VIT®
Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

School of Computer Science and Engineering

J Component report

Programme : MTech Integrated CSE with Spl. In BA

Course Title : Artificial Intelligence and Knowledge Based Systems Course Code: CSE3088

Slot : A1+TA1

Title: FEEDBACK ANALYSIS USING EMOTION RECOGNITION

Team Members: Simran Bohra (20MIA1024)

A Sri Karthik (20MIA1032)

Vinayaka R Srinivas (20MIA1041)

Faculty: Dr.R.Reena Roy

Sign:

Date:

1. INTRODUCTION

AI is a branch of Computer Science concerned with the study and creation of computer systems that exhibit some form of intelligence: systems that learn new concepts and tasks, systems that can reason and draw useful conclusions about the world around us, systems that can understand a natural language or perceive and comprehend a visual scene and systems that perform other types of feats that require human type of intelligence.

Emotion recognition is one of the many facial recognition technologies that have developed and grown through the years. Facial emotion recognition is used to allow a certain program to examine and process the expressions on a human's face. Using advanced image dispensation, this software functions like a human brain that makes it capable of recognizing emotions too. It is AI or "Artificial Intelligence" that detects and studies different facial expressions to use with additional information presented to them. This is useful for a variety of purposes, including investigations and interviews, and allows authorities to detect the emotions of a person with just the use of technology.

1.1 Objective

The objective of emotion recognition is identifying emotions of a human. The emotion can be captured either from the face of a person or from their voice. In our project we focus on identifying human emotion from facial expressions. Emotion recognition is one of the useful tasks and can be used as a base for many real-time applications like Feedback Analysis.

1.2 Abstract

Human emotion recognition plays an important role in the interpersonal relationship. The automatic recognition of emotions has been an active research topic from early eras. Therefore, there are several advances made in this field. Emotions are reflected from speech, hand and gestures of the body and through facial expressions. Hence extracting and understanding of emotion has a high importance of the interaction between human and machine communication. Our project mainly focuses on emotion recognition using facial expressions and then using those expressions for feedback analysis.

1.3 Motivation

The motivation behind choosing this topic specifically lies in the huge investments large corporations do in feedbacks and surveys but fail to get equitable response on their investments. Emotion Detection through facial gestures is a technology that aims to improve product and services performance by monitoring customer behavior to certain products or service staff by their evaluation.

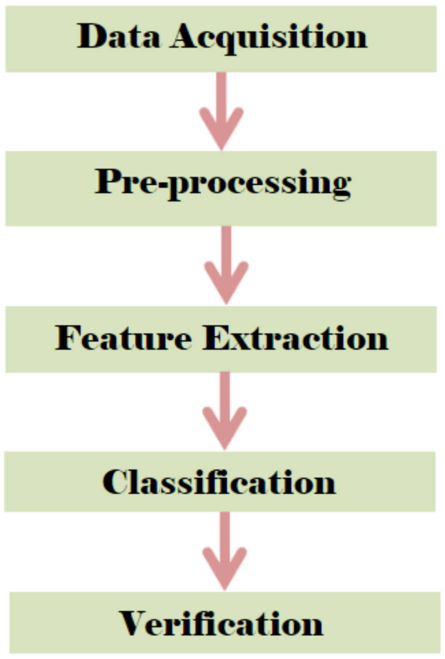
1.4 Applications

Disney uses emotion-detection tech to find out opinion on a completed project, other brands have used it to directly inform advertising and digital marketing.
Kellogg’s is just one high-profile example, having used Affectiva’s software to test audience reaction to ads for its cereal.
Unilever does this, using HireVue’s AI-powered technology to screen prospective candidates based on factors like body language and mood.

1.5 Existing Model

- Data acquisition is the process of gathering, filtering, and cleaning data before the data is put in a data warehouse or any other storage solution.
- Pre-processing is a conventional name for operations with images at the lowest level of abstraction, where both input and output are intensity images.
- Feature extraction is a set of methods that map input features to new output features. It is a process of dimensionality reduction by which an initial set of raw data is reduced to more manageable groups for processing.
- Classification refers to a predictive modeling problem where a class label is predicted for a given example of input data. Classification is a task that requires the use of machine learning algorithms that learn how to assign a class label to examples from the problem domain.

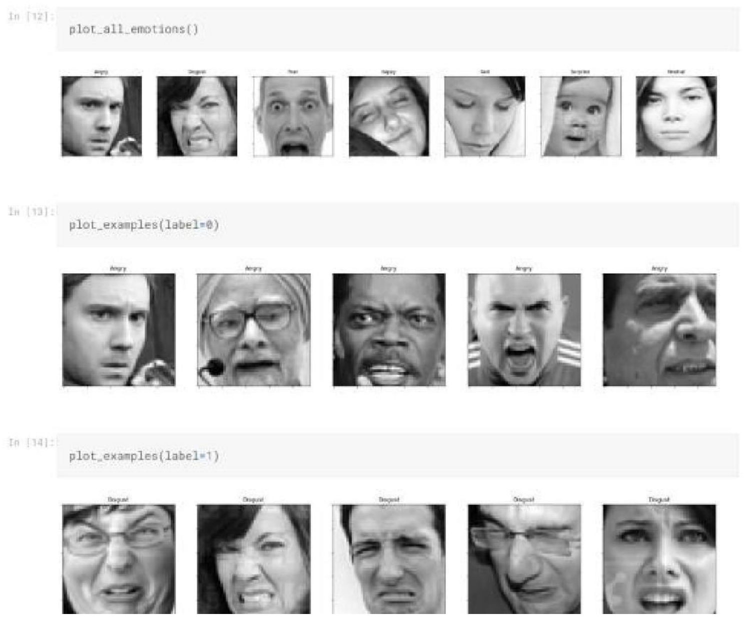
- Verification refers to the output of an algorithm after it has been trained on a historical dataset and applied to new data when forecasting the likelihood of a particular outcome, it mean evaluating the system in several conditions and observing its behavior, watching for defects.



1.6 About the dataset

The data consists of 48x48 pixel grayscale images of faces. The faces have been categorized into facial expression in to one of seven categories (0=Angry, 1=Disgust, 2=Fear, 3=Happy, 4=Sad, 5=Surprise, 6=Neutral). The training set consists of 28,709 example.

The public test set used for the leaderboard consists of 3,589 examples. The final test set, which was used to determine the winner of the competition, consists of another 3,589 examples. This dataset was prepared by Pierre-Luc Carrier and Aaron Courville, as part of an ongoing research project.



```
In [12]: plot_all_emotions()
```

```
In [13]: plot_examples(label=0)
```

```
In [14]: plot_examples(label=1)
```

1.7 Data Pre-Processing

The fer2013.csv consists of three columns namely emotion, pixels and purpose.

- The column in pixel first of all is stored in a list format.
- Since computational complexity is high for computing pixel values in the range of (0-255), the data in pixel field is normalized to values between [0-1].
- The face objects stored are reshaped and resized to the mentioned size of 48X48.
- The respective emotion labels and their respective pixel values are stored in objects.
- We use scikit-learn'strain_test_split () function to split the dataset into training and testing data.
- The test_size being 0.2 meaning, 20% of data is for validation while 80% of the data will be trained.

1.8 Convolutional Neural Networks

Convolutional neural networks, or CNNs, are widely used for image classification, object recognition, and detection. Three types of layers can summarize its structure: convolution, pooling, and classification, as shown. The CNN architecture must be defined according to the application and is usually defined by the number of alternate convolution and pooling players, number of neurons in each layer, and choice of activation function.

1.9 Why CNN

Because of its large number of layers and successful applications, CNNs are one of the preferred techniques for deep learning. Its architecture allows automatic extraction of diverse image features, like edges, circles, lines, and texture. The extracted features are increasingly optimized in further layers. It is important to emphasize that the values of the kernel filters applied in the convolution layers are the result of back propagation during CNN training.

