Face Emotion Recognition

Using Open CV and CNN

MINOR PROJECT

Supervisor - Dr. Ritesh Kumar Mishra

Department of Electronics and Communication Engineering

National Institute of Technology, Patna

Patna – 800005

Semester – 7th

Shreya -- 1904144

Anu Kumari -- 1904069

Sanjay Kumar -- 1904027

Department of Electronics and Communication Engineering
NATIONAL INSTITUTE OF TECHNOLOGY PATNA
Patna, Bihar, India-800005

ACKNOWLEDGEMENT

My group and I express my sincere gratitude to the Department of Electronics & Communication Engineering for extending all kinds of help while carrying out this work.

I am deeply thankful to my guide, Dr. Ritesh Kumar Mishra, who spared his valuable time to review our work and gave us constant encouragement to continue this work.

We are grateful to our family, friends and all the well-wishers for their help and support.

Shreya , Anu Kumari and Sanjay Kumar
Department of Electronics & Communication Engineering

TABLE OF CONTENTS

Objective of the project:

- The main aim of our project is to develop a robust system which can detect as well as recognize human emotions from live feed.
- To explore underlying deep learning algorithm, CV and neural network CNN as well as relevant feature's extraction techniques like Haar Wavelets or Haar Features with some boosting methods, which would help us in accurate identification of the facial emotions.

Abstract:

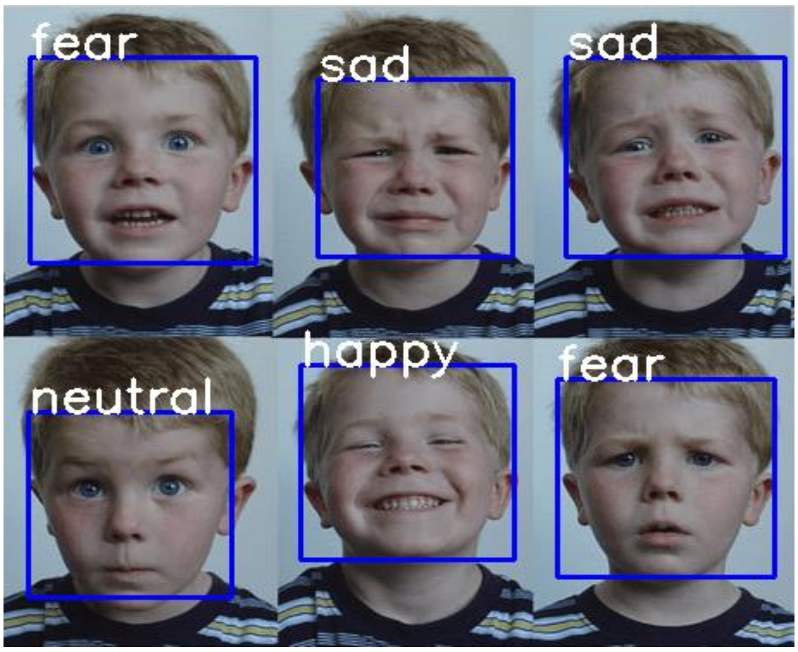
Detecting and recognizing human emotions is a big challenge in computer vision and artificial intelligence. Emotions are a major part of human communication. Most of the communication takes place through emotions. The main aim of our project is to develop a robust system which can detect as well as recognize human emotion from live feed. There are some emotions which are universal to all human beings like angry, sad, happy, surprise, neutral, fear and disgust.

The methodology of this system is based on two stages- facial detection is done by extraction of Haar Cascade features of a face using Viola Jones algorithm and then the emotion is verified and recognized using Convolutional Neural network with DNN under the hood. This has many applications, like in fields of security, surveillance, robotics etc.

Introduction:

The implementation of this project will make use of Deep Learning (DL) Neural Network concepts, more specifically CNN and Computer Vision. It will generate bounding boxes with expressions written as heading on the box relevant to the emotion our model detects.

Human emotion and face expression is one of the most powerful tool of communication. Facial expressions are very convincing. It is found that the linguistic component of a message contributes to only a meagre 7% of the total significance of the effect of the message, whereas the tone indicates about 38% of the total signal and the remaining in turn signified or portrayed 55% of the total message.



Feature Extraction of the facial features is a very widely sought after implementation in the fields of surveillance (video or images), biometrics as well as HCI (human computer interface). Here facial image is used as a medium to read human emotion.

The research on human's emotion can be traced back to the Darwin's pioneer working and since then has attracted a lot of researchers to this area. There are seven basic emotions that are universal to human beings. Namely, neutral, angry, disgust, fear, happy, sad, and surprise and these basic emotions can be recognized from human's facial expression.

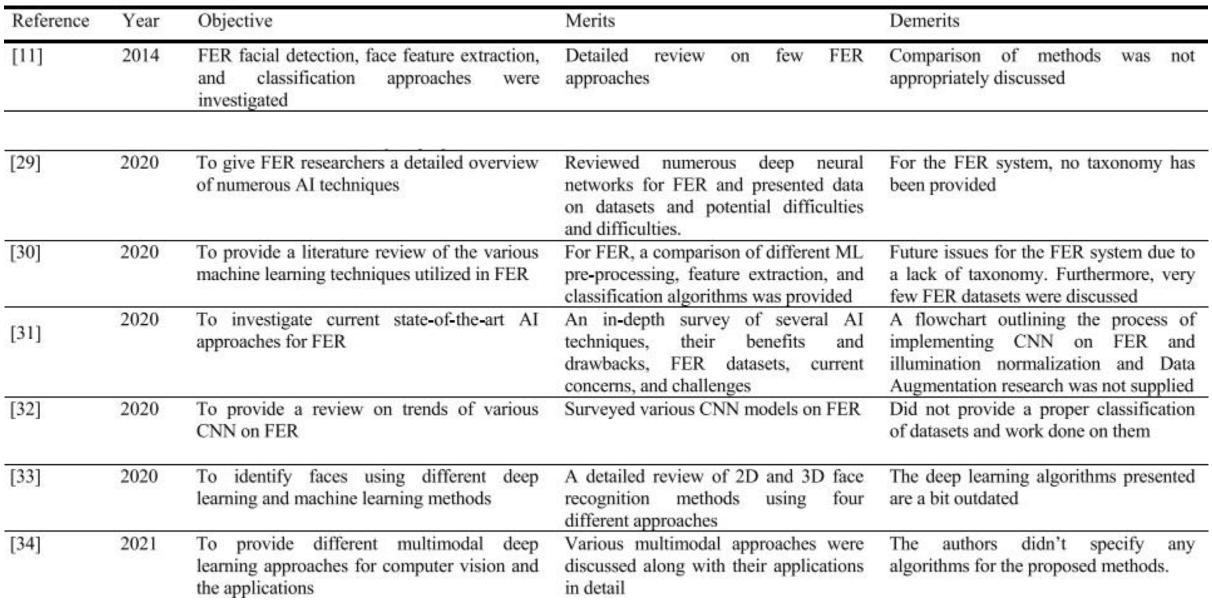
Literature review:

With the help of previous research works, our literature can be divided into three main parts: Face Detection, Feature Extraction and Emotion Classification. Our project is based on the feature extraction technique and emotion detection based on the extracted features.

Emotion recognition can be used as an additional security layer, it can be seen as a second step to facial recognition. This can also be used to recognize if the person is really standing in front of the camera or just a 2-dimensional representation.

Emotion recognition can also be used to provide businesses with better customer response on their products, this could be used as a replacement to the

conventional rating and review system.
Basic Human Emotions are: fear, happy, sad, anger, surprise, neutral, contempt. Detecting the different emotions can be challenging as facial muscle contractions are very minimal in these emotions.
Neural Network and DEEP Learning are used for the purpose of extracting and categorizing these gestures.
We will compare algorithms and feature extraction techniques from different papers.



