Facial Expression Recognition System

Report submitted to

Rajiv Gandhi University of Knowledge Technologies,

Srikakulam. for the fulfilment of mini project

of

Bachelor of Technology

in Computer Science and Engineering

by

M. Laxminarayana (S160307)

G. Rajesh (S160215)

Y. Sai Krishna (S160821)

K. E. Padma Kumar (S160499)

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING

RAJIV GANDHI UNIVERSITY OF KNOWLEDGE TECHNOLOGIES, SRIKAKULAM MAY 2021

RAJIV GANDHI UNIVERSITY OF KNOWLEDGE TECHNOLOGIES

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING

(A.P. Government Act 18 of 2008)

RGUKT-Srikakulam, Srikakulam Dist – 532402

Declaration

We certify that

a. The work contained in this report is original and has been done by us under the guidance of my supervisor(s).

b. The work has not been submitted to any other Institute for any degree or diploma.

c. We have followed the guidelines provided by the Institute in preparing the report.

d. We have conformed to the norms and guidelines given in the Ethical Code of Conduct of the Institute.

e. Whenever We have used materials (data, theoretical analysis, figures, and text) from other sources, We have given due credit to them by citing them in the text of the report and giving their details in the references. Further, We have taken permission from the copyright owners of the sources, whenever necessary.

M. Laxminarayana (S160307)

G. Rajesh (S160215)

Y. Sai Krishna (S160821)

K. E. Padma Kumar (S160499)

RAJIV GANDHI UNIVERSITY OF KNOWLEDGE TECHNOLOGIES

DEPARTMENT OF COMPUTER SCIENCE ENGINEERING

(A.P. Government Act 18 of 2008)

RGUKT-Srikakulam, Srikakulam Dist – 532402

Certificate

This is to certify that the Dissertation Report entitled, "**Facial Expression Recognition System**" submitted by **Mr. M.Laxminarayan, Mr. G.Rajesh, Mr. Y.Sai Krishna, Mr.K.E.Padma Kumar** to Rajiv Gandhi university of Knowledge Technologies, Srikakulam, India, is a record of Bonafide Project work carried out by him/her under my/our supervision and guidance and is worthy of consideration for the fulfilments of mini-project of Bachelor of Technology in computer Science and Engineering of the Institute.

Mr.T. Anil Kumar	**Examiner**
Project Supervisor	Project Examiner
Faculty Dept. of CSE	Faculty Dept. of CSE
RGUKT IIIT Srikakulam	**RGUKT IIIT Srikakulam**

Date:

ACKNOWLEDGEMENT

We would like to express our profound gratitude and deep regards to our guide ***Mr.T. Anil Kumar*** for his exemplary guidance, monitoring and constant encouragement to us throughout this semester. We shall always cherish the time spent with him during the course of this work due to the invaluable knowledge gained in the field of Deep Learning.

We are extremely grateful for the confidence bestowed in us and entrusting our project entitled “**Facial Expression Recognition System”.**

We express gratitude to Mr. Ch.Satish Kumar (HOD of CSE) and other faculty members for being source of inspiration and constant encouragement which helped us in completing the project successfully.

Finally, yet importantly, we would like to express our heartfelt thanks to our beloved God and parents for their blessings, our friends for their help and wishes for the successful completion of this project.

M. Laxminarayana (S160307)

G. Rajesh (S160215)

Y. Sai Krishna (S160821)

K. E. Padma Kumar (S160499)

ABSTRACT

The project Facial Expression Recognition System focuses on providing the information regarding the emotional state of a person by analyzing his facial expression using image processing and deep learning techniques. Human facial expressions convey a lot of information visually rather than articulately. Facial expression recognition plays a crucial role in the area of human and machine interaction. An automatic facial expression recognition system has many applications including human behavior understanding, detection of mental disorders, and synthetic human expressions. Recognition of facial expression by computer with a high recognition rate is still a challenging task. Two popular methods utilized mostly in the literature for the automatic Facial Expression Recognition (FER) systems are based on Geometry and Appearance. Facial Expression Recognition usually performed in four stages. They are Pre-Processing, Face Detection, Feature Extraction, and Expression Classification.

In this project we will apply two deep learning methods - Convolution Neural Networks (CNN) and Transfer Learning to develop a model to identify the Seven key human emotions: Anger, Fear, Disgust, Happy, Sad, Surprise, and Neutrality. By using this model can be various aspects like Automated Security Systems, Mental State Identification of Person, Enhanced Image Capturing.

Keywords: Facial Expression Recognition Systems, Convolutional Neural Networks, Transfer Learning

CONTENTS

1. INTRODUCTION

1.1 Problem Statement

Human facial expressions can be classified into 7 basic emotions: happy, sad, surprise, fear, anger, disgust, and neutral. Facial expressions are expressed through activation of specific set of facial muscles. These expressions are subtle, yet complex signals in an expression contain abundant amount of information about our state of mind. We can use this information to find out a customer interest on a product, A doctor can provide better treatment to his patient, A teacher can deliver better lectures to his/her students.

In Deep learning, Convolutional neural network (CNN) is a neural network that has one or more convolutional layers and are used mainly for image processing and classification. It contains neurons which works in its own receptive field and are connected to other neurons in a way that they cover the entire visual field. And Transfer Learning is a machine learning methodology where we use a pre-trained model as the starting point of the model that we create to solve our own problem. Using These two techniques we are trying to classify the 7 basic emotions of a human being.

1.2 Motivation of the Project

Humans are well-trained in reading the emotions of others, in fact, at Just 14 months old, babies can already tell the difference between happy and sad. In the Current Technological World everyone is using electronic devices like mobiles, computers etc. more than 8 hours a day.

Can we make our computers to understand our emotions? To answer the question, we designed a deep learning neural network that gives machines the ability to make inferences about our emotional states. In other words, “we give them eyes to see what we can see’’.

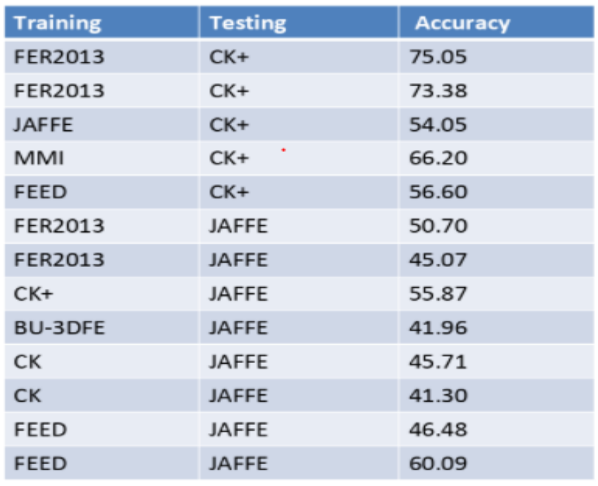
Making a computer that understand Human emotions will bring a drastic change. It can save lives of people by identifying who are being depressed emotionally and prevent them from committing suicide.

1.3 Limitations of the Project

Poor image quality limits the facial emotion recognition’s accuracy. Small image sizes make it difficult to recognize a face. Different angles of a face can affect the reliability of the FER system.

1.4 Existing System

The existing models which are trained on various data sets and methodologies are in the below table. Each of them is having different prediction accuracy.



Training	Testing	Accuracy
FER2013	CK+	75.05
FER2013	CK+	73.38
JAFFE	CK+	54.05
MMI	CK+	66.20
FEED	CK+	56.60
FER2013	JAFFE	50.70
FER2013	JAFFE	45.07
CK+	JAFFE	55.87
BU-3DFE	JAFFE	41.96
CK	JAFFE	45.71
CK	JAFFE	41.30
FEED	JAFFE	46.48
FEED	JAFFE	60.09

1.5 Proposed System

The proposed model will yield the highest prediction accuracy than the existing highest accuracy (FER 2013 75.05)

2. LITERATURE REVIEW

2.1 Study of Related Papers

As per various literature reviews, it is confirmed that for implementing this project four basic steps are required to be performed.

