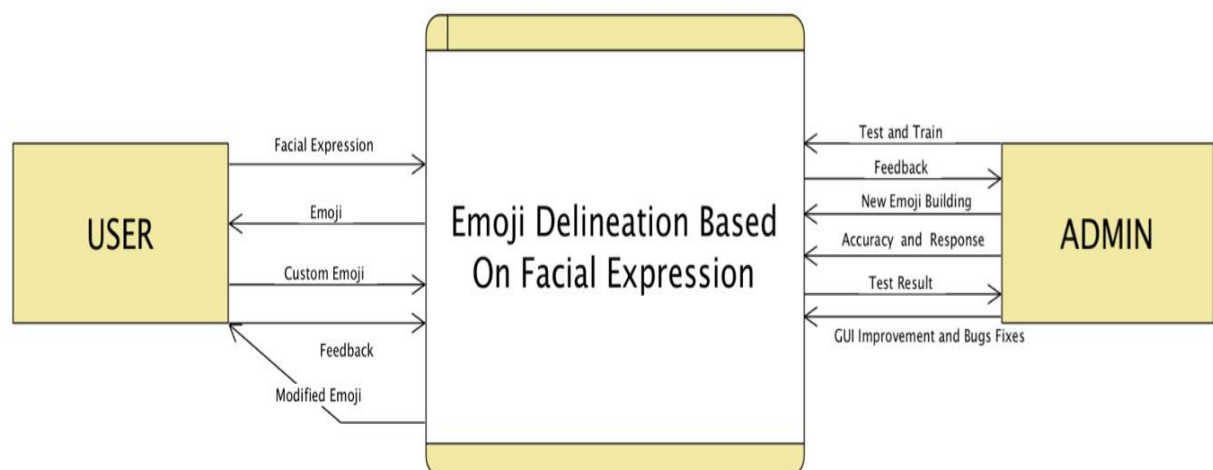


Figure 4-1 DFD Level-0 of MyEmoji

4.3.2 Data Flow Diagram Level-1

The system is further broken into modules. Data Flow Diagram level-1 shown in Figure 4-2 provided details of such modules.

It provided the basic modules required to design the model and their interaction with entities and other modules. The data flows among the different modules as well as between the user and modules.

It gives insight concisely as to how and what works are being done by each module and how are they communicating among themselves to provide the result for the given input data.

The input image provided by the user to system goes to the CNN module which does face detection and emotion analysis and generates an emotion ID that goes to the Emoji Builder module, which is responsible for creation of emoji(s). After mapping of the emotion ID with emoji(s), the corresponding emoji(s) is/are depicted on the standard output device. User can provide feedback to the administrator based on the obtained result.

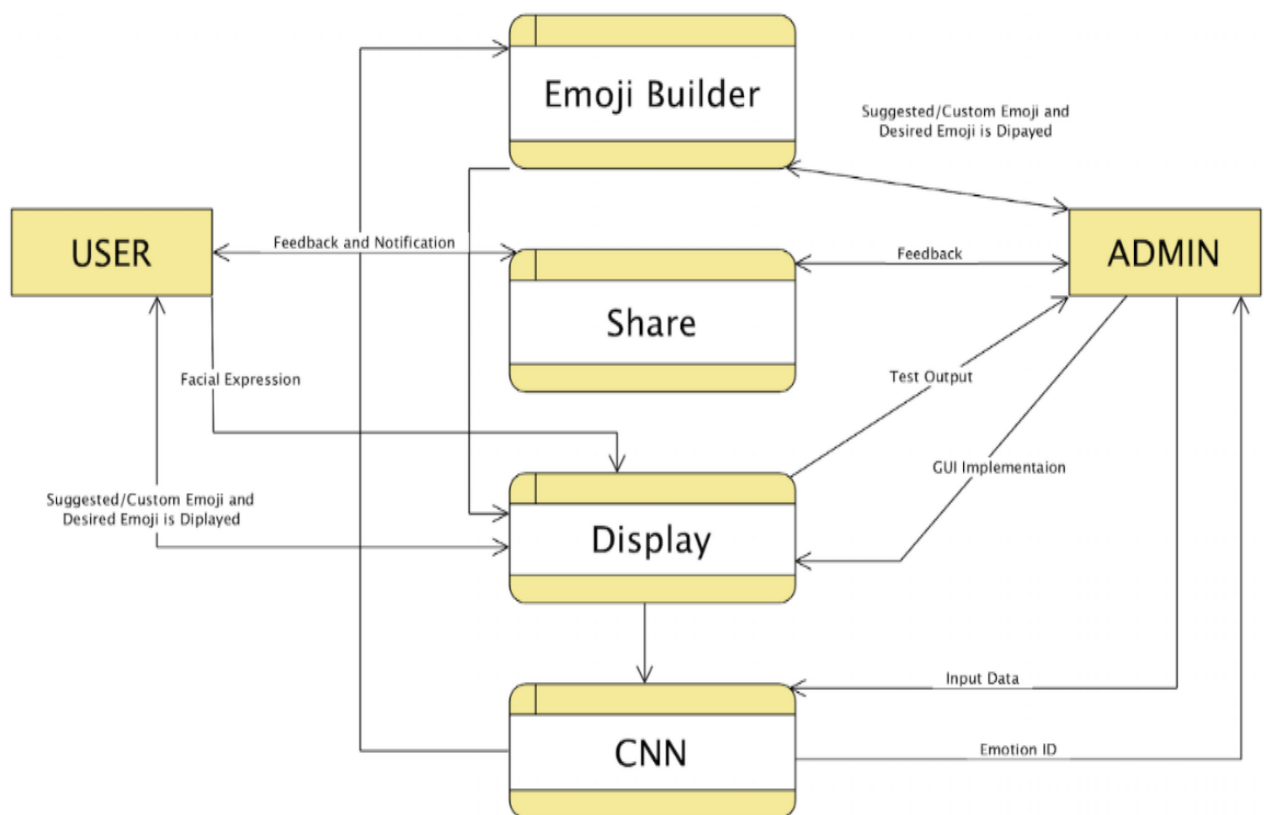


Figure 4-2 DFD Level-1 of MyEmoji

4.3.3 Data Flow Diagram Level-2

On increasing the level of the data flow diagrams, more detailing of the model is required. Data Flow Diagram level-2 shown in Figure 4-3 provides details as to how data is being stored and retrieved for query and processing. Which module is querying which module and what data is being processed at any time by different modules.

The input image provided by the user to system is fed to the input layer of the Deep Neural Network which does face detection and emotion analysis. After the analysis of the image, the parametric weights of deep learning layers are changed to provide an optimal accuracy, an emotion ID is generated that goes to the Emoji Builder component, which is responsible for creation of emoji(s). Each category of emotion is given a unique ID, and the emoji(s) generated are associated with this ID. After mapping of emotion ID with the ID of the class of emoji(s), the corresponding emoji(s) is/are depicted on the standard output device. Any changes made to the emoji is saved on the database. Administrator logs all the feedback provided by the user. Admin can also provide images to the system for testing and validating the system

