

IDENTIFYING HUMAN EMOTION THROUGH AI

MINI PROJECT REPORT
AI MTE 4059

Submitted by

Isha Harish(Reg. No. 200929098)



MANIPAL INSTITUTE OF TECHNOLOGY
MANIPAL
(A constituent unit of MAHE, Manipal)

DEPARTMENT OF MECHATRONICS

APRIL 2023

ABSTRACT

The difficult task of recognising human emotions in images has generated a lot of research in the area of artificial intelligence. In many disciplines, including psychology, marketing, and social media analysis, emotion identification is crucial. The goal of this mini project is to create an AI-based model that can properly recognise human emotions in images. This project's main goal is to show how AI may be used to recognise human emotions and in real-world situations.

For this study, a dataset of facial expressions that depict various emotions, including "Angry", "Disgust", "Fear", "Happy", "Neutral", "Sad", "Surprise" was gathered. The model was then trained using Convolutional Neural Networks (CNN) with multiple layer such as dense layer, hidden layer, dropout layer, output layer, etc. We used optimisers such as Adam and RMSprop. The dataset we used was available in Kaggle therefore we decided to use the Kaggle kernel to train and make the model so that we won't have to download such a large dataset. Then this model was integrated with opencv using jupyter notebook available in anaconda.

The software's/packages we used are keras, tensorflow, python3, OpenCV and anaconda .

Contents

	Page No
Abstract	i
List of Figures	ii
List of Tables	iii
1 INTRODUCTION	1
2 LITERATURE REVIEW AND THEORETICAL BACKGROUND	
3 PROBLEM DEFINITION AND OBJECTIVES	
4 METHODOLOGY	
5 RESULTS AND CONCLUSIONS	
5.1 Results and Discussion	
5.2 Conclusions	
REFERENCES	

Chapter 1: Introduction

In recent years, artificial intelligence (AI) systems that are able to identify and interpret human emotions from photos have garnered a lot of attention. "Emotion recognition" or "affective computing" is a topic of study that has the potential to transform several industries, including marketing and healthcare.

In essence, emotion identification is the act of using computer algorithms to scan facial expressions, body language, and other visual cues to recognize and classify different emotions, such as happiness, sorrow, rage, fear, and surprise. The ability to identify and understand these emotions is typically innate in humans, but using AI to automate this process has the potential to completely transform fields like psychology, education, and customer service.

The applications of artificial intelligence (AI) for emotion recognition are wide-ranging and developing quickly nowadays. For instance, while some businesses are investigating the possibilities of emotion recognition technology in the realms of mental health and wellbeing, others are utilizing it to measure consumer responses to commercials. Emotion recognition technology can be utilized in the healthcare sector to enhance patient care by enabling medical professionals to promptly recognize and assist patients who may be experiencing emotional discomfort.

It is clear why creating AI systems capable of identifying emotions is crucial. Emotions have a key role in shaping the human experience, influencing everything from our interpersonal interactions to our decision-making. By automating processes, we can gain a deeper understanding of human behavior as well as improve our capacity for connection and communication.

Even though the science of AI emotion recognition is still in its early stages, there are many potential applications. But as technology advances and our comprehension of human emotions expands, it is clear that the development of effective emotion identification systems will be crucial to defining the nature of human-technology interaction in the future.

Chapter 3: Problem Definition and Objectives

The current mini-project's focus on applying artificial intelligence to detect human emotions in images is prompted by a number of issues in this field. Among the significant issues are:

Human error and bias: When perceiving emotions, humans are prone to inaccuracy and frequently influenced by their own prejudices. This may result in unreliable findings and inconsistent emotional readings.

Efficiency: It might be challenging to scale up to larger datasets or real-time applications since manual emotion interpretation can be time- and resource-intensive.

Lack of standardisation: Different academics and professionals classify emotions using various methodologies and metrics because there is currently no generally recognised standard for emotion recognition.

Data availability and quality: A significant amount of high-quality data is needed to teach AI systems to recognise emotions. However, gathering such information can be difficult because emotions are frequently irrational and challenging to measure.

Privacy issues and the exploitation of such technology for surveillance or other immoral objectives have been brought up by the usage of emotion recognition technologies.