I. Pre-processing

II. Face detection

III. Facial feature extraction

IV. Emotion classification

❖ Pre-processing

Pre-processing refers to the transformations applied to the data before feeding it to the model. It includes reducing the noise, rescaling the data, and geometric transformations.

I. Reducing the noise

In image processing, we use a technique called Gaussian blur also known as Gaussian smoothing to reduce the noise in the image.

II. Rescaling data

Rescaling the data helps the optimizers plots to be less skewed and converge fast. We call it as **Normalization**. In our case we divide each value with the range (255-0 = 255) to make sure the data is on similar scale.

III. Geometric transformations

Using augmentation techniques like rotating, shearing, scaling, cropping, flipping, padding, and rotating etc...

❖ Face detection

We use Harr cascade algorithm. It is an object detection algorithm used to detect the faces in the images or real-time videos. The algorithm uses edge or line detection features proposed by Viola and Jones.

❖ Facial feature extraction

Feature extraction includes several convolution layers followed by max-pooling and an activation function. In the starting layers basic features like lines and edges are extracted and in the next layers certain complex features are extracted and so on. As the layer's depth increases the complexity of the features also increases.

❖ Emotion Classification

Expression classification is the main part where our trained model comes into picture.

Our model attempts to classify the expression in the give input image based on the features extracted from that image.

2.2 Technology Overview

2.2.1 Google Colaboratory

Colab is a free notebook environment that runs entirely in the cloud. It lets you and your team members edit documents, the way you work with Google Docs. Collab supports many popular machine learning libraries which can be easily loaded in your notebook. Google is quite aggressive in AI research. Over many years, Google developed AI framework called TensorFlow and a development tool called Collaboratory. Today TensorFlow is open-sourced and since 2017, Google made Colaboratory free for public use. Collaboratory is now known as Google Colab or simply called as Colab.

Another attractive feature that Google offers to the developers is the use of GPU. Collab supports GPU and it is free. But in the free version of colab the session expires too often so to develop a system like FER you must train the model hours and hours and you also need to access the fastest GPU's. So, taking a pro membership is mandatory for the work like this.

The introduction of Collab has eased the learning and development of machine learning applications.

Steps to create a new Colab Notebook

1. Open Google Chrome and search for "Google Collaboratory".

2. Open "https://colab.research.google.com".

3.Click on file at the header section and click on new notebook.

4. New colab notebook will be opened.

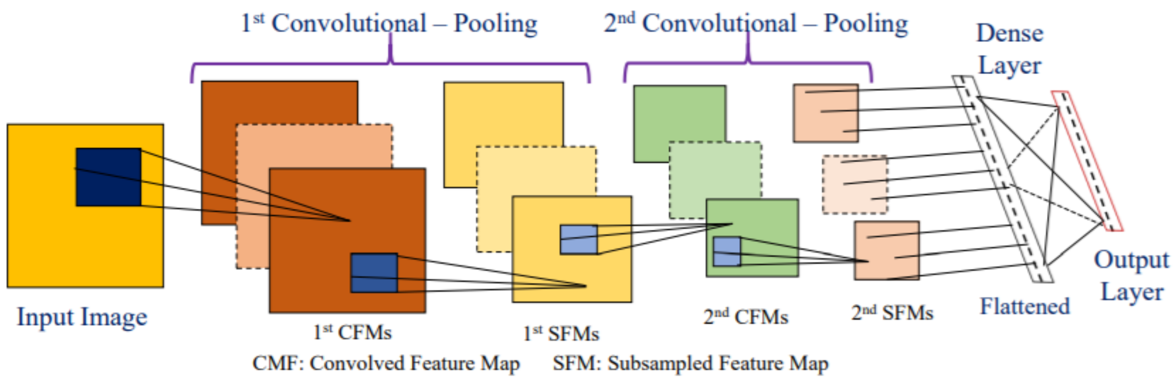
5. Now Start coding in python.

6. You can save the code and download it to your device or else save it to Drive or GitHub.

7. Click on file at the header section, then you will see “Save a copy in Drive”, “Save a copy in GitHub” or else download as. ipynb or .py.

3. IMPLEMENTATION

Convolutional neural network (CNN), a class of artificial neural networks that has become dominant in various computer vision tasks. CNN is designed to automatically and adaptively learn spatial hierarchies of features through backpropagation by using multiple building blocks, such as convolution layers, pooling layers, and fully connected layers.

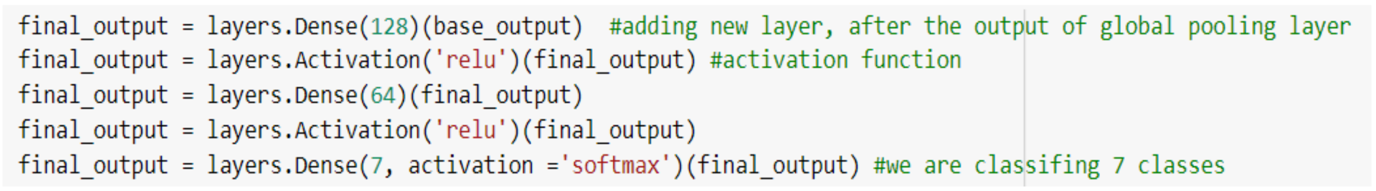

1st Convolutional – Pooling
2nd Convolutional – Pooling
Dense Layer
Output Layer
Input Image
1st CFMs
1st SFMs
2nd CFMs
2nd SFMs
Flattened
CMF: Convolved Feature Map
SFM: Subsampled Feature Map

architecture of a convolutional neural network with two convolutional-pooling layers

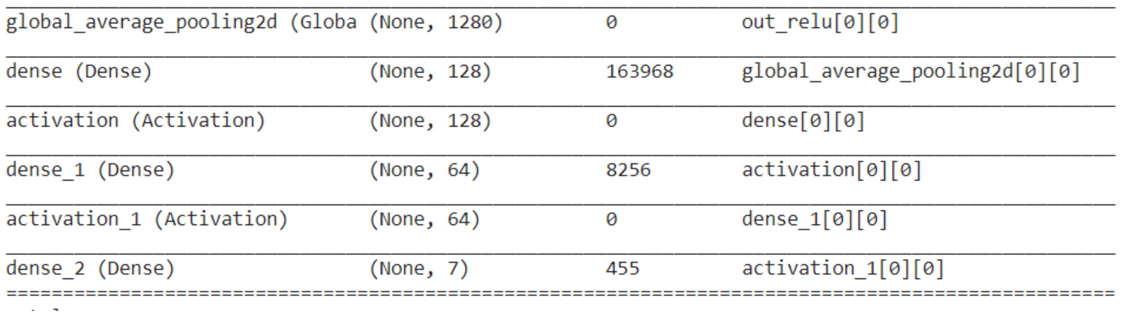
3.1 Facial Emotion Recognition (FER) Using Transfer Learning in Deep CNNs

In our project we are using a pretrained model to build a facial emotion recognition system. We are using MobilenetV2 as the pre-trained model, which is trained on ImageNet dataset.

Actually, the MobilenetV2 model can classify 1000 classes, but our goal is to classify only 7 basic emotions. So, we use the layers in Imagenetv2 up to Global Average pooling. And add some dense layers and an output layer with 7 neurons.



