STRESS DETECTION USING FACIAL FEATURES

A major project report submitted in partial fulfilment of the requirement for the award of degree of

Bachelor of Technology

in

Computer Science & Engineering / Information Technology

Submitted by

Shambhavi Patial (201280)

Mritunjay Chaudhary (201481)

Under the guidance & supervision of

Mr. Aayush Sharma

Ms. Seema Rani

Department of Computer Science & Engineering and Information Technology

Jaypee University of Information Technology, Waknaghat, Solan - 173234 (India)

CERTIFICATE

I hereby declare that the work presented in this report entitled **“Stress Detection using Facial Features”** in partial fulfilment of the requirements for the award of the degree of **Bachelor of Technology in Computer Science and Engineering** submitted in the department of **Computer Science & Engineering and Information Technology**, Jaypee University of Information Technology, Waknaghat is an authentic record of my own work carried out over a period from August 2023 to May 2024 under the supervision of **Mr. Aayush Sharma (**Assistant Professor, Department of Computer Science & Engineering and Information Technology**)** and **Ms. Seema Rani (**Assistant Professor, Department of Computer Science & Engineering and Information Technology**)**. The matter embodied in the report has not been submitted for the award of any other degree or diploma.

Shambhavi Patial (201280) **Mritunjay Chaudhary (201481)**

This is to certify that the above statement made by the candidate is true to the best of my knowledge.

Mr. Aayush Sharma
Assistant Professor
Department of CSE & IT
Dated:

Ms. Seema Rani
Assistant Professor
Department of CSE & IT
Dated:

CANDIDATE’S DECLARATION

We hereby declare that the work presented in this report entitled **‘Stress Detection using Facial Features’** in partial fulfillment of the requirements for the award of the degree of **Bachelor of Technology** in **Computer Science & Engineering / Information Technology** submitted in the Department of Computer Science & Engineering and Information Technology**,** Jaypee University of Information Technology, Waknaghat is an authentic record of my own work carried out over a period from August 2023 to May 2024 under the supervision of **Mr. Aayush Sharma** (Assistant Professor, Department of Computer Science & Engineering and Information Technology) and **Ms. Seema Rani** (Assistant Professor, Department of Computer Science & Engineering and Information Technology).

The matter embodied in the report has not been submitted for the award of any other degree or diploma.

Shambhavi Patial	Mritunjay Chaudhary
Roll No.: 201280	Roll No.: 201481

This is to certify that the above statement made by the candidate is true to the best of my knowledge.

(Supervisor Signature with Date)	(Supervisor Signature with Date)
Supervisor Name: Mr. Aayush Sharma	Supervisor Name: Ms. Seema Rani
Designation: Assistant Professor	Designation: Assistant Professor
Department: CSE & IT	Department: CSE & IT
Dated:	Dated:

ACKNOWLEDGEMENT

First, I express my heartiest thanks and gratefulness to the Lord for His divine blessing to make it possible to complete the project work successfully.

We owe our profound gratitude and indebtedness to our project supervisors **Mr. Aaysuh Sharma** and **Ms. Seema Rani**, who took keen interest and guided us all along in our project work titled **— Stress Detection using Facial Features**, till the completion of our project by providing all the necessary information for developing the project. The project development helped us in research, and we got to know a lot of new things in our domain. We are really thankful to them.

I would also generously welcome each one of those individuals who have helped me straightforwardly or in a roundabout way in making this project a win. In this unique situation, I might want to thank the various staff individuals, educating and non-instructing, who have developed their convenient help and facilitated my undertaking.
Finally, I must acknowledge with due respect the constant support and patients of my parents.

Shambhavi Patial
(201280)

Mritunjay Chaudhary
(201481)

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

Abbreviation	Full Form
CNN	Convolutional Neural Network
FER2013	Facial Emotion Recognition 2013
RESNET	Residual Network
KNN	k-Nearest Neighbor
RAM	Random Access Memory
RMSP	Root Mean Square Propogation

ABSTRACT

Due to the sudden increase in stress related disorders, it is very necessary to develop technologies and applications which are capable of early detection of stress and can prevent disorders related to it. The design and implementation of such a platform is being proposed in this report. It proposes a method of identifying a person's emotional state by using facial features using a camera or an image.

This report discusses the implementation of CNN model for effective facial recognition for emotion detection and stress level detection using the FER2013 dataset which includes training and testing grayscale images of 7 types of emotions i.e., sad, surprised, happy, fear, neutral, angry and disgust.

CHAPTER 1: INTRODUCTION

1.1 INTRODUCTION

Stress management system is necessary to determine the level of stress affecting our world today. Stress is a mental disorder that affects one in four people, according to the World Health Organization (WHO).

Psychological and financial problems, including confusion at work, poor work relationships, depression, and in extreme cases, death, are all symptoms of human anxiety. This creates the need to provide treatment to help anxious people manage their stress. Although it cannot eliminate stress completely, taking precautions will help you cope with stress. Nowadays only people in the medical and psychological fields can decide whether someone is suffering from depression (stress). Questionnaires are one of the best ways to measure stress. The process is based on individual responses; Whether people are worried or not, they may refuse to communicate. Stress screening can reduce the likelihood of health problems and improve public health. This includes developing a scientific method to measure people's stress levels using physiological markers. Because stress has a significant impact on people, many ways to measure stress have been investigated. Ghaderi Tal says this improves people's quality of life.

Stress can be assessed by measuring data such as heart rate (HR), face electromyography (EMG), Galvanic skin response (GSR) foot and hand. Electrocardiograms (ECG or EKG) were also utilized by David Liu and colleagues in order to predict the levels of stress. The effectiveness of these multimodal sensors in detecting stress is still being investigated experimentally and can also point to other natural causes.

Today, the IT industry creates new trends in the market by offering new technologies and products. The research also found that stressed employees also set the bar high. Although many companies offer health benefits to their employees, the issue remains unregulated. In this study, we tried to examine this issue in more detail by analyzing the stress patterns of cooperative employees. We plan to use image processing and deep learning techniques to identify stress patterns and minimize factors that affect stress levels. Deep learning methods such as CNN are used to classify different levels of stress.

1.2 PROBLEM STATEMENT

Stress can be defined as a feeling of emotional or physical tension. It can be a reaction that comes to us when confronted with a challenge or a demand. Stress can be acute, that is an intense and unpleasant reaction that can begin shortly after an overwhelming or traumatic event. It usually does not have long term consequences but repeated episodes can be damaging. Also, it can be chronic, that lasts for a longer period of time or comes back again and again. This can cause high BP, heart diseases, and increase risk of type 2 diabetes. It can also lead to depression, anxiety and other mental health disorders.

India has seen a major growth in the IT sector which is very much needed for its economic growth. The world has become a global village and new jobs are emerging in India, creating a huge demand for people leading stressful jobs. These professionals often work in a rapidly changing environment and have no control over their environment, the pace of their work, or the circumstances they have to deal with. The high absenteeism rates in these jobs indicate that these professionals experience many negative psychological effects due to the stress they experience in their jobs.

The early detection of stress is very important as it can help reduce the chances of brain damage caused by excessive stress and other health problems. It can also affect your body, feelings, thoughts, and behavior. Chronic stress can also lead to many health problems such as: Depression, Diabetes, High BP, Heart disease, Cardiovascular disease, Stroke, Obesity, etc.

1.3 OBJECTIVES

The primary objective of our proposed project is to predict the emotion of a person by its picture irrespective of the atmosphere the image has been taken i.e., indoor or outdoor with different lighting conditions.

- Development of a stable neural network for detecting stress and emotion.
- Implement a RESNET 50V2 model to get a better prediction for the input provided by the user to the model.
- Real world applicability - we will also try to assess the algorithm’s potential for real world applications where different lighting conditions as well as different facial landmarks are varied.
- Emotion Detection - First we will develop an emotion detection model which will be able to detect seven different emotions using various facial landmarks. This model will

further be integrated with a stress detection model where it will help to measure stress levels of the individual.

- Stress Prediction -We will try to include the algorithm to detect the stress levels of a person either via a camera or picture along with the emotion in the second phase of our project.
- Training and validating the model - The testing and validation of the model is an essential stage of a project. so, by contrasting the model prediction with a set of test data we can analyze the model’s performance and validity in predicting the real-life images.
- Web application - In the later phase of the project we will be integrating our prediction model with a frontend application in order to get a more user-friendly environment.
- Improved Mental Health Monitoring: Offer continuous monitoring of individuals' stress levels, contributing to better overall mental health management.

1.4 SIGNIFICANCE & MOTIVATION

AI and Machine Learning (ML) are widely employed in many domains. They have been used to detect insurance fraud, identify patterns in the stock market and detect patterns in other various fields. They have also played a significant role in pattern recognition and classification like FER (Facial Emotion Recognition).

As stress is so prevalent in our society today, and can have many adverse effects on our mental and physical health, it is important to detect it as soon as possible. So, we use the help of above-mentioned techniques along with Deep Learning (DL) models using CNN architecture to make a system that can efficiently detect stress in real-time.

We have also developed an emotion detection model which aids in detecting stress levels of the individuals by identifying seven different emotions by which stress levels can also be altered.

1.5 ORGANIZATION

Our project report consists of 5 main chapters.

The First Chapter includes the main issue and describes the problem. It gives a brief description regarding the proposed model and suggests techniques which can be implemented in future. Chapter 2 includes the literature survey and review that is helpful for determining our project approach. Dataset description, preprocessing approach, and information regarding the framework of our proposed model have been covered in the

Chapter 3. Chapter 4 includes the Software design techniques, tools used, deployed model, system and software requirements. In the last 4th chapter project evaluation, advantages and future scope of this project are covered.

CHAPTER 2: LITERATURE SURVEY

2.1 OVERVIEW OF RELEVANT LITERATURE

2.1.1 Emotion Detection System Using Machine Learning [1]

Author: Prof. Sarika Pawar, Harshada Pisal, Mangesh Shelke, Nitin Nimbalkar

Year: 2022

Summary: The authors have used a machine learning approach in which a KNN classifier is used to identify several emotions such as happy, sad, fear, surprise and anger. They used Haar Cascade to detect the face and its features.

2.1.2 Four-layer ConvNet to facial emotion recognition with minimal epochs and the significance of data diversity [2]

Author: Debnath, T., Reza, M.M., Rahman

Year: 2022

Summary: The authors have employed deep learning through a four-layer Convolutional Neural Network to recognize various emotions using a minimal number of epochs. The use of Local Binary Pattern (LBP) plays an important role to to extract facial features and CNN was used to classify the images according to the emotions. Training model accuracy is 95% and validating accuracy was 91%.

2.1.3 Facial Expression Recognition Via Enhanced Stress Convolution Neural Network for Stress Detection [3]

Author: Wan-Ting Chew, *Siew-Chin Chong, Thian-Song Ong and Lee-Ying Chong

Year: 2022

Summary: The authors of [3] proposed a 50-layer ResNet pre-trained Convolutional Neural Network helps in classification of emotions. ResNet5 architecture is used here which gives an accuracy of 81.67%.

2.1.4 Recognition of Stress Using Face Image and Facial Landmarks [4]

Author: Mohammad Faizal, Nikhil V, , Prajwal C, Aruna Rao B P

Year: 2021

Summary: The authors of [4] also used the machine learning approach of K Nearest Neighbours algorithm for classifying the emotions into various categories. But they used a JAFFE dataset which consists of approximately 213 images which were collected in a more controlled environment.

2.1.5 Extraction of Facial Features for Depression Detection among Students [5]

Author: Dr. D. Venkataraman, Namboodiri Sandhya Parameswaran

Year: 2021

Summary: The above-mentioned authors of [5] have used the Support Vector Machines classifier for the classification of the images into two categories- depressed and not depressed. The presence of some happy features and disgust features, the user can be classified under depressed/ not depressed categories. JAFFE is used here .

2.1.6 Stress Detection Using Image processing and Machine Learning [6]

Author: MS.N.Pavani, P.Supriya, A.SiriChandana, B.Trinetra, S.V.N.S.S.Supriya

Year: 2021

Summary: In [6] a real time stress detection system is used which can detect stress in individuals by allowing the user to upload photograph of themselves on the site and getting the result back in a short period which also indicates the level of stress they have. CNN architecture was used and Flask was used to deploy the model on the web.

2.1.7 Real-time Stress Detection using CNN [7]

Author: A. Srinivas, Sruthi Lanka, Shailaja Preethi Molleti Vijaya Varshini Kallepalli, Divya Kalla, Sruthii Nukala

Year: 2021

Summary: In [7], the authors have used a 5 player CNN model to detect stress in four different levels, they created the model using OpenCV and tensorflow. The four levels of stress are predicted with an accuracy of 73%

2.1.8 Facial Emotion Detection using Convolutional Neural Networks [8]

Author: Mohammad Adnan Adil

Year: 2021

Summary: Mohammad Adnan Adil [8] has proposed two CNN models to classify emotions. The CNN model 2 is somehow similar to Model 1 but has fewer layers. CNN

Model 1 has two convolution layers in the first 3 phases which includes a batch normalization layer followed by pooling, but in case of our CNN Model 2 there exists one convolution and batch normalization layer followed by pooling. Accuracy of Both models is about 80%

2.1.9 Stress **Recognition Using Facial Landmarks and CNN (Alexnet) [9]**

Author: P Ramesh Naidu

Year: 2021

Summary: In [9], a method of combining Local Binary patterns (LBP) and Convolutional Neural Networks (CNN) is used. It deals with the disadvantage of poor stability of CNN grey-scale and helps in identifying the trained network more effectively. Predictions are made according to 3 emotions: if the percentage of angry, disgust and sad emotion is more than 50% the person is considered as under stressed.

2.1.10 Extraction of Facial Features for Depression Detection among Students [10]

Author: Dr. D. Venkataraman, Namboodiri Sandhya Parameswaran

Year: 2020

Summary: In [10], the authors have made a real-time Facial Emotion Detection system using CNN Architecture and also provide the used with relevant recommendations based on results. They only take small videos of the user as inputs.

2.1.11 Stress Detection Using Machine Learning Algorithms [11]

Author: V. R. Archana, B. M. Devaraju

Year: 2020

Summary: The authors of [11] have used a machine learning based approach to determine the stress level of the individual. They employed KNN classifier, Naive Bayes classifier and Decision Tree classifiers to also compare their results.

2.1.12 Detecting Negative Emotional Stress Based on Facial Expression in Real Time [12]

Author: Zhang, J., Mei, X., Liu, H., Yuan, S., & Qian

Year: 2019

Summary: In [12] real time stress detection is done by analyzing the negative emotions . They used Multi-task Cascaded Convolutional Networks for their work.

They achieved an accuracy of 80.4% for Oulu-Casia dataset and an accuracy of 99% for KMU-FED dataset.

2.2 KEY GAPS IN LITERATURE

These are the following gaps found in the literature:

1. **Robustness**

   Face Posture, blurring and occlusion are some of the hurdles that make it difficult to make predictions in real-time.

2. **Emotions are less complex**

   Only the basic five emotions (surprised, angry, fear, sad, happy) or seven emotions (including neutral and disgust) are classified. More complex or mixed emotions such as shocked with pleasure, surprised by frustration, dissatisfied by anger, surprised by sorrow, and so on are not explored.

3. **Scalability**

   Challenges faced during scalability of models are not discussed which can lead to value addition on large datasets

4. **Large computational power required**

   Deep Learning models require a huge labeled dataset, with significant memory and long training and testing times that makes it difficult to implement on regular devices.

5. **Data Bias and Fairness**

   Ongoing issues with ensuring that models do not favor specific demographics or types of images. Fairness, diversity, and bias concerns should be part of research for training data and captioning as well.

6. **Inconsistent evaluation metrics**

   To calculate the performance of the models applied, a lack of standardized evaluation metrics was observed with no facility for benchmarks. In some cases, evaluation metrics were completely ignored and no clear outcome of the research could be stated.

7. **Limited Generalization**

   Most existing models perform well on some datasets; however, they do not usually generalize to other or new data. It remains difficult to guarantee consistent excellent performance across diverse domains and datasets.

8. **No collaboration with experts**

   Only technical aspects are focused on with no collaboration from experts in emotion detection, or traditional medicine to improve the accuracy of the models.

CHAPTER 3: SYSTEM DEVELOPMENT

3.1 REQUIREMENTS AND ANALYSIS

The following requirements and analysis are required:

1. Functional requirements

- Uploading Image and processing
    1) We will be needing a model which is capable of processing the uploaded image by the user and able to save it for future work.
    2) The model should be able to extract only the essential features from the image which are needed for prediction.
- Identifying the image surroundings
    1) The model should be able to identify surroundings in which the image is taken and able to process only the necessary information.
    2) Model should be able to predict both indoor and outdoor images and after analyzing should generate the output.
- Real time processing
    1) The model should be able to predict the output of the image soon after uploading in the minimum time.
- Model Updation
    1) We should be able to retrain the model once the updated dataset is present.

2. Nonfunctional requirements

- Performance

  The model should be able to predict the output under 3 sec of time.
- Reliability

  The model should attain a minimum possible model training loss.
- Scalability and compatibility

  The model should be compatible with the frontend framework Flask in order to make a Web application or website so that it can be accessed by many users easily and efficiently.
- Training Dataset

  We have used a big dataset of 35000 training images in order to get better accuracy.

- Code Maintenance

  The codebase of the model should follow industry standards to increase readability and understandability.
- Image resolution

  The supports all types of common image formats like JPEG, JPG, PNG etc.
- Validation and Testing

  We will be Testing the model on the test dataset and validating it with some random images.