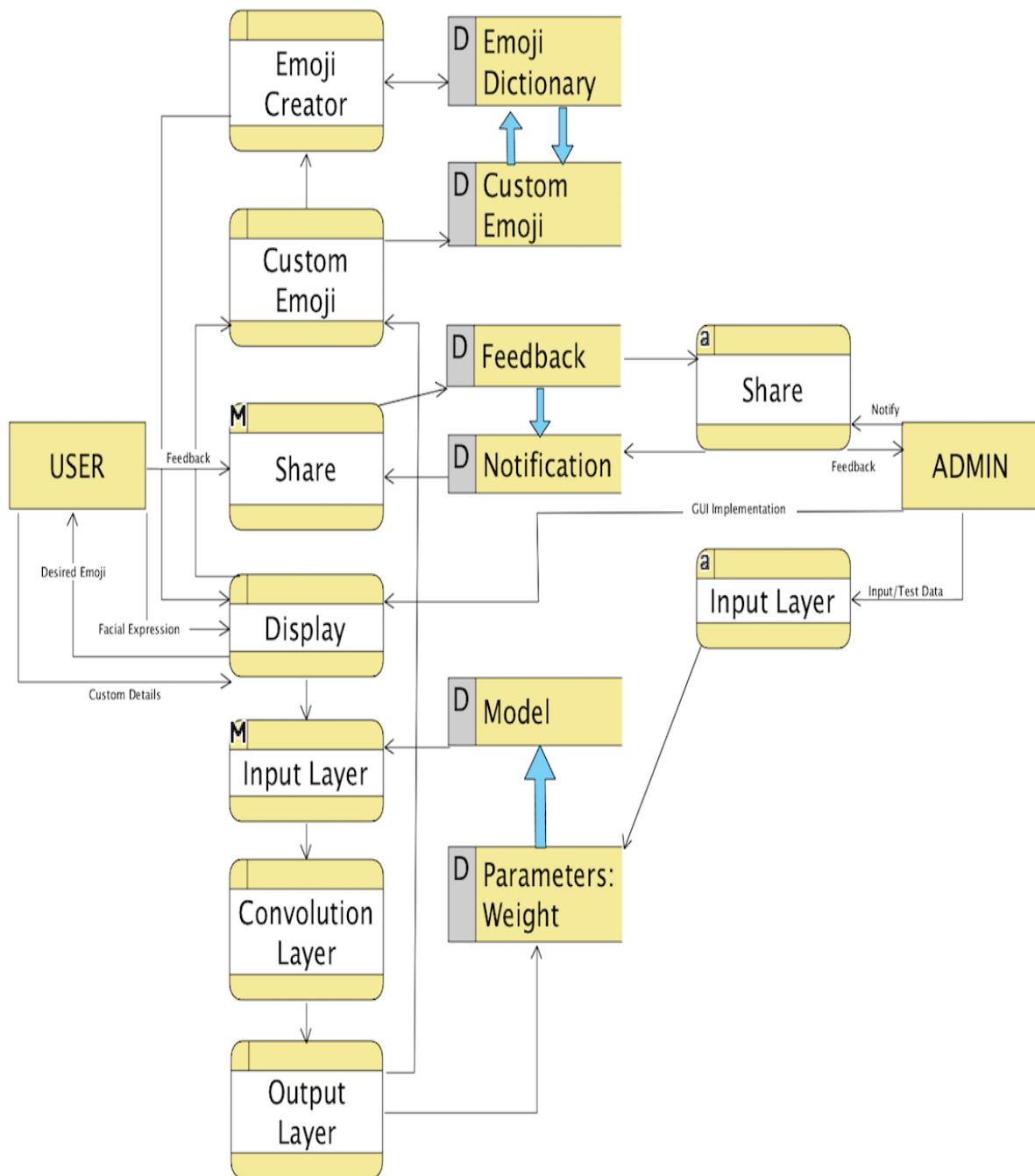


Figure 4-3 DFD Level-2 of MyEmoji

Chapter – 5

Detailed Design of MyEmoji

Detailed Design deals with the internal structure of the model. Low level tells about component internal structure and details about sub components. It provides information about different algorithms to be used the flow of control.

It provides details about structure of the model telling us details about the input facility, output facility and storage facility and how are they functioning. The subsequent section provides information about the low-level design and UML diagram used for providing information about the model.

5.1 UML Diagram

UML (Unified Modelling Language) class diagram provides the internal functionality of the model. It depicts the various classes used in the software. It provides detail information about the internal structure of the class, what features they provide, their attributes and functioning. It depicts the relation among classes. It tells whether class is inheriting or is an abstract class and more details regarding it. In object oriented programing everything is dealt in classes. Classes is a container providing feature of object oriented programing called encapsulation. Hence, class is one of the most important feature. UML provides details about the class thereby providing structural information, how the system is modelled. UML class diagram enable to code the design.

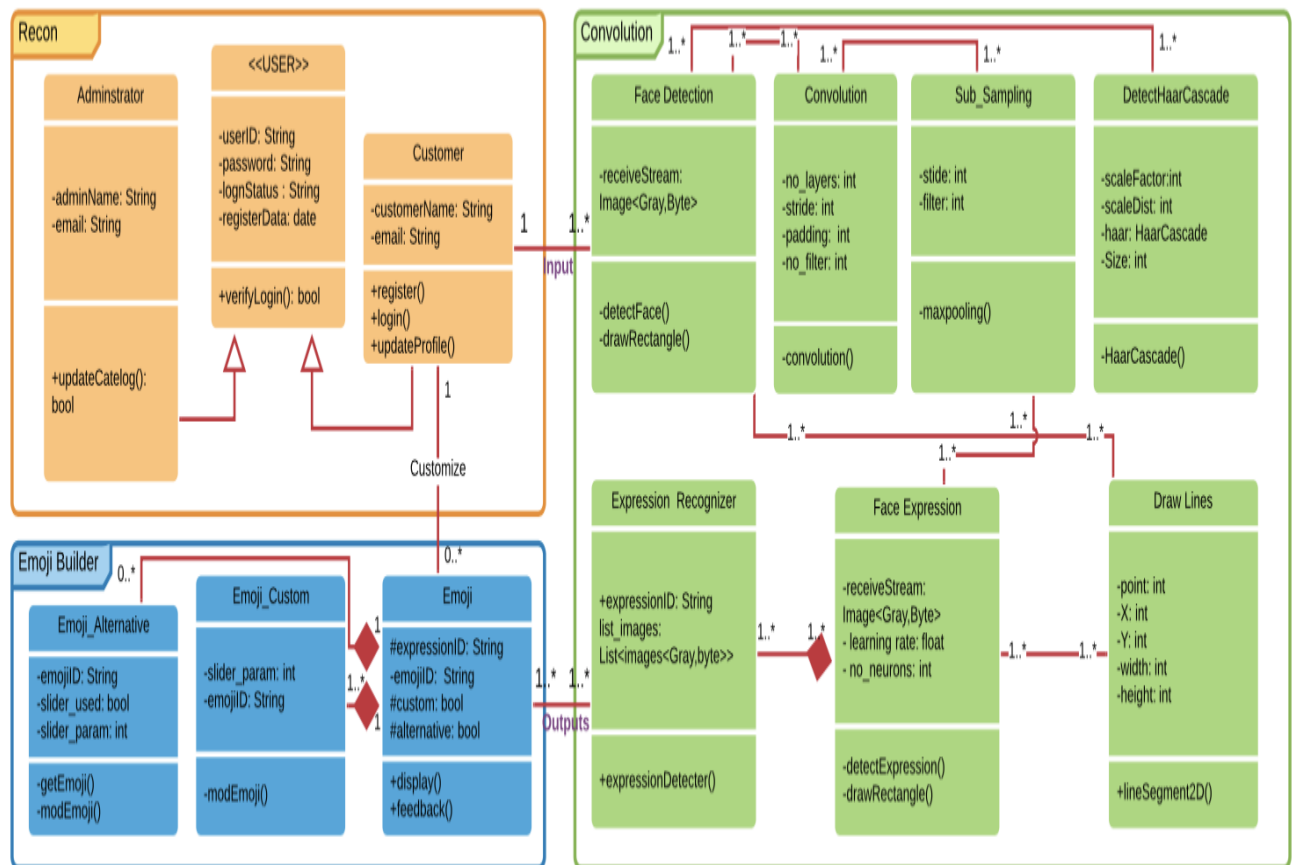


Figure 5-1 UML Class Diagram of MyEmoji

The system is majorly divided into two sub systems. The first one deals with convolution neural network and the second one deal with the creation and display of emoji.

5.1.1 Facial Image Analysis using Deep Learning

The expected outcome from this module is the recognised true facial expression of the given input image. In order for this to happen, the faces need to be detected in the image. Detection involves cropping image to such an extent that it only contains the biggest face in the input image. Haar cascade classifier is used to achieve this. Viola and Jones method makes it possible to detect the faces in a very quick manner. The main idea of this method is to identify the relative colour difference between two segments of the faces. For example, the eyes are darker than region of the cheek. These subtle characteristics of the face is used to make detection very efficient. Once the corpus contains only the face images, the dataset is ready to be used for training.

There are standard deep learning architectures such as AlexNet, LeNet, vGG, ImageNet which have been proved to be great models for Deep Learning applications. For this particular database, the appropriate model has to be chosen. This can be done by training each of these models and seeing which performs better. Keras API enables programmers to make and use these models in a very simple manner.

Another important analysis that has to be made is the number of epochs the model had to undergo to attain a good training. This can again be analysed by checking how the validation error is performing in comparison with training error. It is also important that after all, the test accuracy is what really matters.

5.1.2 Emoji Builder

There are seven classifications of emotion in the system viz. angry, sad, happy, disgust, neutral, surprise and fear.

Emoji Builder involves creation of emoji in all seven fields. For it several classes have been used with each category assigned its own class.

Emoji circle and eclipse for constructing different features of a face with provisions available to modify the features.

The created emoji is passed on for display. Emoji builder provides feature to display the output based on the input data obtained from the Deep Learning component.

The depicted emoji(s) were generated by the Emoji Builder component of the system.