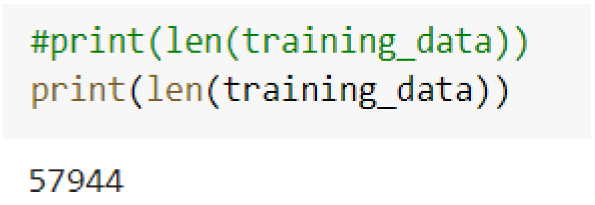
```
final_output = layers.Dense(128)(base_output)  #adding new layer, after the output of global pooling layer
final_output = layers.Activation('relu')(final_output) #activation function
final_output = layers.Dense(64)(final_output)
final_output = layers.Activation('relu')(final_output)
final_output = layers.Dense(7, activation ='softmax')(final_output) #we are classifing 7 classes
```



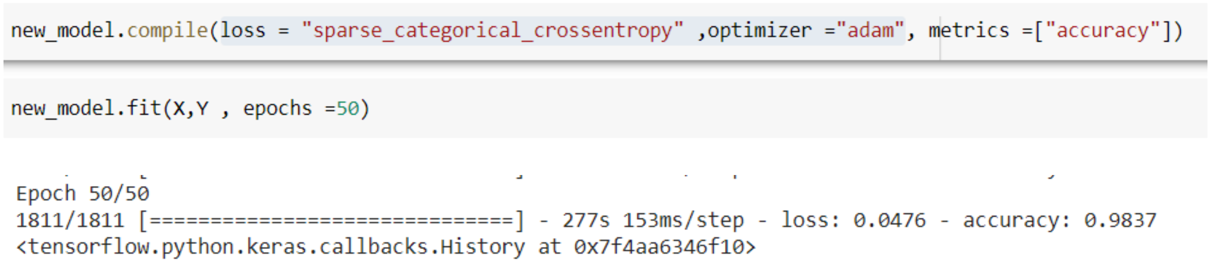
```
________________________________________________________________________________________________
global_average_pooling2d (Globa (None, 1280)         0           out_relu[0][0]
________________________________________________________________________________________________
dense (Dense)                   (None, 128)          163968      global_average_pooling2d[0][0]
________________________________________________________________________________________________
activation (Activation)         (None, 128)          0           dense[0][0]
________________________________________________________________________________________________
dense_1 (Dense)                 (None, 64)           8256        activation[0][0]
________________________________________________________________________________________________
activation_1 (Activation)       (None, 64)           0           dense_1[0][0]
________________________________________________________________________________________________
dense_2 (Dense)                 (None, 7)            455         activation_1[0][0]
================================================================================================
```

We used the FER2013 dataset as input dataset to our new model. In our case we trained the model with 57944 images, loss = "sparse_categorical_crossentropy" and optimizer ="adam".

We trained the model up to 50 epochs, yielding 98% accuracy and 0.0476 loss.



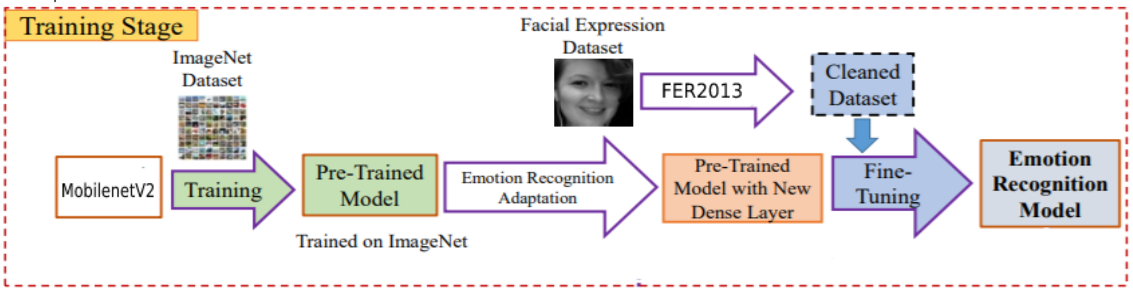
```
#print(len(training_data))
print(len(training_data))

57944
```



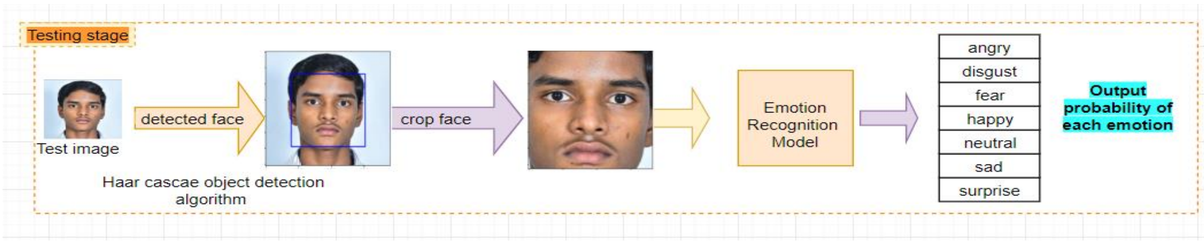
```
new_model.compile(loss = "sparse_categorical_crossentropy" ,optimizer ="adam", metrics =["accuracy"])
```

```
new_model.fit(X,Y , epochs =50)
```

```
Epoch 50/50
1811/1811 [==============================] - 277s 153ms/step - loss: 0.0476 - accuracy: 0.9837
<tensorflow.python.keras.callbacks.History at 0x7f4aa6346f10>
```



Training Stage
ImageNet Dataset
Facial Expression Dataset
FER2013
Cleaned Dataset
MobilenetV2
Training
Pre-Trained Model
Emotion Recognition Adaptation
Pre-Trained Model with New Dense Layer
Fine-Tuning
Emotion Recognition Model
Trained on ImageNet



Testing stage
Test image
detected face
Haar cascae object detection algorithm
crop face
Emotion Recognition Model
angry
disgust
fear
happy
neutral
sad
surprise
Output probability of each emotion

3.2 Sparse_categorical _crossentropy loss

When doing multi-class classification, categorical cross entropy loss is used a lot. It compares the predicted label and true label and calculates the loss. We use sparse categorical crossentropy when your classes are mutually exclusive (e.g., when each sample belongs exactly to one class).

If we use sparse categorical cross entropy then we provide the ground truth as single integer unit only rather than as an n-dimensional vector. Here the integer represents the class of the data.

If we have 7 classes (0–6) and let us say an input belongs to class 2, then the label for this input will be:

categorical: **0 0 1 0 0 0 0 0 0 0** (one-hot-encoding)

sparse: **2** (integer values)