1.10 Steps followed

1. Data Preprocessing
2. Image Augmentation
3. Feature Extraction
4. Training
5. Validation

1.11 Coding

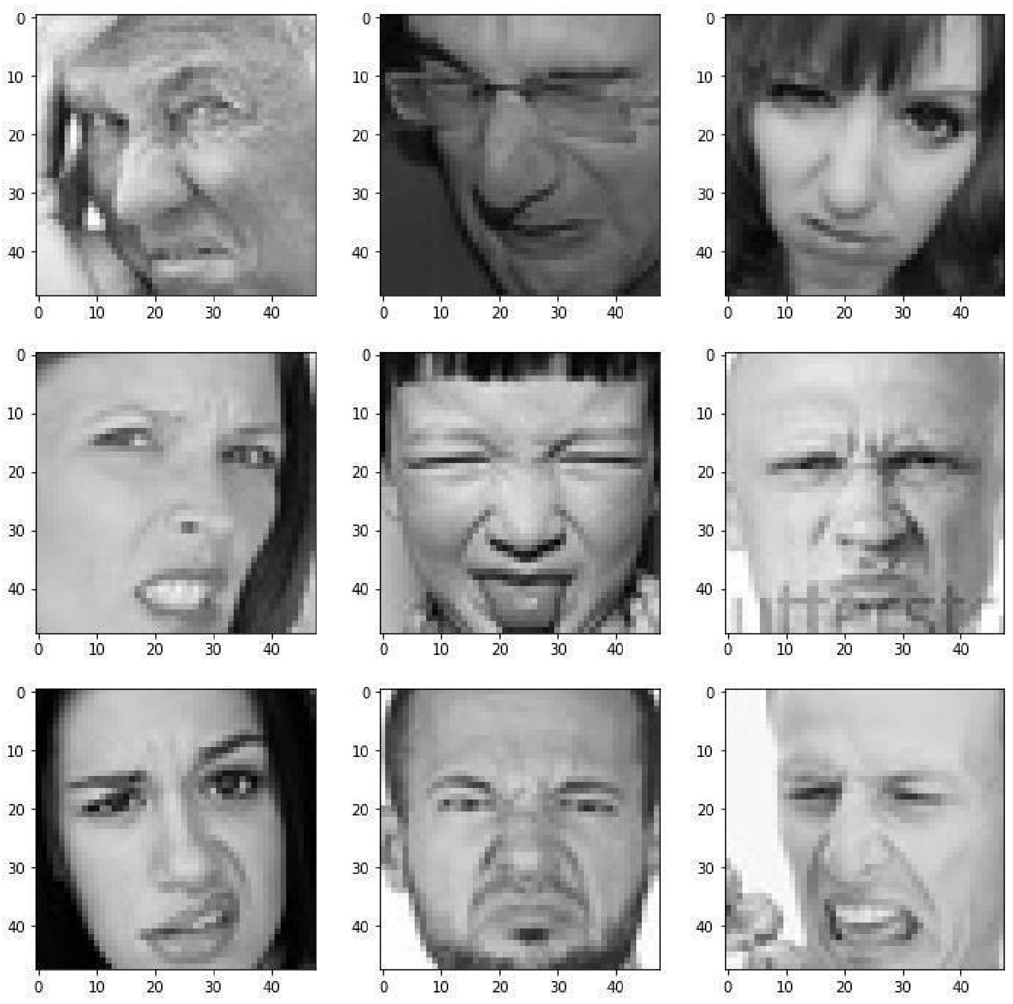
Importing Libraries

In [1]:
```python
import matplotlib.pyplot as plt
import numpy as np
import pandas as pd
import seaborn as sns
import os

# Importing Deep Learning Libraries

from keras.preprocessing.image import load_img, img_to_array
from keras.preprocessing.image import ImageDataGenerator
from keras.layers import Dense,Input,Dropout,GlobalAveragePooling2D,Flatten,Conv2D,BatchNormalization,Activation,MaxPooling2D
from keras.models import Model,Sequential
from keras.optimizers import Adam,SGD,RMSprop
```

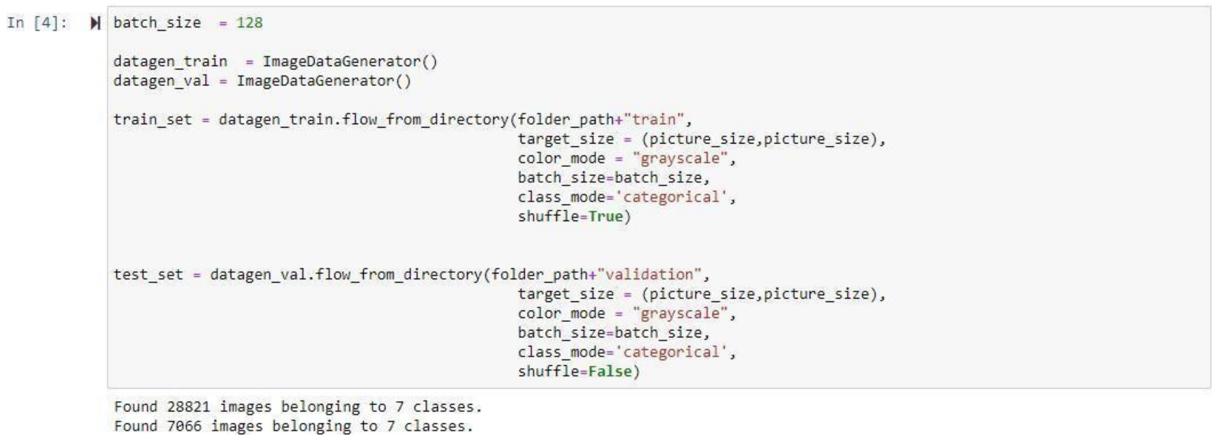
Displaying Images ¶

In [2]:
```python
picture_size = 48
folder_path = "../input/face-expression-recognition-dataset/images/"
```

In [3]:
```python
expression = 'disgust'

plt.figure(figsize= (12,12))
for i in range(1, 10, 1):
    plt.subplot(3,3,i)
    img = load_img(folder_path+"train/"+expression+"/"+
                   os.listdir(folder_path + "train/" + expression)[i], target_size=(picture_size, picture_size))
    plt.imshow(img)
plt.show()
```