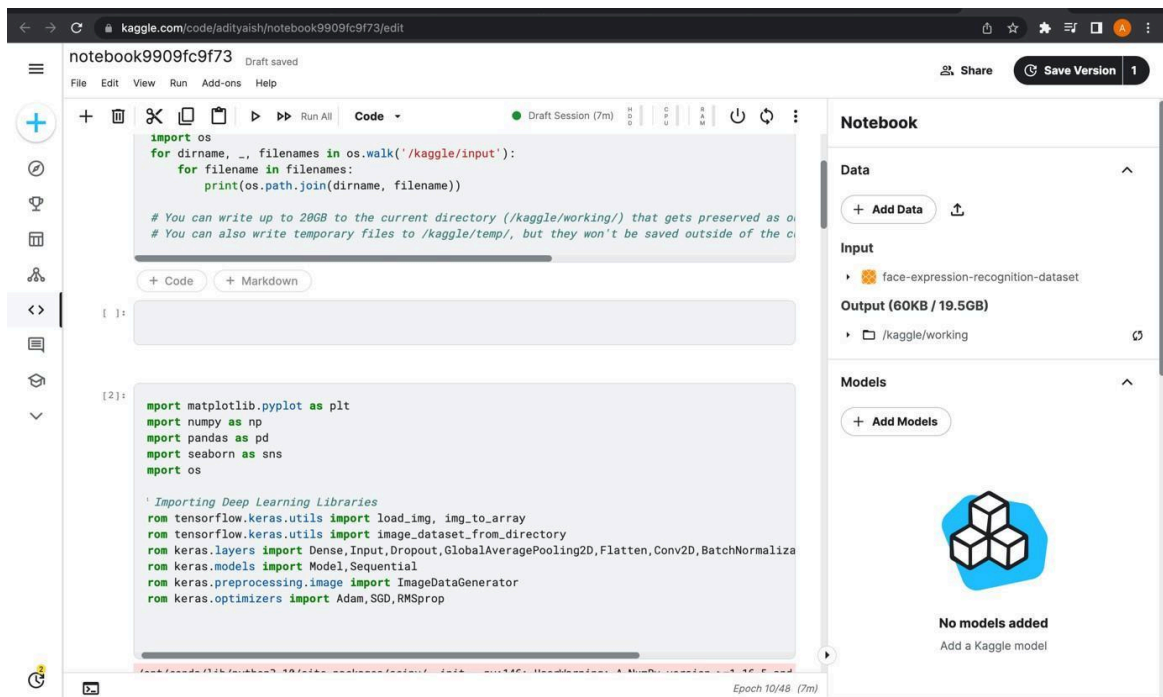
We can solve many of these problems and open up new opportunities in industries like mental health, education, and marketing by creating AI systems that can reliably and effectively recognise human emotions. In addition, by addressing the ethical issues raised by this technology, we can make sure that it is created and used in a way that is ethical and useful to society.

Objective:

- To understand the use of AI to classify human emotions
- To explore the current neural networks , optimisers and image classifiers available today

- To understand the concepts of : Image preprocessing ,feature extraction,, feature classifications and to choose the appropriate model
- To identify the future scope and potential research opportunities in the topic.
- To evaluate the potential applications of this mini project at an industrial level.