$$CE = -\sum_{i}^{C} t_i log(f(s)_i)$$

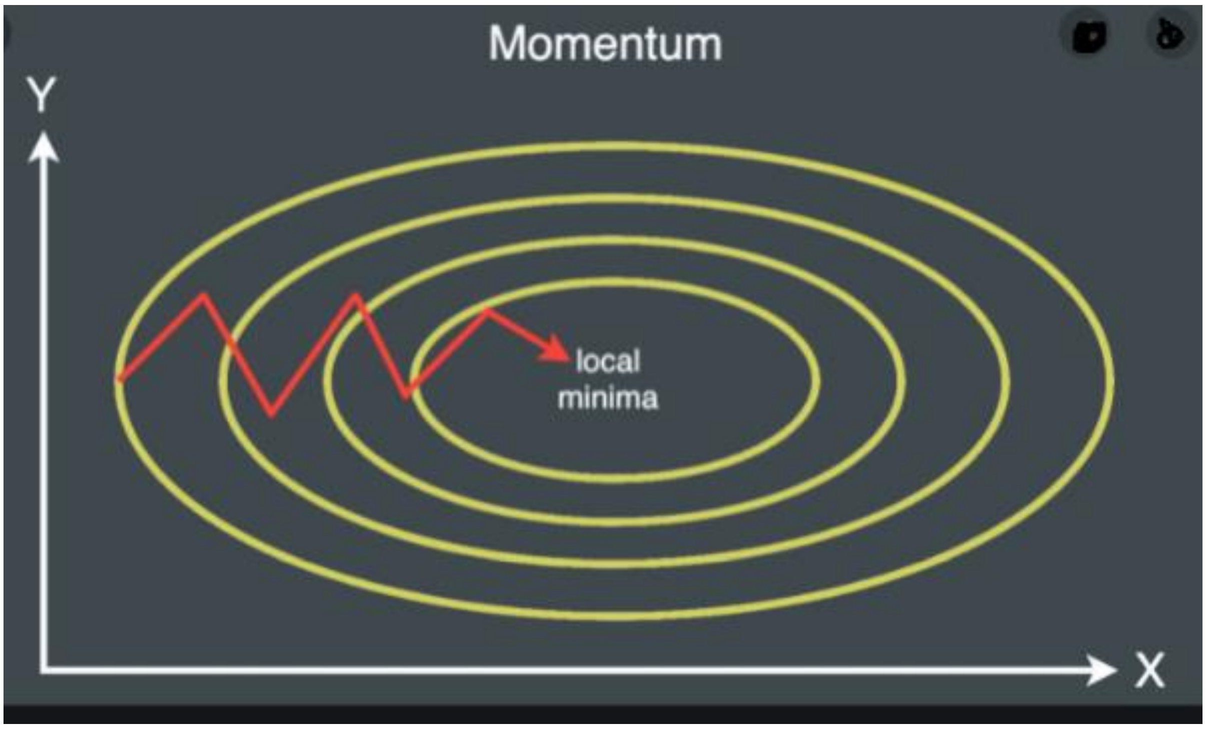
3.3 Adam Optimizer

Picking the right optimizer with the right parameters, can help you squeeze the last bit of accuracy out of your neural network model.

Adam is a replacement **optimization** algorithm for stochastic gradient descent for training deep learning models. **Adam** combines the best properties of the AdaGrad and RMSProp algorithms to provide an **optimization** algorithm that can handle sparse gradients on noisy problems i.e gets the speed from momentum and the ability to adapt gradients in different directions from RMSProp. The combination of the two makes it powerful.

it. It that has recently seen broader adoption for deep learning applications in computer vision.

Adam works well out of the box as an optimizer for multi-class if you want something to give decent performance quickly. Adam uses Momentum and Adaptive Learning Rates to converge faster.



3.4 Relu and SoftMax Activation Functions

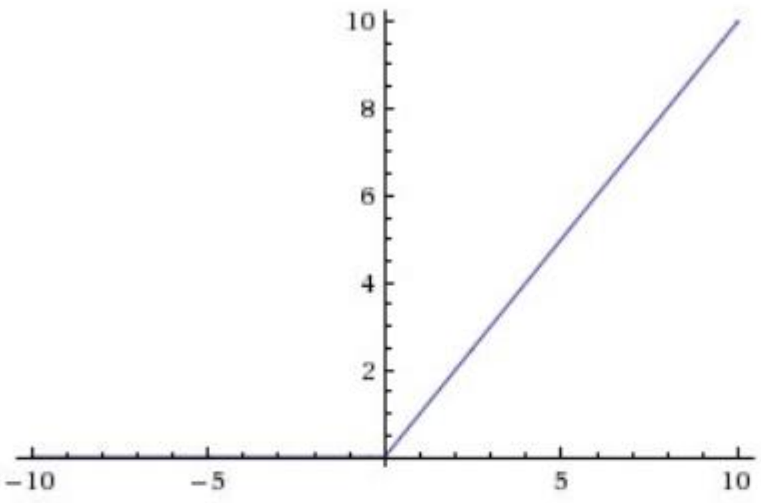
Relu

In a neural network, the activation function is responsible for transforming the summed weighted input from the node into the activation of the node or output for that input.

The **rectified linear activation function** or **ReLU** for short is a piecewise linear function that will output the input directly if it is positive, otherwise, it will output zero. It has become the default activation function for many types of neural networks because a model that uses it is easier to train and often achieves better performance.

It is an activation function defined as the positive part of its argument:
$f(x)=\max(0,x)$

Graphically it can be represented as :



Softmax

It 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 class. It scales numbers/logits into probabilities. The output of a Softmax is a vector with probabilities of each possible outcome.

The softmax function takes as input a vector z of K real numbers, and normalizes it into a probability distribution consisting of K probabilities proportional to the exponentials of the input numbers

$$\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \quad \text{for } i = 1, \dots, K \text{ and } \mathbf{z} = (z_1, \dots, z_K) \in \mathbb{R}^K.$$

σ	= softmax
z	= input vector
e^{z_i}	= standard exponential function for input vector
K	= number of classes in the multi-class classifier
e^{z_j}	= standard exponential function for output vector

4. REQUIREMENT SPECIFICATION

Requirements-Determination is the process, by which analyst gains the knowledge of the organization and apply it in selecting the right technology for a particular application. A Software Requirements Specification (SRS) is a complete description of the behaviour of the system to be developed. It includes a set of use cases that describe all the interactions the users will have with the software. Use cases are also known as functional requirements. In addition to use cases, the SRS also contains non-functional requirements. Non-functional requirements are requirements which impose constraints on the design or implementation (such as quality standards).

4.1 Functional Requirements

In software engineering, a functional requirement defines a function of a software system or its component. A function is described as a set of inputs, the behaviour, and outputs. Functional requirements may be calculations, technical details, data manipulation and processing and other specific functionality that define what a system is supposed to accomplish. Behavioural requirements describing all the cases where the system uses the functional requirements are captured in use cases. Functional requirements are supported by non-functional requirements (also known as quality requirements), which impose constraints on the design or implementation (such as performance requirements, security, or reliability).

Functional Requirements for Present Project

1. Input: Training images and testing images
2. Output: Probabilities saying how much the current image will match with each expression among 7 basic emotions.

3. Process: The model learns the features of the images from the input dataset and classifies a testing image by comparing the learned features.

4.2 Non-Functional Requirements

In systems engineering and requirements engineering, a non-functional requirement is a requirement that specifies criteria that can be used to judge the operation of a system, rather than specific behaviours. This should be contrasted with functional requirements that define specific behaviour or functions In general; functional requirements define what a system is supposed to do whereas non-functional requirements define how a system is supposed to be. Non-functional requirements are often called qualities of a system. Other terms for non-functional requirements are "constraints", "quality attributes", "quality goals" and "quality of service requirements," and "non-behavioural requirements." Qualities, that is, non-functional requirements, can be divided into two main categories one is Execution qualities, such as security and usability, which are observable at run time and Evolution qualities, such as testability, maintainability, extensibility and scalability, which are embodied in the static structure of the software system.

Non-Functional Requirements for Present Project

The faces in the images must be clear and mostly preferred without any accessories as they make extraction of features difficult and may reduce the Accuracy of the model. Images must of dimensions 224 x 224 and with most preferred jpg format.

4.3 System Requirements

To be used efficiently, all computer software needs certain hardware components or other software resources to be present on a computer. These prerequisites are known as (computer) system requirements and are often used as a guideline as opposed to an absolute rule. Industry analysts suggest that this trend plays a bigger part in driving upgrades to existing computer systems than technological advancements.

4.3.1 Hardware Requirements

The most common set of requirements defined by any operating system or software application is the physical computer resources, also known as hardware, a hardware requirements list is often accompanied by a hardware compatibility list (HCL), especially in case of operating systems. An HCL lists tested, compatible, and sometimes incompatible hardware devices for a particular operating system or application. The following sub-sections discuss the various aspects of hardware requirements.