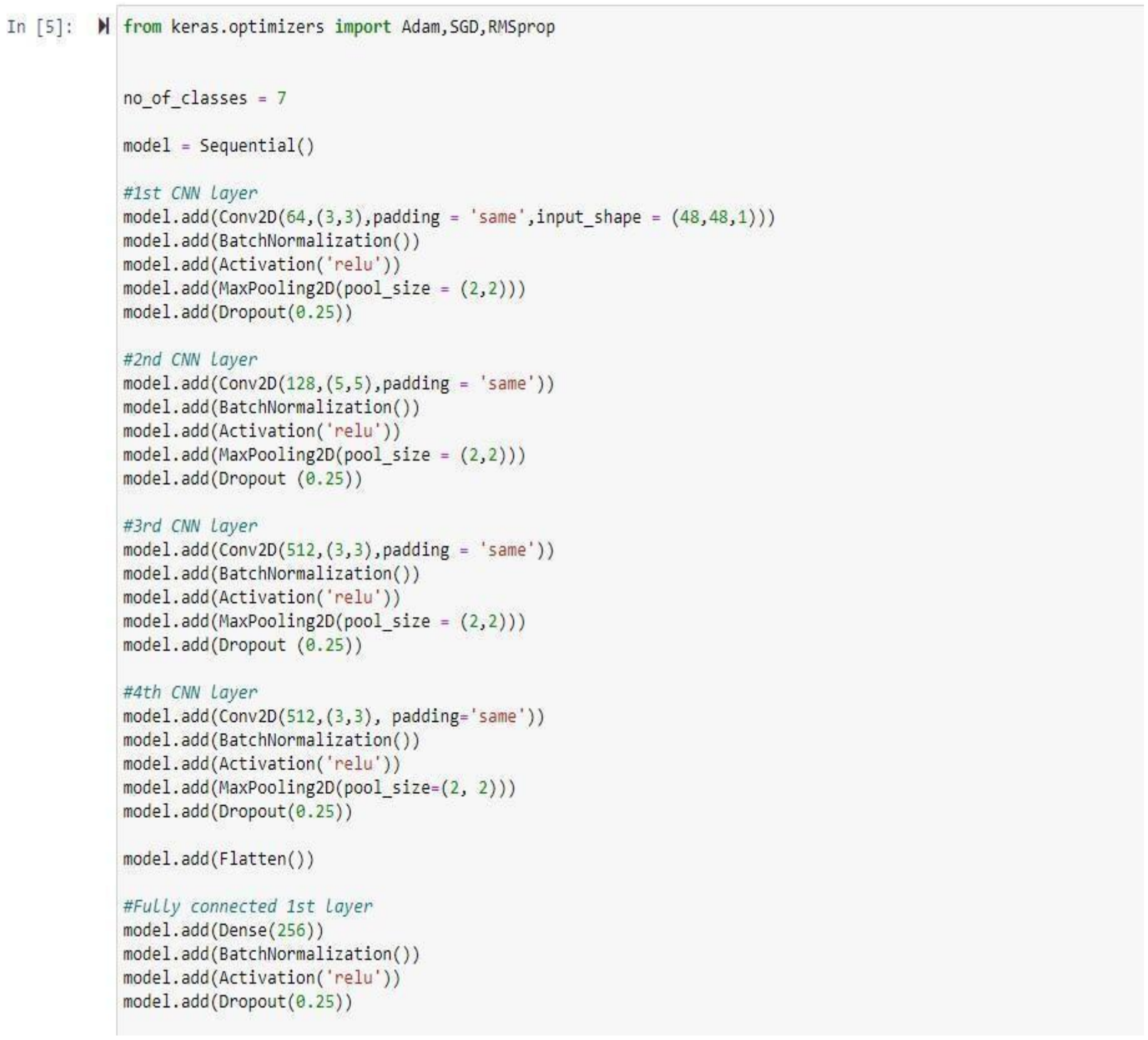


Making Training and Validation Data

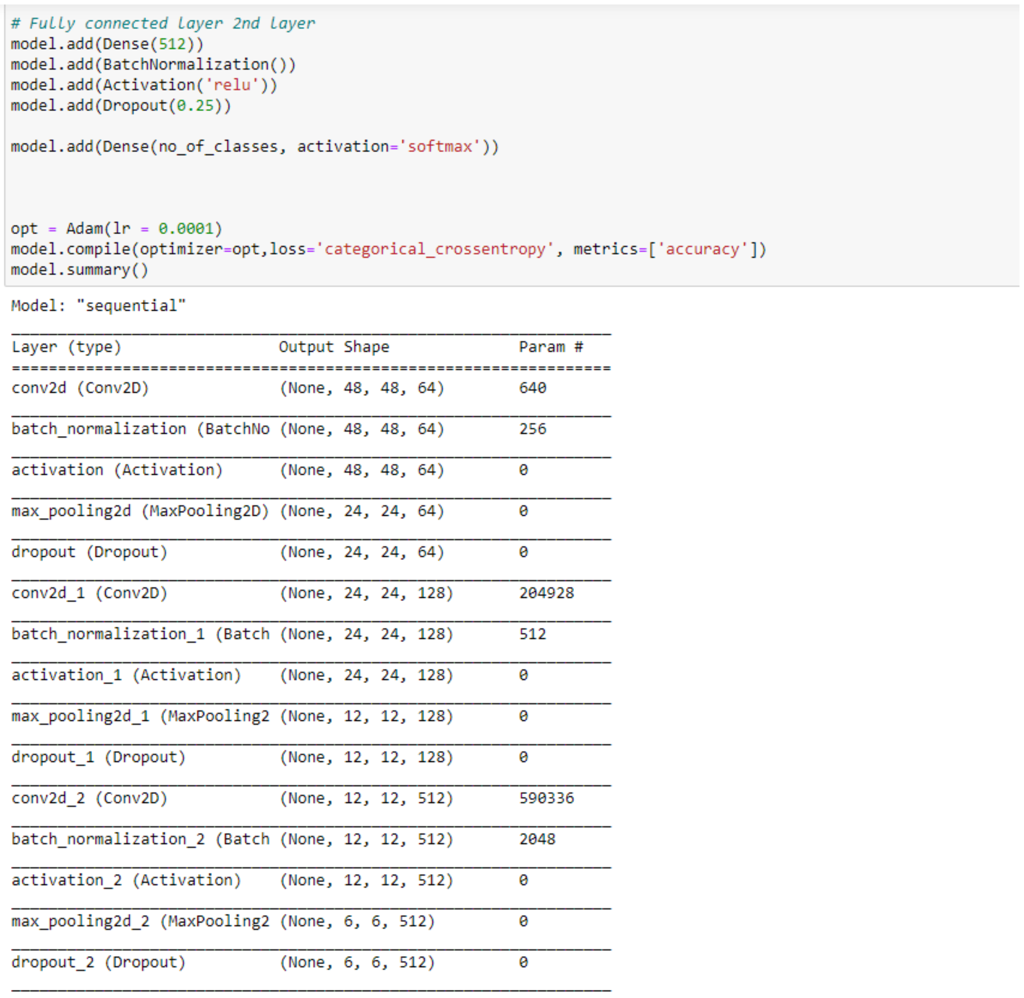


```
In [4]:  batch_size  = 128

         datagen_train  = ImageDataGenerator()
         datagen_val = ImageDataGenerator()

         train_set = datagen_train.flow_from_directory(folder_path+"train",
                                                       target_size = (picture_size,picture_size),
                                                       color_mode = "grayscale",
                                                       batch_size=batch_size,
                                                       class_mode='categorical',
                                                       shuffle=True)


         test_set = datagen_val.flow_from_directory(folder_path+"validation",
                                                       target_size = (picture_size,picture_size),
                                                       color_mode = "grayscale",
                                                       batch_size=batch_size,
                                                       class_mode='categorical',
                                                       shuffle=False)

         Found 28821 images belonging to 7 classes.
         Found 7066 images belonging to 7 classes.
```