3. Hardware requirements

- GPU (s):

  High performance GPUs are needed for efficient training like NVIDIA MAXWELL GPU.
- CPU:

  A multicore CPU is required so that it can efficiently train the model and handle its oppression like the QUAD -Core ARM Cortex -A57 processor.
- RAM:

  Minimum of 16 GB LPDDR4 is required.

4. Software requirements

- Languages

  Python is used throughout the code base.
- Deep learning Frameworks

  We have to chose suitable Deep Learning Frameworks like TensorFlow and Keras for training and Testing of models.
- Version Control

  For the version control we will be using git.
- Development Environment

  We will be using VS Code and Jupyter notebooks as our development environments.

3.2 PROJECT DESIGN AND ARCHITECTURE

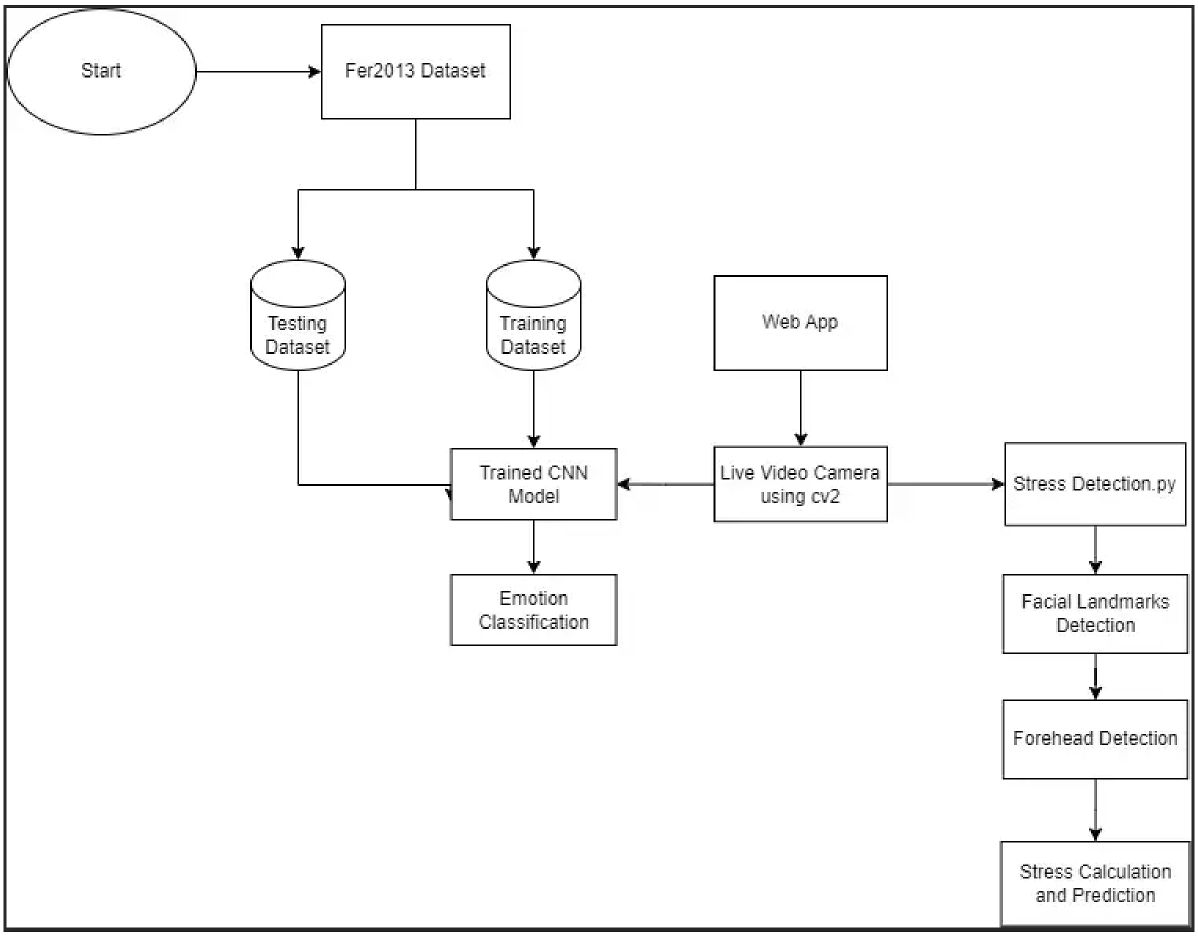


Fig.1 - Project Design

The stress detection method starts with the Fer2013 dataset, which is divided into training and testing sets. The training data is utilized to create a Convolutional Neural Network (CNN) model for emotion classification. Meanwhile, the system incorporates a web application that communicates with consumers using live video camera feeds. The video feed detects facial landmarks, particularly those located on the forehead. These markers are then utilized to estimate stress levels and anticipate emotions in real time. The system's architecture guarantees that all components work seamlessly together, allowing for accurate stress detection.

3.3 DATA PREPARATION

3.3.1 DATA COLLECTION

After serving many existing similar models and research papers we have observed that many of them are using publicly available dataset - **FER2013** from Kaggle.

FER2013 consists of 48*48-pixel grayscale images containing faces. The dataset is prepared in such a way that the faces have been automatically registered, the face is more or less centered and occupies about the same amount of space in each image.

Contents of dataset

The dataset has been categorized into 7 emotions

1) Angry
2) Fear
3) Happy
4) Sad
5) Surprise
6) Neutral
7) Disgust

Training Dataset -

The Training dataset contains total images-29273 for various emotions -

Emotions	**Images count**
NEUTRAL	4982
HAPPY	7164
SAD	4938
FEAR	4103
ANGRY	3993
SURPRISED	3205
DISGUST	436

Table 1. Training Dataset

Testing Dataset -

The Testing dataset contains numerous images-7067 for various emotions -

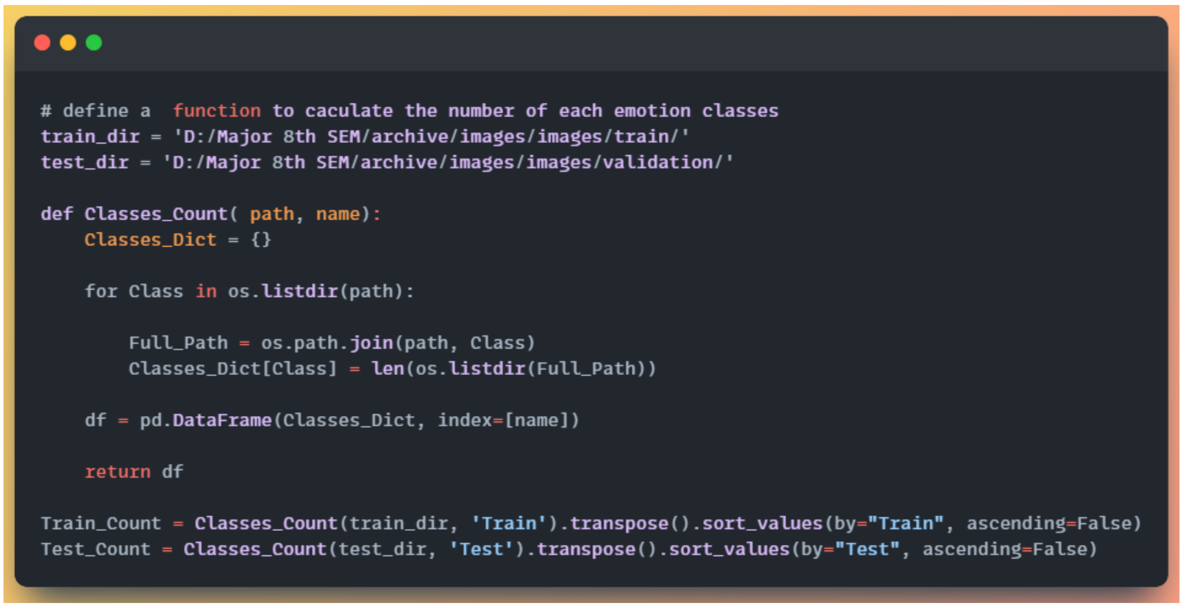
Emotions	**Images count**
NEUTRAL	1216
HAPPY	1825
SAD	1139
FEAR	1018
ANGRY	960
SURPRISED	797
DISGUST	111

Table 2. Testing Dataset

3.3.2 DATA PREPROCESSING

Since the data is raw, we have to process it in order to train our model better. Below are the steps for preprocessing: -

Creating dataframe for training and testing data

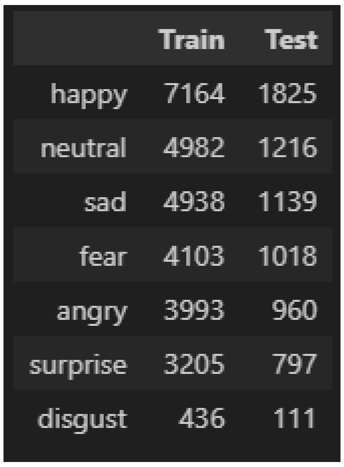


```
# define a  function to caculate the number of each emotion classes
train_dir = 'D:/Major 8th SEM/archive/images/images/train/'
test_dir = 'D:/Major 8th SEM/archive/images/images/validation/'

def Classes_Count( path, name):
    Classes_Dict = {}

    for Class in os.listdir(path):

        Full_Path = os.path.join(path, Class)
        Classes_Dict[Class] = len(os.listdir(Full_Path))

    df = pd.DataFrame(Classes_Dict, index=[name])

    return df

Train_Count = Classes_Count(train_dir, 'Train').transpose().sort_values(by="Train", ascending=False)
Test_Count = Classes_Count(test_dir, 'Test').transpose().sort_values(by="Test", ascending=False)
```

Fig.2 - Creating Dataframe

In this snippet, this code provides a tabular summary of the number of instances in each class for both the training and testing directories, sorted in descending order based on the count. This information might be useful for understanding the distribution of classes in a dataset.

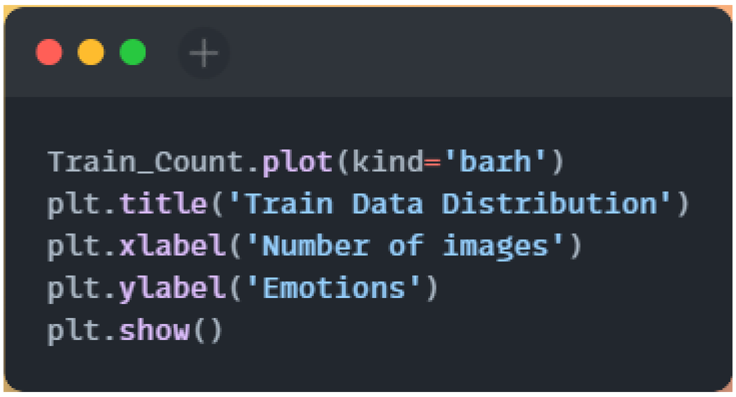
We get the following output:



	Train	Test
happy	7164	1825
neutral	4982	1216
sad	4938	1139
fear	4103	1018
angry	3993	960
surprise	3205	797
disgust	436	111

Fig.3 - Output for the dataframe

Visualizing distribution of images in each class



```
Train_Count.plot(kind='barh')
plt.title('Train Data Distribution')
plt.xlabel('Number of images')
plt.ylabel('Emotions')
plt.show()
```

Fig.4 - Creating bar chart

This code creates a bar chart for training and testing data.

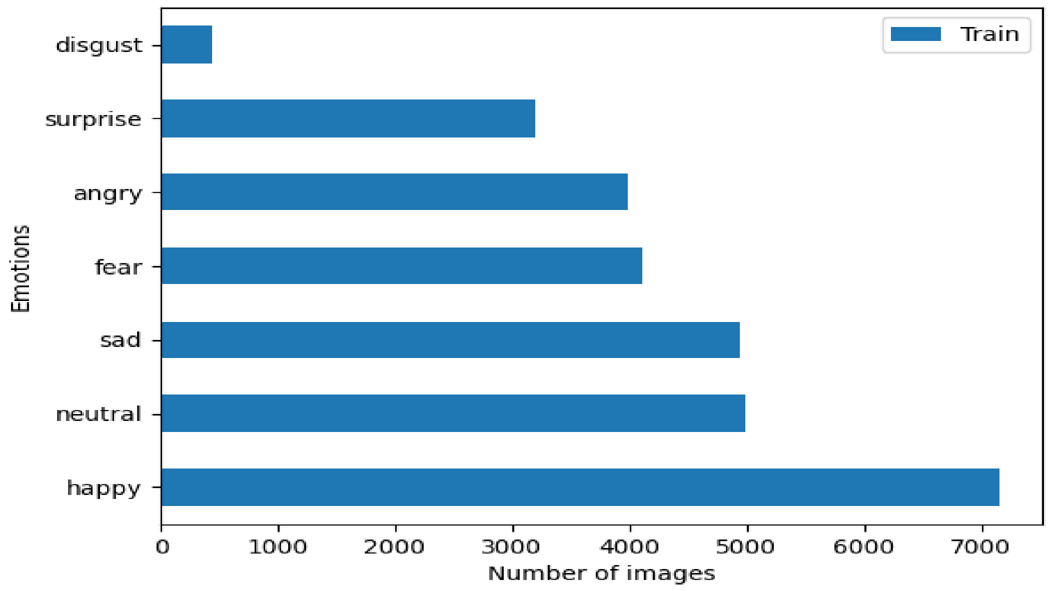


Fig.5 - Distribution for training data

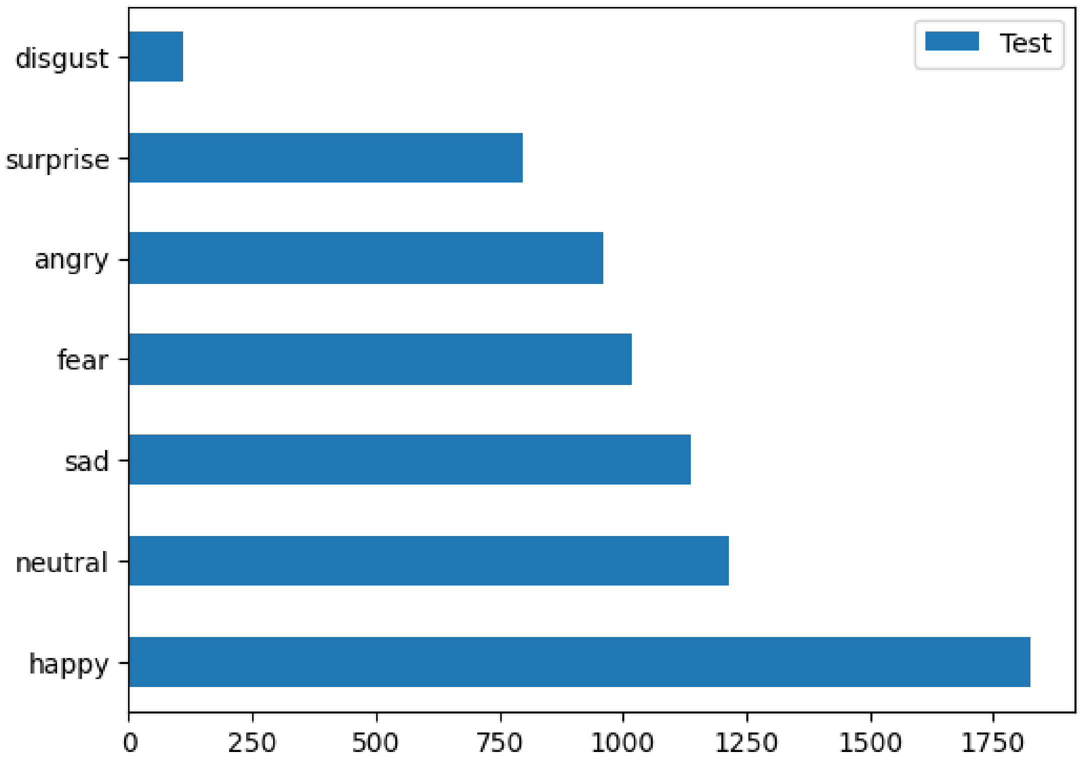
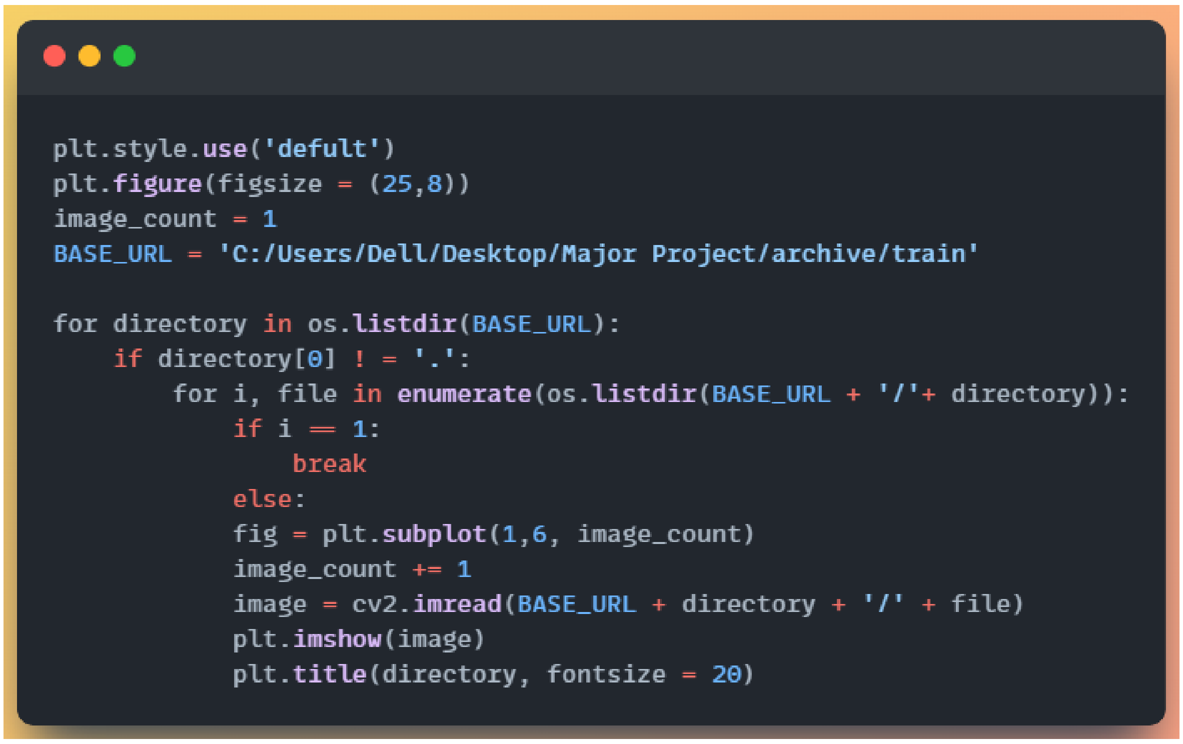