Hardware Requirements for Present Project

1. Processor: Intel(R) Core (TM) i5-6200U CPU @ 2.30GHz
2. RAM: 8GB
3. GPU: Tesla P100(In colab)
4. Storage: 50GB
5. Monitor with 1024*720 resolution

4.3.2 Software Requirements

Software Requirements deal with defining software resource requirements and pre-requisites that need to be installed on a computer to provide optimal functioning of an application. These requirements or pre-requisites are generally not included in the software installation package and need to be installed separately before the software is installed.

Software Requirements for Present Project

1. Operating System: Windows 10
2. Developing Platform: Google Colaboratory with Pro Subscription
3. Languages: Python
4. Browsers: Google Chrome
5. Frameworks: Tensor Flow

5. METHODOLOGY

- Collection of a data set of images. In our project we are using **FER2013** dataset of **35887 pre-cropped, 48 x 48 x 1 pixel grayscale images** of faces each labelled with one of the 7 emotion classes: anger, disgust, fear, happiness, sadness, surprise, and neutral.
- Pre-processing of images.
- Detection of a face from each image.
- The cropped face is converted into grayscale images.
- Separating the data into both training set and testing set.
- The pre-trained (MobileNetV2) model we used accepts 4 dimensions. And it accepts the images of size 224 x 244 x 3. So, we reshape the data.
- The pipeline ensures every image can be fed into the input layer as (None, 224 ,224, 3) NumPy array.
- The NumPy array gets passed into a 53 layered convolutional neural network.
- Convolutions generates feature maps.
- Polling method called Global average pooling generates one feature map for each corresponding category of the classification task in the last convolutional layer. Instead of adding fully connected layers on top of the feature maps, we take the average of each feature map, and the resulting vector is fed directly into the SoftMax layer.
- The SoftMax function presents itself as a probability for each emotion class. The model is able to show the detail probability composition of the emotions in the face.

5.1 MobileNetV2 architecture

MobileNet V2 model has 53 convolution layers and 1 Average Pool with nearly 350 GFLOP (Giga **floating point operations per second**).

There are two types of Convolution layers in MobileNet V2 architecture:

- ❖ 1x1 Convolution
- ❖ 3x3 Depth wise Convolution

Convolutions in MobileNetV2

Following is the list of the 53 Convolution layers in MobileNetV2 architecture with details of different parameters like Input height, Input width, Kernel height output channel etc...:

# Conv	Input H/W	Input Channel	Kernel H/W	Stride H/W	Padding H/W	Output H/W	Output Channel
1	224	3	3	2	0	112	32
2	112	32	3	1	1	112	32
3	112	32	1	1	0	112	16
4	112	16	1	1	0	112	96
5	112	96	3	2	0	56	96
6	56	96	1	1	0	56	24
7	56	24	1	1	0	56	144
8	56	144	3	1	1	56	144
9	56	144	1	1	0	56	24
10	56	24	1	1	0	56	144
11	56	144	3	2	0	28	144
12	28	144	1	1	0	28	32
13	28	32	1	1	0	28	192
14	28	192	3	1	1	28	192
15	28	192	1	1	0	28	32
16	28	32	1	1	0	28	192
17	28	192	3	1	1	28	192
18	28	192	1	1	0	28	32
19	28	32	1	1	0	28	192
20	28	192	3	2	0	14	192
21	14	192	1	1	0	14	64
22	14	64	1	1	0	14	384
23	14	384	3	1	1	14	384
24	14	384	1	1	0	14	64
25	14	64	1	1	0	14	384
26	14	384	3	1	1	14	384
27	14	384	1	1	0	14	64

28	14	64	1	1	0	14	384
29	14	384	3	1	1	14	384
30	14	384	1	1	0	14	64
31	14	64	1	1	0	14	384
32	14	384	3	1	1	14	384
33	14	384	1	1	0	14	96
34	14	96	1	1	0	14	576
35	14	576	3	1	1	14	576
36	14	576	1	1	0	14	96
37	14	96	1	1	0	14	576
38	14	576	3	1	1	14	576
39	14	576	1	1	0	14	96
40	14	96	1	1	0	14	576
41	14	576	3	2	0	7	576
42	7	576	1	1	0	7	160
43	7	160	1	1	0	7	960
44	7	960	3	1	1	7	960
45	7	960	1	1	0	7	160
46	7	160	1	1	0	7	960
47	7	960	3	1	1	7	960
48	7	960	1	1	0	7	160
49	7	160	1	1	0	7	960
50	7	960	3	1	1	7	960
51	7	960	1	1	0	7	320
52	7	320	1	1	0	7	1280
53	1	1280	1	1	0	1	1001

Types of parameters in MobileNetV2:

Total params: 3,538,984

Trainable params: 3,504,872

Non-trainable params: 34,112

6. SYSTEM DESIGN

6.1 Introduction

Grady Booch, James Raumbaugh and Ivar Jacobson have collaborated to combine the best features of their individual object-oriented analysis and design methods into a unified method the unified modeling language, the version 1.0 for the Unified Modeling was released in January 1997 the main parts of UML are based on the Booch, OMT and OOSE methods. The goals of UML are:

1. To model systems using object-oriented concepts
2. To establish an explicit coupling between conceptual as well as executable
3. To address the issues of scale inherent in complex, mission critical system
4. To create a modeling language usable by both humans and machines

Basic Building Blocks of UML

The basic building blocks in UML are things and relationships; these are combined in different ways following different rules to create different types of diagrams. In UML there are nine types of diagrams, below is a list and brief description of them. The more in depth descriptions in the document, will focus on the first five diagrams in the list, which can be seen as the most general, sometimes also referred to as the UML core diagrams.

Use case Diagram: shows a set of use cases, and how actors can use them.

Class Diagram: describes the structure of the system, divided in classes with different connections and relationships

Sequence Diagram: shows the interaction between a set of objects, through the messages that may be dispatched between them.

State chart Diagram: state machines, consisting of states, transitions, events and activities.

Activity Diagram: shows the flow through a program from a defined start point to an end point.

Object Diagram: A set of objects and their relationships, this is a snapshot of instances of the things found in the class objects.

Collaboration Diagram: Collaboration diagram emphasize structural ordering of objects that send and receive messages.

Component Diagram: shows organizations and dependencies among a set of components. These diagrams address the static implementation view of the system.

Deployment Diagram: show the configuration of run-time processing nodes and components that live on them.

6.2 Sequence Diagram

A Sequence diagram is an interaction diagram that shows how processes operate with one another and what is their order. A sequence diagram shows object interactions arranged in time sequence. It depicts the objects and classes involved in the scenario and the sequence of messages exchanged between the objects needed to carry out the functionality of the scenario. A sequence diagram shows, as parallel vertical lines (lifelines), different processes or objects that live simultaneously, and, as horizontal arrows, the messages exchanged between them, in the order in which they occur. The purpose of sequence diagram is to show the flow of functionality through a use case. In other words, we call it a mapping process in terms of data transfers from the actor through the corresponding objects. The key points are:

1. The main purpose is to represent the logical flow of data with respect to a process

2. A sequence diagram displays the objects and not the classes. Messages are written with horizontal arrows with the message name written above them, display interaction. Solid arrow heads represent synchronous calls, open arrow heads represent asynchronous messages, and dashed lines represent reply messages.

When an object destroyed (removed from memory), an X is drawn on top of the lifeline, and the dashed line ceases to be drawn below it (this is not the case in the first example though). It should be the result of a message, either from the object itself, or another.