Model Building

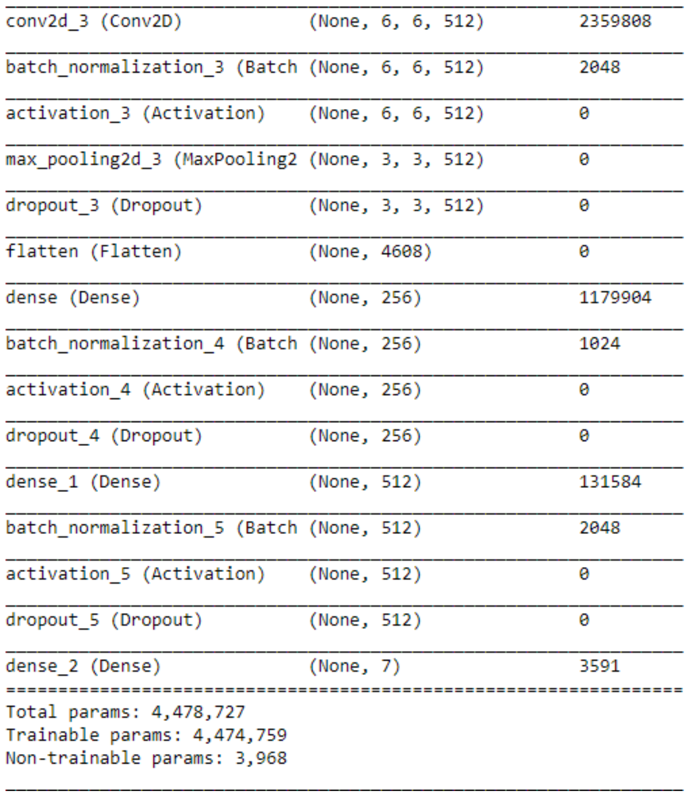


```
In [5]:  from keras.optimizers import Adam,SGD,RMSprop


         no_of_classes = 7

         model = Sequential()

         #1st CNN layer
         model.add(Conv2D(64,(3,3),padding = 'same',input_shape = (48,48,1)))
         model.add(BatchNormalization())
         model.add(Activation('relu'))
         model.add(MaxPooling2D(pool_size = (2,2)))
         model.add(Dropout(0.25))

         #2nd CNN layer
         model.add(Conv2D(128,(5,5),padding = 'same'))
         model.add(BatchNormalization())
         model.add(Activation('relu'))
         model.add(MaxPooling2D(pool_size = (2,2)))
         model.add(Dropout (0.25))

         #3rd CNN layer
         model.add(Conv2D(512,(3,3),padding = 'same'))
         model.add(BatchNormalization())
         model.add(Activation('relu'))
         model.add(MaxPooling2D(pool_size = (2,2)))
         model.add(Dropout (0.25))

         #4th CNN layer
         model.add(Conv2D(512,(3,3), padding='same'))
         model.add(BatchNormalization())
         model.add(Activation('relu'))
         model.add(MaxPooling2D(pool_size=(2, 2)))
         model.add(Dropout(0.25))

         model.add(Flatten())

         #Fully connected 1st layer
         model.add(Dense(256))
         model.add(BatchNormalization())
         model.add(Activation('relu'))
         model.add(Dropout(0.25))
```



```
# Fully connected layer 2nd layer
model.add(Dense(512))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(Dropout(0.25))

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



opt = Adam(lr = 0.0001)
model.compile(optimizer=opt,loss='categorical_crossentropy', metrics=['accuracy'])
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #   
=================================================================
conv2d (Conv2D)              (None, 48, 48, 64)        640       
_________________________________________________________________
batch_normalization (BatchNo (None, 48, 48, 64)        256       
_________________________________________________________________
activation (Activation)      (None, 48, 48, 64)        0         
_________________________________________________________________
max_pooling2d (MaxPooling2D) (None, 24, 24, 64)        0         
_________________________________________________________________
dropout (Dropout)            (None, 24, 24, 64)        0         
_________________________________________________________________
conv2d_1 (Conv2D)            (None, 24, 24, 128)       204928    
_________________________________________________________________
batch_normalization_1 (Batch (None, 24, 24, 128)       512       
_________________________________________________________________
activation_1 (Activation)    (None, 24, 24, 128)       0         
_________________________________________________________________
max_pooling2d_1 (MaxPooling2 (None, 12, 12, 128)       0         
_________________________________________________________________
dropout_1 (Dropout)          (None, 12, 12, 128)       0         
_________________________________________________________________
conv2d_2 (Conv2D)            (None, 12, 12, 512)       590336    
_________________________________________________________________
batch_normalization_2 (Batch (None, 12, 12, 512)       2048      
_________________________________________________________________
activation_2 (Activation)    (None, 12, 12, 512)       0         
_________________________________________________________________
max_pooling2d_2 (MaxPooling2 (None, 6, 6, 512)         0         
_________________________________________________________________
dropout_2 (Dropout)          (None, 6, 6, 512)         0         
_________________________________________________________________
```



```
_________________________________________________________________
conv2d_3 (Conv2D)            (None, 6, 6, 512)         2359808   
_________________________________________________________________
batch_normalization_3 (Batch (None, 6, 6, 512)         2048      
_________________________________________________________________
activation_3 (Activation)    (None, 6, 6, 512)         0         
_________________________________________________________________
max_pooling2d_3 (MaxPooling2 (None, 3, 3, 512)         0         
_________________________________________________________________
dropout_3 (Dropout)          (None, 3, 3, 512)         0         
_________________________________________________________________
flatten (Flatten)            (None, 4608)              0         
_________________________________________________________________
dense (Dense)                (None, 256)               1179904   
_________________________________________________________________
batch_normalization_4 (Batch (None, 256)               1024      
_________________________________________________________________
activation_4 (Activation)    (None, 256)               0         
_________________________________________________________________
dropout_4 (Dropout)          (None, 256)               0         
_________________________________________________________________
dense_1 (Dense)              (None, 512)               131584    
_________________________________________________________________
batch_normalization_5 (Batch (None, 512)               2048      
_________________________________________________________________
activation_5 (Activation)    (None, 512)               0         
_________________________________________________________________
dropout_5 (Dropout)          (None, 512)               0         
_________________________________________________________________
dense_2 (Dense)              (None, 7)                 3591      
=================================================================
Total params: 4,478,727
Trainable params: 4,474,759
Non-trainable params: 3,968
_________________________________________________________________
```

Fitting the Model with Training and Validation Data

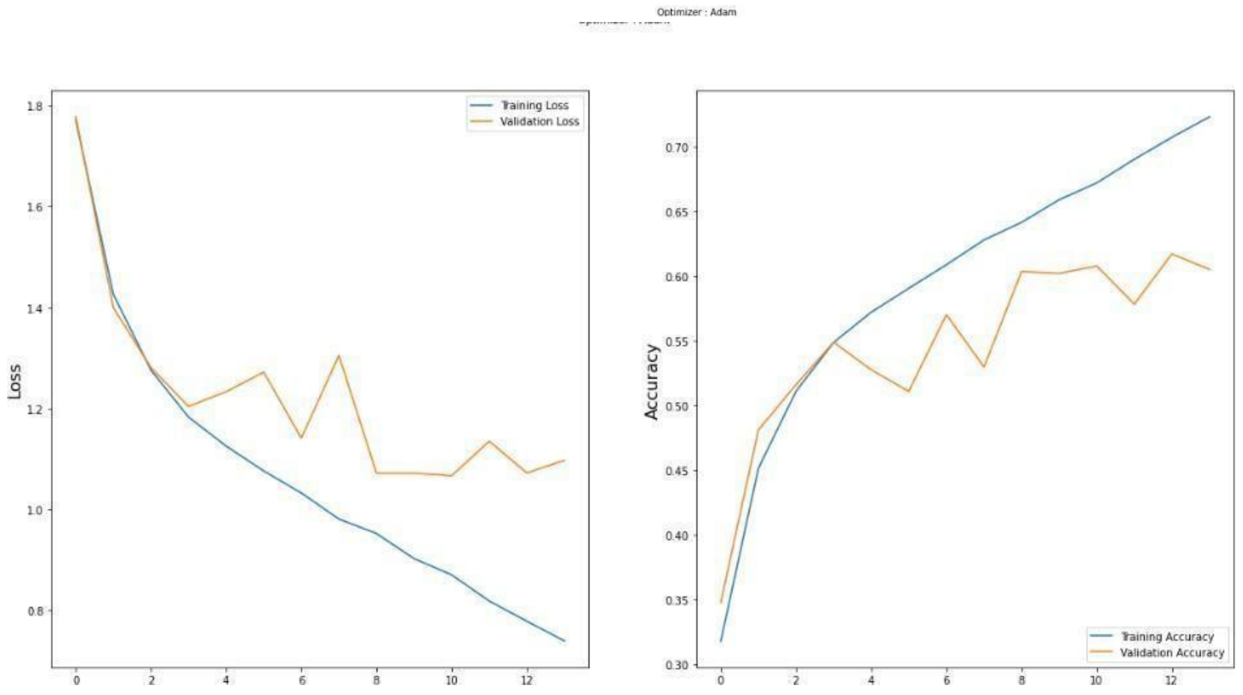
In [8]:
```
from keras.optimizers import RMSprop,SGD,Adam
from keras.callbacks import ModelCheckpoint, EarlyStopping, ReduceLROnPlateau

checkpoint = ModelCheckpoint("./model.h5", monitor='val_acc', verbose=1, save_best_only=True, mode='max')

early_stopping = EarlyStopping(monitor='val_loss',
                          min_delta=0,
                          patience=3,
                          verbose=1,
                          restore_best_weights=True
                          )

reduce_learningrate = ReduceLROnPlateau(monitor='val_loss',
                              factor=0.2,
                              patience=3,
                              verbose=1,
                              min_delta=0.0001)

callbacks_list = [early_stopping,checkpoint,reduce_learningrate]

epochs = 48

model.compile(loss='categorical_crossentropy',
              optimizer = Adam(lr=0.001),
              metrics=['accuracy'])
```