Fig.6 - Distribution for testing data

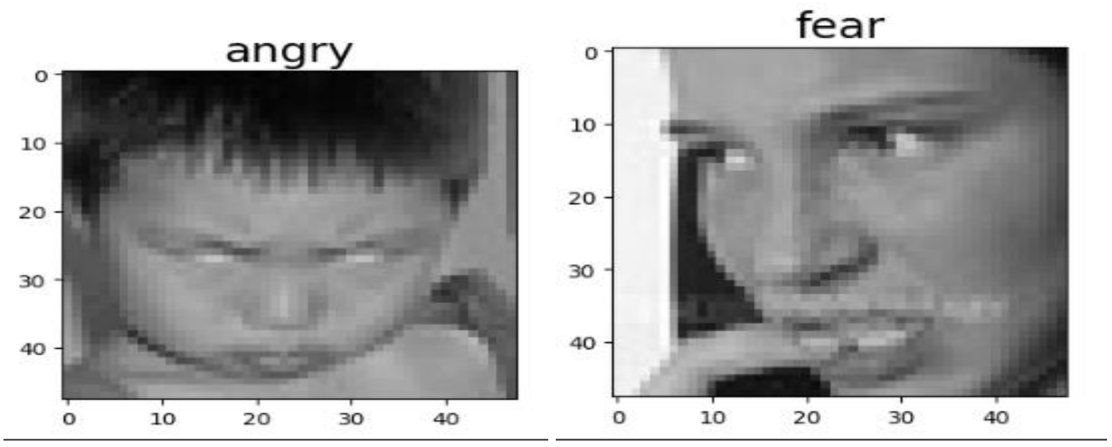
Printing an example image from each class



```
plt.style.use('defult')
plt.figure(figsize = (25,8))
image_count = 1
BASE_URL = 'C:/Users/Dell/Desktop/Major Project/archive/train'

for directory in os.listdir(BASE_URL):
    if directory[0] ! = '.':
        for i, file in enumerate(os.listdir(BASE_URL + '/'+ directory)):
            if i == 1:
                break
            else:
            fig = plt.subplot(1,6, image_count)
            image_count += 1
            image = cv2.imread(BASE_URL + directory + '/' + file)
            plt.imshow(image)
            plt.title(directory, fontsize = 20)
```

Fig.7 - Printing example images

In the previous code snippet is using the Matplotlib library to create a visualization of images from a specified directory. As we have six different types of emotions, we are printing 6 random images.



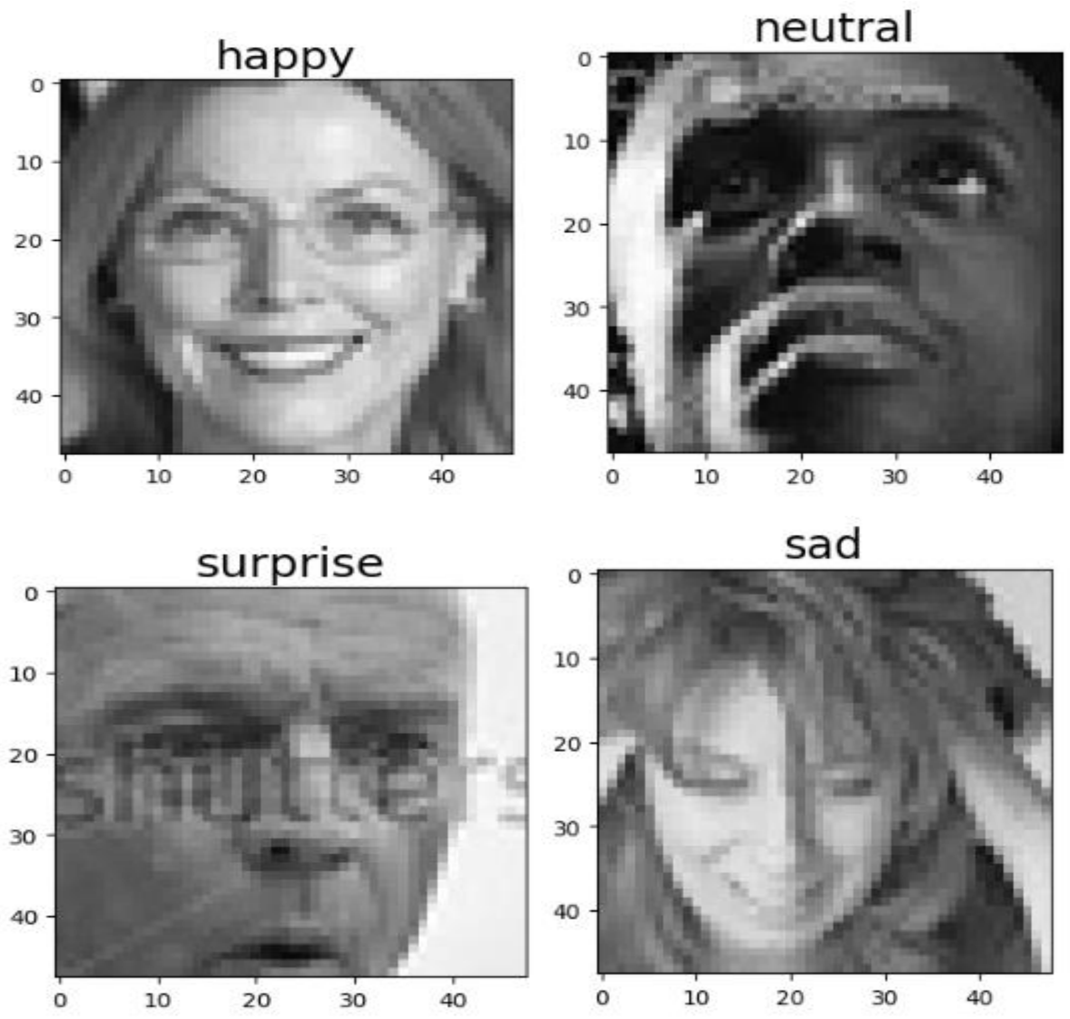
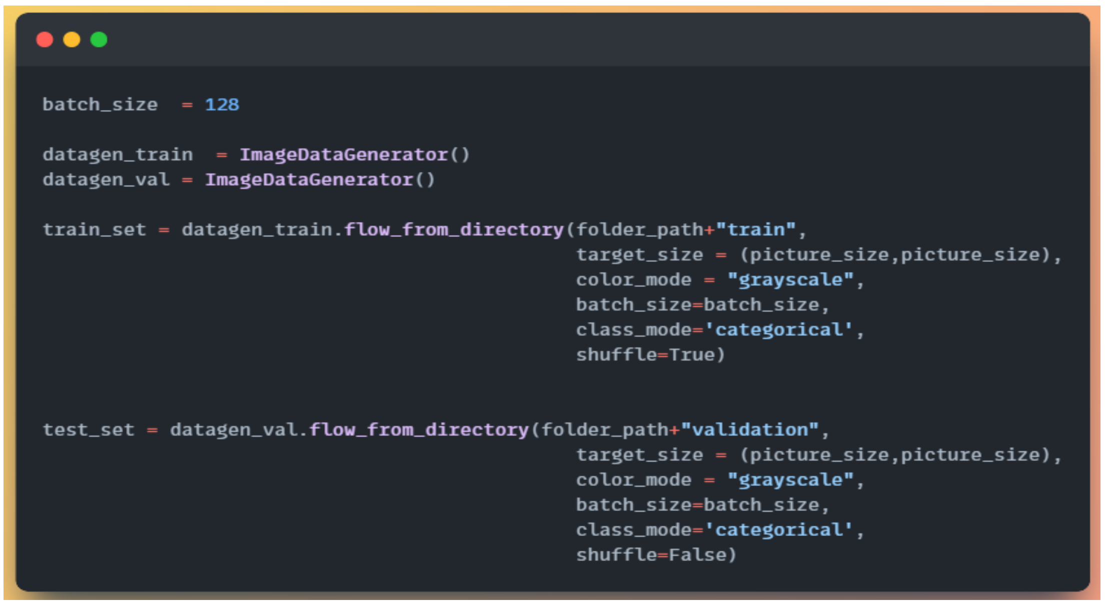


Fig.8 - Example images of six emotions

Creating final datasets



```
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)
```

Fig.9 - Creation of final dataset

In the above code snippet, we create the final dataset each for training data and testing data. Now our preprocessing is completed as the distribution of images in each of the seven classes of emotions are perfectly balanced.

3.4 IMPLEMENTATION

In the codebase we have implemented CNN algorithm for the emotion classification and predicting the stress levels using facial landmarks.

3.4.1 CONVOLUTIONAL NEURAL NETWORK (EMOTION CLASSIFICATION)

Convolutional neural networks, or CNNs or ConvNets, are a subclass of neural networks that are particularly good at processing input with a topology resembling a grid, like images. A binary representation of visual data is what makes up a digital image. It is made up of a grid-like arrangement of pixels with pixel values to indicate the color and brightness of each pixel. A CNN mainly has three layers: convolutional layer, pooling layer, and a fully connected layer.

Convolutional Layer

The fundamental component of the CNN is the convolution layer. It bears the majority of the computational strain on the network. This layer performs a dot product between two matrices, where one matrix is the set of learnable parameters otherwise known as a kernel, and the other matrix is the restricted portion of the receptive field.

Pooling Layer

By calculating a summary statistic from the outputs in the vicinity, the pooling layer substitutes the network's output at specific points. This aids in shrinking the representation's spatial size, which lowers the quantity of computation and weights needed. Each slice of the representation is processed independently for the pooling operation.

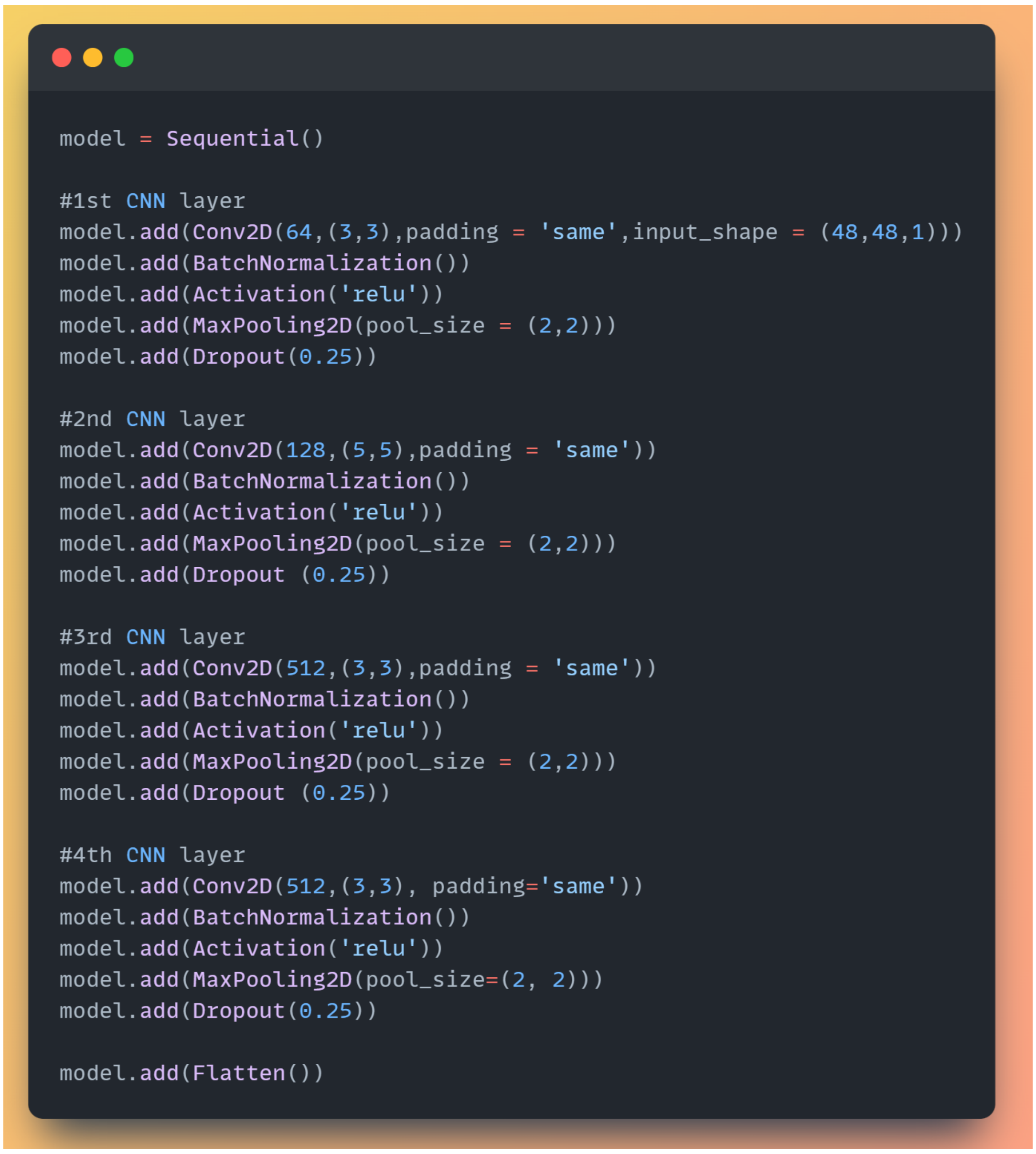
Fully Connected Layer

As in a conventional CNN, all neurons in this layer are fully connected to all neurons in the layer that comes before and after. Because of this, it may be calculated using the standard method of matrix multiplication and bias effect.

Now let's check the implementation of CNN

The CNN model consisting of 6 layers –

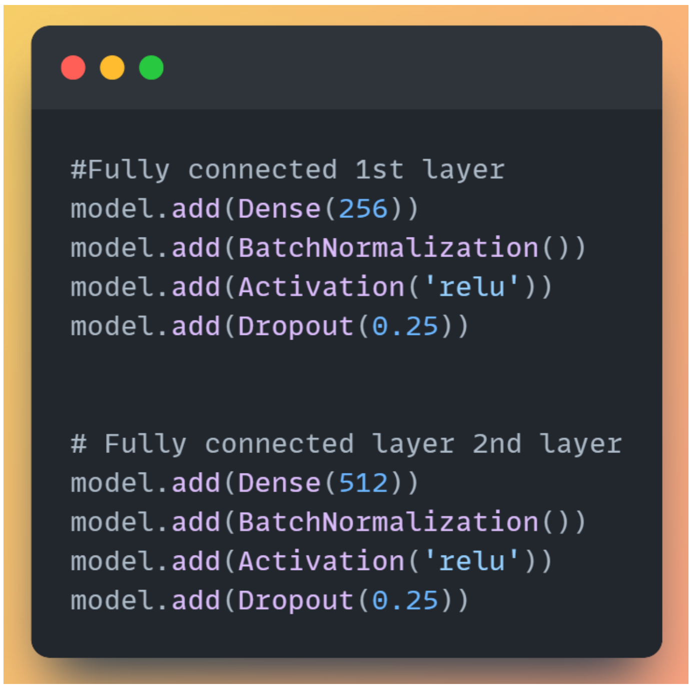
- 4 convolution layers
- 2 fully connected layers



```
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())
```

Fig.10 - Four convolution layer

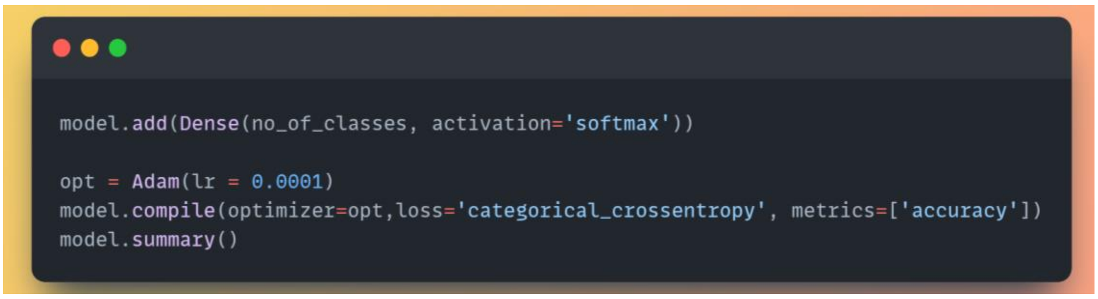
In the above snippet we can observer the four convolution layers that have be applied to build our emotion detection model. Batch normalization, relu activation function, maxpooling layer and dropout layer has been applied after each layer.



```
#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))
```

Fig.11 - Two Fully Connected convolutional layer

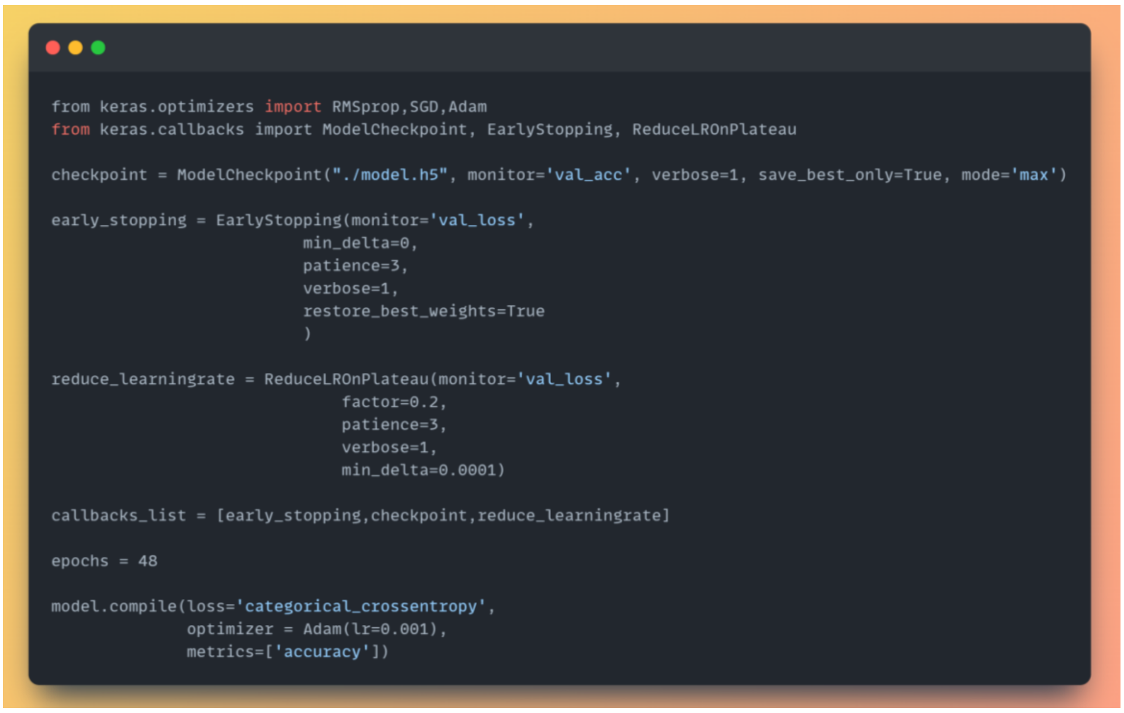
In the above snippets, 4 convolutional layers and 2 fully connected layers have been connected. Now let's create the output layer and make the model.



