A Project Report
on

FACIAL EMOTION RECOGNITION USING ML ALGORITHMS

submitted in partial fulfillment of the requirements for the award of the degree
of
BACHELOR OF TECHNOLOGY
in
COMPUTER SICENCE AND ENGINEERING

by

17WH1A0501	**Ms. V.V.S.S.RAMYA**
17WH1A0526	**Ms. AFRAH SAMREEN**
17WH1A0535	**Ms. SK.AFIFARESHMA**

under the esteemed guidance of
Mr. U.CHANDRASEKHAR
Associate Professor

Department of Computer Science and Engineering
BVRIT HYDERABAD
College of Engineering for Women
(NBA Accredited – EEE, ECE, CSE and IT)
(Approved by AICTE, New Delhi and Affiliated to JNTUH, Hyderabad)
Bachupally, Hyderabad – 500090

June, 2021

DECLARATION

We hereby declare that the work presented in this project entitled **"FACIAL EMOTION RECOGNITION USING ML ALGORITHMS"** submitted towards completion of Project Work in IV year of B.Tech., CSE at 'BVRIT HYDERABAD College of Engineering for Women', Hyderabad is an authentic record of our original work carried out under the guidance of Mr. U.CHANDRASEKHAR, Associate Professor, Department of CSE.

Sign. with date:
Ms. V.V.S.S.RAMYA
(17WH1A0501)

Sign. with date:
Ms. AFRAH SAMREEN
(17WH1A0526)

Sign. with date:
Ms. SK.AFIFARESHMA
(17WH1A0535)

BVRIT HYDERABAD
College of Engineering for Women
(NBA Accredited – EEE, ECE, CSE and IT)
(Approved by AICTE, New Delhi and Affiliated to JNTUH, Hyderabad)
Bachupally, Hyderabad – 500090

Department of Computer Science and Engineering

Certificate

This is to certify that the Project Work report on "FACIAL EMOTION RECOGNITION USING ML ALGORITHMS" is a bonafide work carried out by Ms. V.V.S.S.RAMYA (17WH1A0501); Ms. AFRAH SAMREEN (17WH1A0526); Ms. SK.AFIFARESHMA (17WH1A0535) in the partial fulfillment for the award of B.Tech. degree in **Computer Science and Engineering, BVRIT HYDERABAD College of Engineering for Women, Bachupally, Hyderabad**, affiliated to Jawaharlal Nehru Technological University Hyderabad, Hyderabad under my guidance and supervision.

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

Head of the Department
Dr. Srinivasa Reddy Konda
Professor and HOD,
Department of CSE

Guide
Mr. U. Chandrasekhar
Associate Professor

External Examiner

Acknowledgements

We would like to express our sincere thanks to **Dr. K V N Sunitha, Principal**, **BVRIT HYDERABAD College of Engineering for Women**, for providing the working facilities in the college.

Our sincere thanks and gratitude to our **Dr. Srinivasa Reddy Konda, Head**, Department of CSE, **BVRIT HYDERABAD College of Engineering for Women** for all the timely support and valuable suggestions during the period of our project.

We are extremely thankful and indebted to our internal guide, **Mr. U. Chandrasekhar, Associate Professor**, Department of CSE**, BVRIT HYDERABAD College of Engineering for Women** for his constant guidance, encouragement and moral support throughout the project.

Finally, we would also like to thank our Project Coordinator, all the faculty and staff of **CSE** Department who helped us directly or indirectly, parents and friends for their cooperation in completing the project work.

Ms. V.V.S.S.RAMYA
(17WH1A0501)

Ms. AFRAH SAMREEN
(17WH1A0526)

Ms. SK.AFIFARESHMA
(17WH1A0535)

Contents

ABSTRACT

A facial expression is exhibited by the movement of muscles underneath the face skin. Facial emotion recognition is the process of detecting human emotions from facial expressions. The facial expression for emotion detection has always been a challenging task in achieving through computer algorithms. In the field of Artificial Intelligence, Facial Emotion Recognition (FER) is an active research area, with several recent studies. With the recent advancement in computer vision and machine learning, it is possible to detect emotion from images. The automatic emotion recognition based on facial expression is an interesting research field, which has presented and applied in several areas such as safety, health and in human machine interfaces. The human brain recognizes emotions automatically, and software has now been developed that can recognize emotions as well. This technology is becoming more accurate all the time, and will eventually be able to read emotions as well as our brains do. In this project we propose a technique called facial emotion recognition using convolutional neural networks and haar cascade classifier.

LIST OF FIGURES

1. INTRODUCTION

Humans interact socially with the help of emotions, which are considered as a universal language. These emotions surpass cultural diversities and ethnicity. Facial expressions are responsible for conveying the information, which was difficult to perceive. It gives the mental state of a person that directly relates to his intentions or the physical efforts that he must be applying for performing tasks. Facial expressions are the vital identifiers for human feelings, because it corresponds to the emotions. Most of the times, the facial expression is a non verbal way of emotional expression, and it can be considered as concrete evidence to uncover whether an individual is speaking the truth or not.

Geometric and machine learning based algorithms for an effective recognition are being refined, emphasizing emotion recognition in real-time and not just ideal laboratory conditions. Hence, building a system that is capable of both face detection and emotion recognition has been a crucial area of research. Understanding the different categories of facial expressions of emotion regularly used by us is essential to gain insights into human cognition and affect as well as for the design of computational models and perceptual interfaces. This technique is used to determine the emotion of humans based on their expressions. The current dataset focuses on seven essential facial expression classes reported, which are happy, sad, surprise, fear, neutral, disgust and anger.

To obtain a better understanding of the project, below are some real world projects where the facial emotion recognition is used:

- **Making Cars Safer and Personalized**
  Car manufacturers around the world are increasingly focusing on making cars more personal and safe for us to drive. In their pursuit to build more smart car features, it makes sense for makers to use AI to help them understand the human emotions. Using facial emotion detection smart cars can alert the driver when he is feeling drowsy.

- **Facial Emotion Detection in Interviews**

  A candidate-interviewer interaction is susceptible to many categories of judgment and subjectivity. Such subjectivity makes it hard to determine whether candidate's personality is a good fit for the job. Identifying what a candidate is trying to say is out of our hands because of the multiple layers of language interpretation, cognitive biases, and context that lie in between. That's where AI comes in, which can measure candidate's facial expressions to capture their moods and further assess their personality traits. It will be possible to find whether the candidate is honestly replying to all the questions by measuring the change in emotions during his responses and correlating it the vast amount of knowledge available in this area.

- **Testing for Video Games**

  Video games are designed keeping in mind a specific target audience. Each video game aims to evoke a particular behavior and set of emotions from the users. During the testing phase, users are asked to play the game for a given period and their feedback is incorporated to make the final product. Using facial emotion detection can aid in understanding which emotions a user is going through in real-time as he is playing without analyzing the complete video manually. Such product feedback can be taken by analyzing a live feed of the user and detecting his facial emotions. While feelings of frustration and anger are commonly experienced in advanced video games, making use of facial emotion detection will help understand which emotions are experienced at what points in the game. It is also possible that some unexpected or undesirable emotions are observed during the game.

- **Market Research**

  Traditional market research companies have employed verbal methods mostly in the form of surveys to find the consumers wants and needs. However, such methods assume that consumers can formulate their preferences verbally and the stated preferences correspond to future actions which may not always be right. Facial emotion recognition AI can automatically detect facial expressions on user's faces and automate the video analysis completely. Market research companies can use this technology to scale the data collection efforts and rely on the technology to do analysis quickly.

1.1 Objectives

Facial Emotion Recognition is essential to recognize the emotions in many cases like in criminal investigation, E-learning, gaming and psychology. In criminal investigation, it is used to identify the criminals based on their facial expressions. Similarly in E-learning, this can be used to conclude whether a student is understanding a concept or not and in other applications it ensures the understanding of a person's emotional condition.

The classification of the facial expressions were performed using Convolutional Neural Network and Haar Cascade Classifier Algorithm. The image is uploaded which is then converted as a grey scale image. The face is detected using Haar Cascade Classifier which is then classified by the model that displays the image along with its emotion as the output.

1.2 Methodology

To classify the emotions of different people a large collection of images is required. The dataset is downloaded from the kaggle. In this section the methodology followed is discussed in detail

1.2.1 Dataset

Proper and large dataset is required for all classification research during the training and the testing phase. The dataset for the experiment is downloaded from the Kaggle which contains different facial images and their respective labels. A dataset containing approximately 35,887 images of seven classes is downloaded. Each image size is 48*48 pixels and the dataset link is https://www.kaggle.com/deadskull7/fer2013

These seven classes include:

S.No.	Emotion	Total number of images
1	Angry	4953
2	Disgust	547
3	Fear	5121
4	Happy	8969
5	Sad	6077
6	Surprise	4002
7	Neutral	6198

The dataset has been converted into a CSV file that consists of 48x48 pixel grayscale images of faces. The faces have been automatically registered so that the face is more or less centered and occupies about the same amount of space in each image. The task is to categorize each face based on the emotion shown in the facial expression in to one of seven categories (0=Angry, 1=Disgust, 2=Fear, 3=Happy, 4=Sad, 5=Surprise, 6=Neutral).

train.csv contains two columns, "emotion" and "pixels". The "emotion" column contains a numeric code ranging from 0 to 6, inclusive, for the emotion that is present in the image. The "pixels" column contains a string surrounded in quotes for each image. The contents of this string a space-separated pixel values in row major order. test.csv contains only the "pixels" column and the task is to predict the emotion column.

Fig 1.2.1.1: Seven classes of the dataset

1.2.2 The proposed models:

Convolutional Neural Network (CNN)

CNN architectures vary with the type of the problem at hand. The proposed model consists of three convolutional layers each followed by a maxpooling layer. ReLu (Rectified Linear Unit) activation function is applied to the output of every convolutional layer and fully connected layer.