In [9]:
```
history = model.fit_generator(generator=train_set,
                                steps_per_epoch=train_set.n//train_set.batch_size,
                                epochs=epochs,
                                validation_data = test_set,
                                validation_steps = test_set.n//test_set.batch_size,
                                callbacks=callbacks_list
                                )
```

```
Epoch 1/48
225/225 [==============================] - 180s 799ms/step - loss: 1.7711 - accuracy: 0.3183 - val_loss: 1.7775 - val_accura
cy: 0.3482
Epoch 2/48
225/225 [==============================] - 22s 99ms/step - loss: 1.4261 - accuracy: 0.4517 - val_loss: 1.3999 - val_accurac
y: 0.4814
Epoch 3/48
225/225 [==============================] - 22s 97ms/step - loss: 1.2760 - accuracy: 0.5113 - val_loss: 1.2810 - val_accurac
y: 0.5163
Epoch 4/48
225/225 [==============================] - 22s 100ms/step - loss: 1.1831 - accuracy: 0.5490 - val_loss: 1.2041 - val_accurac
y: 0.5491
Epoch 5/48
225/225 [==============================] - 23s 100ms/step - loss: 1.1257 - accuracy: 0.5724 - val_loss: 1.2334 - val_accurac
y: 0.5278
Epoch 6/48
225/225 [==============================] - 22s 98ms/step - loss: 1.0765 - accuracy: 0.5909 - val_loss: 1.2720 - val_accurac
y: 0.5112
Epoch 7/48
225/225 [==============================] - 23s 101ms/step - loss: 1.0328 - accuracy: 0.6091 - val_loss: 1.1415 - val_accurac
y: 0.5705
Epoch 8/48
225/225 [==============================] - 23s 103ms/step - loss: 0.9813 - accuracy: 0.6282 - val_loss: 1.3052 - val_accurac
y: 0.5300
Epoch 9/48
225/225 [==============================] - 23s 103ms/step - loss: 0.9526 - accuracy: 0.6418 - val_loss: 1.0722 - val_accurac
y: 0.6038
Epoch 10/48
225/225 [==============================] - 27s 120ms/step - loss: 0.9028 - accuracy: 0.6593 - val_loss: 1.0720 - val_accurac
y: 0.6024
Epoch 11/48
225/225 [==============================] - 23s 104ms/step - loss: 0.8707 - accuracy: 0.6724 - val_loss: 1.0670 - val_accurac
y: 0.6081
Epoch 12/48
225/225 [==============================] - 22s 98ms/step - loss: 0.8188 - accuracy: 0.6906 - val_loss: 1.1353 - val_accurac
y: 0.5786
Epoch 13/48
225/225 [==============================] - 23s 104ms/step - loss: 0.7788 - accuracy: 0.7076 - val_loss: 1.0726 - val_accurac
y: 0.6175
Epoch 14/48
225/225 [==============================] - ETA: 0s - loss: 0.7399 - accuracy: 0.7234Restoring model weights from the end of
the best epoch.

Epoch 00014: ReduceLROnPlateau reducing learning rate to 0.00020000000949949026.
225/225 [==============================] - 23s 102ms/step - loss: 0.7399 - accuracy: 0.7234 - val_loss: 1.0975 - val_accurac
y: 0.6054
Epoch 00014: early stopping
```

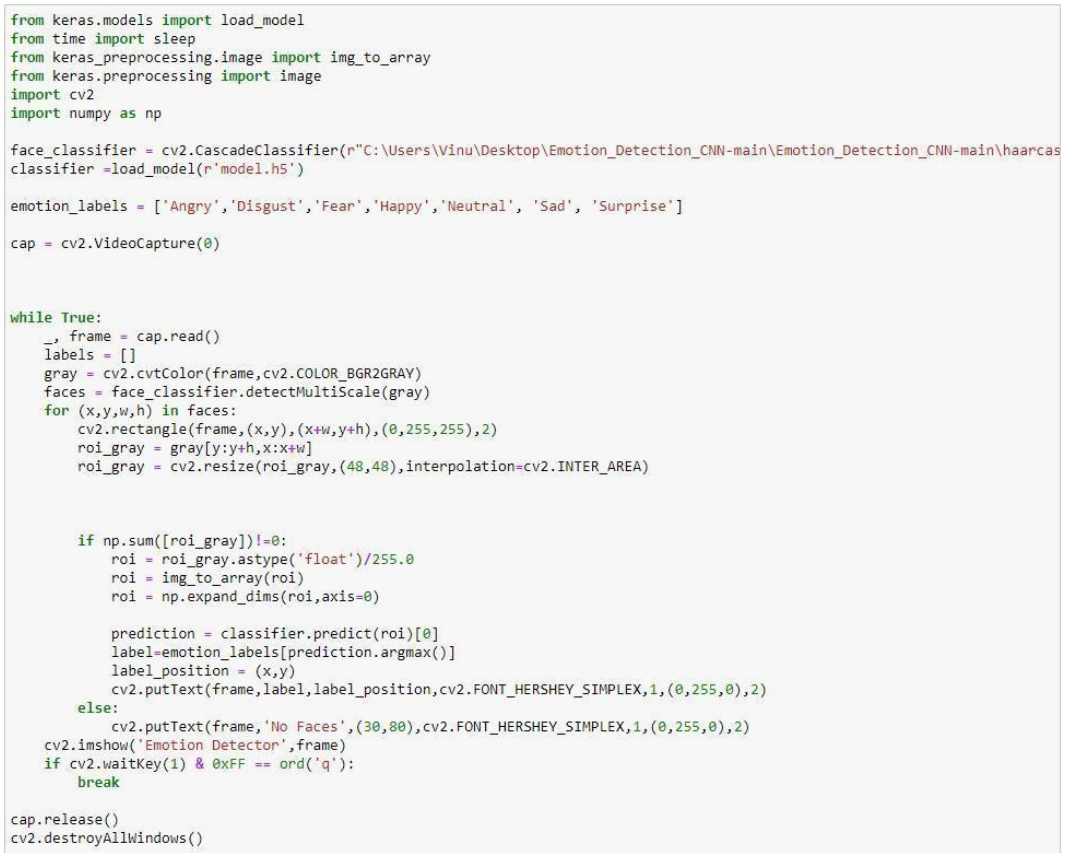
Plotting Accuracy & Loss

In [10]:
```
plt.style.use('dark_background')

plt.figure(figsize=(20,10))
plt.subplot(1, 2, 1)
plt.suptitle('Optimizer : Adam', fontsize=10)
plt.ylabel('Loss', fontsize=16)
plt.plot(history.history['loss'], label='Training Loss')
plt.plot(history.history['val_loss'], label='Validation Loss')
plt.legend(loc='upper right')

plt.subplot(1, 2, 2)
plt.ylabel('Accuracy', fontsize=16)
plt.plot(history.history['accuracy'], label='Training Accuracy')
plt.plot(history.history['val_accuracy'], label='Validation Accuracy')
plt.legend(loc='lower right')
plt.show()
```