Chapter 4: Methodology

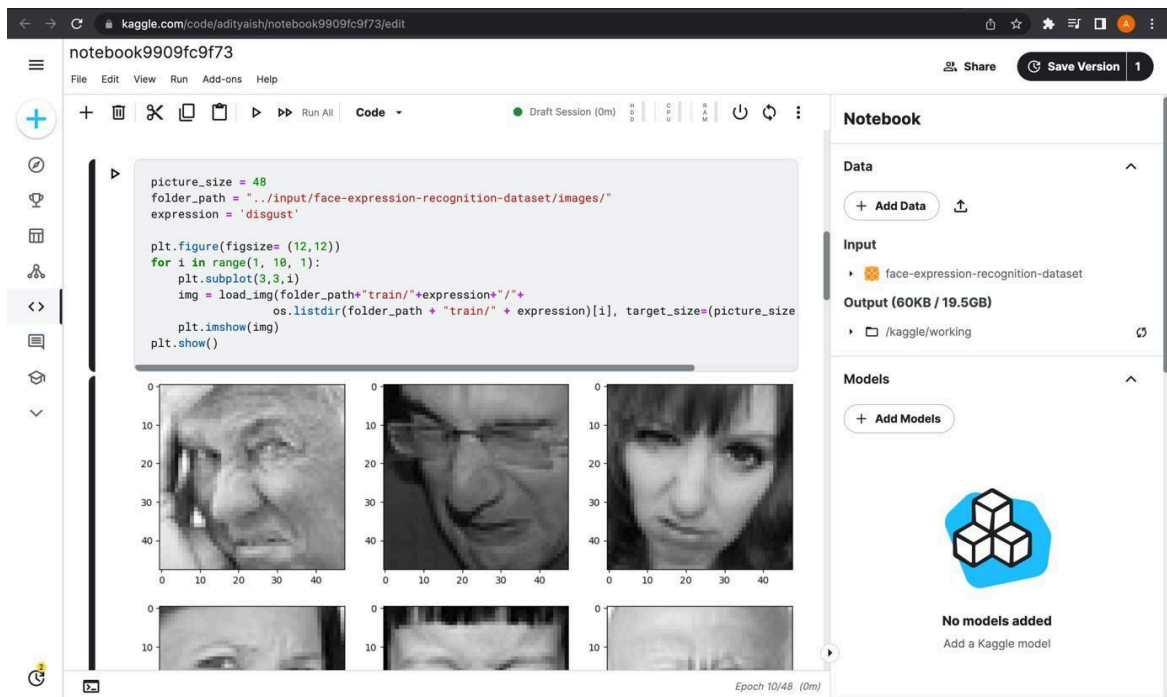


To train and build the model we used a Kaggle kernel .The seven emotions we focused on were happy,anger,surprised,disgust,sad,fear and neutral. The dataset was obtained from Jonathan Oheix in Kaggle which has multiple directories containing thousands of image files .This has both training and validation directories. We used the same kernel to avoid downloading such a huge dataset.

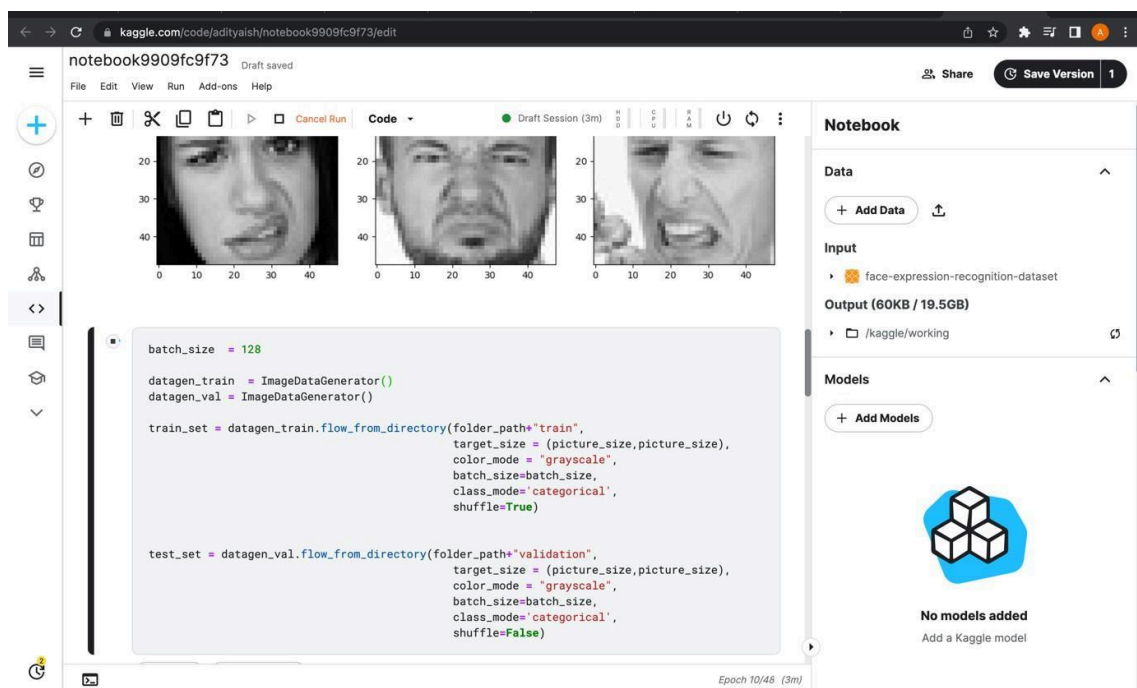
Libraries like matplotlib,numpy were used for the purpose of visualization; load image , image to array are deeplearning libraries. Model to array was used as the model can only decipher images as arrays

The layers from keras included input, dropout, hidden, output, dense ,flatten,conv2D,etc.