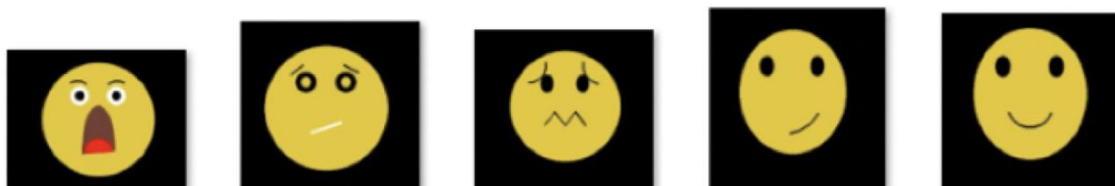


Figure 5-2 Emoji(s) of different Emotions

5.2 Structural Chart

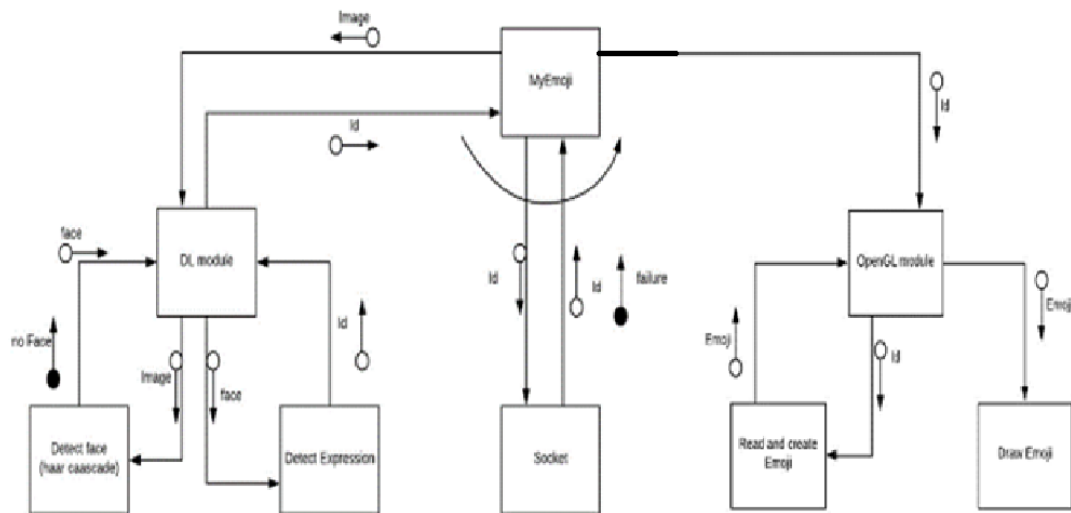


Figure 5-3 Structure Chart of MyEmoji

A Structure Chart in software engineering and organizational theory is a chart which shows the breakdown of a system to its lowest manageable levels. They are used in structured programming to arrange program modules into a tree. Each module is represented by a box, which contains the module's name. The tree structure visualizes the relationships between modules.

The structural chart of MyEmoji is shown in Figure 5-3. There are 3 sub modules. They are Deep learning, Socket and OpenGL modules. The Deep learning module is further divided into 2 sub-modules which are Detect-Face, and Detect Expression. The OpenGL module is divided into 2 sub modules which are read-and-create-emoji and draw-emoji.

5.3 Functional Description of Modules

The section the main 2 modules are further subdivided into four different functional modules. A flow chart depicting the flow of the process is depicted for each one of the functional modules. The working of the different modules involved are explained in subsequent sub sections.

5.3.1 Face Detection Module

The module detects face in the input image. If an input image has multiple faces in it, this module is used to identify the most prominent face of all the faces present in the image.

- **Purpose:** The purpose of the module to detect faces in the input images.
- **Functionality:** It explains how to detect face in the input image and on success pass the emotion id obtained.
- **Input:** Image
- **Output:** Detected Face
- **Flowchart:** The flowchart shown in the Figure 5-4 explains the processes involved in the detection of the face. The input image is uploaded to the system which analyze the frames for faces and after successful detection, pass the detected face for further processing.

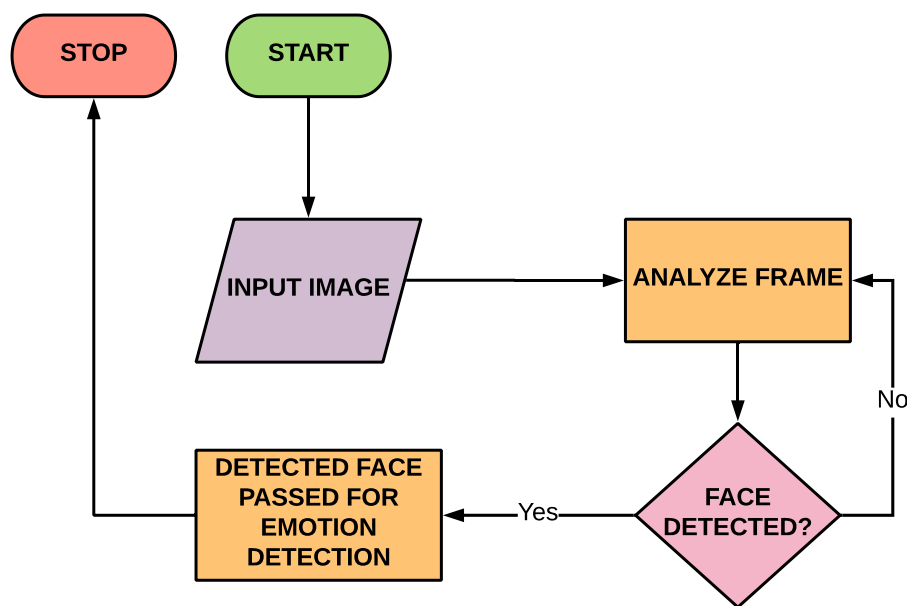


Figure 5-4 Modular Description of Face Detection

5.3.2 Emotion Recognition Module

The module recognizes the facial emotions.

- **Purpose:** The purpose of the module to recognize facial emotion present on the detected face.
- **Functionality:** The functionality of the module is to detect facial emotion and generated emotion ID associated with it.

- **Input:** Image with Detected Face
- **Output:** Emotion ID
- **Flowchart:** The flowchart shown in the Figure 5-5 explains the processes involved in the recognition of the face expression present in the image. The input image is received from the face detection module. After successful detection of the emotion, it generates the emotion ID based on the detected emotion.

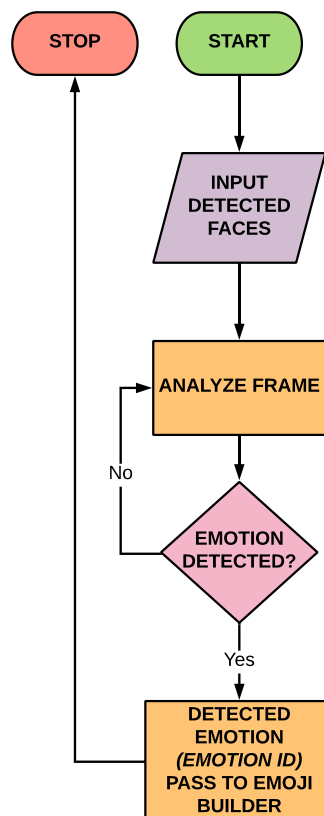


Figure 5-5 Modular Description of Emotion Recognition

5.3.3 Inter-Module Communication

The module transfers the emotion ID for its usage in the emoji building.

- **Purpose:** The purpose of the module to transfer data from client to server.
- **Functionality:** The module transfers emotion ID generated in the client system to the server for emoji building.
- **Input:** Emotion ID in client
- **Output:** Emotion ID in server
- **Flowchart:** The flowchart shown in the Figure 5-6 explains the processes involved in establishing connection between client and server and transfer of emotion ID from

client to server. The connection between client and server is not closed after successful transfer of an emotion ID to support multiple data transfers.

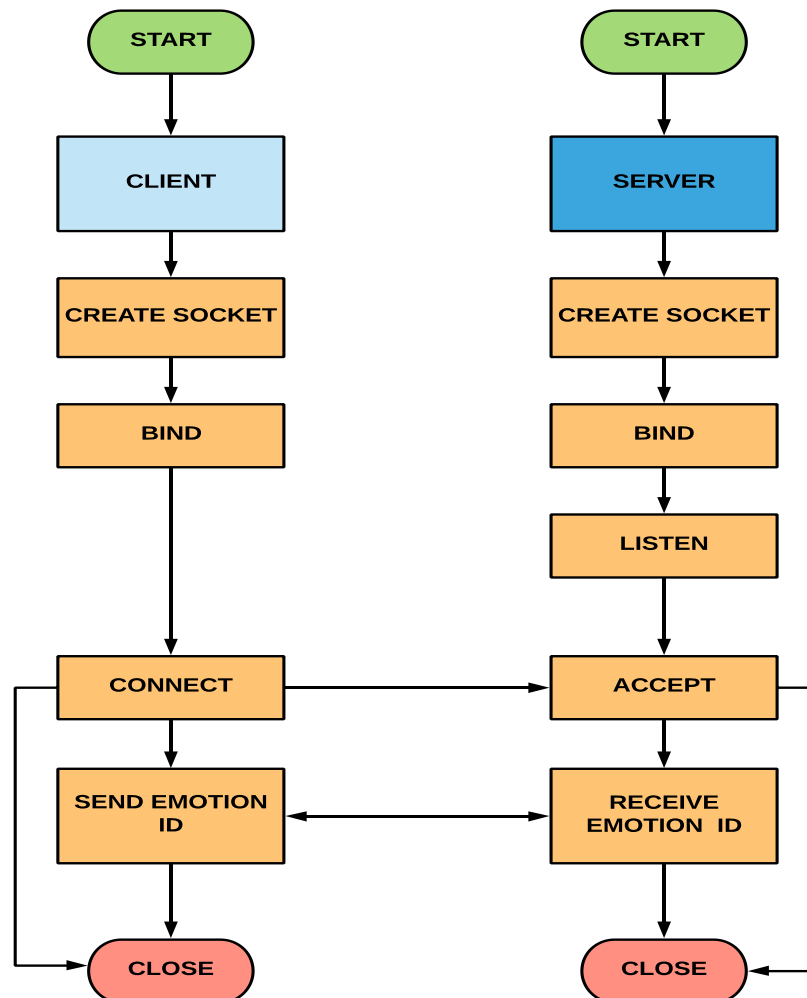


Figure 5-6 Modular Description of Inter-Module Communication

5.3.4 Emoji Builder Module

The module creates emoji and display based on an emotion ID.