Reference	Year	Objective	Merits	Demerits
[11]	2014	FER facial detection, face feature extraction, and classification approaches were investigated	Detailed review on few FER approaches	Comparison of methods was not appropriately discussed
[29]	2020	To give FER researchers a detailed overview of numerous AI techniques	Reviewed numerous deep neural networks for FER and presented data on datasets and potential difficulties and difficulties.	For the FER system, no taxonomy has been provided
[30]	2020	To provide a literature review of the various machine learning techniques utilized in FER	For FER, a comparison of different ML pre-processing, feature extraction, and classification algorithms was provided	Future issues for the FER system due to a lack of taxonomy. Furthermore, very few FER datasets were discussed
[31]	2020	To investigate current state-of-the-art AI approaches for FER	An in-depth survey of several AI techniques, their benefits and drawbacks, FER datasets, current concerns, and challenges	A flowchart outlining the process of implementing CNN on FER and illumination normalization and Data Augmentation research was not supplied
[32]	2020	To provide a review on trends of various CNN on FER	Surveyed various CNN models on FER	Did not provide a proper classification of datasets and work done on them
[33]	2020	To identify faces using different deep learning and machine learning methods	A detailed review of 2D and 3D face recognition methods using four different approaches	The deep learning algorithms presented are a bit outdated
[34]	2021	To provide different multimodal deep learning approaches for computer vision and the applications	Various multimodal approaches were discussed along with their applications in detail	The authors didn't specify any algorithms for the proposed methods.

Tools, Modules & Libraries Used:

Python 3.7.0

Python is an object-oriented programming language created by Guido Rossum in 1989. It is ideally designed for rapid prototyping of complex applications. It has interfaces to many OS system calls and libraries and is extensible to C or C++. Many large companies use the Python programming language, including NASA, Google, YouTube, Bit Torrent, etc.

Python 3 is a newer version of the Python programming language which was released in December 2008. This version was mainly released to fix problems that exist in Python 2. The nature of these changes is such that Python 3 was incompatible with Python 2. It is **backward incompatible**. Some features of Python 3 have been backported to Python 2.x versions to make the migration process easy in Python 3. As a result, for any organization who was using Python 2.x version, migrating their project to 3.x needed lots of changes. These changes not only relate to projects and applications but also all the libraries that form part of the Python ecosystem.

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.

K Keras

A deep learning library

It was developed as part of the research effort of project ONEIROS (Open-ended Neuro-Electronic Intelligent Robot Operating System), and its primary author and maintainer is François Chollet, a Google engineer. Chollet is also the author of the Xception deep neural network model.

Tensorflow

Tensorflow is the most used Deep Learning network and it has pre-trained models that easily help with image classification.

TensorFlow is a multipurpose machine learning framework. TensorFlow can be used anywhere from training huge models across clusters in the cloud to running models locally on an embedded system like your phone/IoT devices.

Matplotlib

Matplotlib produces publication-quality figures in a variety of hardcopy formats and interactive environments across platforms. Matplotlib can be used in Python scripts, Python/IPython shells, web application servers, and various graphical user interface toolkits.

Numpy

NumPy is a Python library used for working with arrays. It also has functions for working in domain of linear algebra, Fourier transform, and matrices.NumPy was created in 2005 by Travis Oliphant. It is an open source project and you can use it freely.NumPy stands for Numerical Python.

In Python we have lists that serve the purpose of arrays, but they are slow to process. NumPy aims to provide an array object that is up to 50x faster than traditional Python lists.The array object in NumPy is called ndarray, it provides a lot of supporting functions that make working with ndarray very easy. Arrays are very frequently used in data science, where speed and resources are very important.

NumPy arrays are stored at one continuous place in memory unlike lists, so processes can access and manipulate them very efficiently. This behaviour is called locality of reference in computer science. This is the main reason why NumPy is faster than lists. Also it is optimized to work with latest CPU architectures.

OpenCV

OpenCV is the huge open-source library for the computer vision, machine learning, and image processing. By using it, one can process images and videos to identify objects, faces, or even handwriting of a human. When integrated with various libraries, such as NumPy, python is capable of processing the OpenCV array structure for analysis. To identify image pattern and its various features we use vector space and perform mathematical operations on these features.

The first OpenCV version was 1.0. OpenCV is released under a BSD license and hence it's free for both **academic** and **commercial** use. It has C++, C, Python and Java interfaces and supports Windows, Linux, Mac OS, iOS and Android. When OpenCV was designed the main focus was real-time applications for computational efficiency. All things are written in optimized C/C++ to take advantage of multi-core processing. There are lots of applications which are solved using OpenCV:

- face recognition
- Automated inspection and surveillance
- number of people – count (foot traffic in a mall, etc)
- Vehicle counting on highways along with their speeds
- Interactive art installations
- Anomaly (defect) detection in the manufacturing process (the odd defective products)
- Street view image stitching
- Video/image search and retrieval
- Robot and driver-less car navigation and control
- object recognition
- Medical image analysis
- Movies – 3D structure from motion
- TV Channels advertisement recognition

Approach to the Problem:

A general face detection process has four stages of process flows: face detection, pre-processing, Feature Extraction, and Face Recognition.

Face Detection

This is the first step in face processing. The main purpose of this step is to detect face from the images from dataset. In this step individual images are taken from the dataset, scanned and then verified whether the image contains a face or just background image. The face determination system determines if the input data (image) is a face. After this step the result is sent for pre-processing so that facial features can be extracted from the face image.

Pre-Processing

Since any unprocessed or raw image data is easily corrupted by noise and hardware issues. This step is done to find smooth face images by removing unwanted noise, blur images and shadowing effects. There are many techniques available for image pre-processing.

Many of them are based on pixel transformation like pixel brightness transformation, geometric transformation, image restoration etc. Without pre-processing good quality images cannot be obtained to achieve a high accuracy detection system. The resultant images are used to extract facial expressions

Feature Extraction and Face Recognition

This step is critical step as it extracts the features using the applied feature extraction algorithm. The steps performs compression of information, reduction irrelevant features as well as removing the noise of the data. After this the facial region is converted into a vector with a given dimension in which the facial features correspond to their locations. After the prior step is done, analysis of the features is done and then the recognition part is used to learn each person's face and then store it in the database.

After the model is trained, then the model is tested against a given input image. All the previous steps like pre-processing and others are performed again. If it works perfectly the model is able to correctly determine the person's identity without the consent of the individual himself. While the evaluation of the model must be able to determine if the assumption is correct or not.

Classification

This is the last step in any image processing system. There are numerous methods and techniques to classify images. Neural network is a very powerful technique for classification. It works for both linear and non-linear dataset. It works even for images not in the dataset as it is a self-learning model which consists of many hidden layers.

In the recent years, many models of artificial neural networks has been used. There are many approaches within ANN like deep convolutional neural network, radial basis function neural network, back propagation feed forward neural network, bilinear neural network etc. There are many other classification techniques like clustering, decision tree, support vector machines etc.

Proposed System:

The problem statements we have are having robust and automated face detection, analysis of the captured image and its meaningful analysis by facial expressions.

Creating data sets for test and training and then the designing and the implementation of perfectly fitted classifiers to learn underlying classifiers to learn the vectors of the facial descriptors.

We propose a model design which is capable of recognising up to six models which are considered universal among all walks of cultures. Mainly being fear, happiness, sadness, surprise, disgust and lastly surprise. Our system would be to understand a face and its characteristics and then make a weighted assumption of the identity of the person.

This algorithm is mainly helped from the most widely used algorithms at this task, known as the Viola-Jones algorithm and CNN.