Sequence Diagram

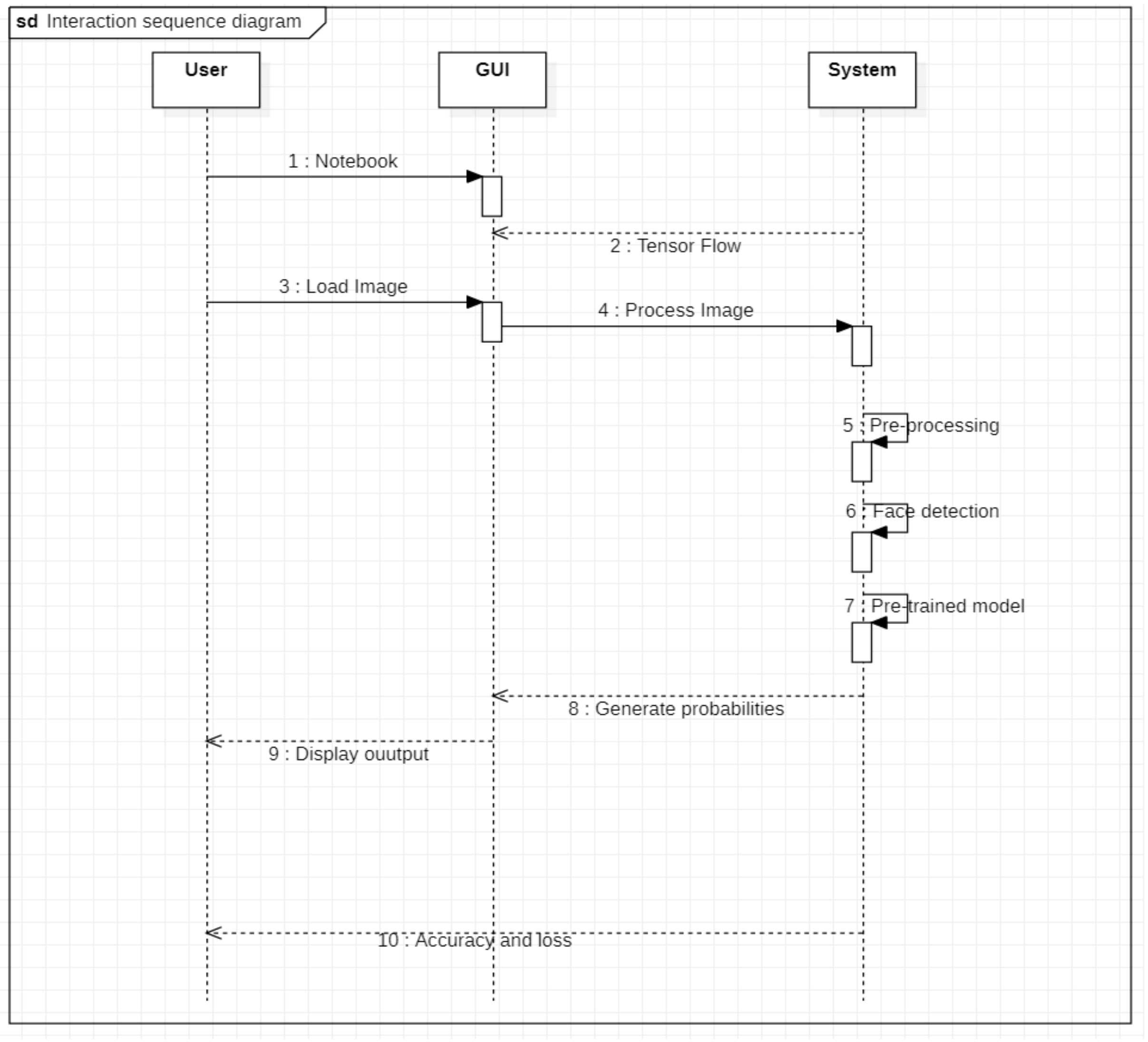
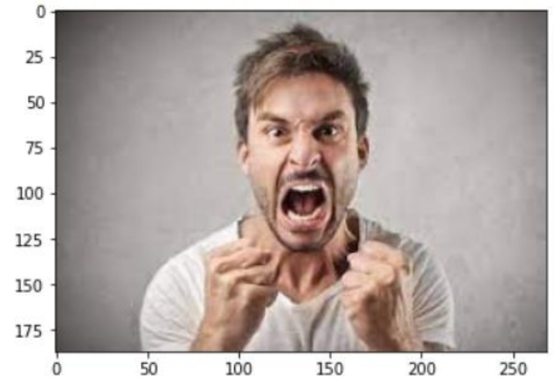


Fig 6.1: Sequence Diagram

7. RESULTS

Input Image

```
<matplotlib.image.AxesImage at 0x7f8c848cddd0>
```

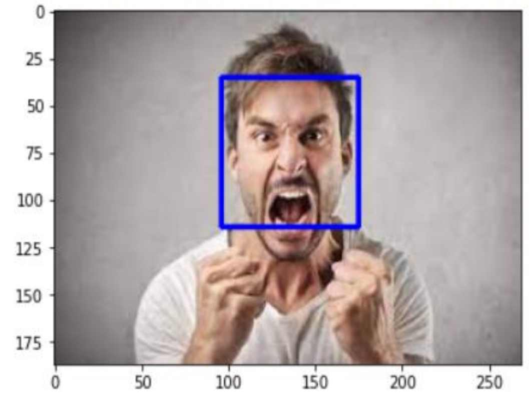


Gray Image

```
<matplotlib.image.AxesImage at 0x7f8c840550d0>
```

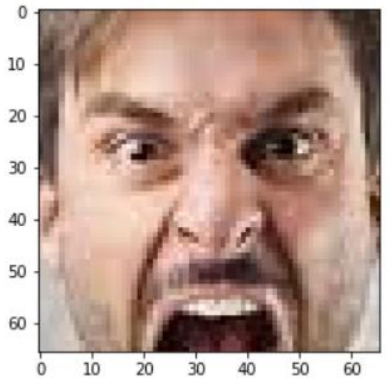
Detected Image

```
<matplotlib.image.AxesImage at 0x7f8cce094f90>
```



Cropped Image

```
<matplotlib.image.AxesImage at 0x7f8cc
```



```
prediction[0]
```

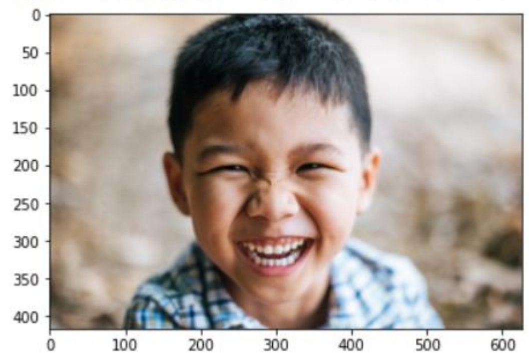
```
array([9.9999523e-01, 2.7406561e-14, 4.7223289e-06, 5.8061955e-11,
       5.7173515e-09, 3.1746681e-09, 3.9668805e-08], dtype=float32)
```

```
The Person in the image is mostly likely to be in Angry
```