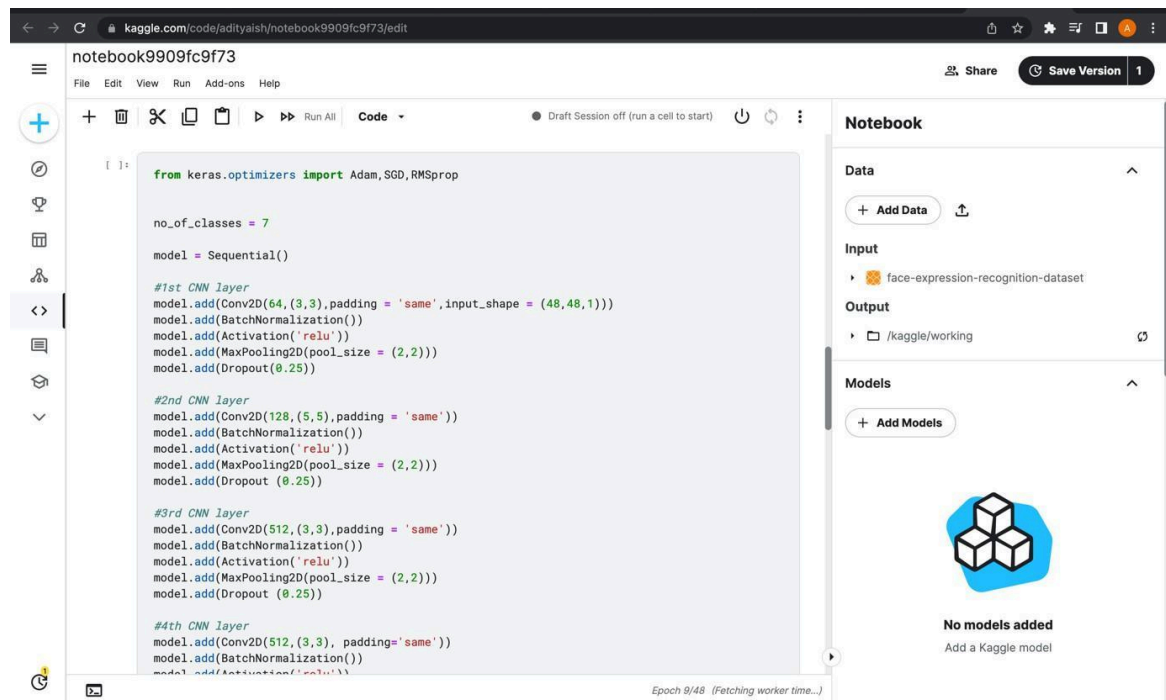
The optimisers used includes adam,SGD and RMSprop.



The images were then standardized to a certain size which remains the same throughout. To check the images we ran a kernel that plotted images from a respected category. In the above image the plot for disgust is shown. This was done to check the standardization.



To make the training and validation dataset batch size was defined. Batch size defines how many training examples should be taken in one iteration. Two variables train and test dataset was defined , they contain the data from the directories



The screenshot shows a Kaggle notebook interface. The main code cell contains the following Python code for building a Keras model:

```
[ ]: 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'))
```

The right sidebar shows the notebook's metadata: Data (Add Data), Input (face-expression-recognition-dataset), Output (/kaggle/working), and Models (Add Models). A message at the bottom of the Models section says "No models added" and "Add a Kaggle model". The bottom status bar indicates "Epoch 9/48 (Fetching worker time...)"

Model Building:

7 classes were defined for seven possible outcomes. The model we chose is sequential.

In the first cnn layer 64 filters with 3x3 kernel size was chosen with the same padding .A relu activation function was used and the poolsize and dropout rate was defined as shown in image .

The rest of the layers were defined in a similar manner as shown in the image.