The first convolutional layer filters the input image with 64 kernels of size 3x3. After max pooling is applied, the output is given as an input for the second convolutional layer with 64 kernels of size 3x3. The last convolutional layer has 128 kernels of size 3x3 followed by a fully connected layer of 1024 neurons. The output of this layer is given to softmax function which produces a probability distribution of the seven output classes. The model is trained using Stochastic Gradient Descent (SGD) with batch size of 16 for 100 epochs.

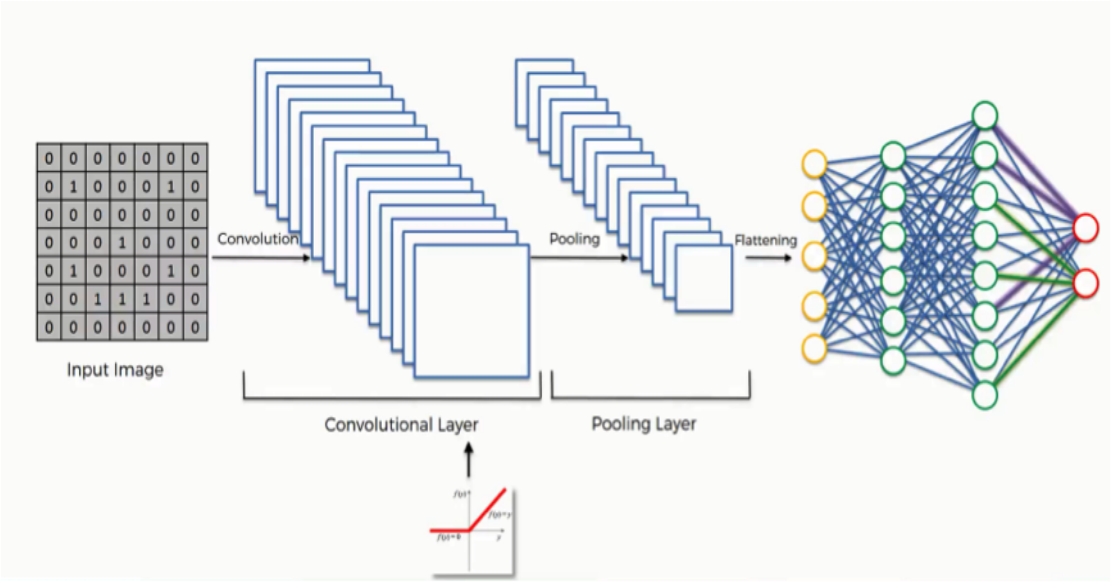


Fig 1.2.2.1: Convolution Neural Network

There are five CNN algorithm steps:

Convolution

The term convolution refers to the mathematical combination of two functions to produce a third function. It merges two sets of information. In the case of a CNN, the convolution is performed on the input data with the use of a filter or kernel to then produce a feature map.

Here are the three elements that enter into the convolution operation:

- Input image
- Feature detector
- Feature map

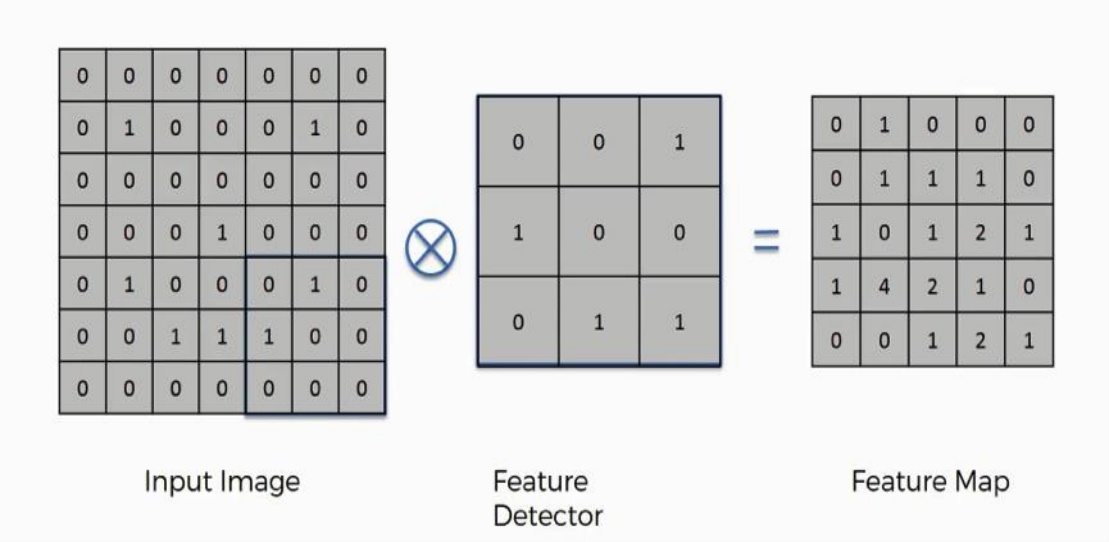


Fig 1.2.2.2: Convolution in CNN

Max pooling

Max pooling is a sample-based discretization process. The objective is to down-sample an input representation (image, hidden-layer output matrix, etc.), reducing its dimensionality and allowing for assumptions to be made about features contained in the sub-regions binned.

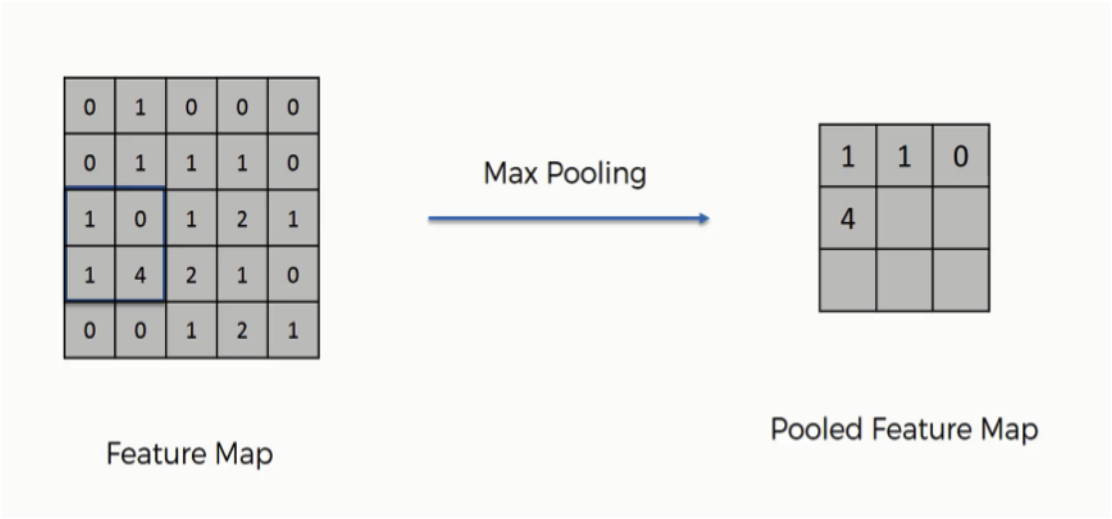


Fig 1.2.2.3: Max Pooling in CNN

Dropout

The term "dropout" refers to dropping out units (hidden and visible) in a neural network. Dropout may be implemented on any or all hidden layers in the network as well as the visible or input layer. It is not used on the output layer. It can be used with most types of layers, such as dense fully connected layers, convolutional layers, and recurrent layers such as the long short-term memory network layer. Dropout is implemented per-layer in a neural network. It is required to prevent overfitting.

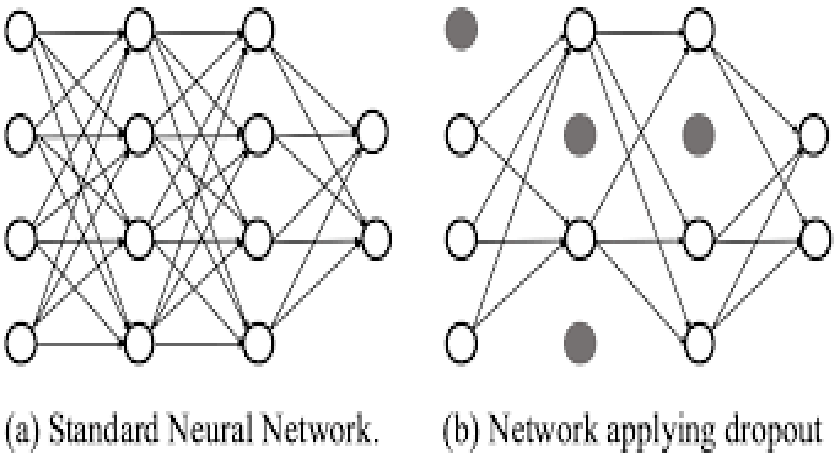


Fig 1.2.2.4: Dropout in CNN

Flattening

Flattening is the process of converting all the resultant 2 dimensional arrays into a single long continuous linear vector.

Fig 1.2.2.5: Flattening in CNN

Full Connection

At the end of a CNN, the output of the last Pooling Layer acts as an input to the so called Fully Connected Layer. There can be one or more of these layers ("fully connected" means that every node in the first layer is connected to every node in the second layer).