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

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

Fig.12 - Output layer and model creation

This code applies an adam optimizer with learning rate of 0.0001 and creates and compiles the model and finally printing the model summary.



```
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'])
```

Fig.13 - Compilation of Model

The above code sets up three callbacks using Keras - EarlyStopping, ModelCheckpoint and ReduceLROnPlateau. These callbacks are used to enhance the training process by adjusting learning rates and avoiding overfitting through early stopping.

- **EarlyStopping** - It is a type of regularization that stops a neural network's training process before it reaches the maximum number of epochs, or iterations. The goal is to monitor the network's performance on a validation set and stop training when the validation error begins to increase or stops improving.
- **ModelCheckpoint** - It is a checkpoint file that saves a model or its weights at a specified interval during training. The model or weights can then be loaded later to continue training from the saved state.
- **ReduceLROnPlateau -** Reduce learning rate when a metric has stopped showing any improvement. Models can generally benefit by reducing the learning rate by a factor of 2-10 when learning stagnates. This callback monitors a quantity and if no improvement is seen for any ‘n" number of epochs, the learning rate is reduced.

Adam Optimizer

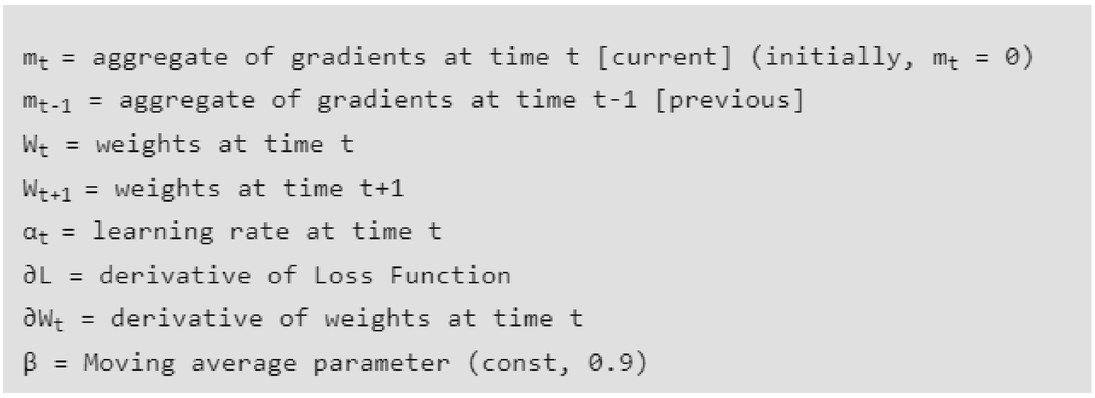
The algorithm for gradient descent optimization is called Adaptive Moment Estimation. When dealing with large problems involving many data points or parameters, the method is extremely effective. It is effective and requires little memory. Essentially, it combines the "gradient descent with momentum" and "RMSP" algorithms.

Adam optimizer combines two gradient descent methods:

- Momentum - This algorithm accelerates the gradient descent algorithm by taking into account the "exponentially weighted average" of the gradients. The algorithm converges to the minima faster when averages are used.

  $w_{t+1} = w_t - \alpha\, m_t$

  where, $m_t = \beta\, m_{t-1} + (1-\beta) \left[\frac{\delta L}{\delta w}\right]$

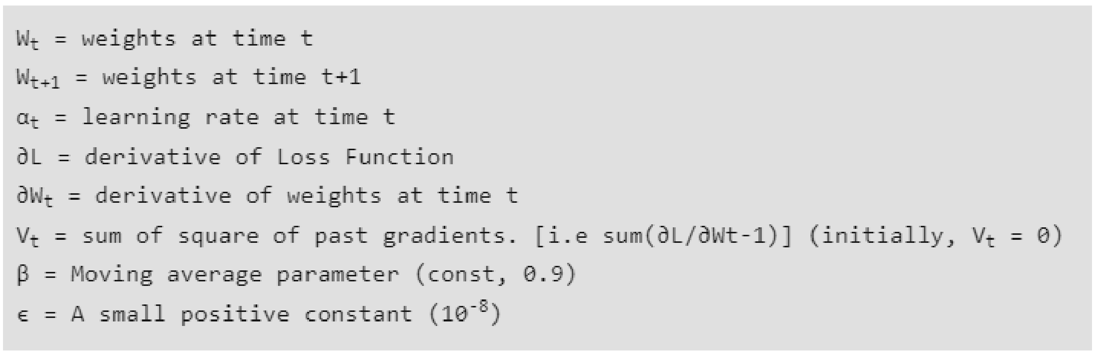


m_t = aggregate of gradients at time t [current] (initially, m_t = 0)
m_{t-1} = aggregate of gradients at time t-1 [previous]
W_t = weights at time t
W_{t+1} = weights at time t+1
α_t = learning rate at time t
∂L = derivative of Loss Function
∂W_t = derivative of weights at time t
β = Moving average parameter (const, 0.9)

Fig.14 - Parameters for momentum

- Root Mean Square Propagation (RMSP) - AdaGrad is intended to be improved by an adaptive learning algorithm known as Root Mean Square Prop, or RMSprop. Instead of using the cumulative sum of squared gradients found in AdaGrad, it makes use of the "exponential moving average".

  $w_{t+1} = w_t - \frac{\alpha_t}{(v_t + \varepsilon)^{1/2}} * \left[\frac{\delta L}{\delta w}\right]$



W_t = weights at time t
W_{t+1} = weights at time t+1
α_t = learning rate at time t
∂L = derivative of Loss Function
∂W_t = derivative of weights at time t
V_t = sum of square of past gradients. [i.e sum(∂L/∂Wt-1)] (initially, V_t = 0)
β = Moving average parameter (const, 0.9)
ϵ = A small positive constant (10^{-8})

Fig.15 - Parameters for RMSP

Haarcascade

The object detection algorithm known as the Haar Cascade Algorithm was proposed by Paul Viola and Michael Jones. A machine learning technique called Haar Cascade uses a large number of different images, both positive and negative, to train the classifier.

- Positive Images: These are types of images that we want our classifier to identify i.e, it contains objects we want to detect.
- Negative Images: These types of images contain something else, i.e., it does not contain the objects we want to detect.

The Haar classifiers, which we have also used here, were also employed by the first real-time face detectors. A machine learning algorithm called a Haar classifier or a Haar cascade classifier locates objects in images and videos.

Haar Cascade Algorithm involves four stages that include:

- Calculation of Haar Features: A Haar feature is just a calculation made on nearby regions at a particular place in a different detection window. The main steps in the computation are adding the intensities of each pixel in each region and figuring out the total of the differences.
- Creation of Integral Images: Creating Integral Images minimizes computation. Rather than performing calculations at every pixel, it generates sub-rectangles, which the array then references in order to compute the Haar Features. When it comes to object detection, the features of an object are the only ones that matter; most other Haar features are not relevant. However, out of hundreds of thousands of Haar features, how do we choose the ones that most accurately represent an object? Adaboost now makes an appearance.
- Adaboost Usage: Adaboost Training combines the "weak classifiers" to create a "strong classifier" that the object detection method can employ. In essence, this is choosing features that are helpful and instructing classifiers on how to use them.
- Implementation of Cascading Classifiers: Here, each sage is essentially a group of unskilled pupils. By using boosting to train weak learners, a highly accurate classifier can be created by averaging the predictions of all weak learners.

Max Pooling Layer

The process of pooling that chooses the maximum element from the area of the feature map that the filter covers is called max pooling. As a result, the feature map that results from the max-pooling layer would include the most noticeable features from the prior feature map.

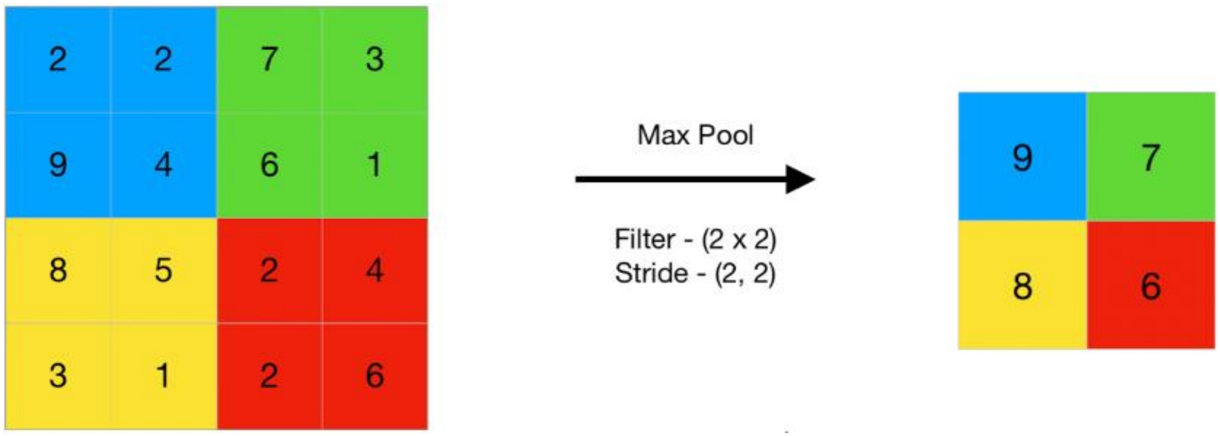


Fig.16 - Max Pooling Layer

Average Pooling Layer

Average pooling computes the average of the elements present in the filter's feature map region. Thus, while max pooling returns the most prominent feature in a given patch of the feature map, average pooling returns the average of all features in that patch.

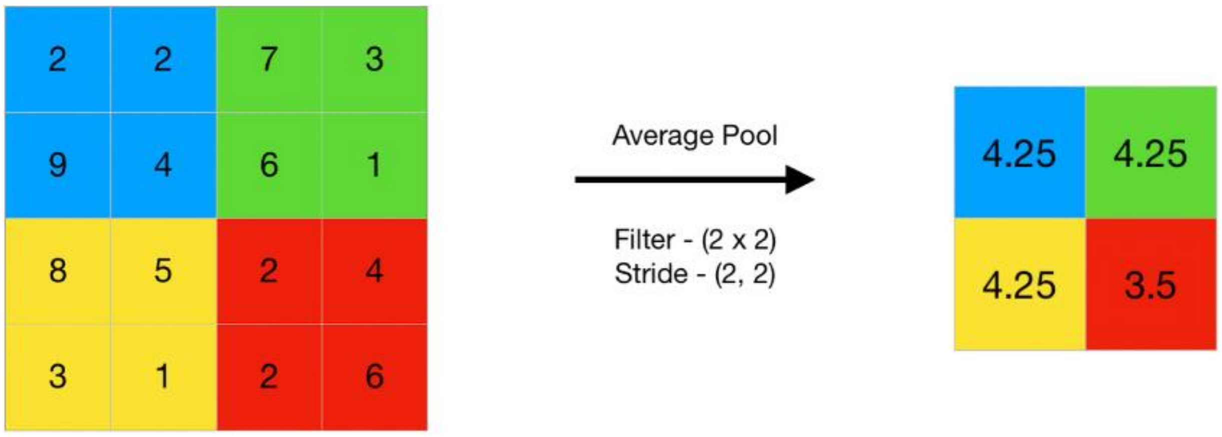
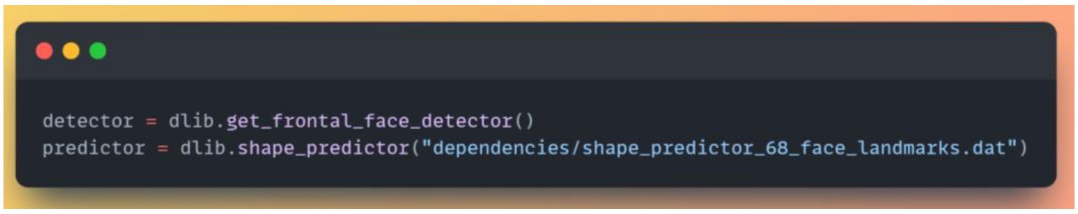


Fig.17 - Average Pooling Layer

3.4.2 STRESS LEVEL PREDICTION

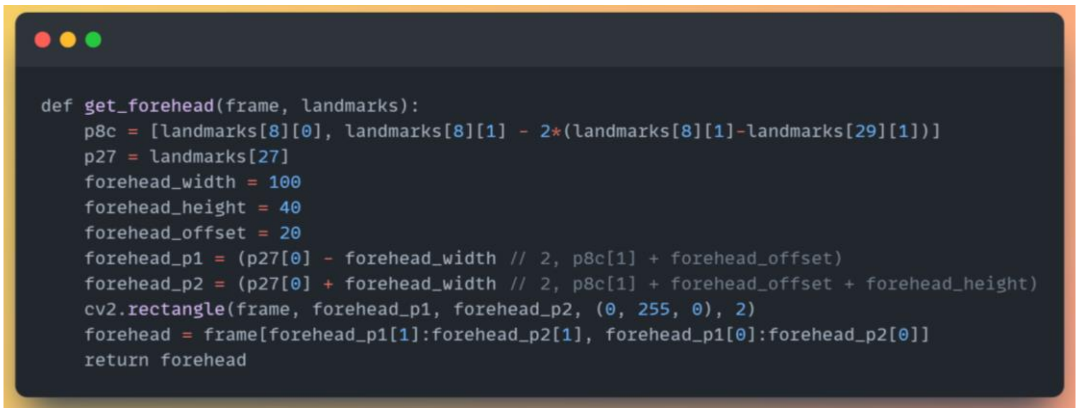
For this project, we have predicted the stress levels using facial landmarks.