Adam optimizer was used with a defined learning rate


```

[14]: 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, m

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 = 5

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

```

The screenshot shows a Kaggle notebook interface. The code cell [14] defines three Keras callbacks: `ModelCheckpoint` to save the best model based on validation accuracy, `EarlyStopping` to stop training when the validation loss stops improving, and `ReduceLROnPlateau` to reduce the learning rate when the validation loss plateaus. These callbacks are combined into a `callbacks_list`. The model is then compiled with categorical crossentropy loss, Adam optimizer, and accuracy metric. The notebook sidebar on the right shows the 'Data' section with the 'face-expression-recognition-dataset' and the 'Models' section with an 'Add Models' button.

To fit the model with training and validation data early stopping ,checkpoints are used .A call back list is used to implement these functions on the model.

The no. of epochs we chose was only 5 due to time constraints but ideally 48 epochs should have been taken for better accuracy.

```

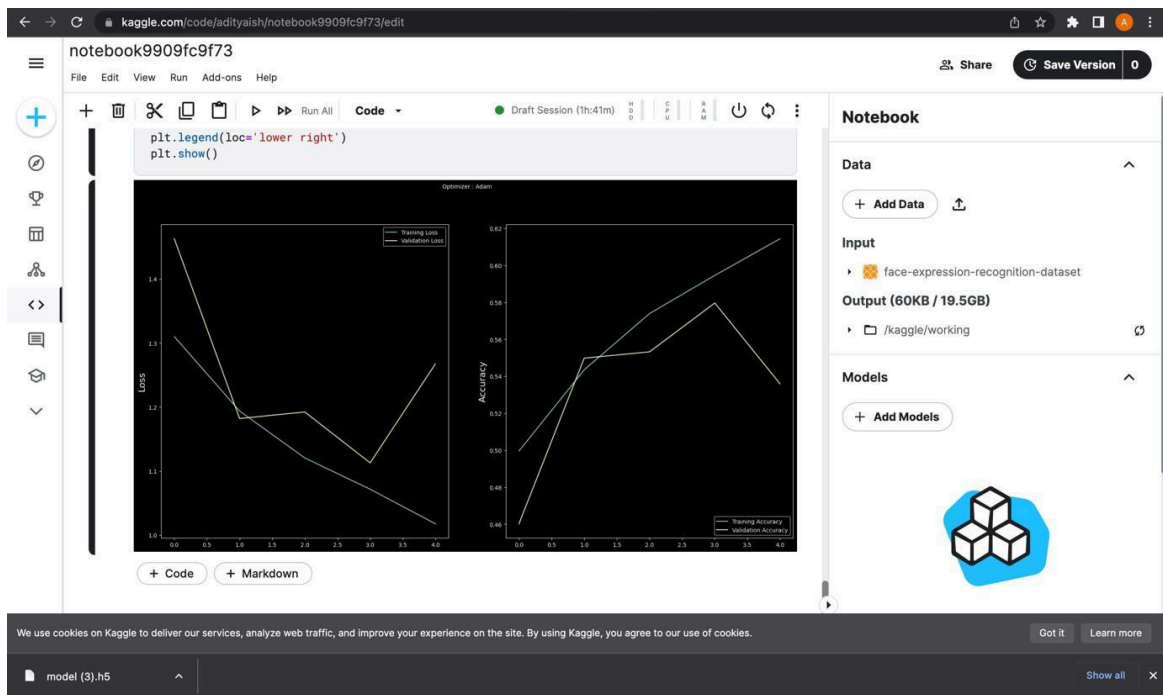
[15]: history = model.fit_generator(generator=train_set,
                                steps_per_epoch=train_set.n//train_set.batch_size,
                                epochs=epochs,
                                validation_data = test_set,
                                validation_steps = test_set.n//test_set.batch_size,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

```

The screenshot shows the execution of the `model.fit_generator` method. The output displays the training history for 5 epochs. A warning message indicates that `Model.fit_generator` is deprecated. The training progress shows the number of steps, loss, accuracy, and validation loss for each epoch. The final accuracy is 0.6145.

Epoch	Steps	Loss	Accuracy	Val Loss
Epoch 1/5	778s 3s/step	1.3101	0.4996	1.4630
Epoch 2/5	772s 3s/step	1.1937	0.5435	1.1819
Epoch 3/5	771s 3s/step	1.1203	0.5740	1.1922
Epoch 4/5	775s 3s/step	1.0714	0.5946	1.1126
Epoch 5/5	767s 3s/step	1.0174	0.6145	1.2674

The notebook sidebar on the right shows the 'Data' section with the 'face-expression-recognition-dataset' and the 'Models' section with an 'Add Models' button.