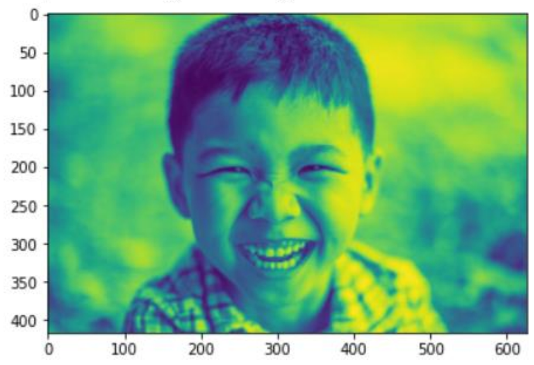
Input Image

Gray Image

```
<matplotlib.image.AxesImage at 0x7f8c813e3f90>
```



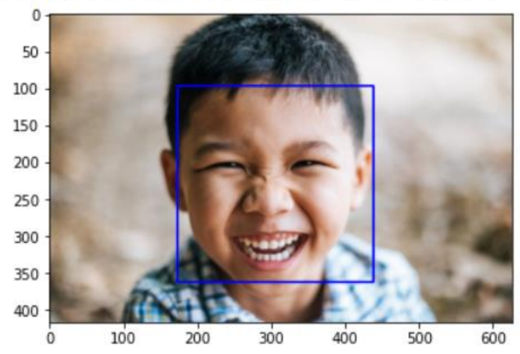
```
<matplotlib.image.AxesImage at 0x7f8c81469b10>
```

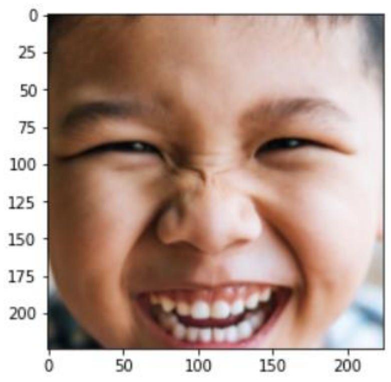


Detected Image

Cropped Image

```
<matplotlib.image.AxesImage at 0x7f8c812b2810>
```





```
prediction[0]
```

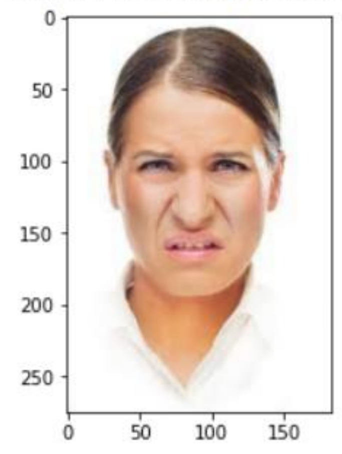
```
array([5.5608880e-02, 1.0645476e-03, 1.1873198e-03, 9.3888730e-01,
       2.0054565e-03, 5.2678655e-04, 7.1968924e-04], dtype=float32)
```

```
The Person in the image is mostly likely to be in Happy
```