As you see from the image below, we have three layers in the full connection step:

- Input layer
- Fully-connected layer
- Output layer

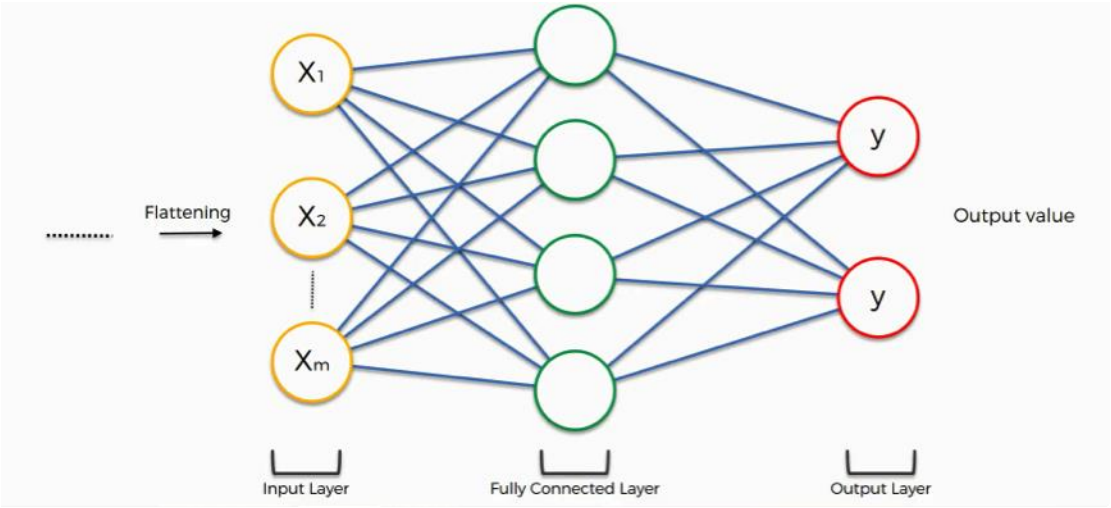


Fig 1.2.2.6: Full Connection in CNN

Haar Cascade Classifier

Haar Cascade classifier is an effective object detection approach used for boosting the implementation. It is an effective object detection approach which was proposed by Paul Viola and Michael Jones in their paper, "Rapid Object Detection using a Boosted Cascade of Simple Features" in 2001. Based on the training it is then used to detect the objects in the other images. The algorithm can be explained in four stages:

Calculating Haar Features:

The first step is to collect the Haar features. A Haar feature is essentially calculations that are performed on adjacent rectangular regions at a specific location in a detection window. The calculation involves summing the pixel intensities in each region and calculating the differences between the sums. Here are some examples of Haar features below:

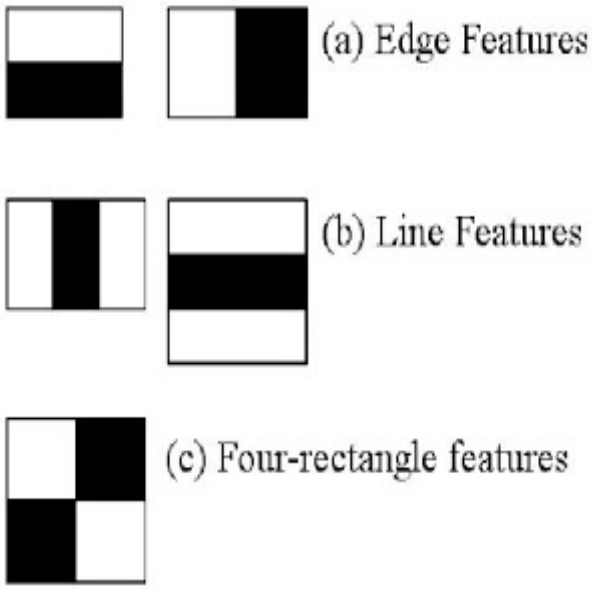


Fig 1.2.2.7: Haar features

These features can be difficult to determine for a large image. This is where integral images come into play.

Creating Integral Images:

Integral images essentially speed up the calculation of these Haar features. Instead of computing at every pixel, it instead creates sub-rectangles and creates array references for each of those sub-rectangles. These are then used to compute the Haar features.

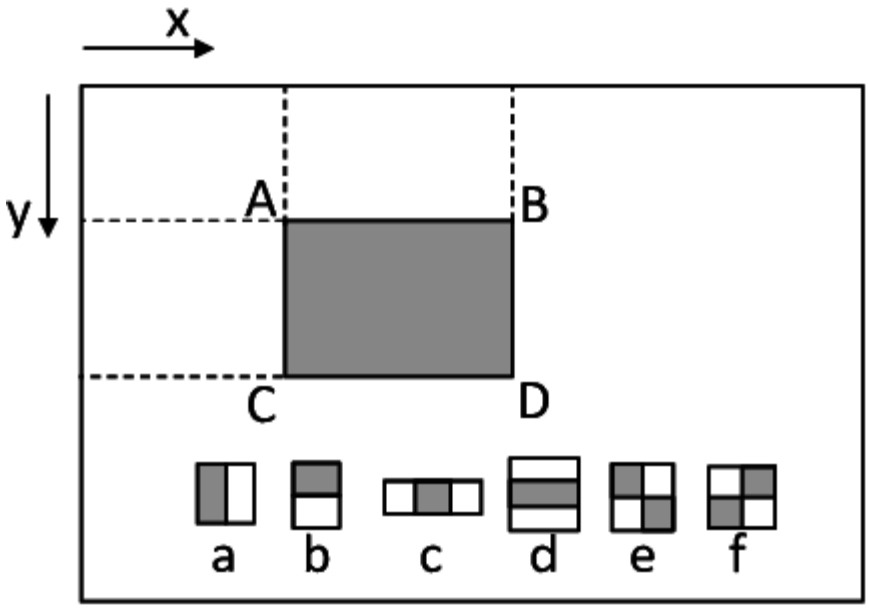


Fig 1.2.2.8: Creating integral images

Using Adaboost:

Adaboost essentially chooses the best features and trains the classifiers to use them. It uses a combination of **"weak classifiers"** to create a **"strong classifier"** that the algorithm can use to detect objects. Weak learners are created by moving a window over the input image, and computing Haar features for each subsection of the image. This difference is compared to a learned threshold that separates non-objects from objects.

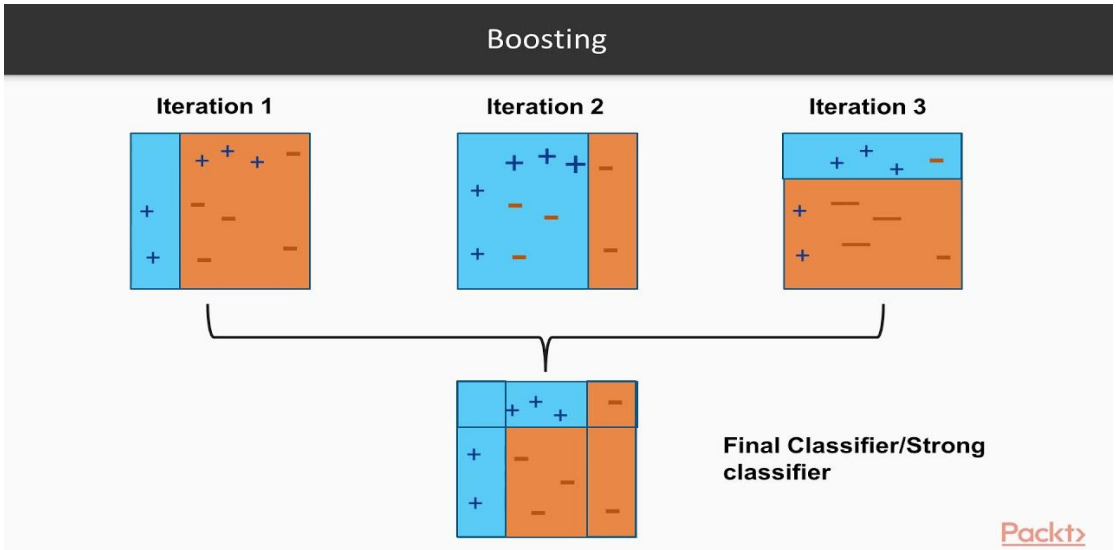


Fig 1.2.2.9: Using Adaboost

Implementing Cascading Classifiers:

The cascade classifier is made up of a series of stages, where each stage is a collection of weak learners. Weak learners are trained using boosting, which allows for a highly accurate classifier from the mean prediction of all weak learners.

1.3 Organization of Project

The technique which is developed is taking a facial image as an input in which the face is detected using Haar Cascade Classifier after which background elimination is done. This image is then classified based on emotion using convolutional neural network model.

We have three modules in our project:

- Model Building
- Image Upload
- Emotion Recognition

2. THEORETICAL ANALYSIS OF THE PROPOSED PROJECT

2.1 Requirements Gathering

2.1.1 Software Requirements

Programming Language : Python

Dataset : fer2013 Dataset

Packages : Tensorflow, Numpy, Pandas, Matplotlib, Scikit-learn, Seaborn, cv2, Keras

Environment : Google Colab

2.1.2 Hardware Requirements

Operating System: Windows 10

Processor : Intel Core i5

CPU Speed : 2.30 GHz

Memory : 8 GB (RAM)

2.2 Technologies Description

Python

Python is an interpreted high-level programming language for general-purpose programming. Created by Guido van Rossum and first released in 1991, Python has a design philosophy that emphasizes code readability, notably using significant whitespace.

Python features a dynamic type system and automatic memory management. It supports multiple programming paradigms, including object-oriented, imperative, functional and procedural, and has a large and comprehensive standard library.

- Python is Interpreted − Python is processed at runtime by the interpreter. You do not need to compile your program before executing it. This is similar to PERL and PHP.
- Python is Interactive − you can actually sit at a Python prompt and interact with the interpreter directly to write your programs.

Python also acknowledges that speed of development is important. Readable and terse code is part of this, and so is access to powerful constructs that avoid tedious repetition of code. Maintainability also ties into this may be an all but useless metric, but it does say something about how much code you have to scan, read and/or understand to troubleshoot problems or tweak behaviors.

This speed of development, the ease with which a programmer of other languages can pick up basic Python skills and the huge standard library is key to another area where Python excels. All its tools have been quick to implement, saved a lot of time, and several of them have later been patched and updated by people with no Python background - without breaking.