- Facial landmarks are detected using the dlib library, especially the shape predictor trained on the 68 facial landmarks dataset.
- Landmarks such as forehead region and distance between the eyebrows.



```
detector = dlib.get_frontal_face_detector()
predictor = dlib.shape_predictor("dependencies/shape_predictor_68_face_landmarks.dat")
```

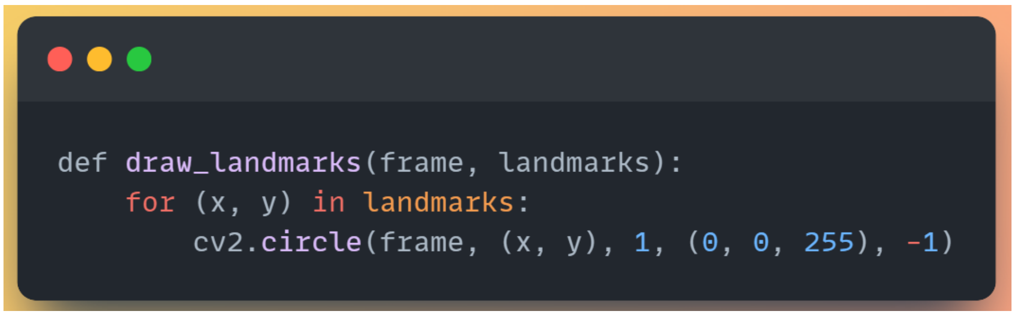
Fig.18 - Loading of the detector and shape predictor



```
def get_forehead(frame, landmarks):
    p8c = [landmarks[8][0], landmarks[8][1] - 2*(landmarks[8][1]-landmarks[29][1])]
    p27 = landmarks[27]
    forehead_width = 100
    forehead_height = 40
    forehead_offset = 20
    forehead_p1 = (p27[0] - forehead_width // 2, p8c[1] + forehead_offset)
    forehead_p2 = (p27[0] + forehead_width // 2, p8c[1] + forehead_offset + forehead_height)
    cv2.rectangle(frame, forehead_p1, forehead_p2, (0, 255, 0), 2)
    forehead = frame[forehead_p1[1]:forehead_p2[1], forehead_p1[0]:forehead_p2[0]]
    return forehead
```

Fig.19 - Getting the coordinates of the forehead

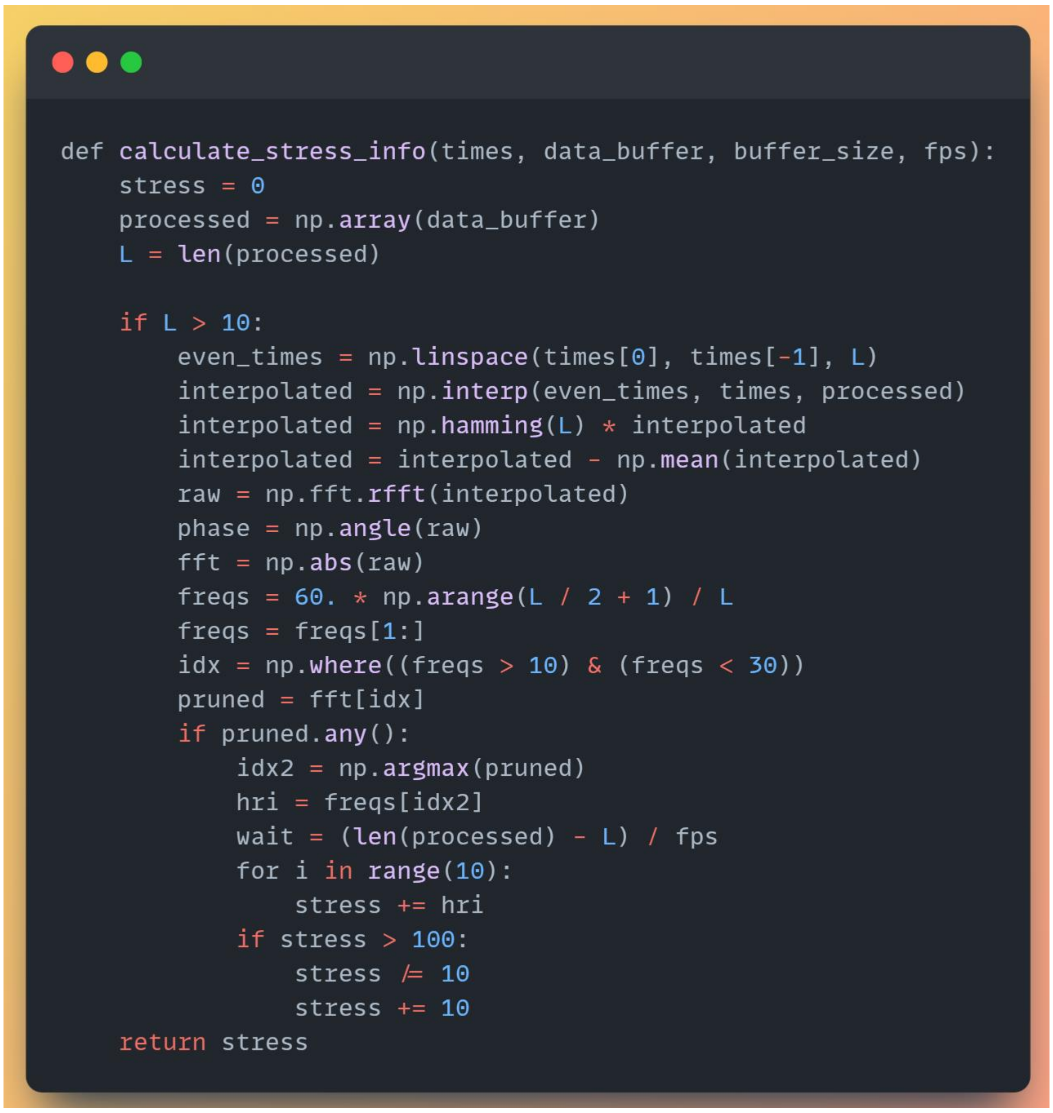
In the above code snippet, we have given frames and landmarks as parameters and using that we are getting the top left and bottom right corner for the forehead. After calculating the coordinates, we are slicing the frame according to that and finally returning that sliced frame.



```
def draw_landmarks(frame, landmarks):
    for (x, y) in landmarks:
        cv2.circle(frame, (x, y), 1, (0, 0, 255), -1)
```

Fig.20 - Drawing the facial landmarks

In the above code snippet, we have drawn the facial landmarks on the current frame.



```python
def calculate_stress_info(times, data_buffer, buffer_size, fps):
    stress = 0
    processed = np.array(data_buffer)
    L = len(processed)

    if L > 10:
        even_times = np.linspace(times[0], times[-1], L)
        interpolated = np.interp(even_times, times, processed)
        interpolated = np.hamming(L) * interpolated
        interpolated = interpolated - np.mean(interpolated)
        raw = np.fft.rfft(interpolated)
        phase = np.angle(raw)
        fft = np.abs(raw)
        freqs = 60. * np.arange(L / 2 + 1) / L
        freqs = freqs[1:]
        idx = np.where((freqs > 10) & (freqs < 30))
        pruned = fft[idx]
        if pruned.any():
            idx2 = np.argmax(pruned)
            hri = freqs[idx2]
            wait = (len(processed) - L) / fps
            for i in range(10):
                stress += hri
            if stress > 100:
                stress /= 10
                stress += 10
    return stress
```

Fig.21 - Calculating the Stress levels

In this code snippet we are calculating the stress levels using the facial landmarks and the distance between the eyebrows.

3.4.3 IMPLEMENTATION OF WEB APPLICATION

To provide a user-friendly interface for stress prediction and emotion classification, we integrated both the models into a web application using Flask, a micro web framework for Python. With this integration, a user can easily visualize the real time stress levels and also detect the emotions through their webcam feed.

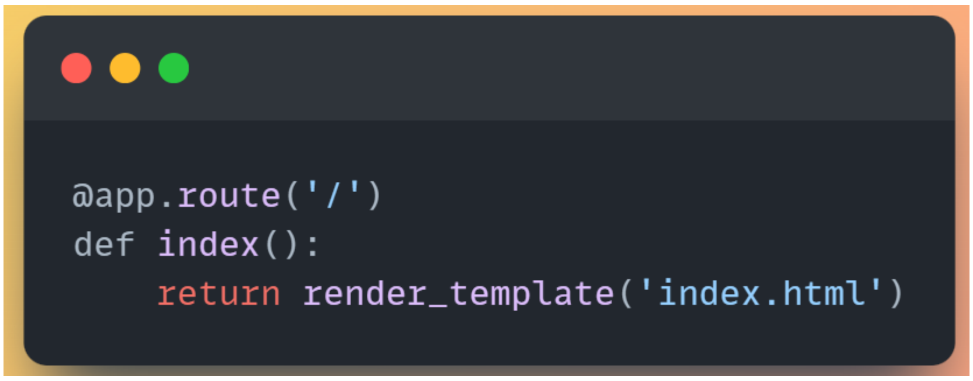
Flask

Flask is a Python micro-web framework for creating web applications. Flask is classified as a microframework because it does not require specific tools or libraries and does not include

features such as database abstraction, form validation, or authentication. Flask, on the other hand, allows you to add application features like object-relational mappers, form validation, and upload handling through extensions.
Flask is ideal for creating REST APIs due to its flexibility and simplicity. It's also a more accessible framework for new developers, as web applications can be built quickly with just one Python file.
Some major companies that use Flask include Airbnb, Lyft, MIT, Netflix, Reddit, Mozilla, Uber, Red Hat, Rackspace, Samsung, Patreon, Mailgun, NGINX, 2market, B2W, and Sieve.

Using Flask, we have created various Routes which are listed below

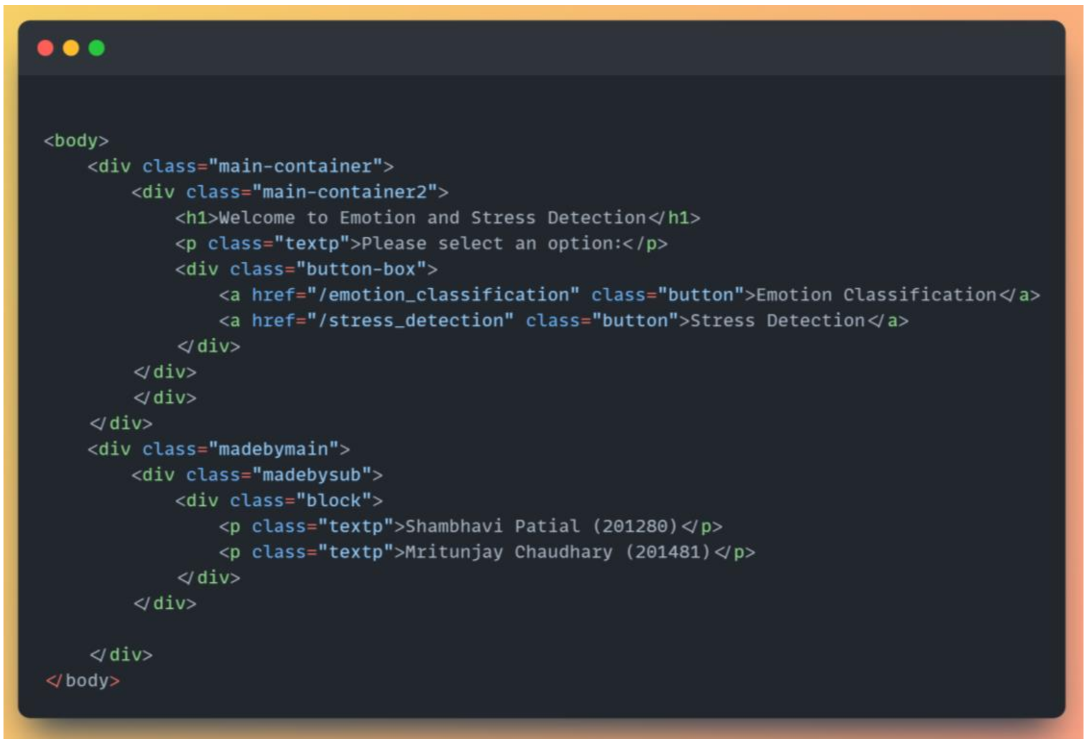
- Default Route for Flask



```
@app.route('/')
def index():
    return render_template('index.html')
```

Fig 22 - Default Route for flask

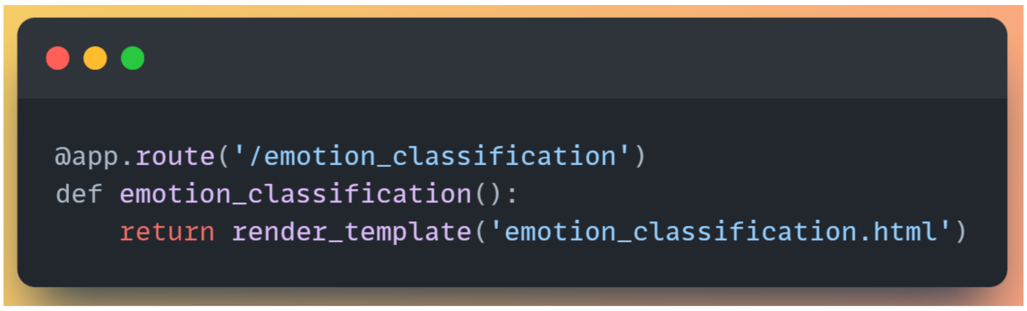
In this code snippet, a default route for the flask application is given which basically renders the index.html file on the webpage, when the flask application is run.



```html
<body>
    <div class="main-container">
        <div class="main-container2">
            <h1>Welcome to Emotion and Stress Detection</h1>
            <p class="textp">Please select an option:</p>
            <div class="button-box">
                <a href="/emotion_classification" class="button">Emotion Classification</a>
                <a href="/stress_detection" class="button">Stress Detection</a>
            </div>
        </div>
        </div>
    </div>
    <div class="madebymain">
        <div class="madebysub">
            <div class="block">
                <p class="textp">Shambhavi Patial (201280)</p>
                <p class="textp">Mritunjay Chaudhary (201481)</p>
            </div>
        </div>

    </div>
</body>
```

Fig 23- HTML code for index.html

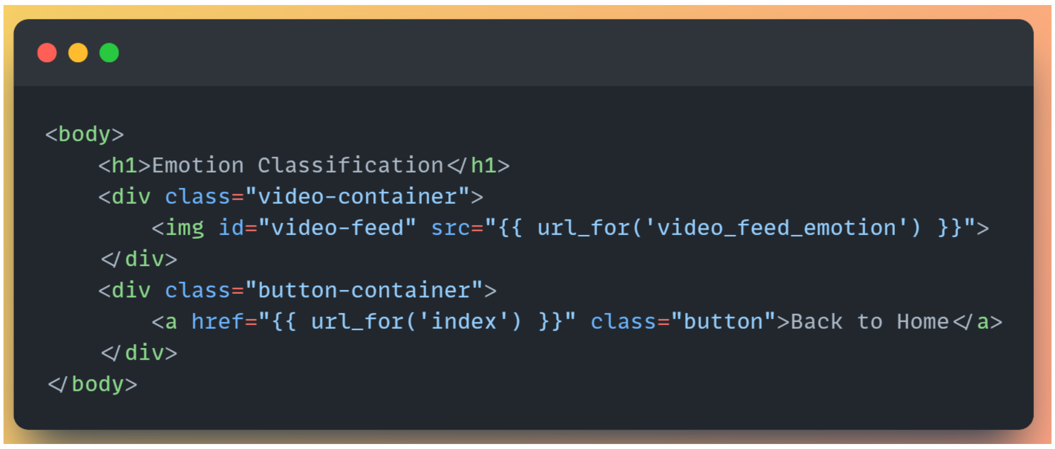
- Route for Emotion Classification



```python
@app.route('/emotion_classification')
def emotion_classification():
    return render_template('emotion_classification.html')
```

Fig.24 - Route for the Emotion Classification

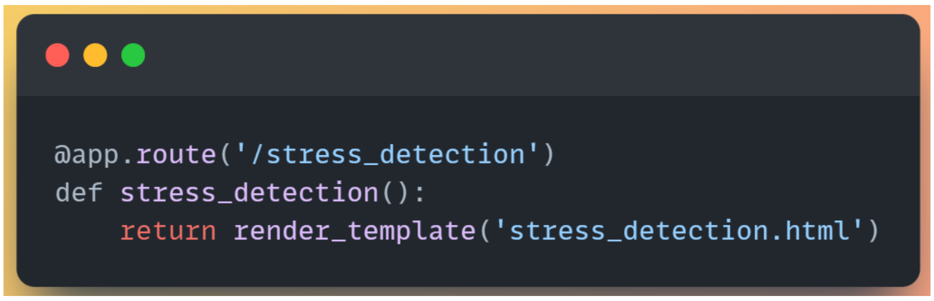
In this code, when the user selects for the emotion classification, the flask application is routed to this route, where it renders the emotion_classification.html.



```
<body>
    <h1>Emotion Classification</h1>
    <div class="video-container">
        <img id="video-feed" src="{{ url_for('video_feed_emotion') }}">
    </div>
    <div class="button-container">
        <a href="{{ url_for('index') }}" class="button">Back to Home</a>
    </div>
</body>
```

Fig.25 - HTML code for emotion_classification.html

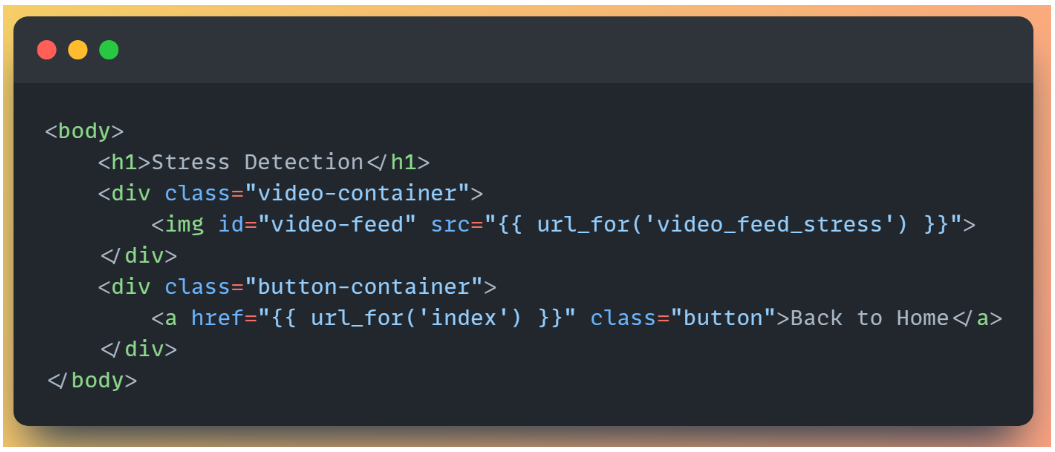
- Route for Stress Level Detection



```
@app.route('/stress_detection')
def stress_detection():
    return render_template('stress_detection.html')
```