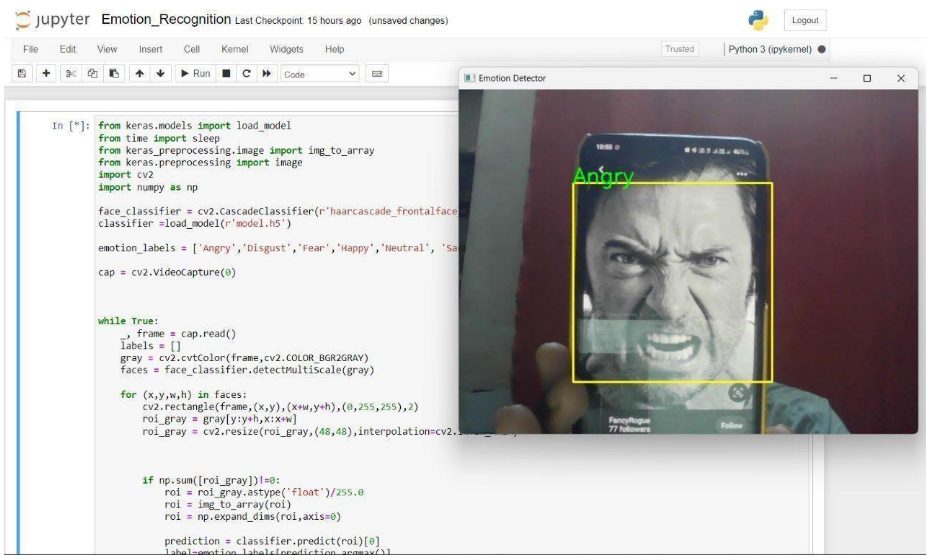
In [3]:
```
!pip install opencv-python
```

```
Requirement already satisfied: opencv-python in c:\users\vinu\anaconda3\lib\site-packages (4.6.0.66)
Requirement already satisfied: numpy>=1.14.5 in c:\users\vinu\anaconda3\lib\site-packages (from opencv-python) (1.21.5)
```

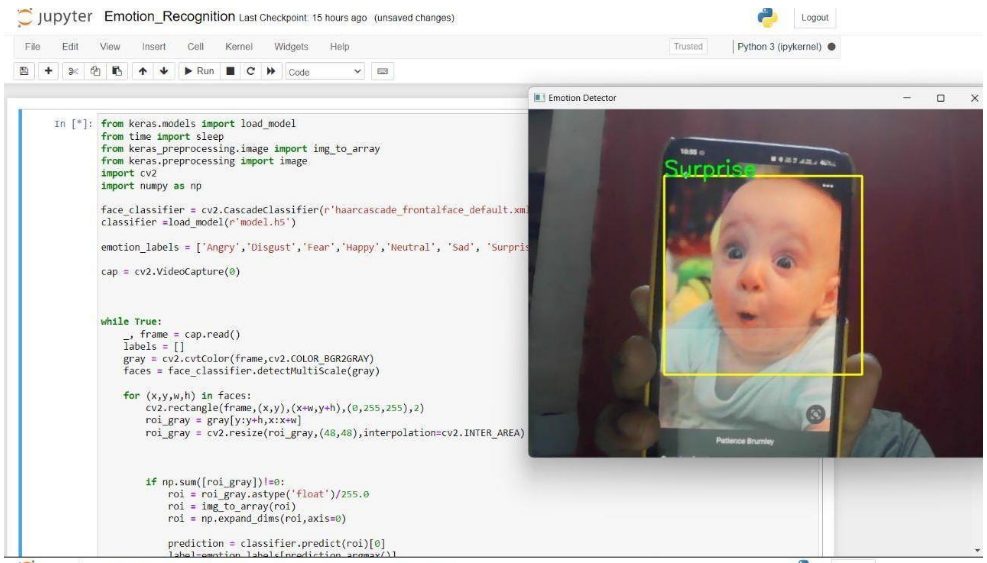


```python
from keras.models import load_model
from time import sleep
from keras_preprocessing.image import img_to_array
from keras.preprocessing import image
import cv2
import numpy as np

face_classifier = cv2.CascadeClassifier(r"C:\Users\Vinu\Desktop\Emotion_Detection_CNN-main\Emotion_Detection_CNN-main\haarcas
classifier =load_model(r'model.h5')

emotion_labels = ['Angry','Disgust','Fear','Happy','Neutral', 'Sad', 'Surprise']

cap = cv2.VideoCapture(0)



while True:
    _, frame = cap.read()
    labels = []
    gray = cv2.cvtColor(frame,cv2.COLOR_BGR2GRAY)
    faces = face_classifier.detectMultiScale(gray)
    for (x,y,w,h) in faces:
        cv2.rectangle(frame,(x,y),(x+w,y+h),(0,255,255),2)
        roi_gray = gray[y:y+h,x:x+w]
        roi_gray = cv2.resize(roi_gray,(48,48),interpolation=cv2.INTER_AREA)



        if np.sum([roi_gray])!=0:
            roi = roi_gray.astype('float')/255.0
            roi = img_to_array(roi)
            roi = np.expand_dims(roi,axis=0)

            prediction = classifier.predict(roi)[0]
            label=emotion_labels[prediction.argmax()]
            label_position = (x,y)
            cv2.putText(frame,label,label_position,cv2.FONT_HERSHEY_SIMPLEX,1,(0,255,0),2)
        else:
            cv2.putText(frame,'No Faces',(30,80),cv2.FONT_HERSHEY_SIMPLEX,1,(0,255,0),2)
    cv2.imshow('Emotion Detector',frame)
    if cv2.waitKey(1) & 0xFF == ord('q'):
        break

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

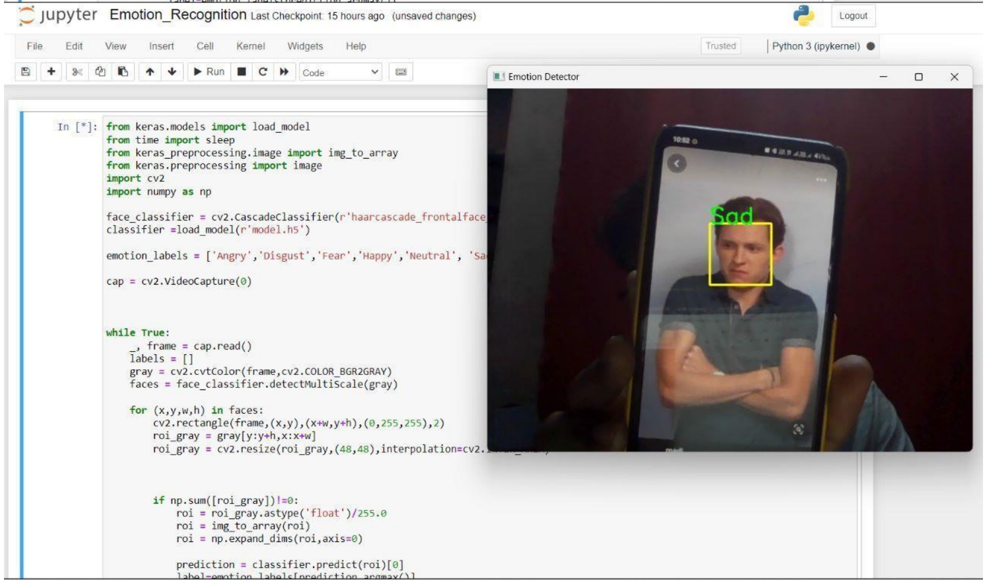
1.12 Real time Recognition



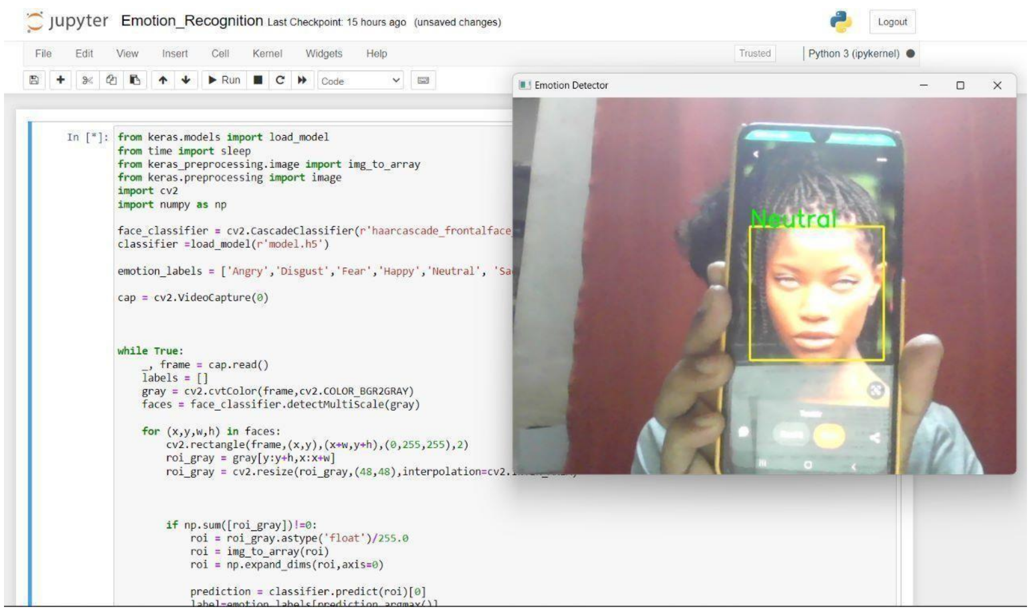
```python
In [*]: from keras.models import load_model
        from time import sleep
        from keras_preprocessing.image import img_to_array
        from keras.preprocessing import image
        import cv2
        import numpy as np