Dataset

The dataset for the experiment is downloaded from the Kaggle which contains different facial images and their respective labels. A dataset containing approximately 35,887 images of seven classes is downloaded. The dataset has been converted into a CSV file that consists of 48x48 pixel grayscale images of faces. train.csv contains two columns, "emotion" and "pixels". The "emotion" column contains a numeric code ranging from 0 to 6 (0=Angry, 1=Disgust, 2=Fear, 3=Happy, 4=Sad, 5=Surprise, 6=Neutral).

Tensorflow

TensorFlow is a free and open-source software library for dataflow and differentiable programming across a range of tasks. It is a symbolic math library, and is also used for machine learning applications such as neural networks. It is used for both research and production at Google.

TensorFlow was developed by the Google Brain team for internal Google use. It was released under the Apache 2.0 open-source license on November 9, 2015.

Numpy

Numpy is a general-purpose array-processing package. It provides a high-performance multidimensional array object, and tools for working with these arrays.

It is the fundamental package for scientific computing with Python. It contains various features including these important ones:

- A powerful N-dimensional array object
- Sophisticated (broadcasting) functions
- Tools for integrating C/C++ and Fortran code
- Useful linear algebra, Fourier transform, and random number capabilities

Besides its obvious scientific uses, Numpy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined using Numpy which allows Numpy to seamlessly and speedily integrate with a wide variety of databases.

Pandas

Pandas is an open-source Python Library providing high-performance data manipulation and analysis tool using its powerful data structures. Python was majorly used for data munging and preparation. It had very little contribution towards data analysis. Pandas solved this problem. Using Pandas, we can accomplish five typical steps in the processing and analysis of data, regardless of the origin of data load, prepare, manipulate, model, and analyze. Python with Pandas is used in a wide range of fields including academic and commercial domains including finance, economics, Statistics, analytics, etc.

Matplotlib

Matplotlib is a Python 2D plotting library which produces publication quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, the Python and IPython shells, the Jupyter Notebook, web application servers, and four graphical user interface toolkits. Matplotlib tries to make easy things easy and hard things possible. You can generate plots, histograms, power spectra, bar charts, error charts, scatter plots, etc., with just a few lines of code. For examples, see the sample plots and thumbnail gallery.

For simple plotting the pyplot module provides a MATLAB-like interface, particularly when combined with IPython. For the power user, you have full control of line styles, font properties, axes properties, etc, via an object oriented interface or via a set of functions familiar to MATLAB users.

Scikit – learn

Scikit-learn provides a range of supervised and unsupervised learning algorithms via a consistent interface in Python. It is licensed under a permissive simplified BSD license and is distributed under many Linux distributions, encouraging academic and commercial use. The library is built upon the SciPy (Scientific Python) that must be installed before you can use scikit-learn. This stack that includes:

- **NumPy**: Base n-dimensional array package
- **SciPy**: Fundamental library for scientific computing
- **Matplotlib**: Comprehensive 2D/3D plotting
- **IPython**: Enhanced interactive console
- **Sympy**: Symbolic mathematics
- **Pandas**: Data structures and analysis
- Extensions or modules for SciPy care conventionally named SciKits. As such, the module provides learning algorithms and is named scikit-learn.

Keras

Keras is an open-source software library that provides a Python interface for artificial neural networks. Keras acts as an interface for the TensorFlow library. Up until version 2.3 Keras supported multiple backends, including TensorFlow, Microsoft Cognitive Toolkit, Theano, and PlaidML. As of version 2.4, only TensorFlow is supported. Designed to enable fast experimentation with deep neural networks, it focuses on being user-friendly, modular, and extensible.

Keras contains numerous implementations of commonly used neural-network building blocks such as layers, objectives, activation functions, optimizers, and a host of tools to make working with image and text data easier to simplify the coding necessary for writing deep neural network code.

Seaborn

Seaborn is an amazing visualization library for statistical graphics plotting in Python. It provides beautiful default styles and color palettes to make statistical plots more attractive. It is built on the top of matplotlib library and also closely integrated to the data structures from pandas.

Seaborn aims to make visualization the central part of exploring and understanding data. It provides dataset-oriented APIs, so that we can switch between different visual representations for same variable for better understanding of dataset.

Seaborn divides plot into the below categories –

- **Relational plots:** This plot is used to understand the relation between two variables.
- **Categorical plots:** This plot deals with categorical variables and how they can be visualized.
- **regression plots:** The regression plots in seaborn are primarily intended to add a visual guide that helps to emphasize patterns in a dataset during exploratory data analysis.
- **Matrix plots:** A matrix plot is an array of scatterplots.
- **Multi-plot grids:** It is a useful approach to draw multiple instances of the same plot on different subsets of the dataset.

Cv2

OpenCV-Python is a library of Python bindings designed to solve computer vision problems.

Earlier, there was only Cv. Later, OpenCV came with both Cv and Cv2. Now, there in the latest releases, there is only the Cv2 module, and Cv is a subclass inside Cv2. In this, everything is returned as NumPy objects like ndarray and native Python objects like lists,tuples,dictionary, etc. So due to this NumPy support, any numpy operation can be performed here. Cv2. imread() method loads an image from the specified file. If the image

cannot be read (because of missing file, improper permissions, unsupported or invalid format) then this method returns an empty matrix.

Google Colab

Google Colaboratory is a free Jupyter notebook environment that runs entirely in the cloud. Most importantly, it does not require a setup and the notebooks that are created can be simultaneously edited by the team members - just the way the documents are edited in Google Docs. Colab supports many popular machine learning libraries which can be easily loaded in your notebook.

As we can perform the following using Google Colab.

- Write and execute code in Python
- Document your code that supports mathematical equations
- Create/Upload/Share notebooks
- Import/Save notebooks from/to Google Drive
- Import/Publish notebooks from GitHub
- Import external datasets e.g. from Kaggle
- Integrate PyTorch, TensorFlow, Keras, OpenCV
- Free Cloud service with free GPU

3. DESIGN

3.1 Introduction

Software design sits at the technical kernel of the software engineering process and is applied regardless of the development paradigm and area of application. Design is the first step in the development phase for any engineered product or system. The designer's goal is to produce a model or representation of an entity that will later be built. Beginning, once system requirement have been specified and analyzed, system design is the first of the three technical activities - design, code and test that is required to build and verify software.

The importance can be stated with a single word "Quality". Design is the place where quality is fostered in software development. Design provides us with representations of software that can assess for quality. Design is the only way that we can accurately translate a customer's view into a finished software product or system. Software design serves as a foundation for all the software engineering steps that follow. Without a strong design we risk building an unstable system – one that will be difficult to test, one whose quality cannot be assessed until the last stage.

During design, progressive refinement of data structure, program structure, and procedural details are developed reviewed and documented. System design can be viewed from either technical or project management perspective. From the technical point of view, design is comprised of four activities – architectural design, data structure design, interface design and procedural design.

3.2 Architecture Diagram

An architecture diagram is a diagram of a system that is used to abstract the overall outline of the software system and the relationships, constraints, and boundaries between components. It is an important tool as it provides an overall view of the physical deployment of the software system and its evolution roadmap. The following are the steps used in the project:

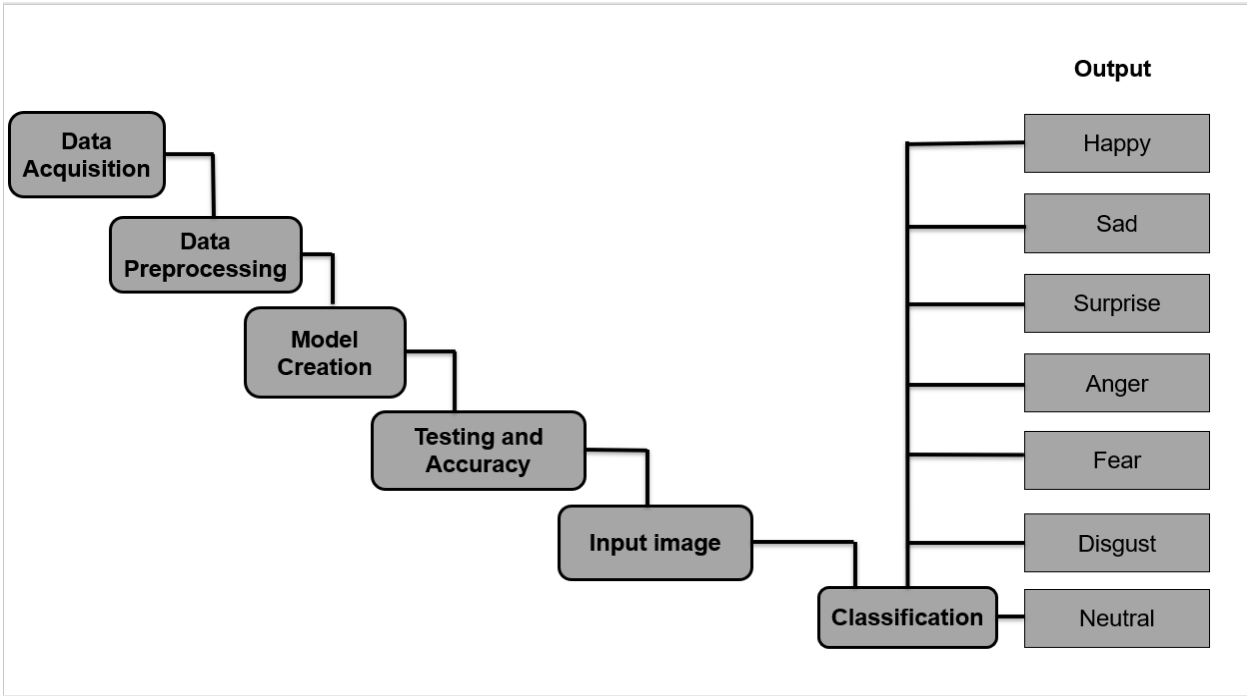


Fig 3.2.1: Architecture Diagram

Data Acquisition

In data acquistion step the data is collected from kaggle. The dataset consist of 35,887 images.The dataset consists of 7 labels, they are happy, sad, disgust, angry, fear, surprise, neutral. A CSV file consisting of train and test images has been used to train the model and classify the images.