- **Purpose:** The purpose of the module is to create and display emoji.
- **Functionality:** The functionality of the module is to create emoji(s) for all seven categories of emotion and display emoji(s) based on the emotion id received.
- **Input:** Emotion ID
- **Output:** Emoji
- **Flowchart:** The flowchart shown in the Figure 5-7 explains the processes involved in generation of the emoji and displaying the emoji(s) associated with the emotion

ID. The module keeps fetching the emotion ID and display it on the standard output device until the displaying window is closed. This mechanism reduces the overheads in running the module.

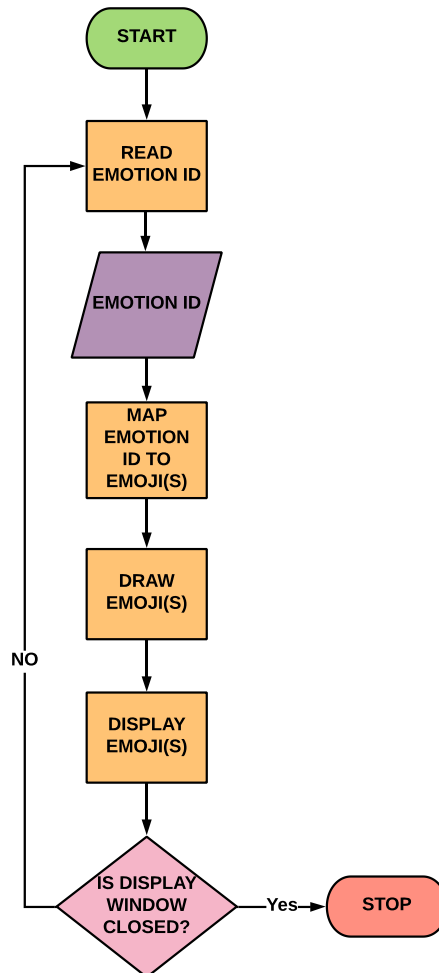


Figure 5-7 Modular Description of Emoji Builder

Chapter – 6

Implementation of MyEmoji

The chapter explains in detail the steps taken towards the implementation of the project. Implementation phase forms a significant phase in the project development. This phase the low level design discussed in the previous chapters is transformed into language specific programs such that all the requirements of the SRS are adhere to.

6.1 Programming language selection

The application required to use an open source library which could be used to deploy on GPU's /TPU's and allow easy prototyping. Keras is thus chosen as the appropriate library which provides both these features. It runs on top of the TensorFlow framework and thus inherits all its advantages. A large number of inbuilt libraries enables the programmer to concentrate on the logic, rather than the numerical computations. Keras is also supported with a plethora of documentation by its community.

6.2 Platform selection

To simulate an end-to-end application interaction, the Deep Learning code which is written in python (using keras) runs on one computer acting as back-end computing. Either windows, iOS, or Ubuntu can support the application provided that TensorFlow framework and Keras are installed along with python.

6.3 Code Conventions

Coding conventions are a set of guidelines which indicate the programming style, practices and methods for each aspect of the program which the programmer used adhere to. In this section the different conventions followed during the implementation of the project are mentioned.

6.3.1 Naming convention

Naming convention helps in keeping the code understandable by anyone who wants to analyse the application code. The conventions followed in this project are as follows: -

- *<model_name.py>*:- Separate classes are developed for each model

- camel Case for functions
- underscore naming scheme for image data variables
- 1 statement per line
- Each function follows a line gap

6.3.2 File organization

The files are organized according to the structured presented below:

Mymoji/

 Main.py

 LeNet.py

 AlexNet.py

 VggNet.py

 customModel.py

 makeNotebook.py

Data/

 ImageFolder/

 <img_name.png>

 ...

 ImageFolder224/

 <img_name.png>

 ...

 Images.npy

 Labels.npy

 Fer2013.csv

ModelWeights/

```
<model_name.h5>

....

Models/

<model_name.json>

....

Haarcascade_files/

<cascade.xml>

...

testImages/

<img_name.png>
```

6.3.3 Properties declaration

Standard declarations conventions are followed. Standard names are given which make it easy to understand the role of each entity declared. Multiple declarations per line are not allowed because of commenting and to reduce ambiguity.

6.3.4 Class declaration

There is a separate class declared for each model being considered in the application. Functions in each class are declared and named in a similar manner to enable easy understanding.

6.3.5 Comments

Comments are absolutely vital to keeping our code readable. Giving sensible names to types and variables is much better than using obscure names that is explained through comments. Single line comments are written using ‘#’ and multiline comments are written using ‘’’ in the python code. In the c++ code for the graphics component the single line comments are written in ‘//’ and the multi-line comments in ‘/*....*/’ form.

6.4 Difficulties Encountered and Strategies Used to Tackle

This section discusses the difficulties encountered in the development of the project. The main difficulties encountered were in the collection of data and the machine learning Algorithms.

6.4.1 Integration of components

Since the application has two modules, there is a requirement of communication between them. Socket programming was the approach chosen to send the emotion number from the back-end computer to the computer running the graphics program.

If both computers were running on the same operating system, the solution would be simple as expected. Since the graphics module is running on windows, it requires the use of WinSock API.

6.4.2 Synchronization between services

The application is composed of two different modules which are to be synchronized to obtain expected results. As every model that is being used is pre-trained, evaluating a facial expression takes very few milliseconds. The transfer between the 2 modules happens very fast since we are sending only on value, i.e., the expression number.

Chapter – 7

Software Testing of MyEmoji

For testing a software it is important that both unit testing and integration testing is done since both are vital for the application to perform as expected. This chapter discusses the test environment used, the results of the unit testing and the results obtained from the integration testing.

7.1 Test Environment

The list mentions the various environments used in the implementation of the project.

- OS: Windows/Linux
- Tool: Keras & GLEW/GLFW/GLM
- Version: Keras– 2.1.5, OpenGL– 3.2
- Dataset: FER-2013

7.1.2 Unit testing of Deep learning modules

Each model was tested so as to determine which model performed the best for the dataset – FER-2013. Table 7-1 depicts the testing of AlexNet Architecture.

Table 7-1 Testing AlexNet Architecture

| Sl. No | |
|------------------------------|---|
| Name of the Test case | Accuracy Testing |
| Feature being Tested | Epoch |
| Description | Finding Optimal Epoch |
| Sample Input | Images |
| Expected Output | Lower Epoch which gives higher accuracy |
| Actual Output | 14 |
| Remarks | Could not identify optimal Epoch |

Optimal epoch couldn't be identified since the relative analysis between validation error and test error was not evident in the extent of testing (20 epochs). Higher epochs training would require GPU access with bigger internal memory allowed. This was not provided by Google colab where all the models are trained for this application. Table 7-2 depicts optimum number of epochs for LeNet.

Table 7-2 Testing LeNet Architecture

| Sl. No | |
|------------------------------|---|
| Name of the Test case | Accuracy Testing |
| Feature being Tested | Epoch |
| Description | Finding Optimal Epoch |
| Sample Input | Images |
| Expected Output | Lower Epoch which gives higher accuracy |
| Actual Output | 8 |
| Remarks | Optimal Epoch found to be 8 and Accuracy of 58% |

LeNet architecture gave reliable results as the graph allowed conclusive analysis

Based on assuming best accuracy that can be reached as around 50-60% with standard architectures, an attempt was made to build a new model which can perform faster by using fewer layers in an optimized manner. The custom model gave an accuracy of 59% which concluded that the model is giving comparable results.

7.2.3 Unit Testing of the Emotion

For the verification of the architecture of the Deep Learning model each of the 7 emotions were individually tested. The output given by the model is in the form of numbers where 0 – Angry, 1 – Disgust, 2 – Fear, 3 – Happy, 4 – Sad, 5 – Surprise, and 6 – Neutral. Different input images are tested and the outputs were tested, the results for each emotion are displayed in the tables below.

Table 7-3 shows the results of testing the emotion happy against the image.

Table 7-3 Testing of Emotion – Happy

| Sl. No | |
|------------------------------|--|
| Name of the Test case | Emotion Testing |
| Feature being Tested | Emotion |
| Description | Verifying the correctness of the model |
| Sample Input | Image |
| Expected Output | 3 |
| Actual Output | 3 |
| Remarks | Image recognised correctly |



Table 7-4 shows the results of testing the emotion neutral against the image.

Table 7-4 Testing of Emotion – Neutral

| Sl. No | |
|------------------------------|--|
| Name of the Test case | Emotion Testing |
| Feature being Tested | Emotion |
| Description | Verifying the correctness of the model |
| Sample Input | Image |
| Expected Output | 6 |
| Actual Output | 6 |
| Remarks | Image recognised correctly |



Table 7-5 shows the results of testing the emotion Surprise against the image.