The paper proposes Viola Jones algorithm to extract Haar like features for detecting a face and to verify and categorize human emotion using neural network.

A wide variety of methodologies have been used in this field to detect human emotions. One of the breakthrough publication on Face detection was given by Ihor Paily. This work shows detection of face by extracting Haar like features and then categorizing them using neural network similar to our work.

In another work the scholars have taken an interesting approach to the problem of mapping facial expressions because of the challenges posed by the 2-dimensionlaity of an image and therefore trying to do a digital image analysis using the region of interest. Here the scholars have used the features of the lips and analyze them according to the expression that they make .

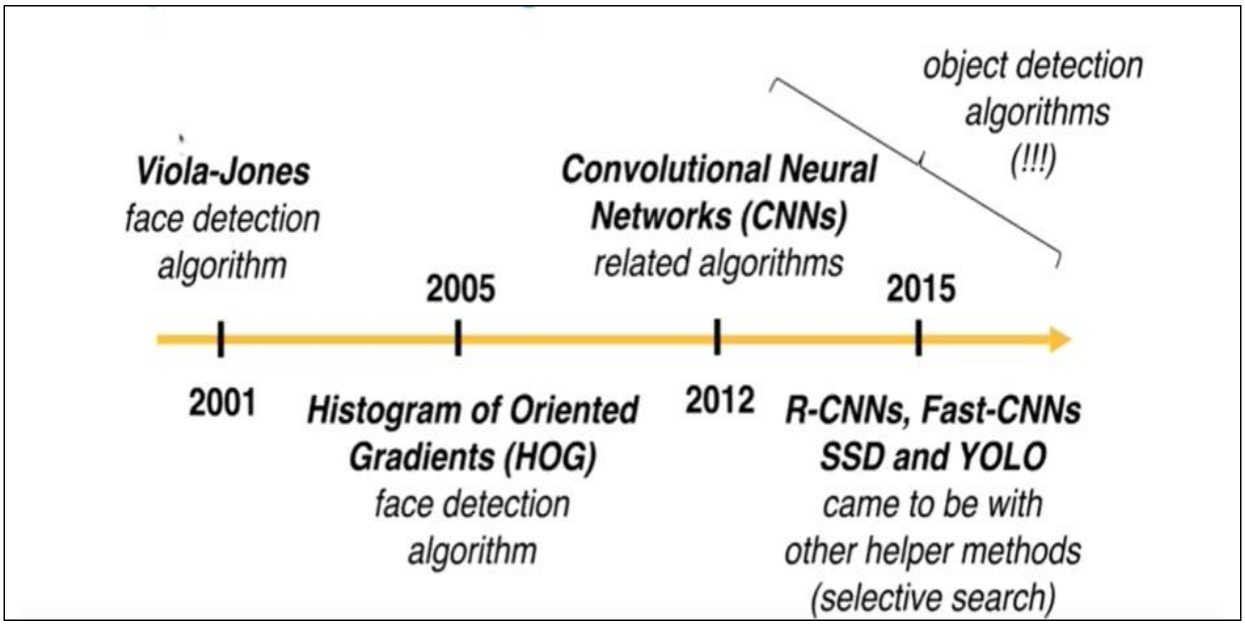
Another work done on facial expression recognition explores statistical unsupervised technique called ICA (independent component analysis) along with the feature optimizing techniques called genetic algorithm uses a concept of evolutionary systems in biology to implement a system to recognize and predict facial emotions.

Algorithms Used:

Computer Vision Algorithms:

The aim of Computer Vision is to identify the given objects on a given image or in a given video.

Evolution of CV algorithms:


object detection algorithms (!!!)
Viola-Jones face detection algorithm
Convolutional Neural Networks (CNNs) related algorithms
2005
2015
2001
Histogram of Oriented Gradients (HOG) face detection algorithm
2012
R-CNNs, Fast-CNNs SSD and YOLO came to be with other helper methods (selective search)

Viola Jones Algorithm:

So basically, the algorithm uses hand coded features such as the distances between the eyes of humans. It can be a given feature or, for example, the width of a human mouth or for example, the distance between the left eye and the nose.

Viola Jones algorithm is named after two computer vision researchers who proposed the method in 2001, Paul Viola and Michael Jones in their paper, “Rapid Object Detection using a Boosted Cascade of Simple Features”. Despite being an outdated framework, Viola-Jones is quite powerful, and its application has proven to be exceptionally notable in real-time face detection. This algorithm is painfully slow to train but can detect faces in real-time with impressive speed.

Given an image (this algorithm works on grayscale image), the algorithm looks at many smaller sub regions and tries to find a face by looking for specific features in each sub region. It needs to check many different positions and scales because an image can contain many faces of various sizes. Viola and Jones used Haar-like features to detect faces in this algorithm.

The Viola Jones algorithm has four main steps:

1. Selecting Haar-like features
2. Creating an integral image
3. Running AdaBoost training
4. Creating classifier cascades

Haar-Wavelets:

It is an Object Detection Algorithm used to identify faces in an image or a real time video. The algorithm uses edge or line detection features proposed by Viola and Jones in their research paper “Rapid Object Detection using a Boosted Cascade of Simple Features” published in 2001.

The algorithm is given a lot of positive images consisting of faces, and a lot of negative images not consisting of any face to train on them. The model created from this training is available at the OpenCV GitHub repository https://github.com/opencv/opencv/tree/master/data/haarcascades.

The repository has the models stored in XML files, and can be read with the OpenCV methods. These include models for face detection, eye detection, upper body and lower body detection, license plate detection etc. Below we see some of the concepts proposed by Viola and Jones in their research.

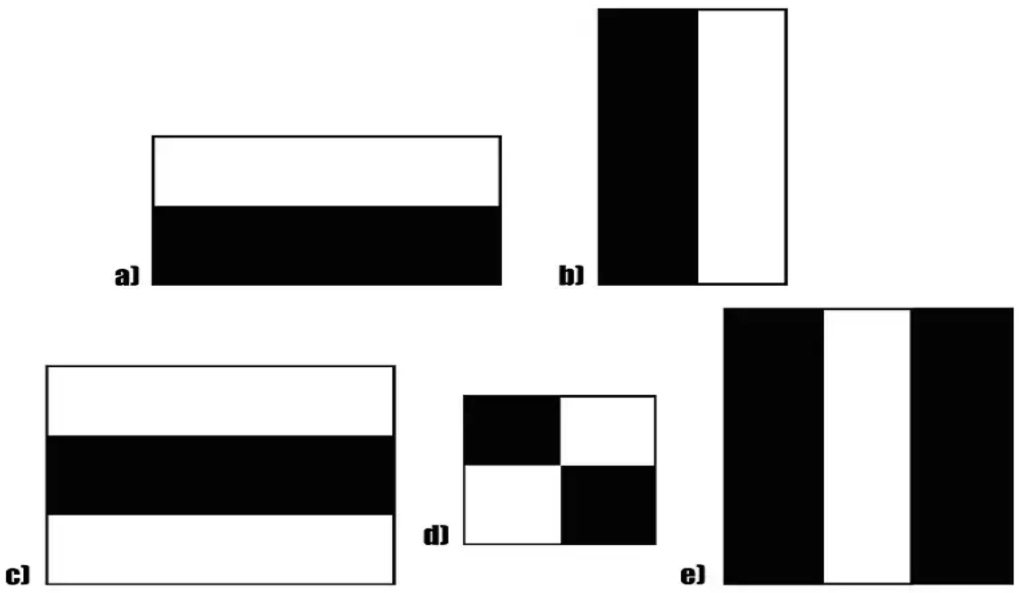


Fig: A sample of Haar features used in the Original Research Paper published by Viola and Jones.