Fig.26 - Route for the detection of stress level

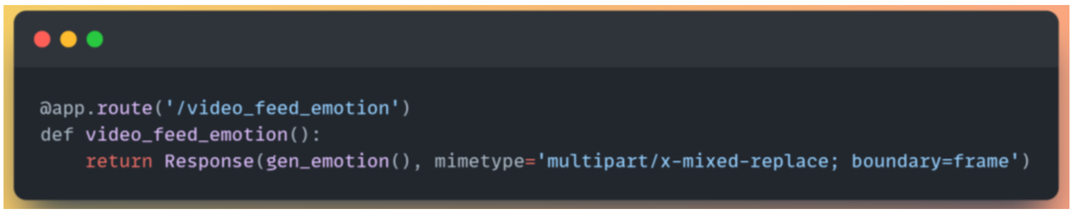
In this code, when the user selects for the Stress detection, the flask application is routed to this route, where it renders the stress_detection.html.



```
<body>
    <h1>Stress Detection</h1>
    <div class="video-container">
        <img id="video-feed" src="{{ url_for('video_feed_stress') }}">
    </div>
    <div class="button-container">
        <a href="{{ url_for('index') }}" class="button">Back to Index</a>
    </div>
</body>
```

Fig.27 - HTML code for stress_detection.html

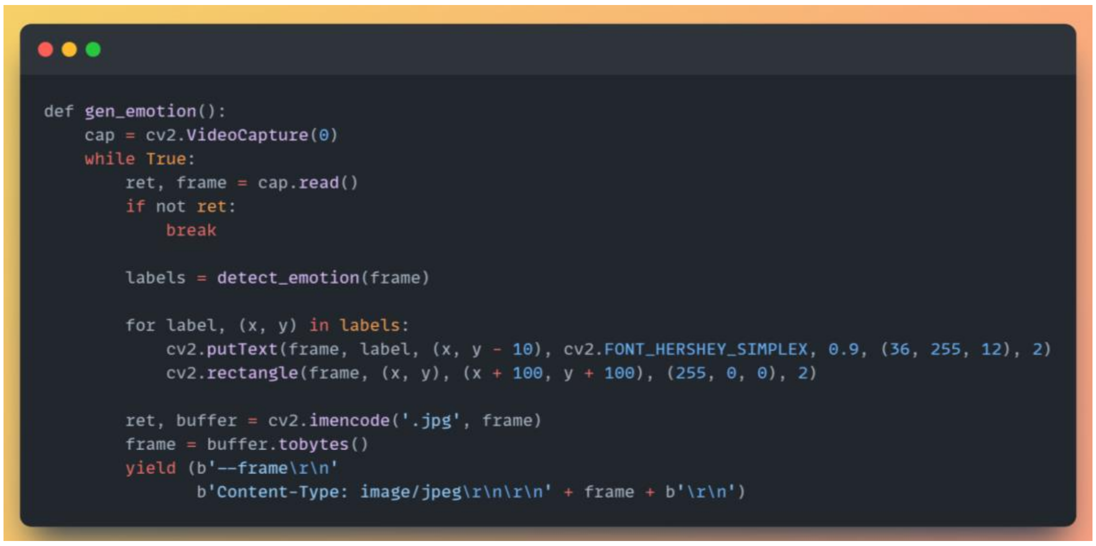
- Route for video feed in Emotion Classification



```python
@app.route('/video_feed_emotion')
def video_feed_emotion():
    return Response(gen_emotion(), mimetype='multipart/x-mixed-replace; boundary=frame')
```

Fig.28 - Route for video feed in Emotion Classification

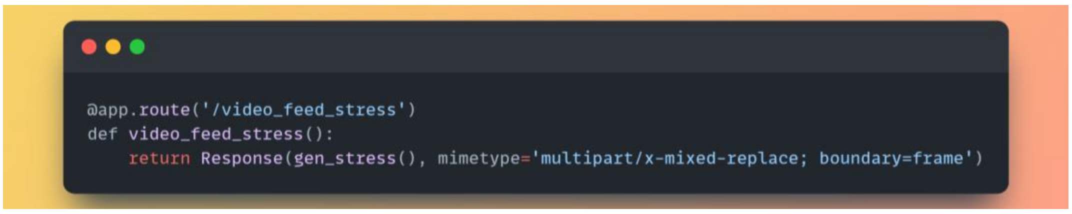
This route streams a video feed of the user's webcam, augmented with overlays indicating the detected emotions. With the help of this user can get to know about his/her emotions in real time.



```python
def gen_emotion():
    cap = cv2.VideoCapture(0)
    while True:
        ret, frame = cap.read()
        if not ret:
            break

        labels = detect_emotion(frame)

        for label, (x, y) in labels:
            cv2.putText(frame, label, (x, y - 10), cv2.FONT_HERSHEY_SIMPLEX, 0.9, (36, 255, 12), 2)
            cv2.rectangle(frame, (x, y), (x + 100, y + 100), (255, 0, 0), 2)

        ret, buffer = cv2.imencode('.jpg', frame)
        frame = buffer.tobytes()
        yield (b'--frame\r\n'
               b'Content-Type: image/jpeg\r\n\r\n' + frame + b'\r\n')
```

Fig.29 - Video Streaming for Emotion Classification

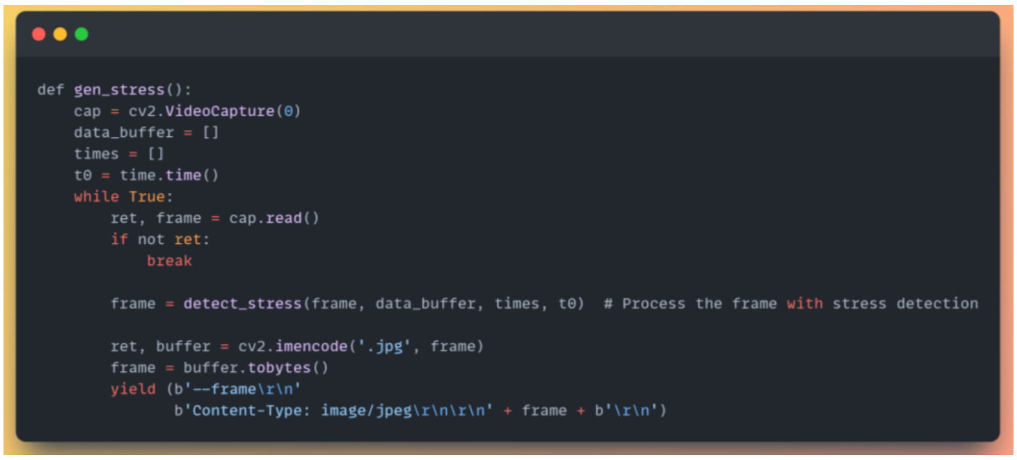
- Route for video feed in Stress Level Detection



```python
@app.route('/video_feed_stress')
def video_feed_stress():
    return Response(gen_stress(), mimetype='multipart/x-mixed-replace; boundary=frame')
```

Fig.30 - Route for video feed in Stress Level Detection

This route streams a video feed of the user's webcam, augmented with overlays indicating the real-time stress level. With the help of this user can get to know about his/her stress levels in real time.



```python
def gen_stress():
    cap = cv2.VideoCapture(0)
    data_buffer = []
    times = []
    t0 = time.time()
    while True:
        ret, frame = cap.read()
        if not ret:
            break

        frame = detect_stress(frame, data_buffer, times, t0)  # Process the frame with stress detection

        ret, buffer = cv2.imencode('.jpg', frame)
        frame = buffer.tobytes()
        yield (b'--frame\r\n'
               b'Content-Type: image/jpeg\r\n\r\n' + frame + b'\r\n')
```

Fig.31 - Video Streaming for Stress Level Detection

3.5 KEY CHALLENGES

We came across various challenges while developing this project. These challenges needed to be addressed in order to ensure that our model is accurate and reliable. Let's discuss the various challenges we encountered and the solutions we found for them.

Challenge: Imbalanced dataset

Solution: As the dataset was highly imbalanced i.e. there were many example images for one type of emotion and on the other hand, very few images for another emotion type. We performed data augmentation for this problem by rotating or flipping the same image to increase the examples.

Challenge: Intra-class variation of FER

Solution: There was a lot of variation between images of the same class. Some of them were even wrong or were not even photos of people. To handle this problem we made our model robust by ensuring that there is no overfitting. Methods like early stopping or using optimizers were performed.

Challenge: Variation in images due to brightness and saturation

Solution: There can be a lot of variation due to brightness and saturation even if there are the same images. Due to difference in brightness or saturation the same emotion can be classified as a completely different emotion. To solve this problem we converted the images to grayscale to remove any variation between images due to brightness and saturation.

Challenge: Deciding parameters to calculate stress level

Solution: One of the problems that we ran into was to decide which parameters should be used to calculate stress levels such that they can be easily implemented and also provide an accurate result. After consulting various research papers, the main focus was put on extracting various facial landmarks and most importantly coordinates of the eyebrows. Difference between the distance of the coordinates of left and right eyebrows was calculated which was used as one of the best parameters to calculate stress levels (in terms of ease of implementation and accuracy).

Challenge: High computation power required

Solution: CNNs require a lot of computational power to get executed. As the complexity of the CNN is increased and/or the number of epochs, the required execution time is also increased. High end systems can execute these networks in a reasonable amount of time.

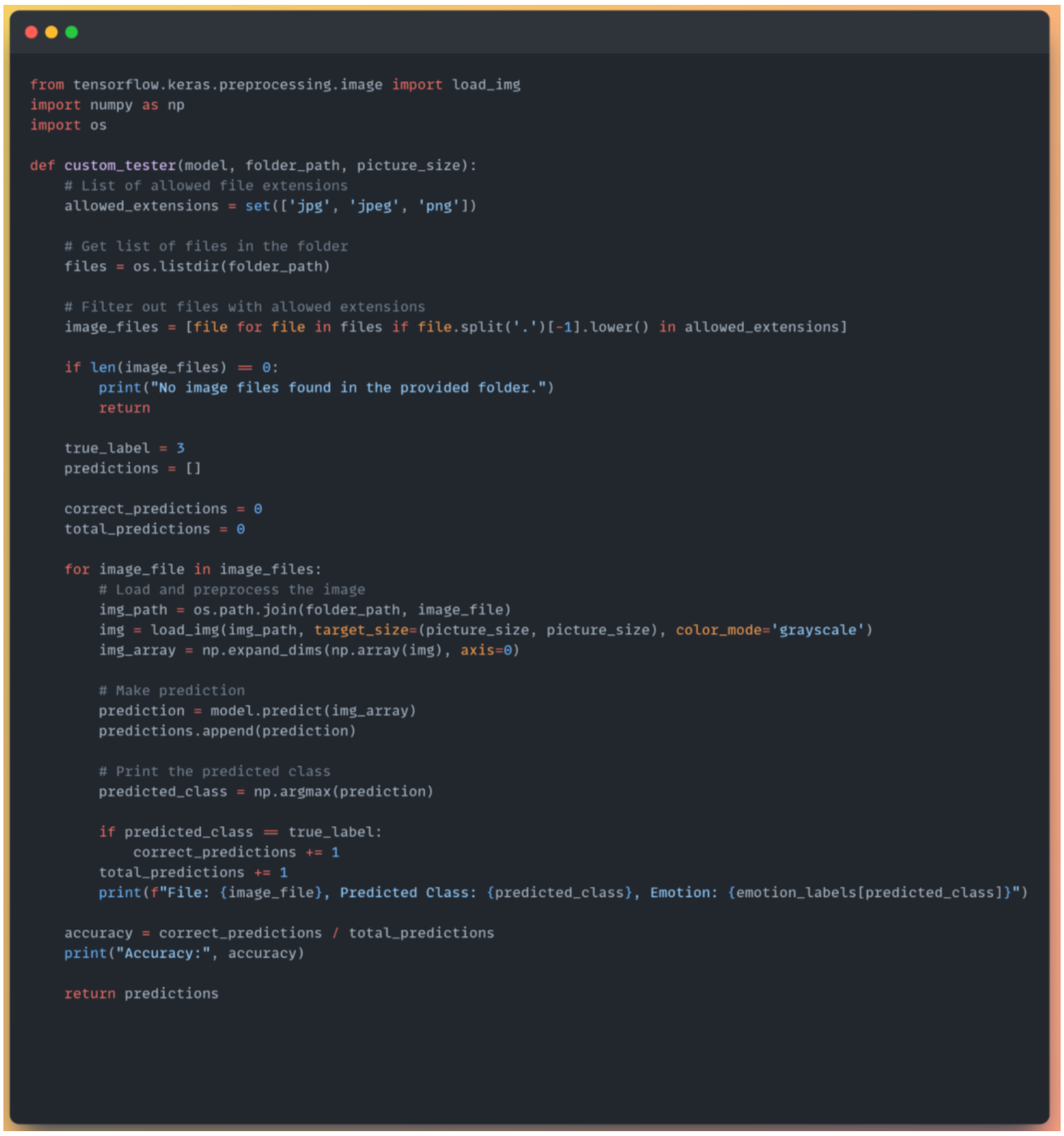
Challenge: The model might perform well on the training data but fail to generalize to new, unseen data.

Solution: Regularly update and retrain the model with new data to improve its generalization capabilities. Use cross-validation techniques and test the model on diverse datasets to ensure it performs well in different scenarios.

CHAPTER 4: TESTING STRATEGY

4.1 TESTING STRATEGY

After building and training our CNN model we now proceed to testing it on all the available emotions.

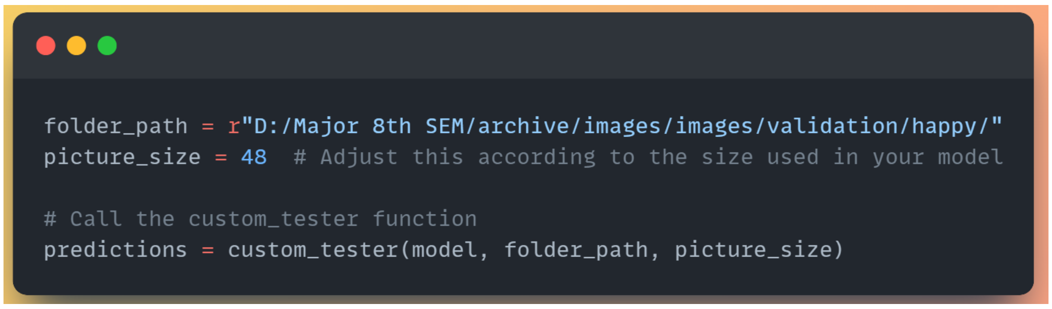


```python
from tensorflow.keras.preprocessing.image import load_img
import numpy as np
import os

def custom_tester(model, folder_path, picture_size):
    # List of allowed file extensions
    allowed_extensions = set(['jpg', 'jpeg', 'png'])

    # Get list of files in the folder
    files = os.listdir(folder_path)

    # Filter out files with allowed extensions
    image_files = [file for file in files if file.split('.')[-1].lower() in allowed_extensions]

    if len(image_files) == 0:
        print("No image files found in the provided folder.")
        return

    true_label = 3
    predictions = []

    correct_predictions = 0
    total_predictions = 0

    for image_file in image_files:
        # Load and preprocess the image
        img_path = os.path.join(folder_path, image_file)
        img = load_img(img_path, target_size=(picture_size, picture_size), color_mode='grayscale')
        img_array = np.expand_dims(np.array(img), axis=0)

        # Make prediction
        prediction = model.predict(img_array)
        predictions.append(prediction)

        # Print the predicted class
        predicted_class = np.argmax(prediction)

        if predicted_class == true_label:
            correct_predictions += 1
        total_predictions += 1
        print(f"File: {image_file}, Predicted Class: {predicted_class}, Emotion: {emotion_labels[predicted_class]}")

    accuracy = correct_predictions / total_predictions
    print("Accuracy:", accuracy)

    return predictions
```

Fig.32- Custom function to get accuracy for each emotions

In this code snippet, we have made a custom function which will calculate the accuracy of a particular emotion.

4.1.1 TESTING FOR HAPPY

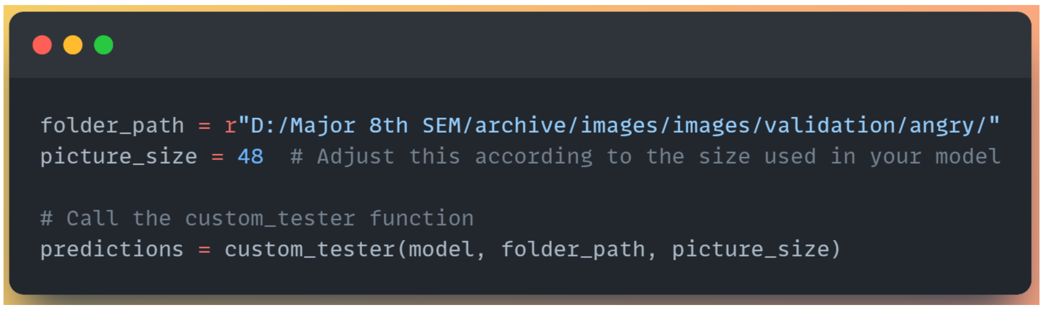


```python
folder_path = r"D:/Major 8th SEM/archive/images/images/validation/happy/"
picture_size = 48  # Adjust this according to the size used in your model

# Call the custom_tester function
predictions = custom_tester(model, folder_path, picture_size)
```

Fig.33 - Testing for Happy

In this snippet we are calling that custom function to calculate the accuracy for the happy emotion.