Table 7-5 Testing of Emotion – Surprise

| Sl. No | |
|------------------------------|--|
| Name of the Test case | Emotion Testing |
| Feature being Tested | Emotion |
| Description | Verifying the correctness of the model |
| Sample Input | Image |
| Expected Output | 5 |
| Actual Output | 5 |
| Remarks | Image recognised correctly |



Table 7-6 shows the results of testing the emotion Disgust against the image.

Table 7-6 Testing of Emotion – Disgust

| Sl. No | |
|------------------------------|--|
| Name of the Test case | Emotion Testing |
| Feature being Tested | Emotion |
| Description | Verifying the correctness of the model |
| Sample Input | Image |
| Expected Output | 1 |
| Actual Output | 1 |
| Remarks | Image recognised correctly |



Table 7-7 shows the results of testing the emotion Sad against the image.

Table 7-7 Testing of Emotion – Sad

| Sl. No | |
|------------------------------|--|
| Name of the Test case | Emotion Testing |
| Feature being Tested | Emotion |
| Description | Verifying the correctness of the model |
| Sample Input | Image |
| Expected Output | 4 |
| Actual Output | 4 |
| Remarks | Image recognised correctly |



Table 7-8 shows the results of testing the emotion Angry against the image.

Table 7-8 Testing of Emotion – Angry

| Sl. No | |
|------------------------------|--|
| Name of the Test case | Emotion Testing |
| Feature being Tested | Emotion |
| Description | Verifying the correctness of the model |
| Sample Input | Image |
| Expected Output | 0 |
| Actual Output | 0 |
| Remarks | Image recognised correctly |



Table 7-9 shows the results of testing the emotion Fear against the image.

Table 7-9 Testing of Emotion – Fear

| Sl. No | |
|------------------------------|--|
| Name of the Test case | Emotion Testing |
| Feature being Tested | Emotion |
| Description | Verifying the correctness of the model |
| Sample Input | Image |
| Expected Output | 2 |
| Actual Output | 2 |
| Remarks | Image recognised correctly |



7.3 Integration testing of the modules

The system consists of two sub module viz. Emoji Builder and Deep Learning. The two - modules are integrated to obtain a fully functional system.

The Emoji Builder module runs on a system with required specification and the Deep Learning module runs on a different platform on a different system.

Integration of the two modules are done through sockets. The Deep Learning module processes the data i.e. facial expression and send it to the Emoji Builder module where the corresponding emoji is depicted. This interaction is through sockets and file read and write.

The connection is properly tested for between the modules and behaviour of the connection was observed on passing different value. File handling is checked for bugs.

Mapping of data sent from Deep Learning layer to Emoji builder layer is checked for proper functionality of the system to resolve any issues that may arise during processing.

Finally the respective emoji is displayed based on the data sent by the Deep Learning layer. Some of the checks performed are: window creation, window positioning, and rasterization checks.

7.4 System testing

System is tested to check whether the application is working well as a whole.

Table 7-10 provides the false test case where the Expected output was supposed to be Happy but it turns out to be Fear.

Table 7-10 System Testing - Happy




| Sl. No | | |
|------------------------------|--|---|
| Name of the Test case | Emotion Recognition |  Test Image |
| Feature being Tested | Emotion | |
| Description | Validation of the Emotion |  |
| Sample Input | Image | |
| Expected Output | Happy |  |
| Actual Output | Fear | |
| Remarks | The dataset does not have sufficient baby faces. | Expected Output: Happy |

Table 7-11 provides the false test case where the Expected output was supposed to be Sad but it turns out to be Angry.

Table 7-11 System Testing - Sad

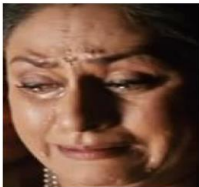


| Sl. No, | | |
|------------------------------|--|---|
| Name of the Test case | Emotion Recognition |  Test Image |
| Feature being Tested | Emotion | |
| Description | Validation of the Emotion |  |
| Sample Input | Image | |
| Expected Output | Sad |  |
| Actual Output | Angry | |
| Remarks | Inherent similarity between fear and sad emotion leads to the wrong result | Expected output: Sad |

Table 7-12 provides the true test case where the Expected output is happy and it turns out to be Angry.

Table 7-12 System Testing - Happy

| Sl. No, | |
|------------------------------|---------------------------|
| Name of the Test case | Emotion Recognition |
| Feature being Tested | Emotion |
| Description | Validation of the Emotion |
| Sample Input | Image |
| Expected Output | Happy |
| Actual Output | Happy |
| Remarks | Correct output |



Test Image

Actual output: Happy



Expected output: Happy

Table 7-13 provides the false test case where the Expected output was supposed to be fear but it turns out to be surprise.

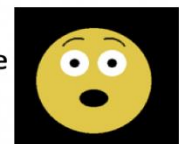
Table 7-13 System Testing - Fear

| Sl. No, | |
|------------------------------|--|
| Name of the Test case | Emotion Recognition |
| Feature being Tested | Emotion |
| Description | Validation of the Emotion |
| Sample Input | Image |
| Expected Output | Fear |
| Actual Output | Surprise |
| Remarks | Inherent similarity between fear and sad emotion leads to the wrong result |



Test Image

Actual output: Surprise



Expected output: Fear

Chapter – 8

Experimental Analysis and Results of MyEmoji

The chapter lists the results of the experiment conducted and the inferences that were made from the testing. The evaluation metrics have been listed and the results have been accordingly listed out in this chapter. The results were obtained from a combination of the information acquired through initial process discovery as well as statistics gathered from multiple runs of in the system. The input to the application is a facial image. The true output would be the actual expression of that face. The output we get is compared with the true output to check how often it is correct.

8.1 Evaluation Metric

The system was mainly designed to reduce human effort in a nearly deterministic process. There are two metrics which is involved in deciding whether the model is trained to a good extent or whether the model itself has to be changed:

- Validation error
- Validation accuracy

The train error and train accuracy cannot be used to judge the optimal epoch as it hasn't seen a new set of data yet. Validation accuracy is more reliable as it runs on the validation set in order to adjust the hyper-parameters. The accuracy is expected to stabilize once the hyper-parameters are tweaked.

The main evaluation metric in the first module of the project was the accuracy given by a particular architecture of the CNN model. Here there was a trade-off between the numbers of epochs to be selected vs. the training time of the model. As the increase in the number of epochs the training time was also increasing. To find an optimal epoch the model was tested for different number of epochs and based on this analysis an optimal epoch was identified and the corresponding accuracy was considered. For the second module of the project the time taken to perform the creation of the different emoji(s) was taken as the evaluation matrix.

8.2 Experimental Dataset

The experimental dataset in this project includes a large dataset of images categorised into the seven different emotions namely angry, sad, happy, fear, disgust, surprise and neutral. The dataset is called as Facial Expression Recognition Challenge (FERC-2013). The data consists of 48×48 pixel grayscale images of faces. The faces have been automatically registered so that the face is more or less centred and occupies about the same amount of space in each image. The dataset is in the form of a .csv file having 2 columns, one indicating the emotion and the other having the 48×48 number of pixel values of each image. The emotion are written in form of numbers from 0-6 where 0=Angry, 1=Disgust, 2=Fear, 3=Happy, 4=Sad, 5=Surprise, and 6=Neutral. The training set in the dataset consists of 28,709 example images and the validation set consists of 3,589 example images.

8.3 Performance Analysis

The analysis of the result is done through the accuracy given by each architecture for the corresponding number of epoch. The 3 different models used during the testing step gave a validation accuracy of around 58%-59%. A summary of this results is shown in table-2.

Table 8-1 Summary of the different architectures

| Architecture | Accuracy | Number of Epochs |
|---------------------|----------|------------------|
| LeNet | 58% | 8 |
| AlexNet | 57% | 14 |
| Custom architecture | 59% | 7 |

Each model was analysed by generating a graph of epoch vs. accuracy. Higher accuracy with lower epoch was the aim of the application. Validation error – VE, Training error - TE

The model's evaluation is based is based on 3 categories: -

- Under-fitting – both VE and TE are high
- Good-fit – VE is little higher that TE but both are low
- Over-fitting – TE is low and VE is high

8.4 Inference from the Result

Figure 8-1 shows model accuracy and model loss analysis of the LeNet architecture. Since the LeNet architecture consists of only 3 layers (2 convolution layers followed by max-pooling layers and 1 fully connected layer) the training time for the 20 epochs is relatively lesser. As it can be seen from the image, the training accuracy regularly increases as the number of epochs increases whereas the validation accuracy reaches a threshold value near the 8 epoch mark and further increase in the number of epochs doesn't increase the value of the validation accuracy by a significant margin. The similar analysis can be shown through the model loss graph which is the mirror image of the accuracy graph.