These features on the image makes it easy to find out the edges or the lines in the image, or to pick areas where there is a sudden change in the intensities of the pixels.

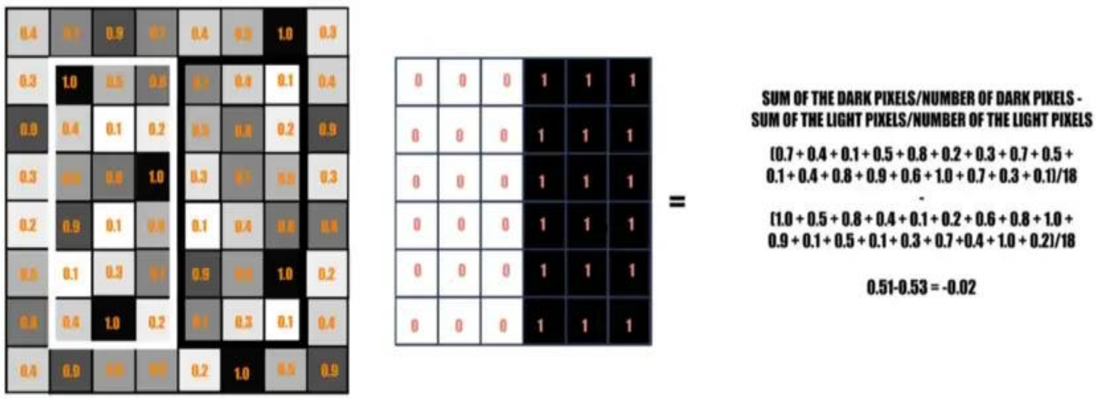


Fig: The rectangle on the left is a sample representation of an image with pixel values 0.0 to 1.0. The rectangle at the centre is a haar kernel which has all the light pixels on the left and all the dark pixels on the right.

The haar calculation is done by finding out the difference of the average of the pixel values at the darker region and the average of the pixel values at the lighter region**. If the difference is close to 1, then there is an edge detected by the haar feature.**

Haar Cascade Detection is one of the oldest yet powerful face detection algorithms invented. It has been there since long, long before Deep Learning became famous. Haar Features were not only used to detect faces, but also for eyes, lips, license number plates etc. The models are stored on GitHub, and we can access them with OpenCV methods.

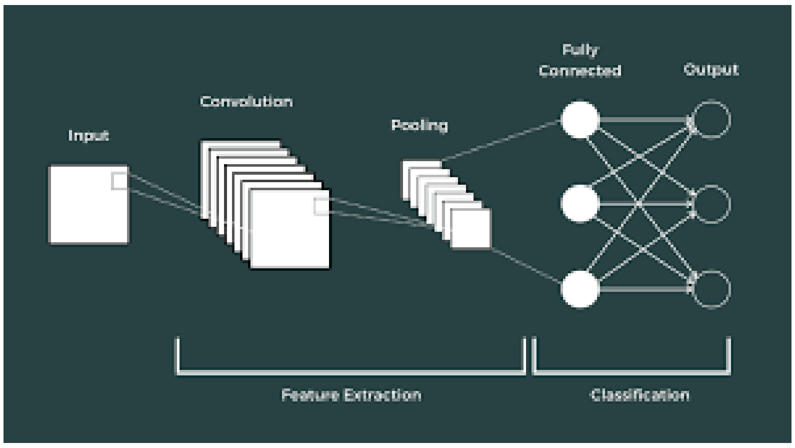
Histogram of Oriented Gradients:

Then four years later, the so-called histogram of oriented gradients method outperformed Viola Jones 'method, which means that it is working better as far as detecting faces is concerned.

Convolutional Neural Networks (CNN):

Then in 2012, convolutional neural networks came to be. As far as object detection algorithms are concerned, convolutional neural networks can outperform any previous algorithms.

Introduction



In the past few decades, Deep Learning has proved to be a very powerful tool because of its ability to handle large amounts of data. The interest to use hidden layers has surpassed traditional techniques, especially in pattern recognition. One of the most popular deep neural networks is Convolutional Neural Networks.

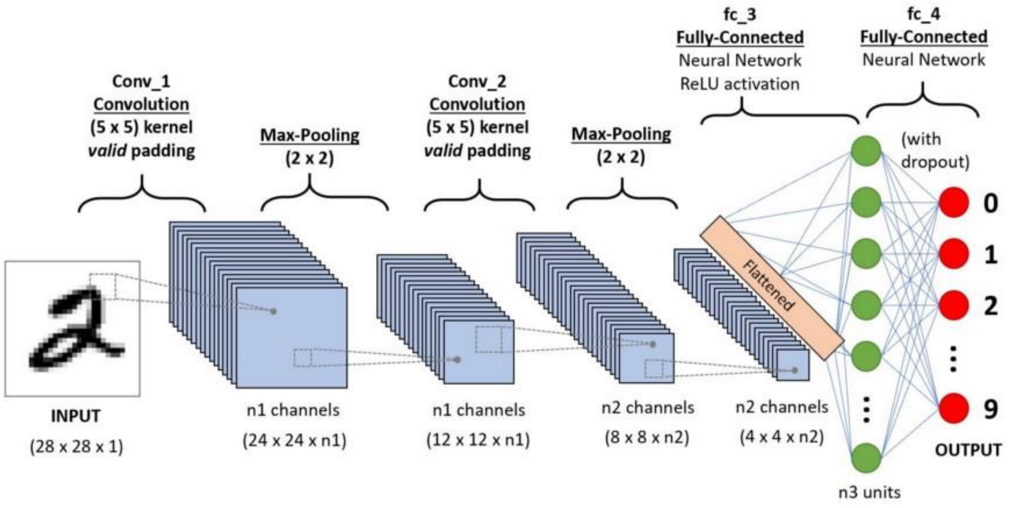
Since the 1950s, the early days of AI, researchers have struggled to make a system that can understand visual data. In the following years, this field came to be known as Computer Vision. In 2012, computer vision took a quantum leap when a group of researchers from the University of Toronto developed an AI model that surpassed the best image recognition algorithms and that too by a large margin.

The AI system, which became known as AlexNet (named after its main creator, Alex Krizhevsky), won the 2012 ImageNet computer vision contest with an amazing 85 percent accuracy. The runner-up scored a modest 74 percent on the test.

At the heart of AlexNet was Convolutional Neural Networks a special type of neural network that roughly imitates human vision. Over the years CNNs have become a very important part of many Computer Vision applications and hence a part of any computer vision course online. So let's take a look at the workings of CNNs

Background of CNNs

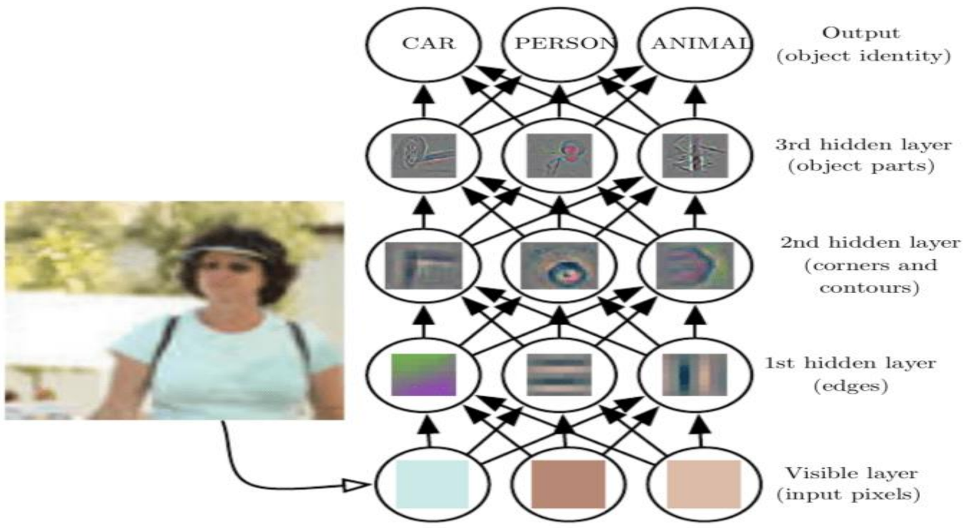
CNN's were first developed and used around the 1980s. The most that a CNN could do at that time was recognize handwritten digits. The important thing to remember about any deep learning model is that it requires a large amount of data to train and also requires a lot of computing resources. This was a major drawback for CNNs at that period.