Data Preprocessing

Data preprocessing is a data mining technique that involves transforming raw data into an understandable format. Real-world data is often incomplete, inconsistent, and/or lacking in certain behaviors or trends, and is likely to contain many errors.

When creating a machine learning project, it is not always a case that we come across the clean and formatted data and while doing any operation with data, it is mandatory to clean it and put in a formatted way. The data preprocessing has been done for this purpose. The images have been reshaped as part of this in the project and the uploaded image has also been preprocessed by performing the background elimination.

Model Creation

An appropriate and best model needs to be created for any Machine Learning project. Convolutional Neural Network (CNN) model is the most suitable model for image classification. The model type used is Sequential. Sequential is the easiest way to build a model in Keras. It allows to build a model layer by layer. The 'add()' function is used to add layers to the model.

Kernel size is the size of the filter matrix for the convolution. So a kernel size of 3 means a 3x3 filter matrix. Activation is the activation function for the layer. The activation function that has been used for the first 3 layers is the ReLU, or Rectified Linear Activation. This activation function has been proven to work well in neural networks. In between the Conv2D layers and the dense layer, there is a 'Flatten' layer. Flatten serves as a connection between the convolution and dense layers. 'Dense' is the layer type used for the output layer. Dense is a standard layer type that is used in many cases for neural networks. The activation function used for the output layer is 'softmax'. It makes the output sum up to 1 so the output can be interpreted as probabilities.

Training and Accuracy

Training a neural network is the process of finding the values for the weights and biases. During training, the test data is not used at all. After training completes, the accuracy of the resulting neural network model's weights and biases are applied just once to the test data. Emotion recognition training is a novel technique which targets the recognition of facial expressions of emotions.

Accuracy is one metric for evaluating the classification models. Informally, accuracy is the fraction of predictions a model got right. Formally, accuracy has the following definition:

Accuracy = Number of correct predictions / Total number of predictions

Input Image

An image needs to be given as an input to the model to check whether the model is predicting correctly or not. This image is at first converted into a grey scale image. The entire background is eliminated from the image and the face is detected using the Haar Cascade Classifier after which it is fed into the model to predict the appropriate emotion.

Classification

The input image is being classified by the CNN model into one of the seven emotions which are angry, disgust, fear, happy, sad, surprise and neutral. In the classification part, the images are tested according to the given categories and produces the final predicted emotion.

Output

The final predicted emotion is obtained along with the given input image.

4. IMPLEMENTATION

4.1 Coding

#Import the necessary libraries

```
import pandas as pd
import numpy as np
from keras.models import Sequential
from keras.layers.core import Flatten, Dense, Dropout
from keras.layers.convolutional import Conv2D, MaxPooling2D, ZeroPadding2D
from keras.optimizers import SGD
import cv2
```

#Fetch the dataset from Dropbox

Fetching the dataset from dropbox enables the easy and fast loading of the data into the project since the size of the dataset is large. It is simple and can be done by just adding a link obtained after uploading the dataset into dropbox.

After unzipping the dataset, the CSV file is read using Pandas

```
!wget https://www.dropbox.com/s/qjhdcuzbysxb5vf/fer2013.zip?dl=0
!unzip fer2013.zip?dl=0

x_train,y_train,x_test,y_test=[],[],[],[]

df = pd.read_csv('/content/fer2013.csv')
df.shape
```

```
df = pd.read_csv('/content/fer2013.csv')
df.shape
```

```
(35887, 3)
```

df.Usage.value_counts()

```
df.Usage.value_counts()
```

```
Training       28709
PublicTest      3589
PrivateTest     3589
Name: Usage, dtype: int64
```

#Determine the total number of images

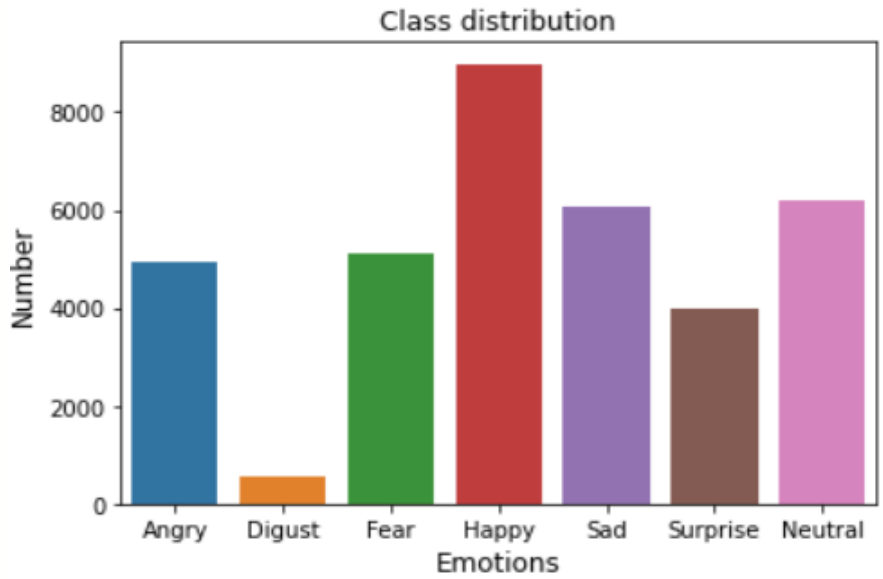
```
emotion_map = {0: 'Angry', 1: 'Digust', 2: 'Fear', 3: 'Happy', 4: 'Sad', 5: 'Surprise', 6: 'Neutral' }
emotion_counts = df['emotion'].value_counts(sort=False).reset_index()
emotion_counts.columns = ['emotion', 'number']
emotion_counts['emotion'] = emotion_counts['emotion'].map(emotion_map)
emotion_counts
```

	emotion	number
0	Angry	4953
1	Digust	547
2	Fear	5121
3	Happy	8989
4	Sad	6077
5	Surprise	4002
6	Neutral	6198

#Plot the class distribution graph

The class distribution gives an overview on the number of images present in each individual class of emotions.

```
import seaborn as sns
import matplotlib.pyplot as plt
plt.figure(figsize=(6,4))
sns.barplot(emotion_counts.emotion, emotion_counts.number)
plt.title('Class distribution')
plt.ylabel('Number', fontsize=12)
plt.xlabel('Emotions', fontsize=12)
plt.show()
```



#Extract the images into train and test arrays

```
for index, row in df.iterrows():
  val = row['pixels'].split(" ")
  try:
    if 'Training' in row['Usage']:
      x_train.append(np.array(val, 'float32'))
      y_train.append(row['emotion'])
    elif 'PublicTest' in row['Usage']:
      x_test.append(np.array(val, 'float32'))
      y_test.append(row['emotion'])
  except:
    print(f"error occured at index :{index} and row:{row}")
```

#Initialize the required parameters

The required variables like batch size, number of features, number of epochs along with the height and width are initialized so as to pass them during the training of the model.

```
num_features = 64
num_labels = 7
batch_size = 16
epochs = 100
width, height = 48, 48
```

```
x_train = np.array(x_train,'float32')
y_train = np.array(y_train,'float32')
x_test = np.array(x_test,'float32')
y_test = np.array(y_test,'float32')
```

```
from keras.utils.np_utils import to_categorical
y_train=to_categorical(y_train, num_classes=num_labels)
y_test=to_categorical(y_test, num_classes=num_labels)
```

```
x_train -= np.mean(x_train, axis=0)
x_train /= np.std(x_train,axis=0)
x_test -= np.mean(x_test,axis=0)
x_test /= np.std(x_test, axis=0)
```

#Reshape the train and test arrays

The train and test sets are reshaped so that all the images are of the same dimensions and it becomes easy for the model to train.

```
x_train = x_train.reshape(x_train.shape[0], 48, 48, 1)
x_test = x_test.reshape(x_test.shape[0], 48, 48, 1)
```

#Import from Tensorflow library

```
from tensorflow.keras.layers import Dense, Conv2D, Flatten, Dropout, MaxPooling2D, Activ
ation
from tensorflow.keras.preprocessing import image
import matplotlib.pyplot as plt
from tensorflow.keras.losses import categorical_crossentropy
from tensorflow.keras import layers
from tensorflow import keras
```

#Build the CNN model

```
model = Sequential()
model.add(Conv2D(64, kernel_size=(3, 3), activation='relu', input_shape=(x_train.shape[1:])
))
model.add(Conv2D(64, kernel_size=(3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2), strides=(2,2)))
model.add(Dropout(0.5))
```

#2nd convo layer

```
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(Conv2D(64, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2), strides=(2,2)))
model.add(Dropout(0.5))
```

#3rd convo layer

```
model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(Conv2D(128, (3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2), strides=(2,2)))

model.add(Flatten())
```

#fully connected neural networks

```
model.add(Dense(1024, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(1024, activation='relu'))
model.add(Dropout(0.2))

model.add(Dense(num_labels, activation='softmax'))

model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
conv2d_6 (Conv2D)            (None, 46, 46, 64)        640
_________________________________________________________________
conv2d_7 (Conv2D)            (None, 44, 44, 64)        36928
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 22, 22, 64)        0
_________________________________________________________________
dropout_4 (Dropout)          (None, 22, 22, 64)        0
_________________________________________________________________
conv2d_8 (Conv2D)            (None, 20, 20, 64)        36928
_________________________________________________________________
conv2d_9 (Conv2D)            (None, 18, 18, 64)        36928
_________________________________________________________________
max_pooling2d_4 (MaxPooling2 (None, 9, 9, 64)          0
_________________________________________________________________
dropout_5 (Dropout)          (None, 9, 9, 64)          0
_________________________________________________________________
conv2d_10 (Conv2D)           (None, 7, 7, 128)         73856
_________________________________________________________________
conv2d_11 (Conv2D)           (None, 5, 5, 128)         147584
_________________________________________________________________
max_pooling2d_5 (MaxPooling2 (None, 2, 2, 128)         0
_________________________________________________________________
flatten_1 (Flatten)          (None, 512)               0
_________________________________________________________________
dense_3 (Dense)              (None, 1024)              525312
_________________________________________________________________
dropout_6 (Dropout)          (None, 1024)              0
_________________________________________________________________
dense_4 (Dense)              (None, 1024)              1049600
_________________________________________________________________
dropout_7 (Dropout)          (None, 1024)              0
_________________________________________________________________
dense_5 (Dense)              (None, 7)                 7175
=================================================================
Total params: 1,914,951
Trainable params: 1,914,951
Non-trainable params: 0
```

Above is the summary of the model obtained which has been built using CNN.

model.summary() prints a useful summary of the model, which includes:

- Name and type of all layers in the model.
- Output shape for each layer.
- Number of weight parameters of each layer.
- If the model has general topology, the inputs each layer receives
- The total number of trainable and non-trainable parameters of the model.