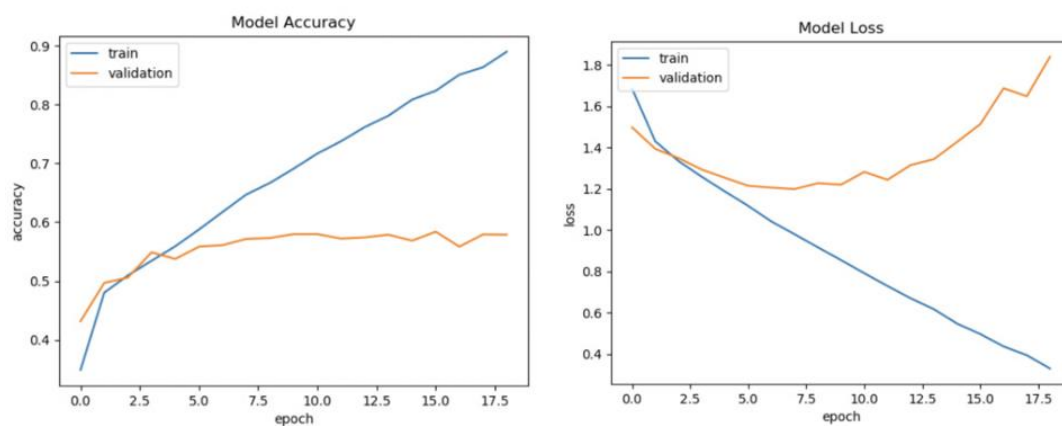


Figure 8-1 Analysis of the LeNet Architecture

Figure 8-2 shows model accuracy and model loss analysis of the AlexNet architecture. Since the AlexNet architecture has more number of layers the training time is relatively more as the number of epochs increases. As seen in the Figure 8-2, the training accuracy regularly increases as the number of epochs increases whereas the validation accuracy is inconsistent as the number of epochs increases. At few points the accuracy falls as the number of epochs increases and at other points the accuracy increases as the number of epochs increases. For this reason in the 20 epochs tested the optimal epoch is found to be 14, as at this point the maximum accuracy is achieved.

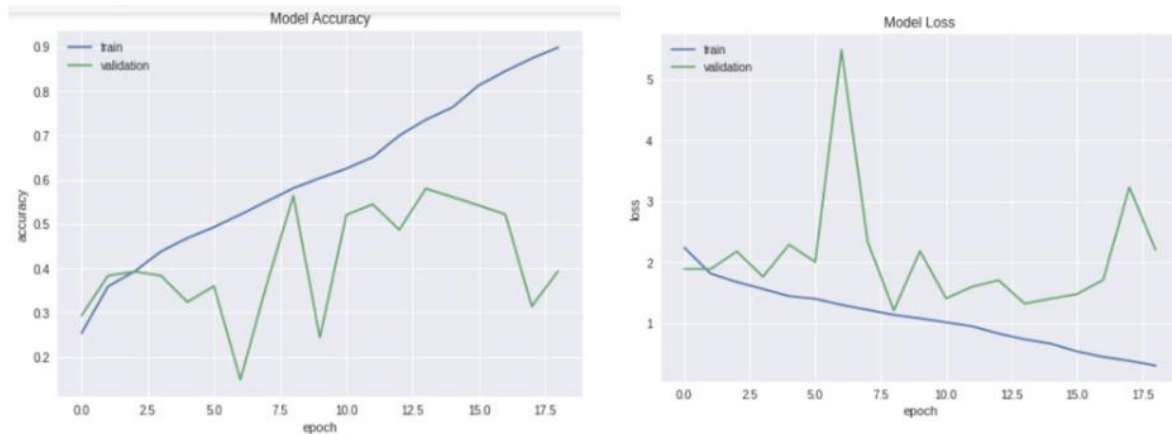


Figure 8-2 Analysis of the AlexNet Architecture

Figure 8-3 shows model accuracy and model loss analysis of the Custom architecture. The validation accuracy generally increases with the increase in the number of epochs and the optimal epoch is selected to be 5.

Custom Architecture

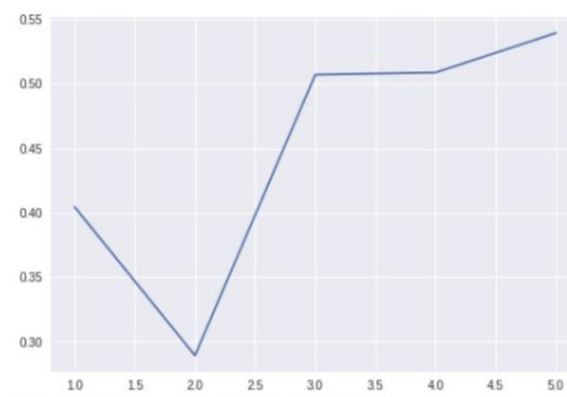


Figure 8-3 Analysis of the Custom Architecture

Chapter – 9

Conclusion

The proposed system efficiently reduces human effort in the process of selection of a relevant emoji. The system provides a user friendly approach for the selection of the emoji, by suggesting emoji(s) to the user based on his/her facial expression. Further in the era of technology, where there are continuous updates to products, new integrations and releases, a process with reduced time and effort is highly appreciated and it has more scope for usage.

The system can further be enhanced to use in totally different approach as a real time feedback mechanism by analysing the facial expression of the customers on spot to see the behaviour and even observe the liking of the customer. The proposed system will provide great results in the real time analysis of expression and usage in chat services.

9.1 Limitations of the project

Identifying limitations of the current application helps in developing the next phase/version of the application. Few of the identified limitations are listed below:

- Custom architecture is suitable only for FERC-2013 data set.
- More false positive errors encountered.
- Lack of better rendering technique for emoji and customization options.
- The system gives better result when a photo of an individual person is used rather than a group of people.

9.2 Future Enhancement

Based on the identified limitations of the current application, future enhancements, both in terms of performance improvements and addition of new features is laid out. Few of them are listed below:

- A slider should appear once the emoji suggestion is shown. The user should be given the option to increase or decrease the intensity of the expression.
- Increase the accuracy in the custom architecture by training it over multiple data sets.

- A lightweight mobile application can be used to collect the input image from the user in real time. The app can make use of the front camera to do the same.
- Obtain a better result when the system is used on an image containing a group of people.

References

- [1] T. Ahsan, T. Jabid, and U.P. Chong, “*Facial expression recognition using local transitional pattern on gabor filtered facial images*”, IETE Technical Review., 30, (1), pp. 47-52, 2013.
- [2] D. Ciresan, U. Meier, J. Schmidhuber, “*Multi-column deep neural networks for image classification*”, In Computer Vision and Pattern Recognition (CVPR) IEEE Conference, pages 3642-3649, 2012.
- [3] A. Krizhevsky, I. Sutskever, G. E. Hinton, “*Image classification with deep convolutional neural networks*”, In Advances in neural information processing systems, pages 1097-1105, 2012.
- [4] Shabab Bazrafkan¹, Tudor Nedelcu¹, Pawel Filipczuk, “*Deep Learning for facial expression recognition*”, IEEE International Conference on Consumer Electronics (ICCE), 38, (4), pp. 551–558, Jan 2017.
- [5] Dilbag Singh, “*Human Emotion Recognition System*”, I.J. Image, Graphics and Signal Processing, 8, pp. 50-56, 2012.
- [6] Denis Sokolov and Mikhail Patkin, “*Real-Time Emotion Recognition on Mobile Devices*” 13th IEEE International Conference on Automatic Face & Gesture Recognition, 2018.
- [7] Yiliang Xie, Hongyuan Jin, and Eric C.C. Tsang, “*Improving the Lenet with batch normalization and online hard example mining for digits recognition*”, International conference on wavelet analysis and pattern recognition, 2017.
- [8] Lisha Xiao, and Qin Yan, Scene, “*Classification with Improved AlexNet Model*”, 12th International Conference on Intelligent Systems and Knowledge Engineering (ISKE), 2017.
- [9] Matthew Zeiler and Rob Fergus, “*Visualizing and Understanding Convolution Networks*”, IEEE Conference on Computer Vision and Pattern Recognition, 2013.
- [10] Kaiming He, Xiangyu Zhang, Shaoqing Ren and Jian Sun, “*Deep Residual Learning for Image Recognition*”, IEEE Conference on Computer Vision and Pattern Recognition, 2016.
- [11] Ke Shan, Junqi Guo, Wenwan You, “*Automatic facial expression recognition based on a deep convolutional-neural-network structure*”, IEEE 15th International