In 2012 Alex Krizhevsky realized that it was time to bring back the branch of deep learning that uses multi-layered neural networks. The availability of large sets of data, to be more specific ImageNet datasets with millions of labeled images and an abundance of computing resources enabled researchers to revive CNNs.

How does it work?

Convolutional neural networks are composed of multiple layers of artificial neurons. Artificial neurons, a rough imitation of their biological counterparts, are mathematical functions that calculate the weighted sum of multiple inputs and outputs an activation value. When you input an image in a ConvNet, each layer generates several activation functions that are passed on to the next layer.



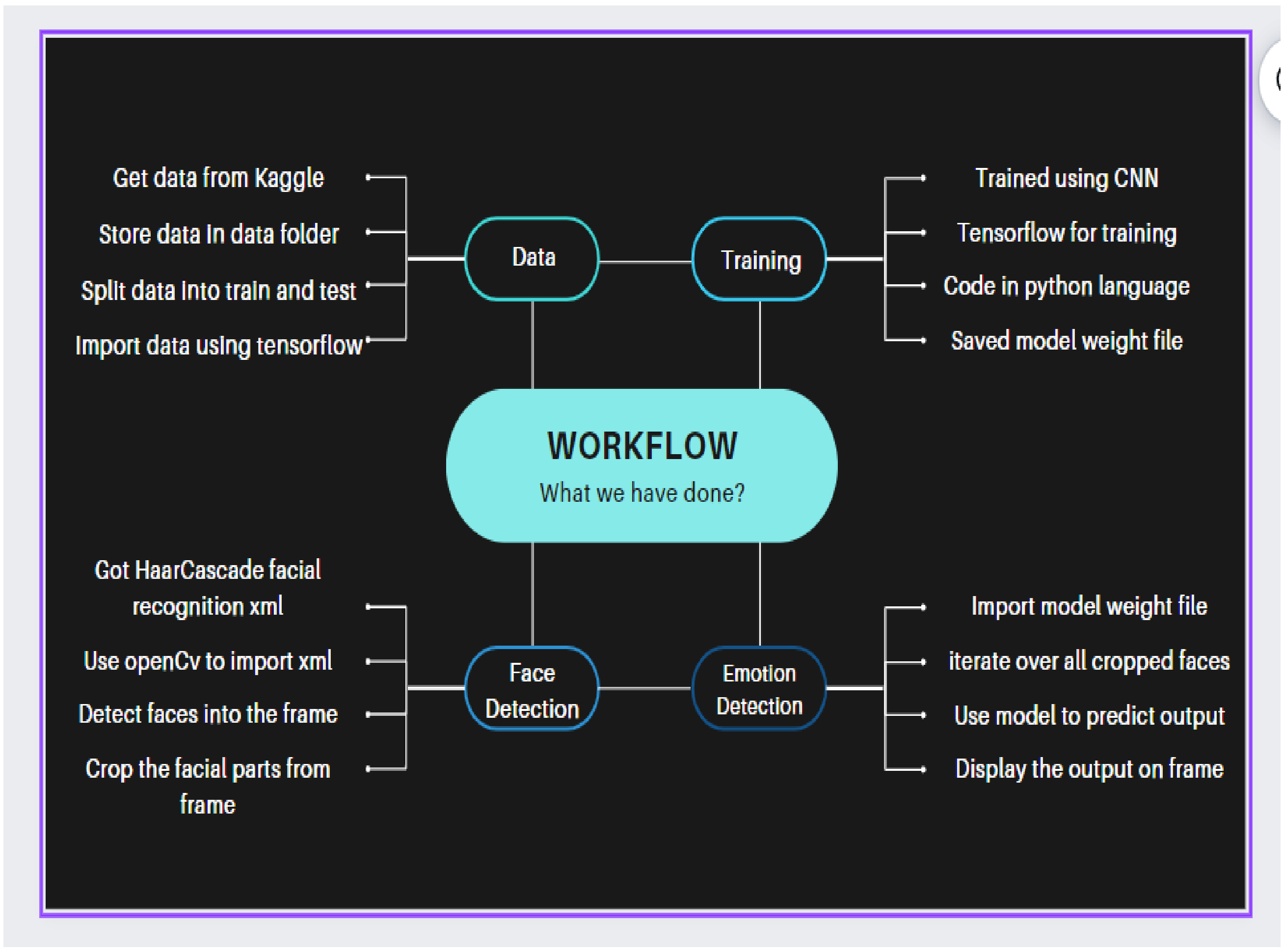
The first layer usually extracts basic features such as horizontal or diagonal edges. This output is passed on to the next layer which detects more complex features such as corners or combinational edges.

Based on the activation map of the final convolution layer, the classification layer outputs a set of confidence scores (values between 0 and 1) that specify how likely the image is to belong to a "class." For instance, if you have a ConvNet that detects cats, dogs, and horses, the output of the final layer is the possibility that the input image contains any of those animals.

Further Evolution:

Back in 2015, several computer vision related approaches have been constructed, such as our fast-CNN, faster CNN and then Yolo algorithm ,You only look once and the so-called assets, the single shot multiple detection algorithm. Of course, there are several approaches, such as selective search that makes these algorithms extremely powerful. So these are the so-called object detection algorithms.

Working:


Get data from Kaggle
Store data In data folder
Split data Into train and test
Import data using tensorflow
Data
Training
Trained using CNN
Tensorflow for training
Code in python language
Saved model weight file
WORKFLOW
What we have done?
Got HaarCascade facial recognition xml
Use openCv to import xml
Detect faces into the frame
Crop the facial parts from frame
Face Detection
Emotion Detection
Import model weight file
iterate over all cropped faces
Use model to predict output
Display the output on frame

- First we need to train our model so that it can detect various emotions, to do that we will use keras library which is built above tensorflow for deep learning.

- We will use emotions dataset from Kaggle and then split it into training and testing set with a ratio of 4:1 so that we can have enough data to test and thus to avoid any overfitting.

- To do that so we will first create a folder named **data** within our source folder and then create two **subfolders train and test**, train will contain the training data whereas test will contain the testing data.

- Now we will import the training and testing data into python code using **flow_from_directory** function within **ImageGenerato**r module which also comes under tensorflow library.