        face_classifier = cv2.CascadeClassifier(r'haarcascade_frontalface
        classifier =load_model(r'model.h5')

        emotion_labels = ['Angry','Disgust','Fear','Happy','Neutral', 'Sa

        cap = cv2.VideoCapture(0)



        while True:
            _, frame = cap.read()
            labels = []
            gray = cv2.cvtColor(frame,cv2.COLOR_BGR2GRAY)
            faces = face_classifier.detectMultiScale(gray)

            for (x,y,w,h) in faces:
                cv2.rectangle(frame,(x,y),(x+w,y+h),(0,255,255),2)
                roi_gray = gray[y:y+h,x:x+w]
                roi_gray = cv2.resize(roi_gray,(48,48),interpolation=cv2.



                if np.sum([roi_gray])!=0:
                    roi = roi_gray.astype('float')/255.0
                    roi = img_to_array(roi)
                    roi = np.expand_dims(roi,axis=0)

                    prediction = classifier.predict(roi)[0]
```


jupyter Emotion_Recognition Last Checkpoint: 15 hours ago (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code
Emotion Detector
Surprise
Patience Brumley

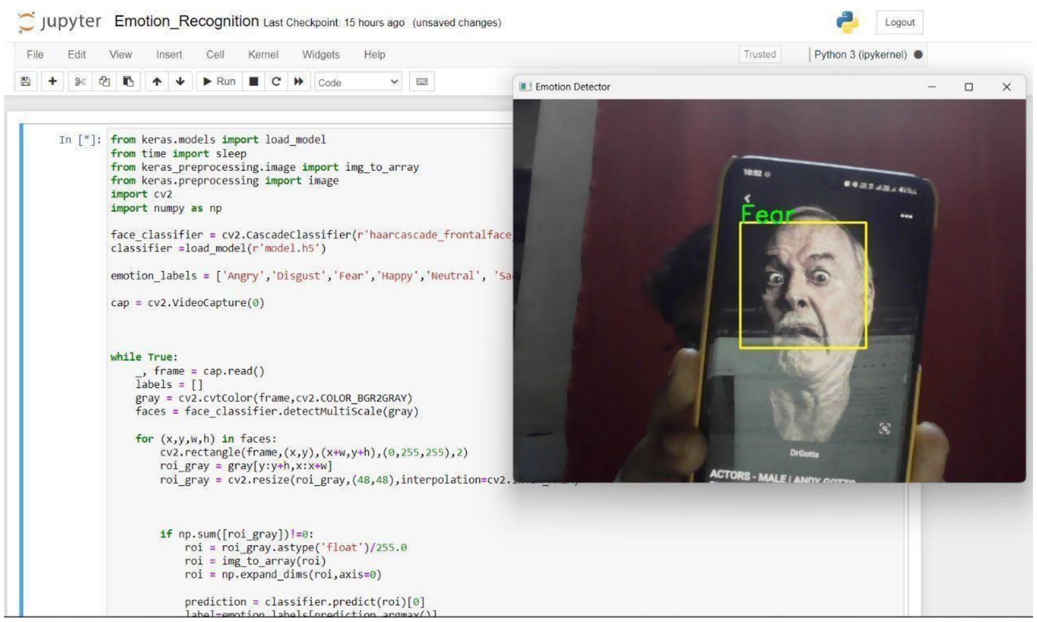
```
In [*]: from keras.models import load_model
        from time import sleep
        from keras_preprocessing.image import img_to_array
        from keras.preprocessing import image
        import cv2
        import numpy as np

        face_classifier = cv2.CascadeClassifier(r'haarcascade_frontalface_default.xml
        classifier =load_model(r'model.h5')

        emotion_labels = ['Angry','Disgust','Fear','Happy','Neutral', 'Sad', 'Surpris

        cap = cv2.VideoCapture(0)



        while True:
            _, frame = cap.read()
            labels = []
            gray = cv2.cvtColor(frame,cv2.COLOR_BGR2GRAY)
            faces = face_classifier.detectMultiScale(gray)

            for (x,y,w,h) in faces:
                cv2.rectangle(frame,(x,y),(x+w,y+h),(0,255,255),2)
                roi_gray = gray[y:y+h,x:x+w]
                roi_gray = cv2.resize(roi_gray,(48,48),interpolation=cv2.INTER_AREA)



                if np.sum([roi_gray])!=0:
                    roi = roi_gray.astype('float')/255.0
                    roi = img_to_array(roi)
                    roi = np.expand_dims(roi,axis=0)