- Conference on Software Engineering Research, Management and Applications (SERA), 48, (2), pp. 224-228 2017.
- [12] Mohsen Davarynejad, Sobhan Davarynejad, Jos Vrancken, Jan van den Berg. "*Granular value-function approximation for road network traffic control*", 2010 International Conference on Networking, Sensing and Control (ICNSC), 2010.
- [13] Dinh Viet Sang, Nguyen Van Dat, Do Phan Thuan. "*Facial expression recognition using deep convolutional neural networks*", 2017 9th International Conference on Knowledge and Systems Engineering (KSE), 2017.
- [14] Ionescu, Radu Tudor, Andreea-Lavinia Popescu, Marius Popescu, and Dan Popescu. "*BiomassID: A biomass type identification system for mobile devices*", Computers and Electronics in Agriculture, 2015.
- [15] Vibha. V. Salunke, C.G. Patil. "*A New Approach for Automatic Face Emotion Recognition and Classification Based on Deep Networks*", 2017 International Conference on Computing, Communication, Control and Automation (ICCUBE), 2017.
- [16] Khaled S. Younis, Waed Ayyad, Abdallah Al- Ajlony. "*Embedded system implementation for material recognition using Deep learning* ", 2017 IEEE Jordan Conference on Applied Electrical Engineering and Computing Technologies (AEECT), 2017.
- [17] Jia Xiang, Gengming Zhu. "*Joint Face Detection and Facial Expression Recognition with MTCNN*", 2017 4th International Conference on Information Science and Control Engineering (ICISCE), 2017.
- [18] Rosa Delima, Febrian Galih, and Argo Wibowo, "*Development of Crop and Farmer Activity Information System*", Researchers World: Journal of Arts, Science and Commerce, 2017.
- [19] C. L. Lisetti, and D. E. Rumelhart, "*Facial Expression Recognition Using a Neural Network*", In Proceedings of the 11th International FLAIRS Conference, pp. 328—332, 2004.
- [20] C. Busso, Z. Deng, S. Yildirim, M. Bulut, C.M. Lee, A. Kazemzadeh, S.B. Lee, U. Neumann, and S. Narayanan "*Analysis of Emotion Recognition using Facial Expression*" , Speech and Multimodal Information, ICMI'04, PY, USA, 2004.

Appendix

```
ksav@ksav-Inspiron-7520:~/Desktop/MPImp$ python main.py
/home/ksav/anaconda3/lib/python3.6/site-packages/h5py/__init__.py:36: FutureWarning: Conversion of the second argument of 'issubdtype' from 'float' to '
np.floating' is deprecated. In future, it will be treated as 'np.float64 == np.dtype(float).type'.
  from ._conv import register_converters as _register_converters
Using TensorFlow backend.

-----Emotion Detection using DL-----
Choose Model:
1.LeNet-5
2.AlexNet
3.VggNet(could not train. Please don't choose)
4.Custom Model
Enter choice:
1

What do you intend to do?
1.Detect Emotion
2.Analyze Model Performance
Enter choice:
1
Which image to load?
sad9
loaded model from disk
Sad
4
ksav@ksav-Inspiron-7520:~/Desktop/MPImp$
```

Figure A-1 Emotion detection using LeNet

The user is given the option to choose the model through which he wants the emotion to be detected. From this point, either the pertained model can be used or the user is allowed to train the model himself before going for emotion detection.

On choosing Detect Emotion option the input image name is entered. The detected emotion is displayed along with its emotion id. Simultaneously, this emotion id is sent over to the computer which is running the graphics module using socket programming.


```

File Edit View Search Terminal Help
ksav@ksav-Inspiron-7520:~/Desktop/MyEmoji$ python main.py
/home/ksav/anaconda3/lib/python3.6/site-packages/h5py/__init__.py:36: FutureWarning: Conversion of the second argument of 'issubdtype' from 'float' to 'np.float64' is deprecated. In future, it will be treated as 'np.float64 == np.dtype(float).type'.
  from ._conv import register_converters as _register_converters
Using TensorFlow backend.

-----Emotion Detection using DL-----
Choose Model:
1.LetNet-5
2.AlexNet
3.VggNet(could not train. Please don't choose)
4.Custom Model
Enter choice:
2

What do you intend to do?
1.Detect Emotion
2.Analyze Model Performance
Enter choice:
1

Which image to load?
happy?
loaded model from disk
Happy
1

```

Figure A-2 Emotion detection using AlexNet

```

File Edit View Search Terminal Help
ksav@ksav-Inspiron-7520:~/Desktop/MyEmoji$ python main.py
/home/ksav/anaconda3/lib/python3.6/site-packages/h5py/__init__.py:36: FutureWarning: Conversion of the second argument of 'issubdtype' from 'float' to 'np.float64' is deprecated. In future, it will be treated as 'np.float64 == np.dtype(float).type'.
  from ._conv import register_converters as _register_converters
Using TensorFlow backend.

-----Emotion Detection using DL-----
Choose Model:
1.LetNet-5
2.AlexNet
3.VggNet(could not train. Please don't choose)
4.Custom Model
Enter choice:
4

What do you intend to do?
1.Detect Emotion
2.Analyze Model Performance
Enter choice:
1

Which image to load?
surprised?
loaded model from disk
Surprise
5

```

Figure A-3 Emotion detection using custom model

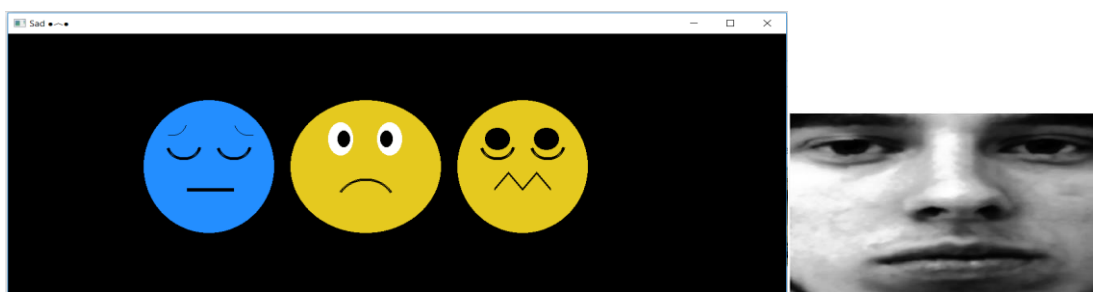


Figure A-4 Sad Emoji(s)

Sad faces are rendered when the emotion diagnosed was sad.

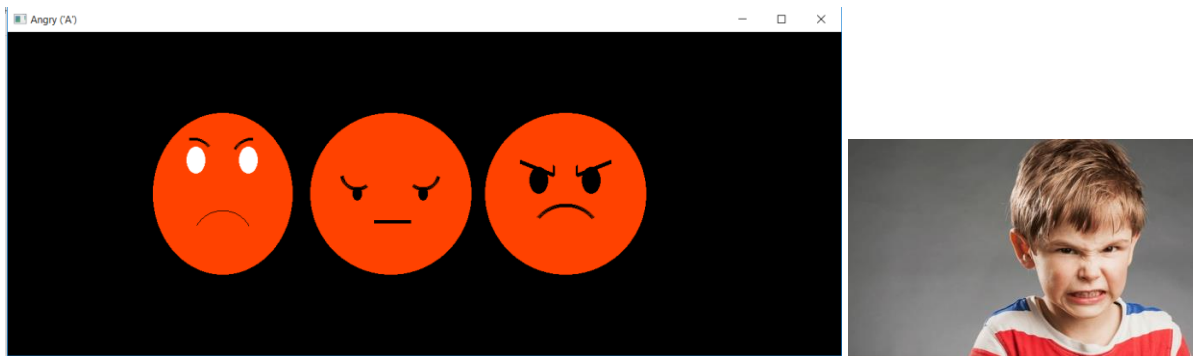


Figure A-5 Angry Emoji(s)

Angry faces are rendered when the emotion diagnosed was angry.

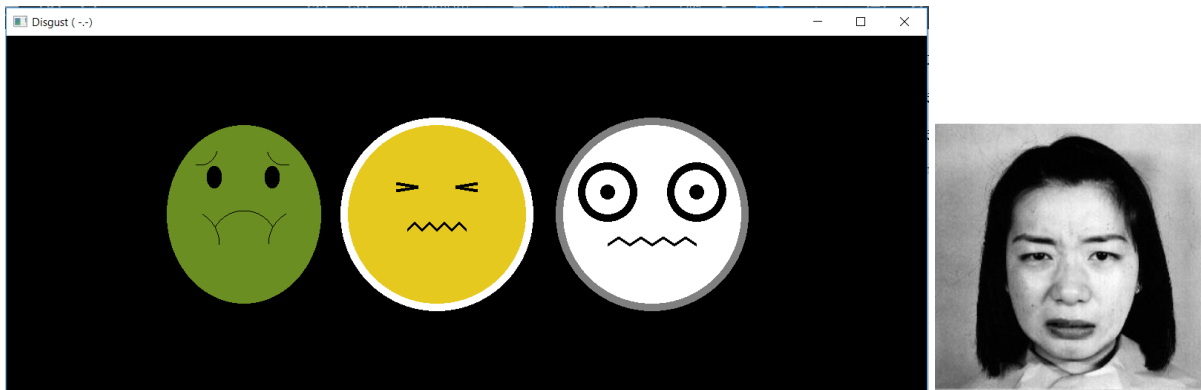


Figure A-6 Disgust Emoji(s)

Disgust faces are rendered when the emotion diagnosed shows disgust.