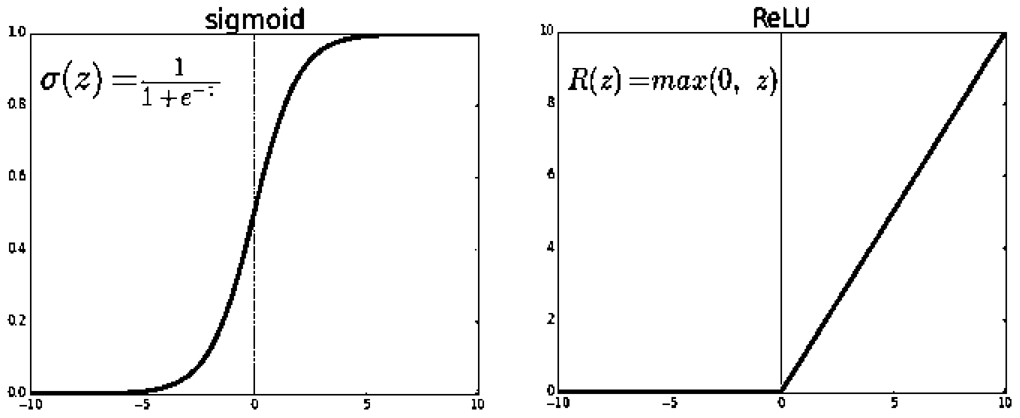
- So we have both training and testing dataset ready for training. Now we will prepare layers to train our deep learning using model function from **Sequential** imported from **tensorflow.keras.models.**

- We will create a four layered deep learning model where one will be the input layer and the last one will be the output layer which will predict the output of the model. Here our activation function used is `relu` .

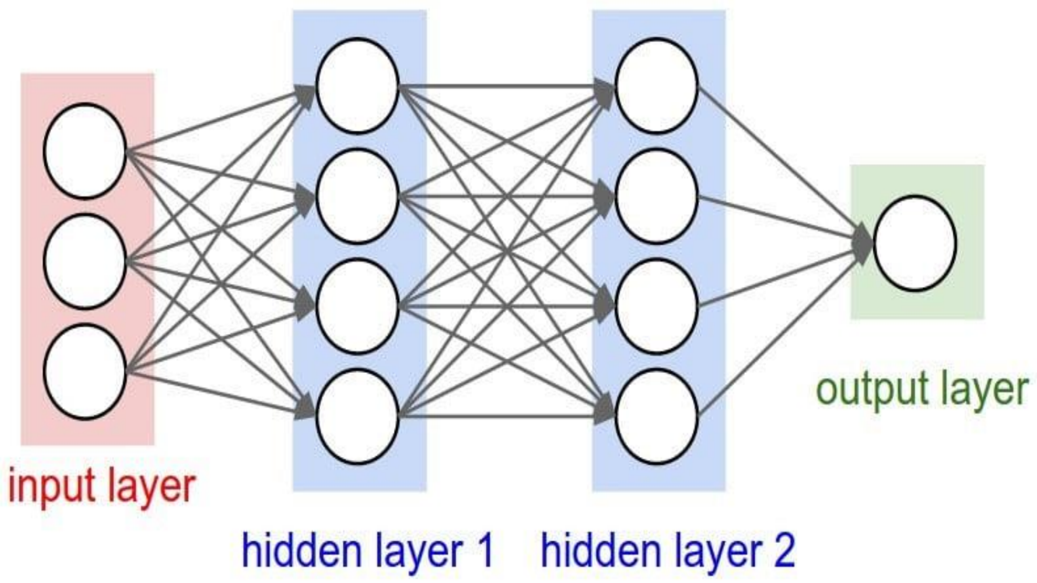
- **ReLU Activation Function:**

  It activates a node only if input is above a certain threshold. F(x)=max(0,x) ; ReLU maths function

  The gradient is either 0 or constant, so it can solve the vanishing gradient issue in the sigmoid activation function.



- Now the layer looks similar to one below:



- TensorFlow internally implement all these layers and trains our model and makes our work much more easier. Our images will pass through all these layers again and again to make the model more accurate and thus predict the emotions.

- After the model is trained we will save this model in a weigh file so that we can import this model in future and use it for prediction to do that we will use model.save('name here') functionality provided by tensorflow.

- Now in order to predict real-time emotion detection we will import the above model again into our code **model.load_weight('name here')** function and to detect the emotion first we need a face.

- To do that we need a Computer Vision library and for that we will use OpenCV library which is written in C++ and available for python. OpenCV will help us to use camera and load the images from it.

- Now to detect the faces within an image we need an image detection algorithm that can do it. There is an old but famous algorithm for such kind of detection named Viola Jones Algorithm that detect facial features using Haar cascade method.

- There are pre-trained xml models available for facial detections using haar cascade, we will be using those models and detect faces in frame.

- Once a face is detected using haar-cascade method, we will take the coordinates of the rectangular bounding box and then capture that face.

- Now we use it to predict the emotion by using model.pridict('image') method. This will then pass the image through various layers we have already implemented to predict the emotion. These layers are pre-trained and will use the existing weights and the emotion that will have the highest probability will come as an output of the prediction.

- Then it will use OpenCv to put image in rectangular box and put the predicted emotion text above the box. The prediction will actually be a number and we have to map that number to the emotion as per the order of the train folder.

- If there are multiple faces detected then we will have an array of coordinates and we will iterate over those coordinates and predict those images one-by-one.
- Finally to close the openCv window we will use the standard key 'q', which is user-defined in code.

- This is how we will be able to predict the emotions using deep learning and haar-cascade face detection in real-time.

How to run application:

To run the application we have to following steps:-

1. First install **Python 3.7.0**
2. Now change directory (cd) into the project folder and install the required python packages using *"pip install -r requirements.txt".*
3. Once all the packages are installed we will then go into "*src*" folder.

4. Now, if you haven't already trained the model you have to train it first to do that write *"python emotions.py –mode train"*.
5. Training the model will take a bit time because it will train it for 50 epochs.
6. After the training is completed we will move forward to real-time prediction and to do this we will use command *"python emotions.py –mode display"*.
7. After executing the above command an OpenCv window will appear and you will be able to see working prediction of our model in real-time.

Data Collection:

Working Codes:

dataset_prepare.py file

```
import numpy as np
import pandas as pd
from PIL import Image
from tqdm import tqdm
import os

# convert string to integer
def atoi(s):
    n = 0
    for i in s:
        n = n*10 + ord(i) - ord("0")
    return n

# making folders
outer_names = ['test','train']
inner_names = ['angry', 'disgusted', 'fearful', 'happy', 'sad',
'surprised', 'neutral']
os.makedirs('data', exist_ok=True)
for outer_name in outer_names:
    os.makedirs(os.path.join('data',outer_name), exist_ok=True)
    for inner_name in inner_names:
        os.makedirs(os.path.join('data',outer_name,inner_name),
exist_ok=True)

# to keep count of each category
angry = 0
disgusted = 0
fearful = 0
happy = 0
sad = 0
```

```
surprised = 0
neutral = 0
angry_test = 0
disgusted_test = 0
fearful_test = 0
happy_test = 0
sad_test = 0
surprised_test = 0
neutral_test = 0

df = pd.read_csv('./fer2013.csv')
mat = np.zeros((48,48),dtype=np.uint8)
print("Saving images...")

# read the csv file line by line
for i in tqdm(range(len(df))):
    txt = df['pixels'][i]
    words = txt.split()

    # the image size is 48x48
    for j in range(2304):
        xind = j // 48
        yind = j % 48
        mat[xind][yind] = atoi(words[j])

    img = Image.fromarray(mat)

    # train
    if i < 28709:
        if df['emotion'][i] == 0:
            img.save('train/angry/im'+str(angry)+'.png')
            angry += 1
        elif df['emotion'][i] == 1:
            img.save('train/disgusted/im'+str(disgusted)+'.png')
            disgusted += 1
        elif df['emotion'][i] == 2:
            img.save('train/fearful/im'+str(fearful)+'.png')
            fearful += 1
        elif df['emotion'][i] == 3:
            img.save('train/happy/im'+str(happy)+'.png')
            happy += 1
        elif df['emotion'][i] == 4:
            img.save('train/sad/im'+str(sad)+'.png')
            sad += 1
        elif df['emotion'][i] == 5:
            img.save('train/surprised/im'+str(surprised)+'.png')
            surprised += 1
        elif df['emotion'][i] == 6:
            img.save('train/neutral/im'+str(neutral)+'.png')
            neutral += 1

    # test
    else:
        if df['emotion'][i] == 0:
            img.save('test/angry/im'+str(angry_test)+'.png')
            angry_test += 1
        elif df['emotion'][i] == 1:
            img.save('test/disgusted/im'+str(disgusted_test)+'.png
')
            disgusted_test += 1
        elif df['emotion'][i] == 2:
            img.save('test/fearful/im'+str(fearful_test)+'.png')
            fearful_test += 1
        elif df['emotion'][i] == 3:
            img.save('test/happy/im'+str(happy_test)+'.png')
            happy_test += 1
        elif df['emotion'][i] == 4:
            img.save('test/sad/im'+str(sad_test)+'.png')
            sad_test += 1
        elif df['emotion'][i] == 5:
            img.save('test/surprised/im'+str(surprised_test)+'.png
')
            surprised_test += 1
        elif df['emotion'][i] == 6:
            img.save('test/neutral/im'+str(neutral_test)+'.png')
            neutral_test += 1

print("Done!")
```