The loss and accuracy of the datasets were plotted using python plotting libraries. As seen in the image it is clear that the loss decreases and accuracy increases as time elapses

The image shows a Jupyter notebook interface for a notebook titled 'Untitled5'. The code cell contains the following Python code:

```
In [7]: from keras.models import load_model
from time import sleep
from tensorflow.keras.utils import img_to_array
from keras.preprocessing import image
import cv2
import numpy as np

face_classifier = cv2.CascadeClassifier(r'/Users/adityagirish/Desktop/ai/haarcascade-frontalface-default.xml')
classifier = load_model(r'/Users/adityagirish/Desktop/ai/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)

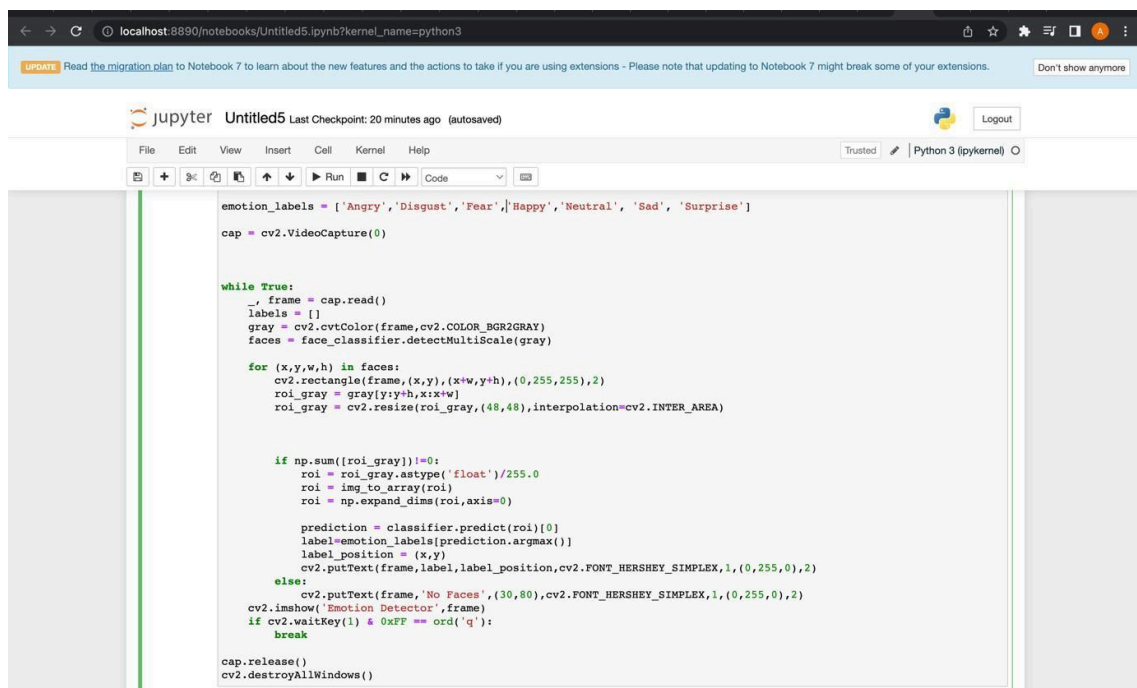
    for (x,y,w,h) in faces:
        cv2.rectangle(frame, (x,y), (x+w,y+h), (0,255,255), 2)
        roi_gray = gray[y:y+h, x:x+w]
        roi_gray = cv2.resize(roi_gray, (48,48), interpolation=cv2.INTER_AREA)

        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]
```

In the main program (which was built in jupyter notebook) the raw paths of all the above files containing the models was added using load model function .The emotion labels were then defined which would be displayed on the webcam by opencv .The while loop is used to define the webcam parameters. The for loop is used so that multiple faces can be detected and the rectangle is drawn around them.roi function (region of interest) was used to place the focus only on the face and ignore the background.