#Compiling the model

Categorical crossentropy is used to represent the loss between actual labels and predictions whereas SGD (Stochastic Gradient Descent) is used to reduce the loss and takes only a subset for computation.

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

#Training the model

The model is trained using the batch size, number of epochs and various other parameters initialized earlier in order to gain its accuracy and make it ready for testing.

```
history=model.fit(x_train, y_train,
      batch_size = batch_size,
      epochs = epochs,
      verbose=1,
      validation_data = (x_test, y_test),
      shuffle=True)
```

```
Epoch 1/20
1795/1795 [==============================] - 57s 8ms/step - loss: 1.8249 - accuracy: 0.2448 - val_loss: 1.7978 - val_accuracy: 0.2497
Epoch 2/20
1795/1795 [==============================] - 12s 7ms/step - loss: 1.7609 - accuracy: 0.2694 - val_loss: 1.7059 - val_accuracy: 0.3070
Epoch 3/20
1795/1795 [==============================] - 12s 7ms/step - loss: 1.6944 - accuracy: 0.3190 - val_loss: 1.6220 - val_accuracy: 0.3683
Epoch 4/20
1795/1795 [==============================] - 13s 7ms/step - loss: 1.6359 - accuracy: 0.3563 - val_loss: 1.6105 - val_accuracy: 0.3670
Epoch 5/20
1795/1795 [==============================] - 13s 7ms/step - loss: 1.5962 - accuracy: 0.3689 - val_loss: 1.5086 - val_accuracy: 0.4160
Epoch 6/20
1795/1795 [==============================] - 12s 7ms/step - loss: 1.5397 - accuracy: 0.4043 - val_loss: 1.4827 - val_accuracy: 0.4210
Epoch 7/20
1795/1795 [==============================] - 12s 7ms/step - loss: 1.4799 - accuracy: 0.4282 - val_loss: 1.4010 - val_accuracy: 0.4589
Epoch 8/20
1795/1795 [==============================] - 13s 7ms/step - loss: 1.4400 - accuracy: 0.4382 - val_loss: 1.7218 - val_accuracy: 0.3572
Epoch 9/20
1795/1795 [==============================] - 13s 7ms/step - loss: 1.3912 - accuracy: 0.4682 - val_loss: 1.3412 - val_accuracy: 0.4795
Epoch 10/20
1795/1795 [==============================] - 12s 7ms/step - loss: 1.3523 - accuracy: 0.4755 - val_loss: 1.2999 - val_accuracy: 0.5035
Epoch 11/20
1795/1795 [==============================] - 12s 7ms/step - loss: 1.3159 - accuracy: 0.4975 - val_loss: 1.3965 - val_accuracy: 0.4703
Epoch 12/20
1795/1795 [==============================] - 13s 7ms/step - loss: 1.3013 - accuracy: 0.4978 - val_loss: 1.3010 - val_accuracy: 0.5057
Epoch 13/20
1795/1795 [==============================] - 12s 7ms/step - loss: 1.2705 - accuracy: 0.5127 - val_loss: 1.2406 - val_accuracy: 0.5269
Epoch 14/20
1795/1795 [==============================] - 12s 7ms/step - loss: 1.2495 - accuracy: 0.5214 - val_loss: 1.2319 - val_accuracy: 0.5297
Epoch 15/20
1795/1795 [==============================] - 12s 7ms/step - loss: 1.2140 - accuracy: 0.5348 - val_loss: 1.2573 - val_accuracy: 0.5272
Epoch 16/20
1795/1795 [==============================] - 13s 7ms/step - loss: 1.1902 - accuracy: 0.5510 - val_loss: 1.1995 - val_accuracy: 0.5372
Epoch 17/20
1795/1795 [==============================] - 12s 7ms/step - loss: 1.1802 - accuracy: 0.5510 - val_loss: 1.2460 - val_accuracy: 0.5249
Epoch 18/20
1795/1795 [==============================] - 12s 7ms/step - loss: 1.1673 - accuracy: 0.5543 - val_loss: 1.2002 - val_accuracy: 0.5517
Epoch 19/20
1795/1795 [==============================] - 13s 7ms/step - loss: 1.1435 - accuracy: 0.5621 - val_loss: 1.1913 - val_accuracy: 0.5603
Epoch 20/20
1795/1795 [==============================] - 12s 7ms/step - loss: 1.1223 - accuracy: 0.5727 - val_loss: 1.1610 - val_accuracy: 0.5578
```

#Determine the accuracy

Accuracy is one metric for evaluating the classification models. It is the fraction of predictions a model got right. Formally, accuracy has the following definition:

Accuracy = Number of correct predictions / Total number of predictions

```
from sklearn.metrics import accuracy_score
test_true = np.argmax(y_test, axis=1)
test_pred = np.argmax(model.predict(x_test), axis=1)
print("Model Accuracy: {:.4f}".format(accuracy_score(test_true, test_pred)))
```

#Performance visualization

Visualization of the performance of any machine learning model is an easy way to make sense of the data being poured out of the model and make an informed decision about the changes that need to be made on the parameters or hyperparameters that affects the Machine Learning model.

```
plt.figure(0)
plt.plot(history.history['accuracy'], label='training accuracy')
plt.plot(history.history['val_accuracy'], label='val accuracy')
plt.title('Accuracy')
plt.xlabel('epochs')
plt.ylabel('accuracy')
plt.legend()


plt.figure(1)
plt.plot(history.history['loss'], label='training loss')
plt.plot(history.history['val_loss'], label='val loss')
plt.title('Loss')
plt.xlabel('epochs')
plt.ylabel('loss')
plt.legend()
```

#Plot the confusion matrix

A confusion matrix is a table that is often used to describe the performance of a classification model on a set of test data for which the true values are known. It is used to visualize important predictive analytics like recall, specificity, accuracy, and precision.

```
! pip install -q scikit-plot
import scikitplot as skplt
from sklearn.metrics import classification_report
yhat_valid = model.predict_classes(x_test)
skplt.metrics.plot_confusion_matrix(np.argmax(y_test, axis=1), yhat_valid, figsize=(7,7))
#pyplot.savefig("confusion_matrix_dcnn.png")

print(f'total wrong validation predictions: {np.sum(np.argmax(y_test, axis=1) != yhat_valid)}
\n\n')
print(classification_report(np.argmax(y_test, axis=1), yhat_valid))
```

#Save the model

```
fer_json = model.to_json()
with open("fer.json", "w") as json_file:
  json_file.write(fer_json)
model.save_weights("fer.h5")
```

#Import the necessary libraries for image processing

```
import os
import cv2
import numpy as np
from keras.models import model_from_json
from keras.preprocessing import image
```

#Load the model

```
model = model_from_json(open("fer.json", "r").read())
model.load_weights('fer.h5')
face_haar_cascade = cv2.CascadeClassifier('/content/haarcascade_frontalface_default.xml')
```

#Upload the image

```
import cv2
from google.colab.patches import cv2_imshow
test_image = cv2.imread('/content/baby.jpg')
cv2_imshow(test_image)
```

#Convert to grey scale image

```
gray_image = cv2.cvtColor(test_image, cv2.COLOR_BGR2GRAY)
cv2_imshow(gray_image)
```