emotions.py file

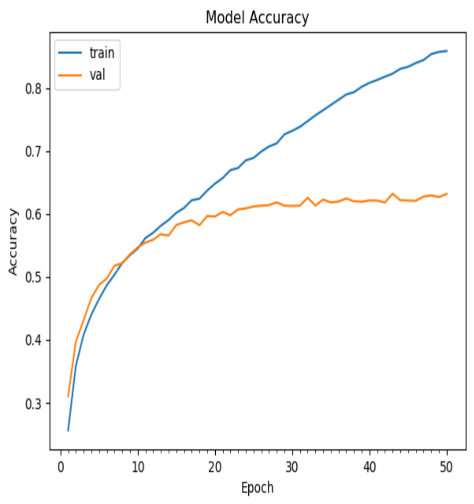
```
import numpy as np
import argparse
import matplotlib.pyplot as plt
import cv2
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense, Dropout, Flatten
from tensorflow.keras.layers import Conv2D
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.layers import MaxPooling2D
from tensorflow.keras.preprocessing.image import
ImageDataGenerator
```

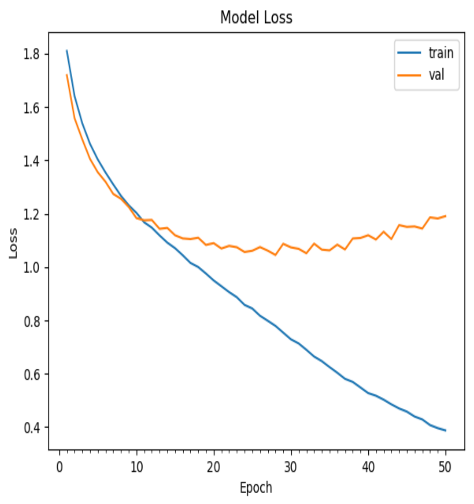
```python
import os
os.environ['TF_CPP_MIN_LOG_LEVEL'] = '2'

# command line argument
ap = argparse.ArgumentParser()
ap.add_argument("--mode",help="train/display")
mode = ap.parse_args().mode

# plots accuracy and loss curves
def plot_model_history(model_history):
    """
    Plot Accuracy and Loss curves given the model_history
    """
    fig, axs = plt.subplots(1,2,figsize=(15,5))
    # summarize history for accuracy
    axs[0].plot(range(1,len(model_history.history['accuracy'])+1),
model_history.history['accuracy'])
    axs[0].plot(range(1,len(model_history.history['val_accuracy'])
+1),model_history.history['val_accuracy'])
    axs[0].set_title('Model Accuracy')
    axs[0].set_ylabel('Accuracy')
    axs[0].set_xlabel('Epoch')
    axs[0].set_xticks(np.arange(1,len(model_history.history['accur
acy'])+1),len(model_history.history['accuracy'])/10)
    axs[0].legend(['train', 'val'], loc='best')
    # summarize history for loss
    axs[1].plot(range(1,len(model_history.history['loss'])+1),mode
l_history.history['loss'])
    axs[1].plot(range(1,len(model_history.history['val_loss'])+1),
model_history.history['val_loss'])
    axs[1].set_title('Model Loss')
    axs[1].set_ylabel('Loss')
    axs[1].set_xlabel('Epoch')
    axs[1].set_xticks(np.arange(1,len(model_history.history['loss'
])+1),len(model_history.history['loss'])/10)
    axs[1].legend(['train', 'val'], loc='best')
    fig.savefig('plot.png')
    plt.show()

# Define data generators
train_dir = 'data/train'
val_dir = 'data/test'

num_train = 28709
num_val = 7178
batch_size = 64
num_epoch = 50
```

```python
train_datagen = ImageDataGenerator(rescale=1./255)
val_datagen = ImageDataGenerator(rescale=1./255)

train_generator = train_datagen.flow_from_directory(
        train_dir,
        target_size=(48,48),
        batch_size=batch_size,
        color_mode="grayscale",
        class_mode='categorical')

validation_generator = val_datagen.flow_from_directory(
        val_dir,
        target_size=(48,48),
        batch_size=batch_size,
        color_mode="grayscale",
        class_mode='categorical')

# Create the model
model = Sequential()

model.add(Conv2D(32, kernel_size=(3, 3), activation='relu',
input_shape=(48,48,1)))
model.add(Conv2D(64, kernel_size=(3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Conv2D(128, kernel_size=(3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(128, kernel_size=(3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.25))

model.add(Flatten())
model.add(Dense(1024, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(7, activation='softmax'))

# If you want to train the same model or try other models, go for
this
if mode == "train":
    model.compile(loss='categorical_crossentropy',optimizer=Adam(l
r=0.0001, decay=1e-6),metrics=['accuracy'])
    model_info = model.fit_generator(
            train_generator,
            steps_per_epoch=num_train // batch_size,
            epochs=num_epoch,
            validation_data=validation_generator,
            validation_steps=num_val // batch_size)
    # plot_model_history(model_info)
    model.save_weights('model.h5')

# emotions will be displayed on your face from the webcam feed
elif mode == "display":
    model.load_weights('model.h5')

    # prevents openCL usage and unnecessary logging messages
    cv2.ocl.setUseOpenCL(False)

    # dictionary which assigns each label an emotion (alphabetical
order)
    emotion_dict = {0: "Angry", 1: "Disgusted", 2: "Fearful", 3:
"Happy", 4: "Neutral", 5: "Sad", 6: "Surprised"}

    # start the webcam feed
    cap = cv2.VideoCapture(0)
    while True:
        # Find haar cascade to draw bounding box around face
        ret, frame = cap.read()
        if not ret:
            break
        facecasc =
cv2.CascadeClassifier('haarcascade_frontalface_default.xml')
        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        faces = facecasc.detectMultiScale(gray,scaleFactor=1.3,
minNeighbors=5)

        for (x, y, w, h) in faces:
            cv2.rectangle(frame, (x, y-50), (x+w, y+h+10), (255,
0, 0), 2)
            roi_gray = gray[y:y + h, x:x + w]
            cropped_img =
np.expand_dims(np.expand_dims(cv2.resize(roi_gray, (48, 48)), -1),
0)
            prediction = model.predict(cropped_img)
            maxindex = int(np.argmax(prediction))
            cv2.putText(frame, emotion_dict[maxindex], (x+20, y-
60), cv2.FONT_HERSHEY_SIMPLEX, 1, (255, 255, 255), 2, cv2.LINE_AA)

        cv2.imshow('Video',
cv2.resize(frame,(1600,960),interpolation = cv2.INTER_CUBIC))
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break

    cap.release()
    cv2.destroyAllWindows()
```