4.1.2 TESTING FOR ANGRY

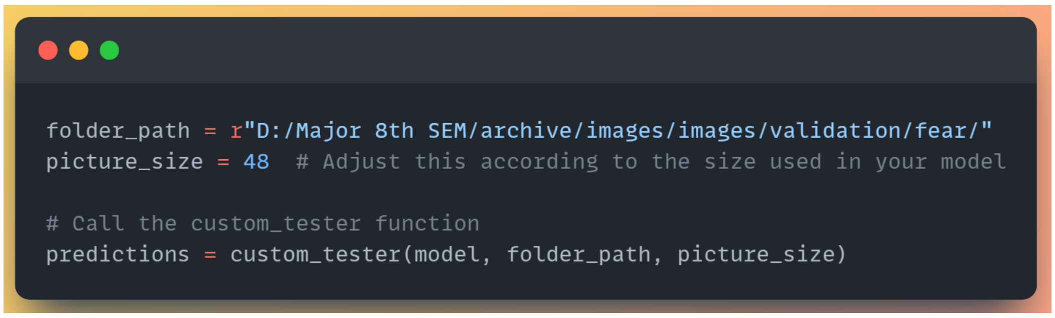


```python
folder_path = r"D:/Major 8th SEM/archive/images/images/validation/angry/"
picture_size = 48  # Adjust this according to the size used in your model

# Call the custom_tester function
predictions = custom_tester(model, folder_path, picture_size)
```

Fig.34 - Testing for Angry

In this snippet we are calling that custom function to calculate the accuracy for the Angry emotion.

4.1.3 TESTING FOR FEAR

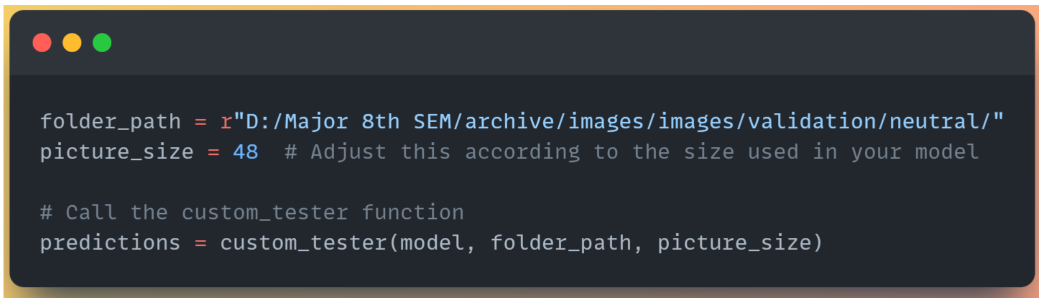


```python
folder_path = r"D:/Major 8th SEM/archive/images/images/validation/fear/"
picture_size = 48  # Adjust this according to the size used in your model

# Call the custom_tester function
predictions = custom_tester(model, folder_path, picture_size)
```

Fig.35 - Testing for Fear

In this snippet we are calling that custom function to calculate the accuracy for the fear emotion.

4.1.4 TESTING FOR NEUTRAL

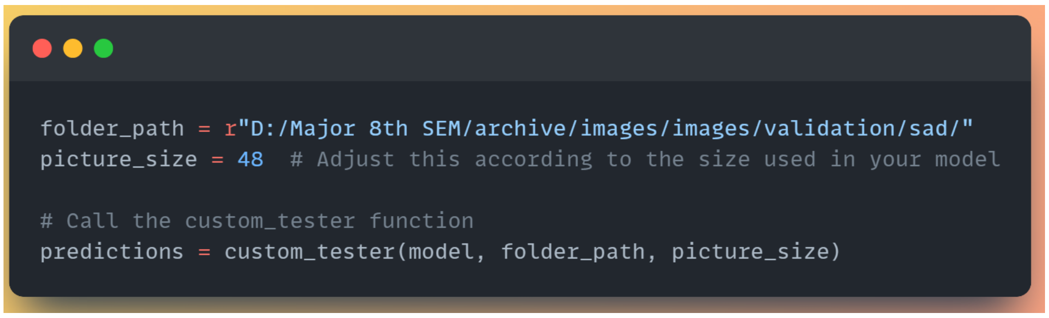


```python
folder_path = r"D:/Major 8th SEM/archive/images/images/validation/neutral/"
picture_size = 48  # Adjust this according to the size used in your model

# Call the custom_tester function
predictions = custom_tester(model, folder_path, picture_size)
```

Fig.36 - Testing for Neutral

In this snippet we are calling that custom function to calculate the accuracy for the neutral emotion.

4.1.5 TESTING FOR SAD

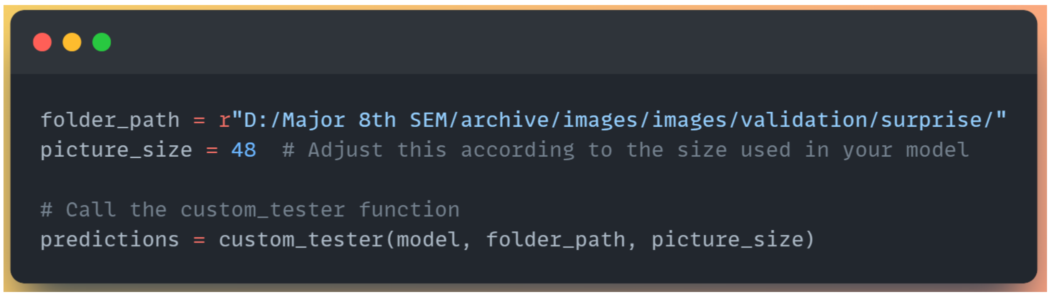


```python
folder_path = r"D:/Major 8th SEM/archive/images/images/validation/sad/"
picture_size = 48  # Adjust this according to the size used in your model

# Call the custom_tester function
predictions = custom_tester(model, folder_path, picture_size)
```

Fig.37 - Testing for Sad

In this snippet we are calling that custom function to calculate the accuracy for the sad emotion.

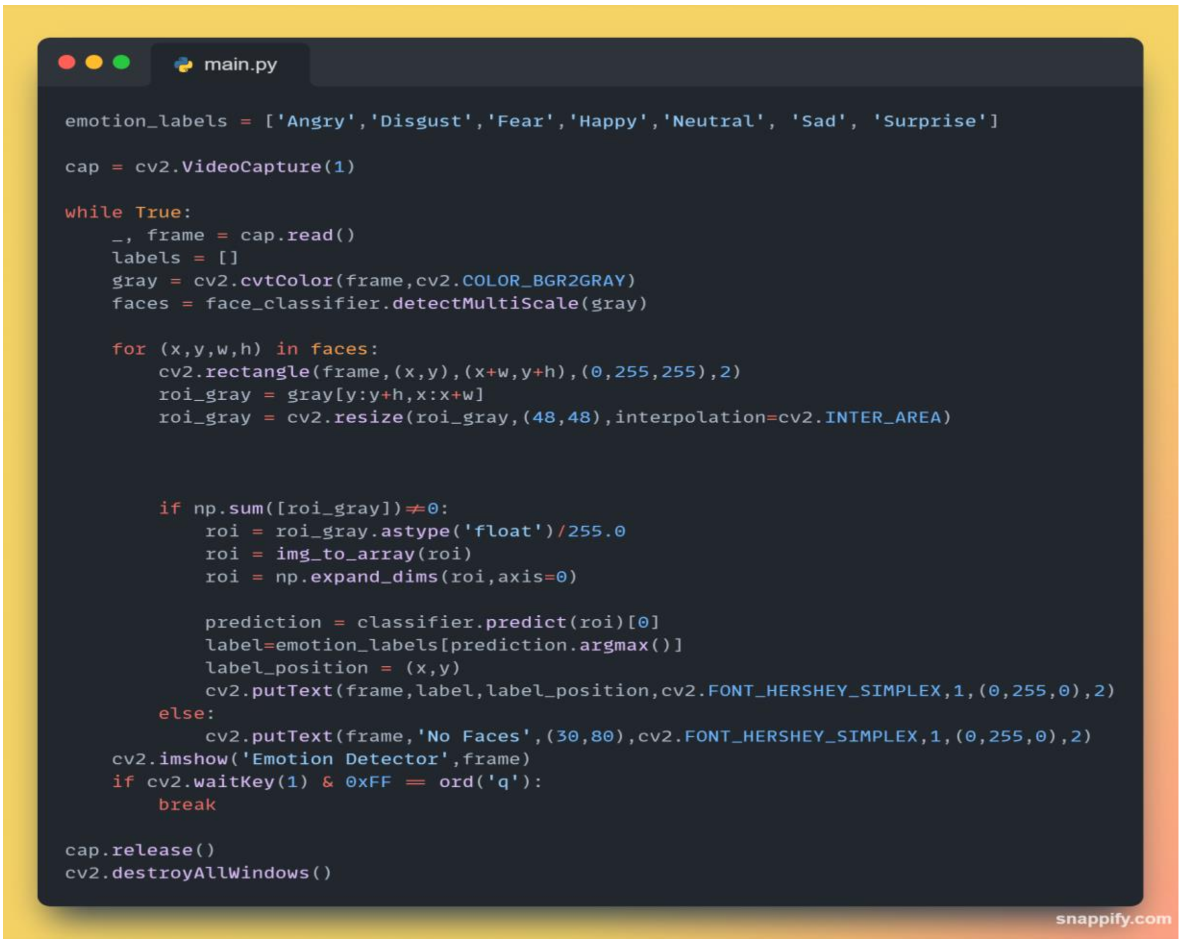
4.1.6 TESTING FOR SURPRISE



```python
folder_path = r"D:/Major 8th SEM/archive/images/images/validation/surprise/"
picture_size = 48  # Adjust this according to the size used in your model

# Call the custom_tester function
predictions = custom_tester(model, folder_path, picture_size)
```

Fig.38 - Testing for Surprise

In this snippet we are calling that custom function to calculate the accuracy for the surprise emotion.



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

cap = cv2.VideoCapture(1)

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()
```

Fig.39 - Testing the CNN model

In this snippet, we are testing the CNN model using the live feed from the webcam. The model will predict the emotion of the person present in the frame. After predicting it will overlay the result on the frame itself.

4.2 TEST CASES AND OUTCOMES

We have also tested our model on some random frames using live feed from a webcam.

4.2.1 EMOTION CLASSIFICATION

For the Emotion classification

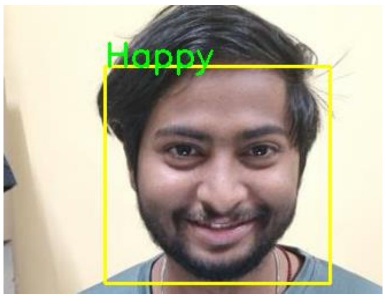


Fig.40 - Prediction for the test subject 1

In the previous output it is seen that the uploaded test image is of a Happy face and our model is correctly predicting the same for it.

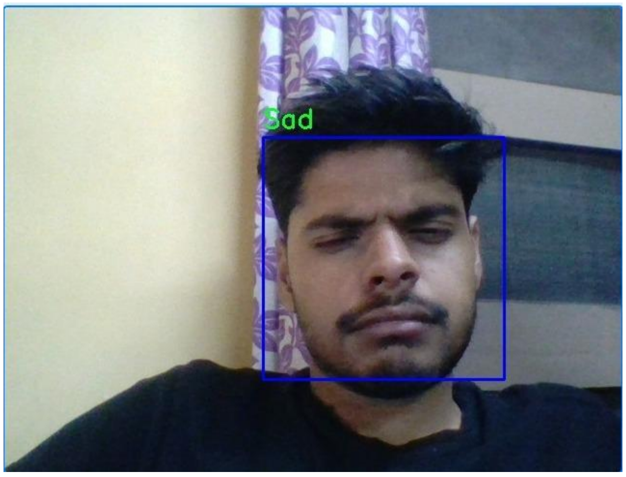


Fig.41 - Prediction for the test subject 2

In this it is seen that the uploaded test image is of a Sad face and our model is correctly predicting the same for it.

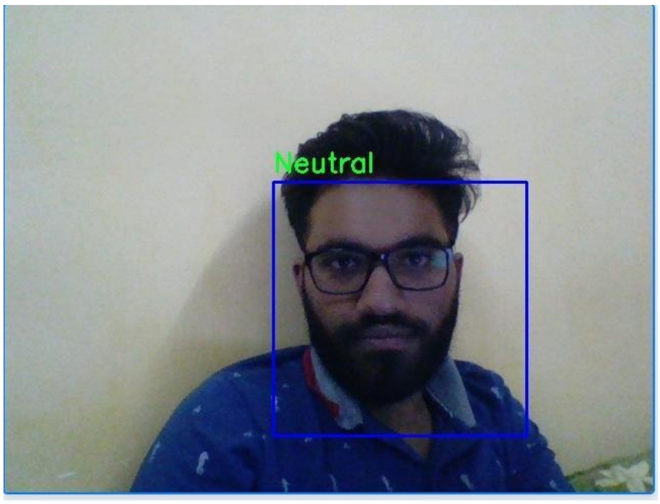


Fig.42 - Prediction for the test subject 3

In this it is seen that the uploaded test image is of a Neutral face and our model is correctly predicting the same for it.

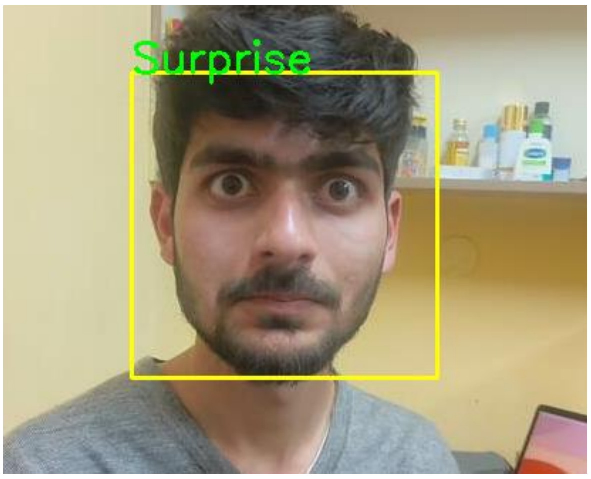


Fig.43 - Prediction for the test subject 4

In this it is seen that the uploaded test image is of a Surprised face and our model is correctly predicting the same for it.

4.2.2 STRESS LEVEL DETECTION

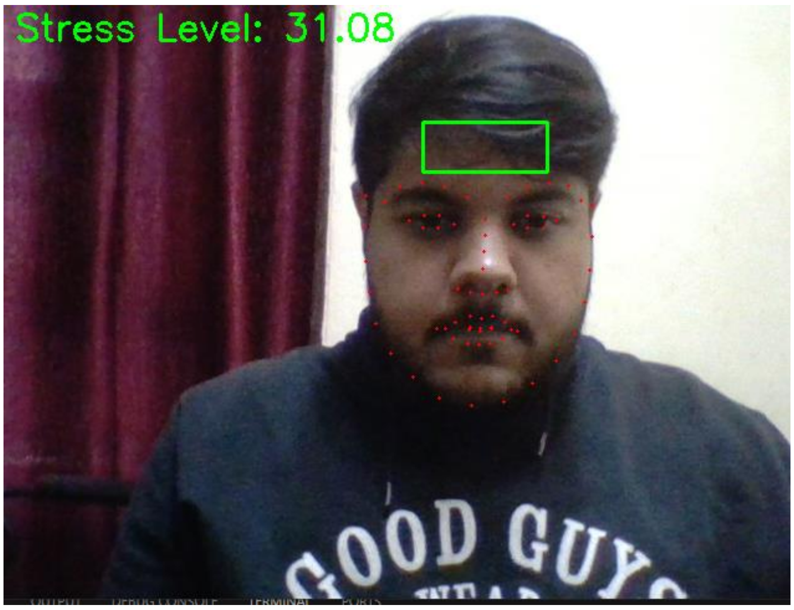


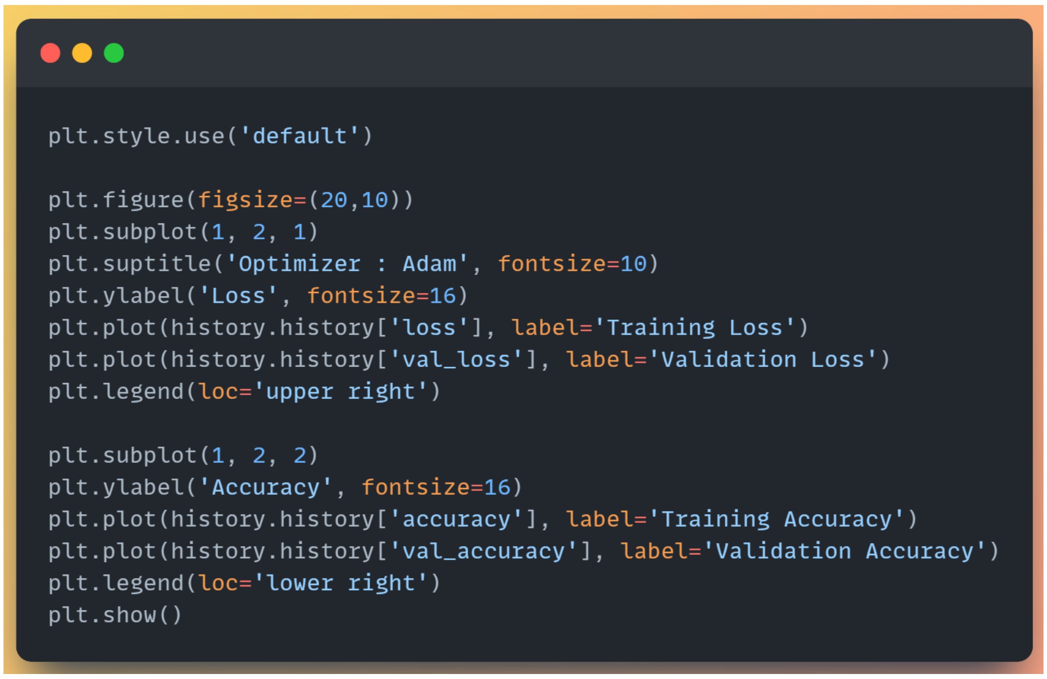
Fig.44 - **Detecting** stress level on test subject 5

In this it is seen that the model is detecting the stress levels from the live feed of the webcam.

CHAPTER 5: RESULTS AND EVALUATION

5.1 RESULTS

After successful training of the model i.e. CNN, now we will be validating it and analyzing their performance and accuracy on testing dataset.