                    prediction = classifier.predict(roi)[0]
                    label=emotion_labels[prediction.argmax()]
```


jupyter Emotion_Recognition Last Checkpoint: 15 hours ago (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code
Emotion Detector
Sad

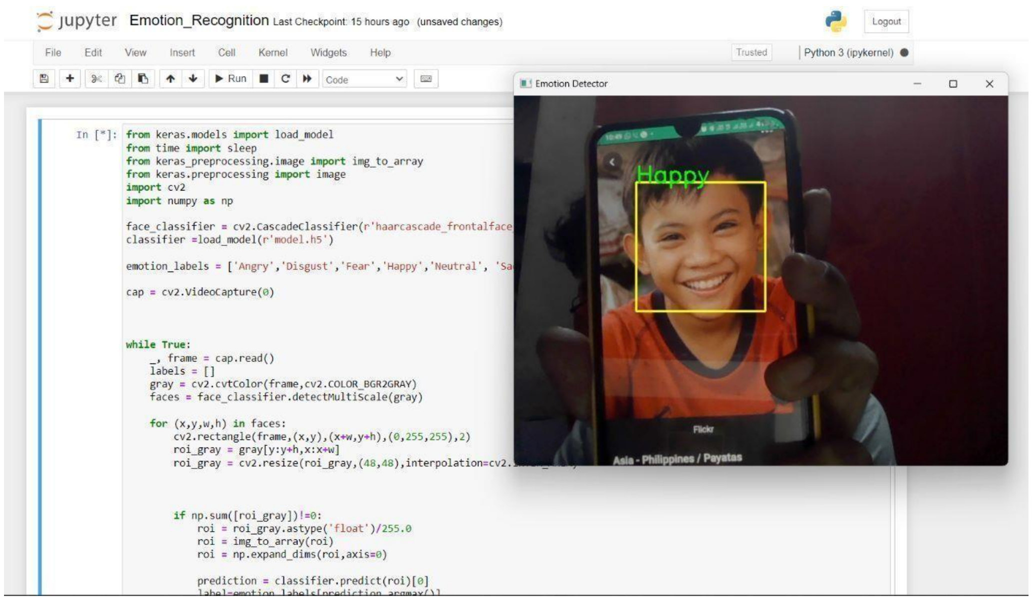
```
In [*]: from keras.models import load_model
        from time import sleep
        from keras_preprocessing.image import img_to_array
        from keras.preprocessing import image
        import cv2
        import numpy as np

        face_classifier = cv2.CascadeClassifier(r'haarcascade_frontalface
        classifier =load_model(r'model.h5')

        emotion_labels = ['Angry','Disgust','Fear','Happy','Neutral', 'Sa

        cap = cv2.VideoCapture(0)



        while True:
            _, frame = cap.read()
            labels = []
            gray = cv2.cvtColor(frame,cv2.COLOR_BGR2GRAY)
            faces = face_classifier.detectMultiScale(gray)

            for (x,y,w,h) in faces:
                cv2.rectangle(frame,(x,y),(x+w,y+h),(0,255,255),2)
                roi_gray = gray[y:y+h,x:x+w]
                roi_gray = cv2.resize(roi_gray,(48,48),interpolation=cv2.



                if np.sum([roi_gray])!=0:
                    roi = roi_gray.astype('float')/255.0
                    roi = img_to_array(roi)
                    roi = np.expand_dims(roi,axis=0)

                    prediction = classifier.predict(roi)[0]
                    label=emotion_labels[prediction.argmax()]
```


jupyter Emotion_Recognition Last Checkpoint: 15 hours ago (unsaved changes)
Logout
File Edit View Insert Cell Kernel Widgets Help
Trusted
Python 3 (ipykernel)
Run
Code
Emotion Detector
Neutral

```
In [*]: from keras.models import load_model
        from time import sleep
        from keras_preprocessing.image import img_to_array
        from keras.preprocessing import image
        import cv2
        import numpy as np

        face_classifier = cv2.CascadeClassifier(r'haarcascade_frontalface
        classifier =load_model(r'model.h5')

        emotion_labels = ['Angry','Disgust','Fear','Happy','Neutral', 'Sa

        cap = cv2.VideoCapture(0)



        while True:
            _, frame = cap.read()
            labels = []
            gray = cv2.cvtColor(frame,cv2.COLOR_BGR2GRAY)
            faces = face_classifier.detectMultiScale(gray)

            for (x,y,w,h) in faces:
                cv2.rectangle(frame,(x,y),(x+w,y+h),(0,255,255),2)
                roi_gray = gray[y:y+h,x:x+w]
                roi_gray = cv2.resize(roi_gray,(48,48),interpolation=cv2.



                if np.sum([roi_gray])!=0:
                    roi = roi_gray.astype('float')/255.0
                    roi = img_to_array(roi)
                    roi = np.expand_dims(roi,axis=0)

                    prediction = classifier.predict(roi)[0]
                    label=emotion_labels[prediction.argmax()]
```



```
In [*]: from keras.models import load_model
        from time import sleep
        from keras_preprocessing.image import img_to_array
        from keras.preprocessing import image
        import cv2
        import numpy as np

        face_classifier = cv2.CascadeClassifier(r'haarcascade_frontalface
        classifier =load_model(r'model.h5')

        emotion_labels = ['Angry','Disgust','Fear','Happy','Neutral', 'Sa

        cap = cv2.VideoCapture(0)



        while True:
            _, frame = cap.read()
            labels = []
            gray = cv2.cvtColor(frame,cv2.COLOR_BGR2GRAY)
            faces = face_classifier.detectMultiScale(gray)

            for (x,y,w,h) in faces:
                cv2.rectangle(frame,(x,y),(x+w,y+h),(0,255,255),2)
                roi_gray = gray[y:y+h,x:x+w]
                roi_gray = cv2.resize(roi_gray,(48,48),interpolation=cv2.



                if np.sum([roi_gray])!=0:
                    roi = roi_gray.astype('float')/255.0
                    roi = img_to_array(roi)
                    roi = np.expand_dims(roi,axis=0)

                    prediction = classifier.predict(roi)[0]
                    label=emotion_labels[prediction.argmax()]
```



```
In [*]: from keras.models import load_model
        from time import sleep
        from keras_preprocessing.image import img_to_array
        from keras.preprocessing import image
        import cv2
        import numpy as np

        face_classifier = cv2.CascadeClassifier(r'haarcascade_frontalface
        classifier =load_model(r'model.h5')

        emotion_labels = ['Angry','Disgust','Fear','Happy','Neutral', 'Sa

        cap = cv2.VideoCapture(0)



        while True:
            _, frame = cap.read()
            labels = []
            gray = cv2.cvtColor(frame,cv2.COLOR_BGR2GRAY)
            faces = face_classifier.detectMultiScale(gray)

            for (x,y,w,h) in faces:
                cv2.rectangle(frame,(x,y),(x+w,y+h),(0,255,255),2)
                roi_gray = gray[y:y+h,x:x+w]
                roi_gray = cv2.resize(roi_gray,(48,48),interpolation=cv2.



                if np.sum([roi_gray])!=0:
                    roi = roi_gray.astype('float')/255.0
                    roi = img_to_array(roi)
                    roi = np.expand_dims(roi,axis=0)

                    prediction = classifier.predict(roi)[0]
                    label=emotion_labels[prediction.argmax()]
```

CONCLUSION

Eight real valued and seven binary parameters were successfully extracted from different subjects for seven different expressions (neutral, angry, disgust, fear, sad, happy and surprised). An integrated committee neural network system was developed incorporating a generalized neural network committee and a specialized neural network committee.

Several generalized neural networks (with different initial weights, structure, etc) were trained to classify the image into seven different expressions (neutral, angry, disgust, fear, sad, happy and surprised). Similarly, several specialized neural networks were trained to classify the image into four different expressions (angry, disgust, fear and sad).

REFERENCES

https://www.geeksforgeeks.org/convolutional-neural-network-cnn-in-mach ine-learning/

https://paperswithcode.com/task/facial-expression-recognition

https://www.frontiersin.org/articles/10.3389/fpsyg.2021.759485/full

https://towardsdatascience.com/introduction-to-convolutional-neural-netw
ork-cnn-de73f69c5b83

https://www.kaggle.com/datasets/jonathanoheix/face-expression-recogniti on-dataset

https://www.kaggle.com/code/gauravsharma99/facial-emotion-recognition