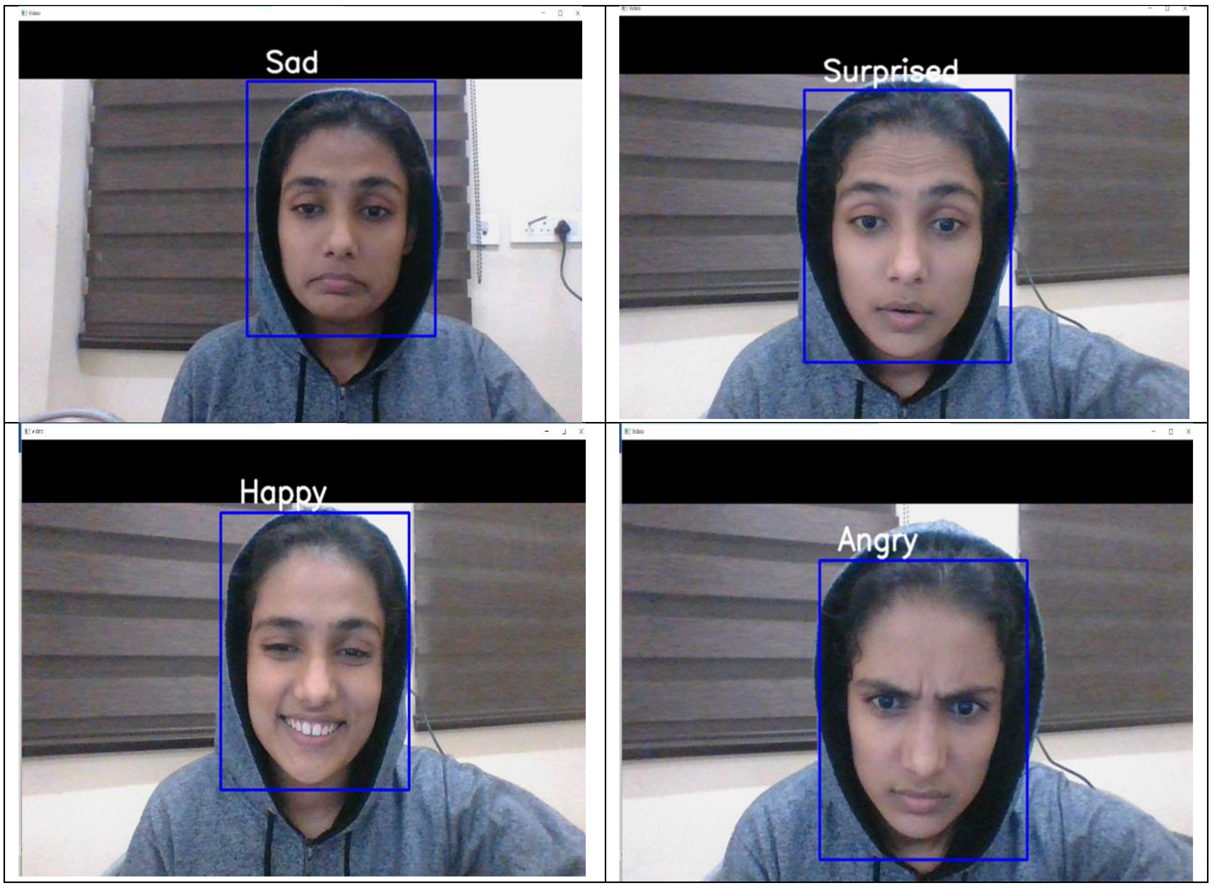
RESULTS:

Following will be the result after training. The below graph is plotted using matplotlib library of python.





Detected Emotions:



Challenges & Issues:

- There is a lot of improvement we can do in this project. First thing we can improve is the facial detection algorithm we are using, although Viola Jones Face detection algorithm is widely used and easy to implement. It works fine with frontal faces but fails to detect faces in any other position or orientation.
- There are a lot of other face detection models available which are highly accurate and can detect multiple faces at different distance into the frame with much more accuracy. Some of those models are **YOLO (You only look once), SSD (Single shot detection).** These are some pre-trained model available that can be used to detect multiple objects including facial features but the only limitation of using these models are that although they are powerful but require a lot of computation power and you will need a powerful GPU to render these models with high frame-rate.
- Regarding the emotion detection model we are using simple CNN through keras-tensorflow here. There is also a scope of improvement here that we can use a hybrid model that can train out own neural network. That will be a hybrid one and it will provide a better accuracy with less loss. To do this we must have a deep knowledge of Machine Learning and Deep learning along with some maths. Also we can improve our dataset and research for a much better and large dataset that can contribute in increasing our present accuracy.
- Apart from all these improvements we can also setup our system to use GPU (Graphical Processing unit), if present, for better training speed and frame rate. This can be done using CUDA software provided by Nvidia.

Conclusion:

We built our CNN model which also uses dense neural nets under the hood. We tested our data with several trained models with the help of Tensorflow and finally successfully detected human face emotions with help of Voila Jones and CNN algorithm all implemented with Tensorflow library's in-built functions.

We can of course go for better methods when we have grasped good knowledge of ML, DL and OpenCV higher algorithms like YOLO and SSD.

References

[1] Rajesh K.M, Naveenkumar M, “A Robust Method for Face Recognition and Face Emotion Detection System using Support Vector Machines”ž in 2016 International Conference on Electrical, Electronics, Communication, Computer and Optimization Techniques (ICEECCOT).

[2] Raluca Boia, Radu Dogaru, Laura Florea, “A comparison of several Classifiers for Eye Detection on Emotion Expressing Faces”ž in 2013 4th International Symposium on Electrical and Electronics Engineering (ISEEE).

[3] Hadi Seyedarabi, Ali Aghagolzadeh, Soharb Khanmihammadi, 2004, “Recognition of six basic facial expression By feature-points tracking using RBF Neural network and fuzzy inference system,2004 IEEE International Conference on Multimedia and Expo (ICME), pp 1219-1222.

[4] B. Fasel and J. Luettin. Automatic facial expression analysis: A survey. Pattern Recognition, 2003.

[5] Bhaskar Gupta, Sushant Gupta, Arun Kumar Tiwari, Face Detection Using Gabor Feature Extraction and Artificial Neural Network.

[6] Anchal Garg, Dr. Rohit Bajaj, Facial expression recognition & classification using hybridization of ICA, GA, and Neural Network for Human-Computer Interaction, Journal of Network Communications and Emerging Technologies (JNCET), 2015.

[7] Namrata Mahajan, Harshad Mahajan, Emotion Detection Algorithm, International Journal of Electrical and Electronics Research, Vol. 2, Issue 2, pp: (56–60), Month: April - June 2014.

[8] Govind U. Kharat, Sanjay V. Dudul, Emotion Recognition from Facial Expression Using Neural Networks, IEEE 2008.

[9] Debishree Dagar, Abir Hudait, H. K. Tripathy, M. N. Das, Automatic Emotion Detection Model from Facial Expression, International Conference on Advanced Communication Control and Computing Technologies 2016.

[10] What are Convolution Neural Network ? , https://www.analyticsvidhya.com/blog/2021/05/convolutional-neural-networks-cnn/

[11] Tensorflow Documents: https://www.tensorflow.org/api_docs/python/tf/keras

[12] Vamshi Krishna Gudipati, Oindrila Ray Barman, Mofica Gaffoor, Harshagandha, Abdelshakour Abuzneid, Efficient Facial Expression Recognition Using Adaboost and Haar Cascade Classifiers, 2016 IEEE.