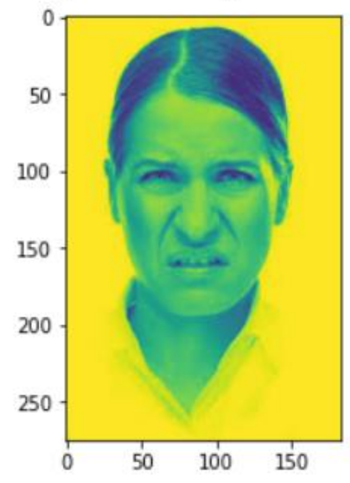
Input Image

Gray Image

```
<matplotlib.image.AxesImage
```



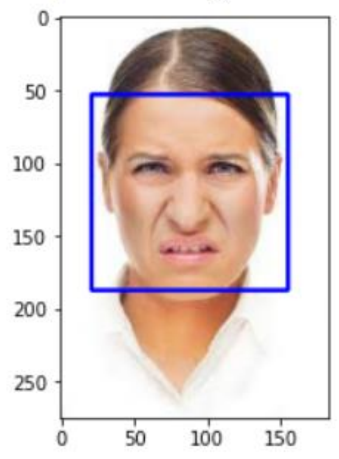
```
<matplotlib.image.AxesImage
```

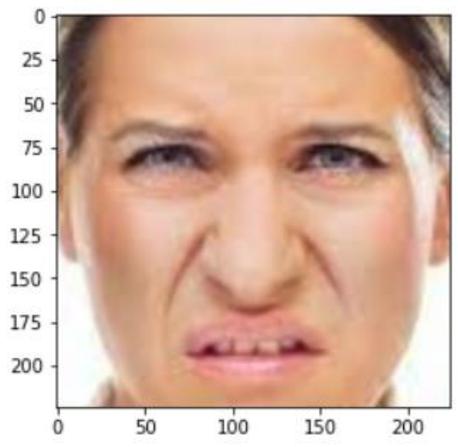


Detected Image

Cropped Image

```
<matplotlib.image.AxesImage
```



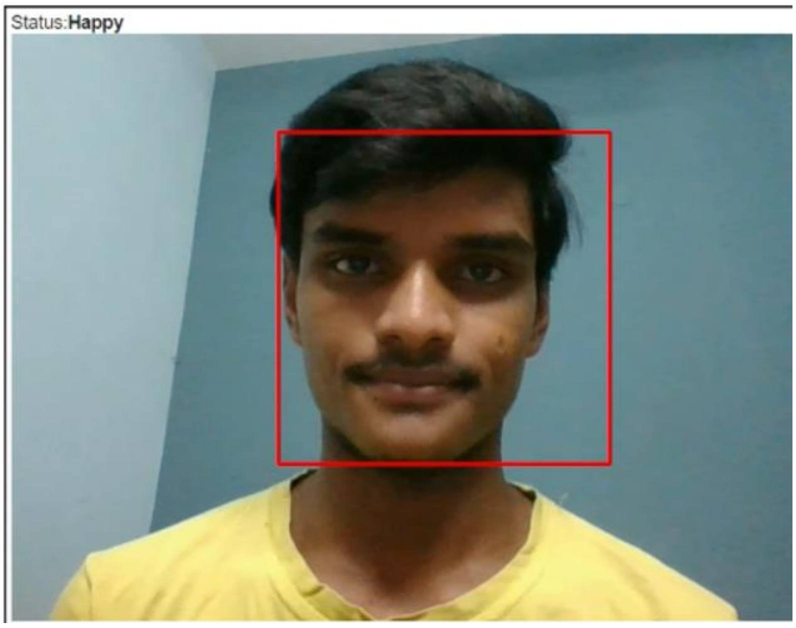


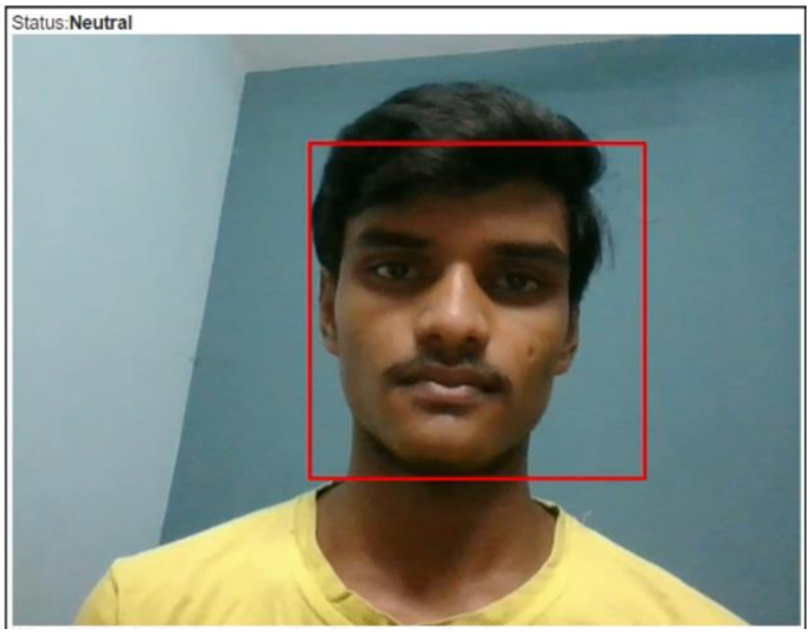
```
prediction[0]
```

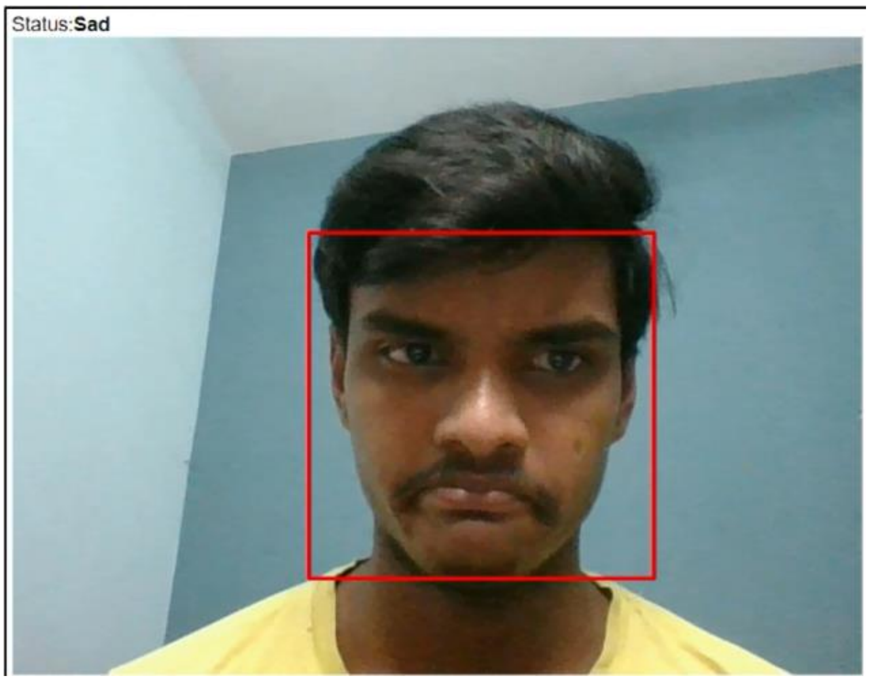
```
array([8.48936806e-12, 1.00000000e+00, 2.37978637e-10, 9.63971007e-17,
       3.30080732e-17, 1.96718475e-11, 1.29391315e-08], dtype=float32)
```

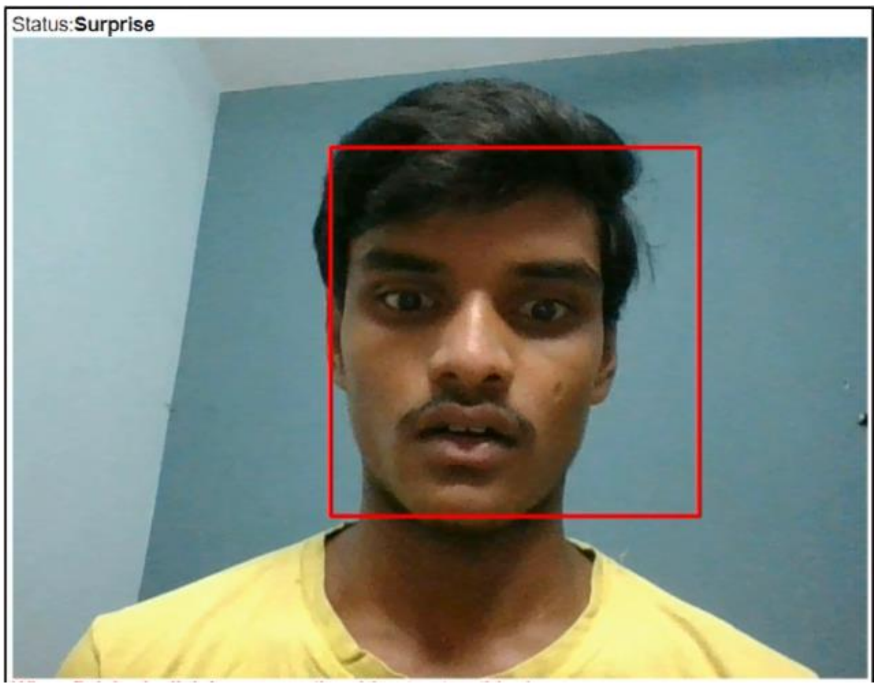
```
The Person in the image is mostly likely to be in Disgust
```

Results through live web cam






Status:**Sad**


Status:**Surprise**

8. TESTING AND VALIDATION

8.1 Introduction

Software testing is an investigation conducted to provide stakeholders with information about the quality of the product or service under test. Software testing also provides an objective, independent view of the software to allow the business to appreciate and understand the risks at implementation of the software. Test techniques include, but are not limited to, the process of executing a program or application with the intent of finding software bugs. Software testing can also be stated as the process of validating and verifying that a software program/application/product meets the business and technical requirements that guided its design and development, Works as expected and can be implemented with the same characteristics. Software testing, depending on the testing method employed, can be implemented at any time in the development process. However, most of the test effort occurs after the requirements have been defined and the coding process has been completed. As such, the methodology of the test is governed by the software development methodology adopted.

Different software development models will focus the test effort at different points in the development process. Newer development models, such as Agile, often employ test driven development and place an increased portion of the testing in the hands of the developer, before it reaches a formal team of testers. In a more traditional model, most of the test execution occurs after the requirements have been defined and the coding process has been completed. Testing can never completely identify all the defects within software. Instead, it furnishes a criticism or comparison that compares the state and behaviour of the product against oracles principles or mechanisms by which someone might recognize a problem. These oracles may include (but are not limited to) specifications, contracts, comparable products, past versions of the same product, inferences about intended or expected purpose, user or customer expectations, relevant standards, applicable laws, or other criteria.

8.2 Types of Testing

8.2.1 Unit Testing

Unit Testing is done on individual modules as they are completed and become executable. It is confined only to the designer's requirements.

8.2.2 Integration Testing

Integration testing ensures that software and subsystems work together as a whole. It tests the interface of all the modules to make sure that the modules behave properly when integrated together.

8.3 Validation

The system has been tested and implemented successfully and thus ensured that all the requirements as listed in the software requirements specification are completely fulfilled. In case of erroneous input corresponding error messages are displayed.

Table 1: Validation Reports

Test No.	Test Case	Expected Output	Actual Output	Result
1	Access Drive: Accessing files from the drive	Mounted by /content/drive	Mounted by /content/drive	Passed
2	Detecting face	Face detected	Face detected	Passed
3	Classify emotion	Emotion in the given image	Emotion in the given image	Passed

9. Conclusion

In our project we used a CNN using Transfer Learning for emotion recognition from facial images. According to the experimental results, using Image augmentation and the Kaggle dataset FER2013, the proposed method shows very high accuracy. In our proposed system we experimented with few confusing images, mostly hand covered images are mis classified.

Further fine-tuning hyperparameters of individual pre-trained models and extending special attention to profile views might enhance the classification accuracy. Our project will be compatible with broader real-life industry applications, such as monitoring patients in the hospital or surveillance security. Moreover, the idea of facial emotion recognition may be extended to emotion recognition from speech or body movements to cover emerging industrial applications.

References

[1] https://www.researchgate.net/publication/343443531_Facial_Emotion_Recognition

[2] https://iq.opengenus.org/mobilenetv2-architecture/

[3] https://machinelearningmastery.com/how-to-configure-image-data-augmentation-when-training-deep-learning-neural-networks/

Appendix

The proposed system can classify an emotion in the given testing image. It compares the learned features from the input dataset and results the probability of each emotion.