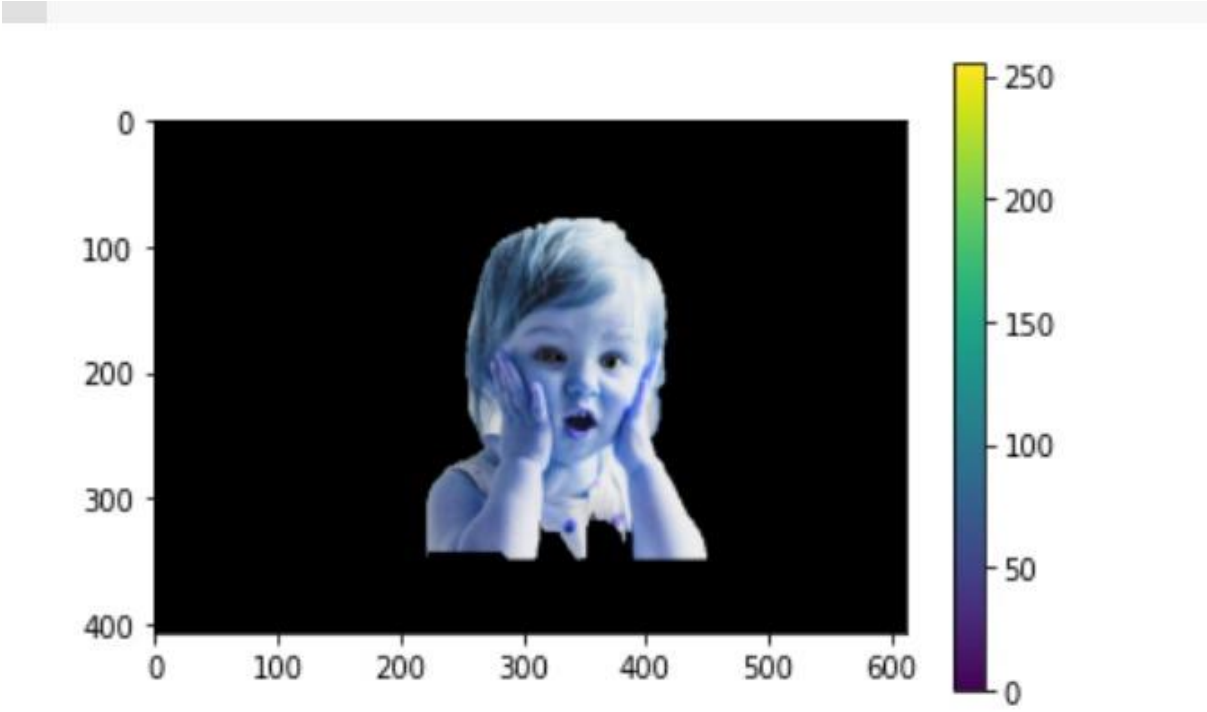
#Background elimination

```
import numpy as np
import cv2 as cv
from matplotlib import pyplot as plt

img = cv.imread('baby.jpg')
mask = np.zeros(img.shape[:2],np.uint8)
bgdModel = np.zeros((1,65),np.float64)
fgdModel = np.zeros((1,65),np.float64)
rect = (50,50,500,300)

cv.grabCut(img,mask,rect,bgdModel,fgdModel,5,cv.GC_INIT_WITH_RECT)
mask2 = np.where((mask==2)|(mask==0),0,1).astype('uint8')
img = img*mask2[:,:,np.newaxis]

plt.imshow(img),plt.colorbar(),plt.show()
```

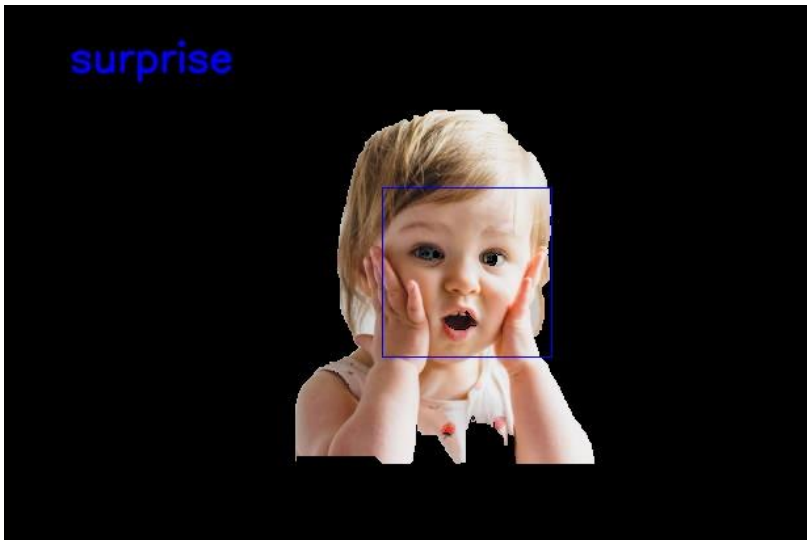


```
import cv2
face_cascade = cv2.CascadeClassifier('/content/haarcascade_frontalface_default.xml')
from keras.preprocessing.image import img_to_array
cv2_imshow(img)
```

#Final output

```
import cv2
face_cascade = cv2.CascadeClassifier('/content/haarcascade_frontalface_default.xml')
faces = face_cascade.detectMultiScale(gray_image, 1.1, 4)
for (x,y,w,h) in faces:
  cv2.rectangle(img,(x,y), (x+w,y+h), (255,0,0))
  roi_gray =  gray_image[y:y+w, x:x+h]
  roi_gray = cv2.resize(roi_gray,(48,48))
  image_pixels = img_to_array(roi_gray)
  image_pixels = np.expand_dims(image_pixels, axis = 0)
  image_pixels /= 255
  predictions = model.predict(image_pixels)
  max_index = np.argmax(predictions[0])
  emotion_detection = ('angry', 'disgust', 'fear', 'happy', 'sad', 'surprise', 'neutral')
  emotion_prediction = emotion_detection[max_index]
  print(emotion_prediction)
  font = cv2.FONT_HERSHEY_SIMPLEX
  org = (50, 50)
```

```
  fontScale = 1
  color = (255, 0, 0)
  thickness = 2
  image = cv2.putText(img, emotion_prediction, org, font, fontScale, color, thickness, cv2.LI
NE_AA)
  cv2_imshow(image)
```



5. RESULTS AND DISCUSSIONS

So the accuracy of the model obtained is 60.30%

```
#accuracy
#loss_and_metrics = model.evaluate(x_train,y_train)
#print(loss_and_metrics)
from sklearn.metrics import accuracy_score
test_true = np.argmax(y_test, axis=1)
test_pred = np.argmax(model.predict(x_test), axis=1)
print("Model Accuracy: {:.4f}".format(accuracy_score(test_true, test_pred)))
```

```
Model Accuracy: 0.6030
```

Fig 5.1: Accuracy of CNN model

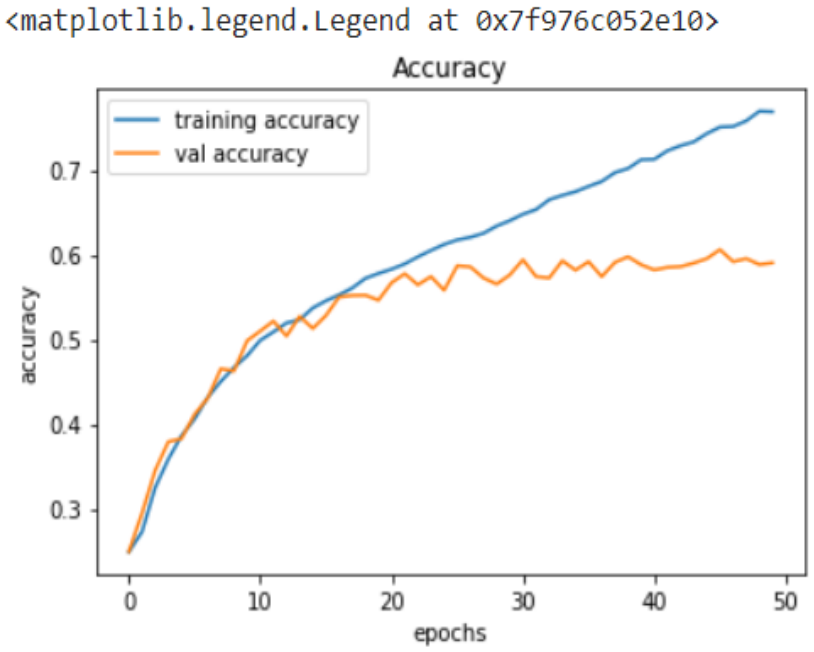


Fig 5.2: Accuracy graph

The above graph shows the comparison between training and validation accuracy.
The validation accuracy is the accuracy which is calculated on the data set that cannot be used for training, whereas training accuracy is the accuracy calculated only on the training set.

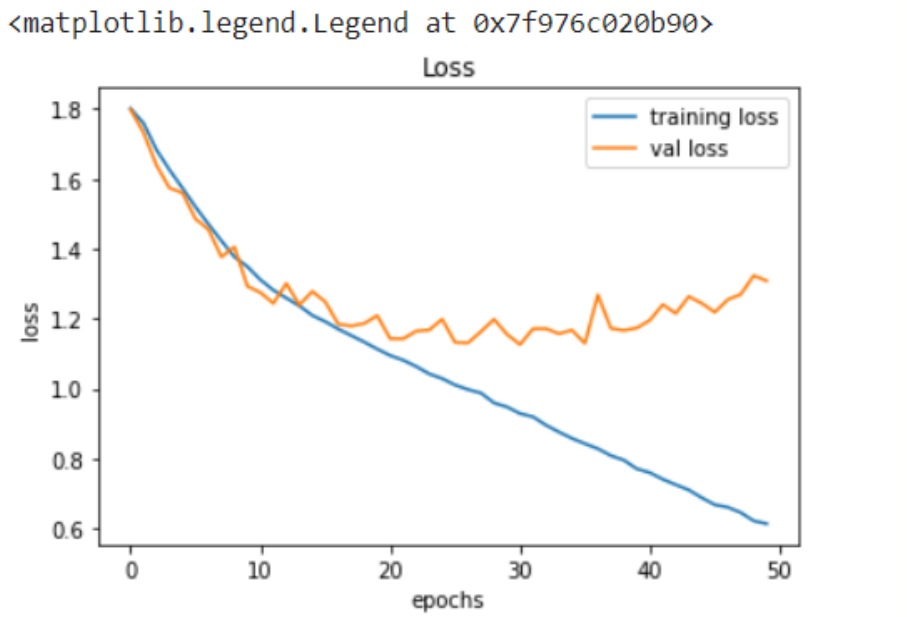


Fig 5.3: Loss graph

The above graph shows a comparison between training loss and validations loss.