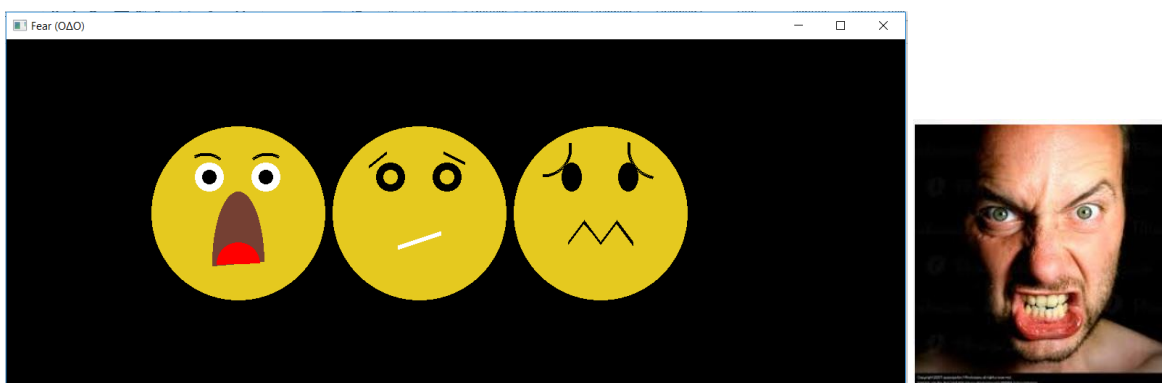


Figure A-7 Fear Emoji(s)

Fear faces are displayed when the module diagnosed emotion as fear.

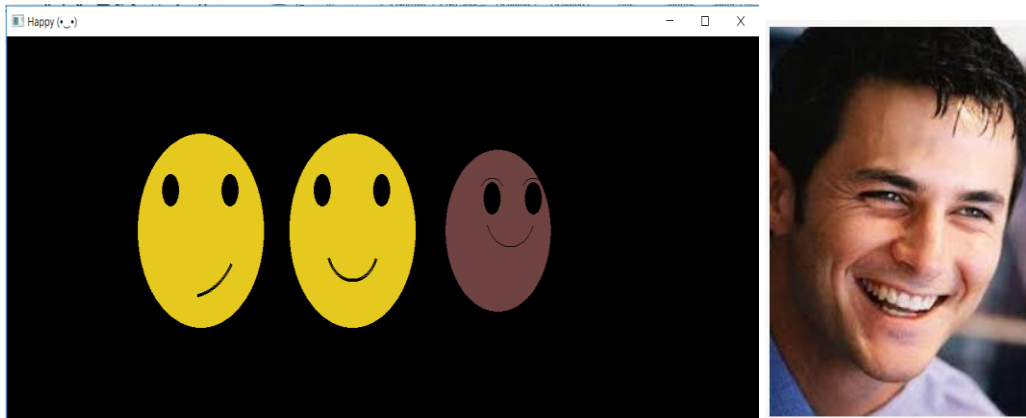


Figure A-8 Happy Emoji(s)

Happy faces are displayed when the module diagnosed emotion as happy or exited.

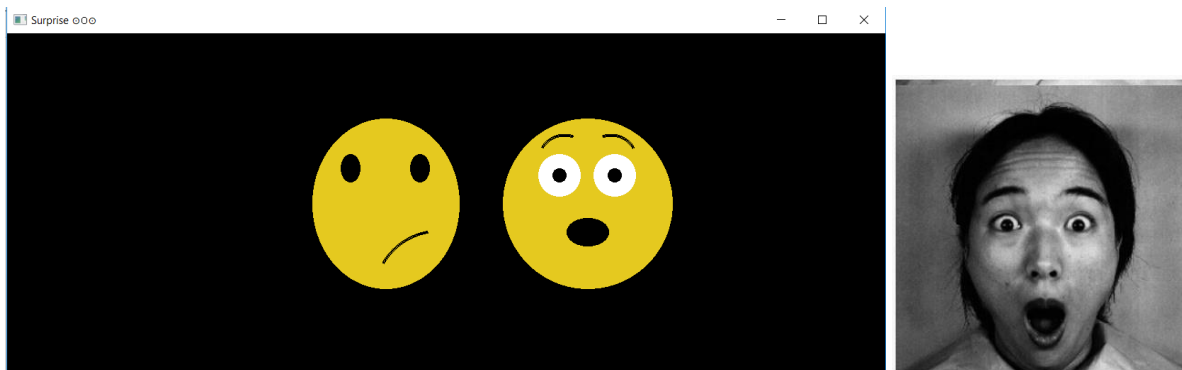


Figure A-9 Surprise Emoji(s)

Surprise faces are displayed when the module diagnosed emotion as surprise or shock.

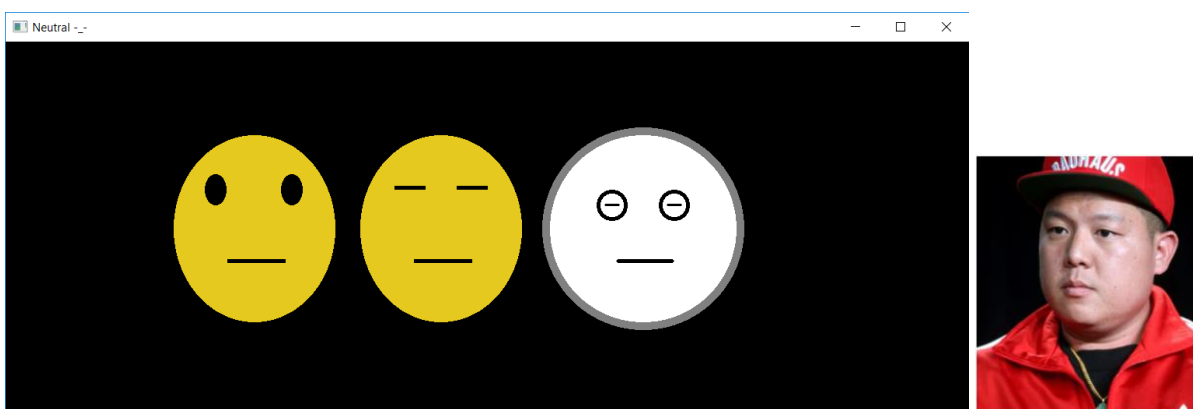


Figure A-10 Neutral Emoji(s)

Neutral Emoji(s) are rendered when the emotion diagnosed was neutral.

ORIGINALITY REPORT

8%

SIMILARITY INDEX

3%

INTERNET SOURCES

5%

PUBLICATIONS

5%

STUDENT PAPERS

PRIMARY SOURCES

1

Submitted to University of Florida

Student Paper

1%

2

www.seenets.com

Internet Source

1%

3

Dinh Viet Sang, Nguyen Van Dat, Do Phan Thuan. "Facial expression recognition using deep convolutional neural networks", 2017 9th International Conference on Knowledge and Systems Engineering (KSE), 2017

Publication

1%

4

Submitted to University of Southern California

Student Paper

1%

5

huifangseo.com

Internet Source

1%

6

Submitted to Panola College

Student Paper

1%

7

Submitted to Icon College of Technology and Management

Student Paper

<1%

8

Mohsen Davarynejad, Sobhan Davarynejad, Jos Vrancken, Jan van den Berg. "Granular value-function approximation for road network traffic control", 2010 International Conference on Networking, Sensing and Control (ICNSC), 2010

Publication

<1 %

9

prezi.com
Internet Source

<1 %

10

Ionescu, Radu Tudor, Andreea-Lavinia Popescu, Marius Popescu, and Dan Popescu. "BiomassID: A biomass type identification system for mobile devices", Computers and Electronics in Agriculture, 2015.

Publication

<1 %

11

Submitted to Fresno Pacific University
Student Paper

<1 %

12

Submitted to The University of Manchester
Student Paper

<1 %

13

Vibha. V. Salunke, C.G. Patil. "A New Approach for Automatic Face Emotion Recognition and Classification Based on Deep Networks", 2017 International Conference on Computing, Communication, Control and Automation (ICCUBEA), 2017

Publication

<1 %

| | | |
|----|--|------|
| 14 | "Neural Information Processing", Springer Nature, 2017 Publication | <1 % |
| 15 | web.it.kth.se Internet Source | <1 % |
| 16 | sir.stikom.edu Internet Source | <1 % |
| 17 | Khaled S. Younis, Waed Ayyad, Abdallah Al-Ajlony. "Embedded system implementation for material recognition using deep learning", 2017 IEEE Jordan Conference on Applied Electrical Engineering and Computing Technologies (AEECT), 2017 Publication | <1 % |
| 18 | Submitted to University of Surrey Student Paper | <1 % |
| 19 | Jia Xiang, Gengming Zhu. "Joint Face Detection and Facial Expression Recognition with MTCNN", 2017 4th International Conference on Information Science and Control Engineering (ICISCE), 2017 Publication | <1 % |
| 20 | Rosa Delima -, Febrian Galih -, Argo Wibowo -. "DEVELOPMENT OF CROP AND FARMER ACTIVITY INFORMATION SYSTEM", Researchers World : Journal of Arts, Science | <1 % |

and Commerce, 2017

Publication

21

homepage.cs.uiowa.edu

Internet Source

<1%

Exclude quotes Off

Exclude matches Off

Exclude bibliography On