Using classifier the model outcome was predicted



The screenshot displays a Jupyter Notebook environment in a web browser. The browser's address bar shows the URL `localhost:8890/notebooks/Untitled5.ipynb?kernel_name=python3`. A notification bar at the top indicates an update to Notebook 7. The Jupyter interface includes a menu bar (File, Edit, View, Insert, Cell, Kernel, Help) and a toolbar with icons for file operations, running, and saving. The notebook title is "Untitled5" with a last checkpoint of 20 minutes ago. The kernel is identified as "Python 3 (ipykernel)". The code cell contains a Python script for emotion detection:

```
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)

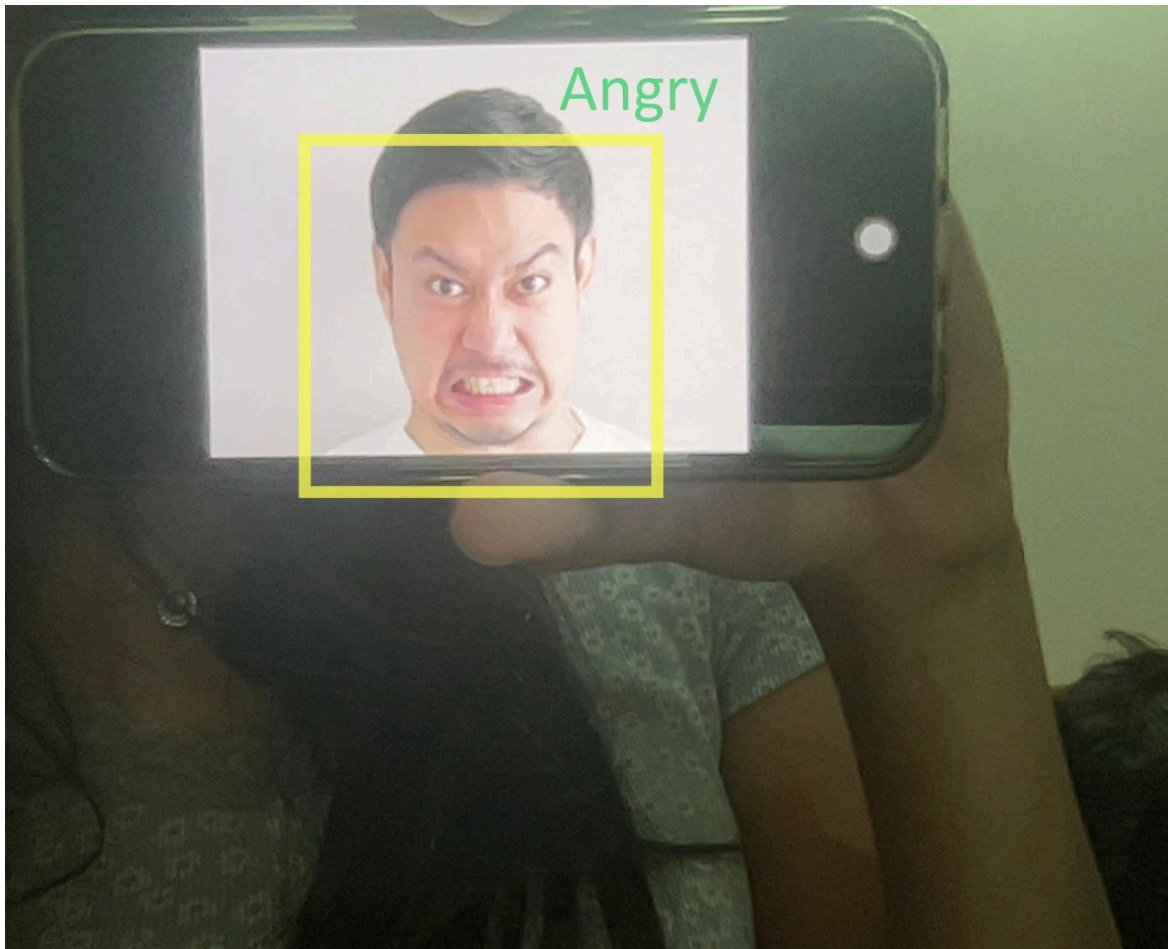
        if np.sum(roi_gray) != 0:
            roi = roi_gray.astype('float')/255.0
            roi = roi.toarray()
            roi = np.expand_dims(roi, axis=0)

            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()
```


Chapter 5: Results and Conclusions

The AI model accurately predicted the emotion displayed by the picture on my phone .With higher number of epochs this accuracy can be increased.



In conclusion, the little experiment on AI-assisted human emotion recognition in images has shown encouraging results and has the potential to revolutionise a number of industries. We can solve many of the difficulties involved with manually interpreting emotions, such as human error, bias, and inefficiency, by utilising the power of AI.