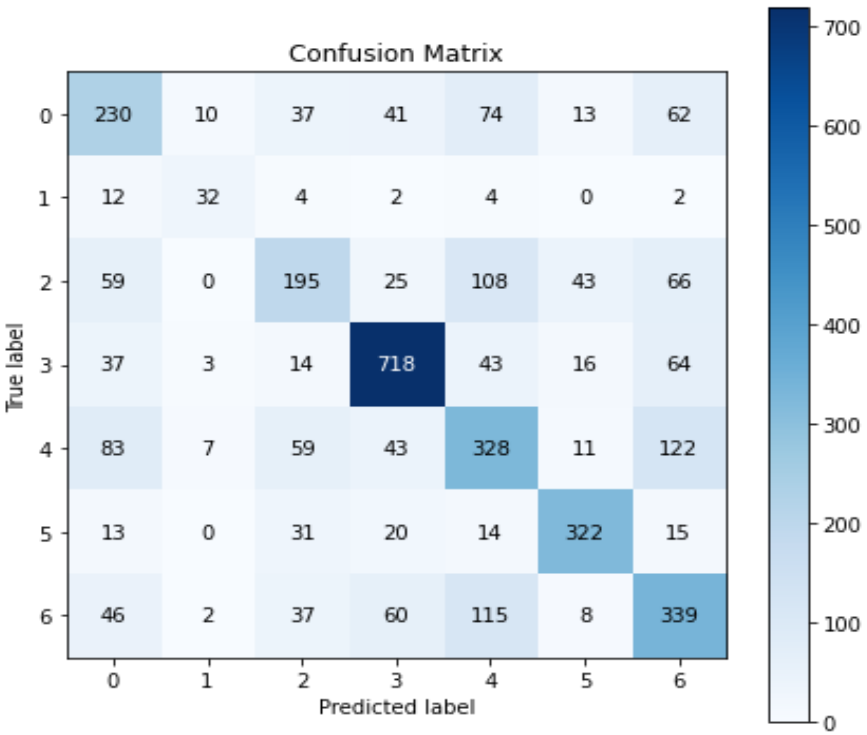


Fig 5.4: Confusion matrix

Background elimination technique:

Background subtraction (BS) is a common and widely used technique for generating a foreground mask (namely, a binary image containing the pixels belonging to moving objects in the scene) by using static cameras.

As the name suggests, BS calculates the foreground mask performing a subtraction between the current frame and a background model, containing the static part of the scene or, more in general, everything that can be considered as background given the characteristics of the observed scene.

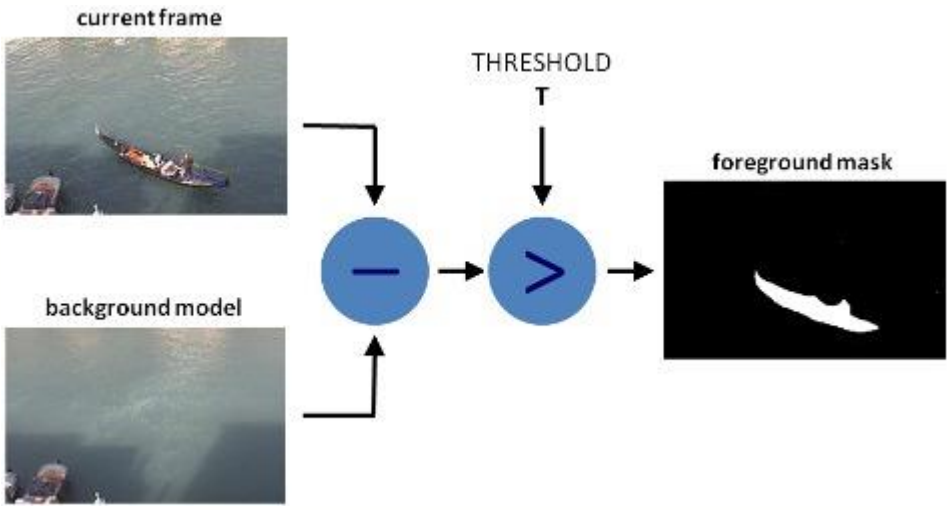


Fig 5.5: Background elimination

Background modeling consists of two main steps:

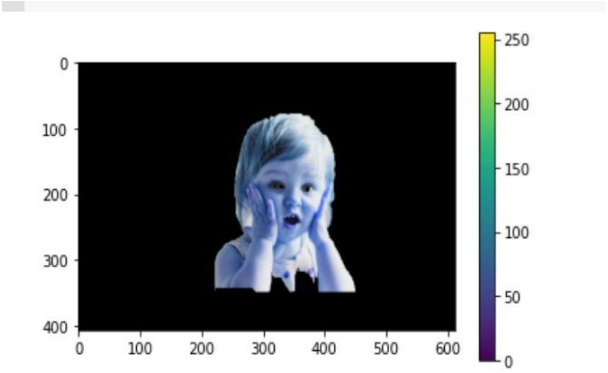
- Background Initialization;
- Background Update.

In the first step, an initial model of the background is computed, while in the second step that model is updated in order to adapt to possible changes in the scene.

This technique works in the following way for this model:

Input

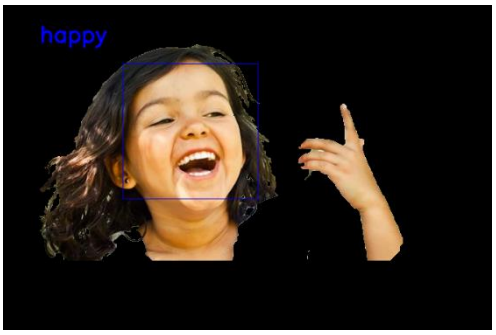
Background elimination



6. TEST CASES

Some of the tested emotions are as follows:

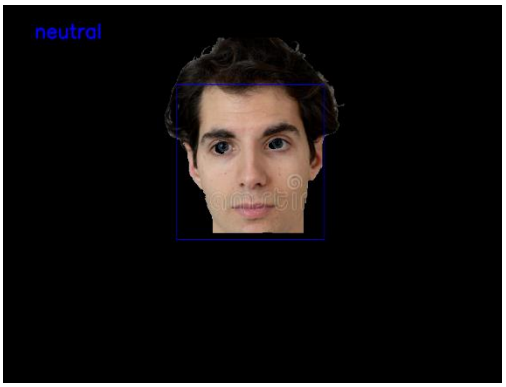
Input **Output**



Input **Output**

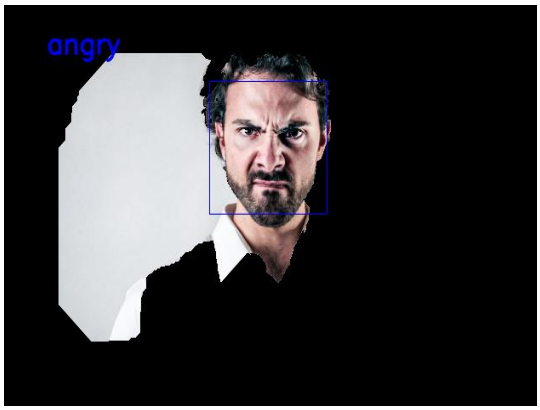
Input

Output



Input

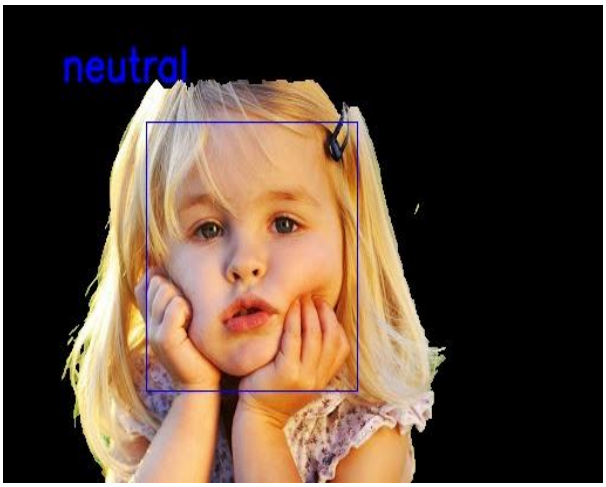
Output



The images are misclassified in some cases when mixed emotions exist in the uploaded image. This misclassification can be observed in the below images:

The given image can be considered as consisting of two different emotions which can be viewed as neutral and sad. So, the model tries to give any one of these emotions as the final predicted emotion.

Input **Output**



7. CONCLUSION

The purpose of this work was not to get conclusive results but to bear out the main challenges and difficulties involved in emotion detection from facial expressions. The project helps in identifying the different emotions of a person which are angry, disgust, happy, sad, fear, surprise and neutral. These facial emotions can be mainly used in E-learning, criminal investigation, online gaming, psychology and many other applications. Using a public dataset of train and test samples for seven different classes of human emotions, we build and train a convolutional neural network model and use it along with haar cascade classifier to produce the final predicted emotion of an uploaded image.

8. FUTURE SCOPE

As emotion recognition technology becomes more sophisticated and more deeply embedded in our array of devices, it will become expected that our computers and phones provide us with a continual progression of customized triggers and messaging. The technology will be found even in future car dashboards, refrigerator doors, and conference room walls -- essentially any surface will become a possible means for detection of emotions. Ultimately, expect emotion recognition to be just another core component of marketing, similar to how "digital marketing" is now really just "marketing."

The future enhancements of this project are:

- To increase the accuracy
- To predict the compound emotions

This work can be carried out by others in order to convert it into a model for predicting the emotions of a person by using the real time capturing of the face. Similarly, the other advancement of this project may include the extraction of images from the videos and predicting the emotions from those images.

9. REFERENCES

[1] Shichuan Du, Yong Tao, and Aleix M. Martinez1 Department of Electrical and Computer Engineering, and Center for Cognitive and Brain Sciences, The Ohio State University, Columbus, OH 43210 Edited by David J. Heeger, New York University, New York, NY, and approved February 28, 2014

[2] Rzayeva, Z, & Emin Alasgarov. 2020. Facial Emotion Recognition using Convolutional Neural Networks. IEEE Xplore (Auckland University of Technology)

[3] Chibelushi, C. C., & Bourel, F. (2003). Facial expression recognition: A brief tutorial overview. CVonline: On-Line Compendium of Computer Vision, 9.

[4] Y.-I. Tian, T. Kanade, and J. F. Cohn, "Recognizing action units for facial expression analysis," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 23, no. 2, pp. 97–115, 2001.

[5] Mendel, N. 2020. Facial Emotion Recognition Using Convolutional Neural Networks (FERC).