```
plt.style.use('default')

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()
```

Fig.45 - Graphs for model loss and accuracy

In the above code snippet, we are plotting two graphs - for visualizing loss and for visualizing accuracy. These graphs are shown on the next page.

The two graphs that we are plotting are: -

- Model Loss - It shows the model loss over the epochs

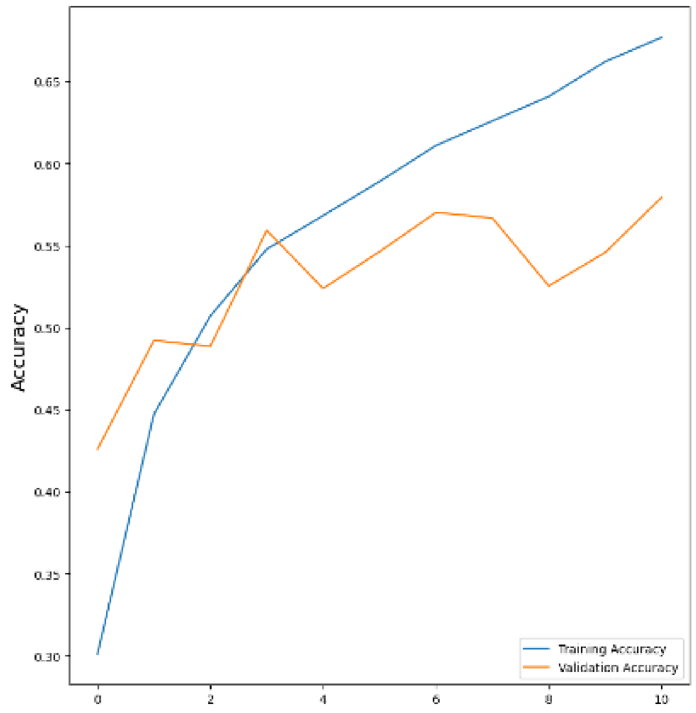


Fig.46 - Model loss vs epoch

- Model accuracy - It shows the model accuracy over the epochs

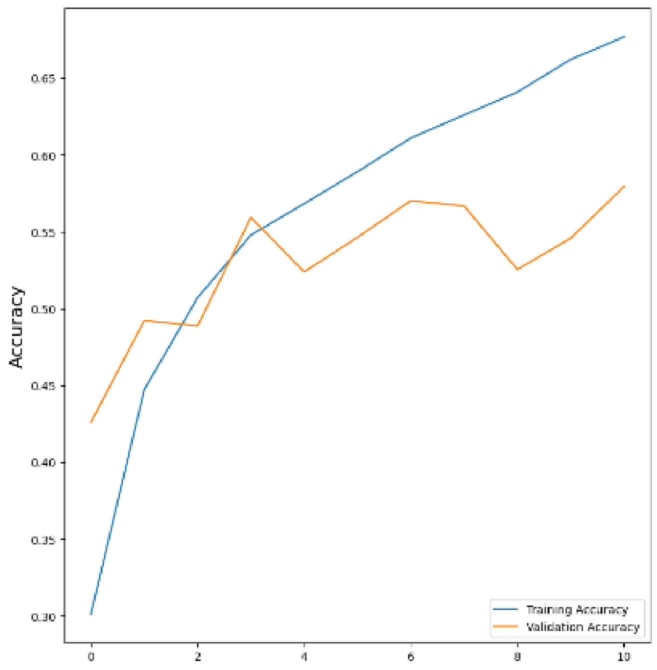


Fig.47 - Model accuracy vs epoch

Model accuracy - It shows the model accuracy over the epochs

- Our functional requirement of having a codebase which is cable of processing the uploaded image and able to utilize it for future use has been fulfilled
- Requirements for the model to be able to predict both indoor and outdoor images have been fulfilled.

Our frontend integration of stress prediction and emotion categorization models into a web interface has given encouraging usability and accessibility outcomes. Users can interact with the application via their web browsers, providing an easy way to track their mental health in real time. The following screenshots depict the user interface of our web application:

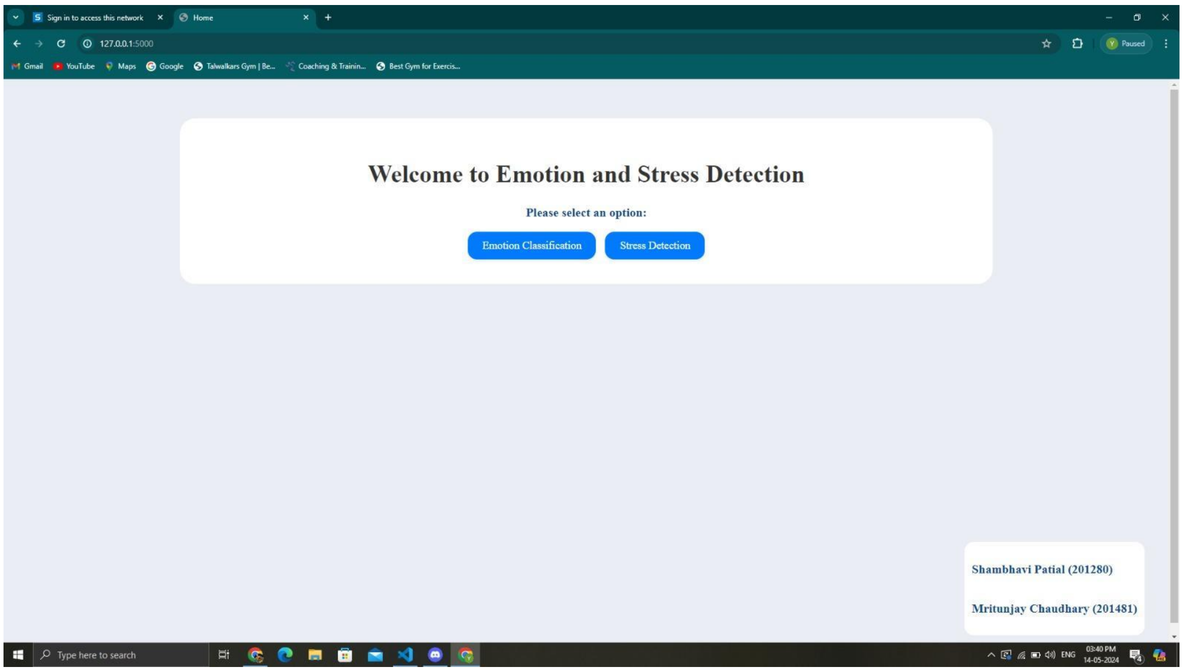

Welcome to Emotion and Stress Detection
Please select an option:
Emotion Classification
Stress Detection
Shambhavi Patial (201280)
Mritunjay Chaudhary (201481)

Fig.48 - Home Page of the our application

Here the user can have the option to choose between stress detection and emotion classification.

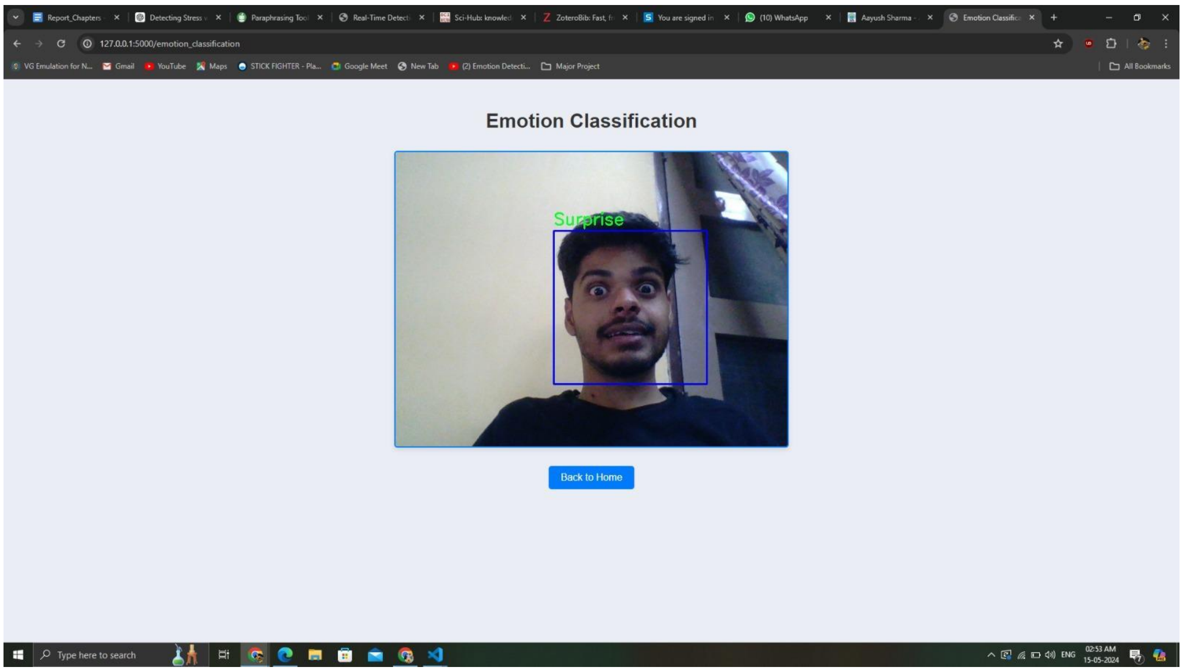


Fig.49 - Emotion Classification Page

The Emotion Classification page shows a webcam feed with real-time emotion classification overlays. Users can see how their facial expressions are analyzed to determine their emotional states, which encourages self-reflection and awareness.

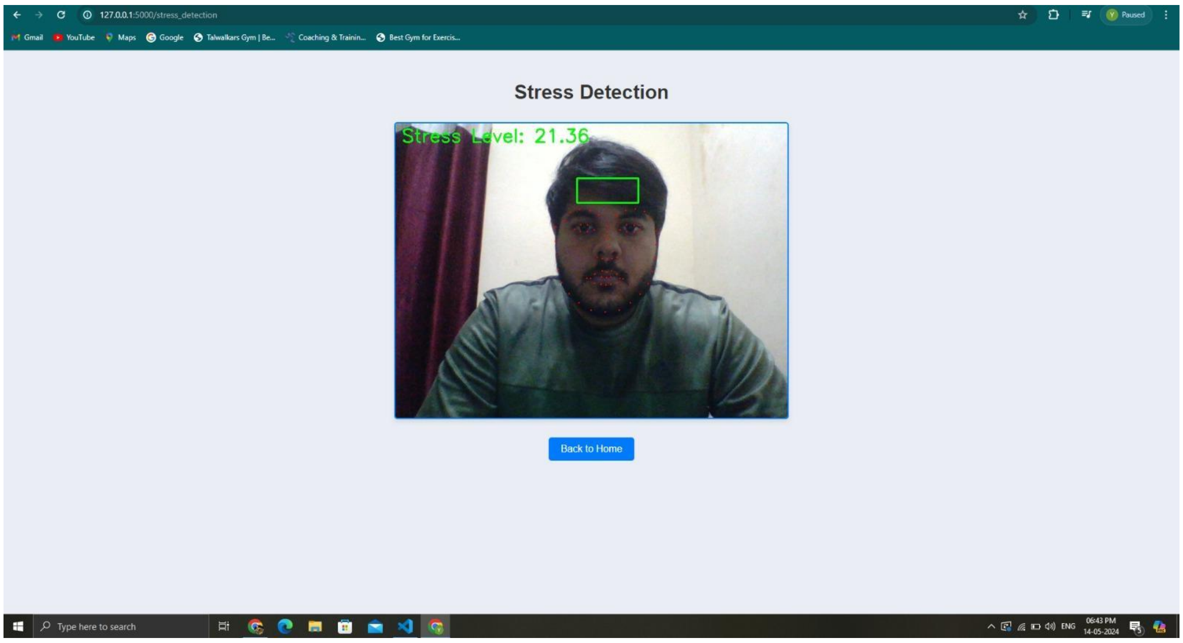


Fig.50 - Stress Level Detection Page [A] - Low Stress

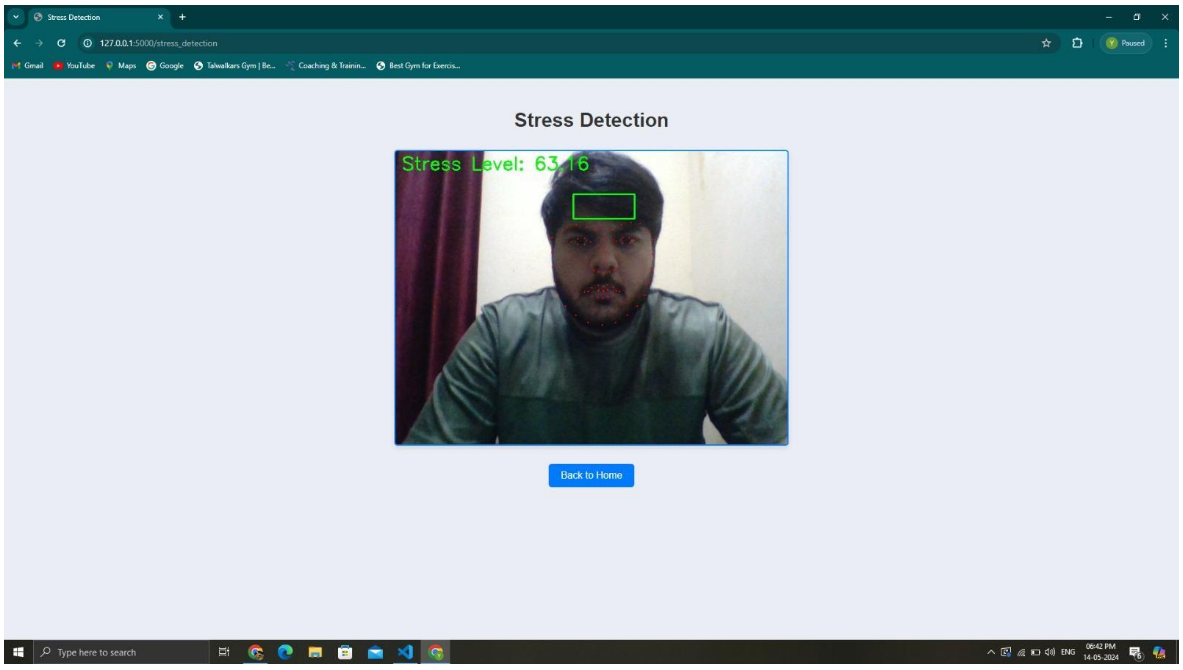


Fig.51 - Stress Level Detection Page [B] - High Stress

The Stress Detection page displays the camera feed alongside real-time stress level indicators. Users can visualize changes in their stress levels and take proactive steps to successfully manage stress, hence enhancing mental health.

Both the emotion classification and stress detection methods performed in real time, ensuring a low latency between recording the webcam feed and delivering the analysis findings.

CHAPTER 6: CONCLUSION AND FUTURE SCOPE

6.1 CONCLUSION

The implementation of CNN in the Stress detection model Prediction has yielded significant results in terms of accuracy, speed and other parameters which are being used to determine the overall performance of our model.

Stress detection can act as a powerful tool in various fields. This web application can also be used to predict the natural reflex and current condition of a person. One of its major applications includes that it can also be used to predict unusual behavior and helps to analyze threats in various situations such as hostage situation and can be used by Police and armed forces in case of lie detector to check if a person is lying or not. In general society tends to ignore stress and consider it as a part of daily life but if a person is suffering from prolonged stress it can lead to many health issues regarding blood pressure, cardiovascular diseases, depression, Anxiety, sleep disorder related issues, deteriorating mental health, and loss of memory.

Our web application is based on detecting various emotions based on some features which will contribute to detect stress levels on an individual user. We have tried our best to get accurate results based on individual characteristics of a person.

Prediction of stress is a significant step towards alarming people regarding various health issues and chronic diseases. Our CNN model is providing precise results based on the pixel information which is being sent through the input layer to other convolutional layers.

6.2 FUTURE WORK

The Future work includes focusing on further refining our existing project for better results Although we have tried our best to give precise results, we will try to improve efficiency and explore other strategies.

In the future we will make a model for precise stress detection using facial landmarks and making a user-friendly graphic user interface in the form of a website or an application for the people in which the user needs to fill his/ her details and integrate our frontend with our backend.

Deployment of our model on the Cloud platform so that it can be accessible by many people.

REFERENCES

[1] D. Mehta, M. F. H. Siddiqui, and A. Y. Javaid, “Recognition of Emotion Intensities Using Machine Learning Algorithms: A Comparative Study,” Sensors (Basel), vol. 19, no. 8, p. 1897, Apr. 2019, doi: 10.3390/s19081897.

[2] J. He, K. Li, X. Liao, P. Zhang, and N. Jiang, “Real-Time Detection of Acute Cognitive Stress Using a Convolutional Neural Network From Electrocardiographic Signal,” IEEE Access, vol. 7, pp. 42710–42717, 2019, doi: 10.1109/ACCESS.2019.2907076.

[3] A. Jaiswal, A. Krishnama Raju and S. Deb, "Facial Emotion Detection Using Deep Learning," 2020 International Conference for Emerging Technology (INCET), Belgaum, India, 2020, pp. 1-5, doi: 10.1109/INCET49848.2020.9154121.,

[4]Mohammad Failzal, Nikhil V, , Prajwal C, Aruna Rao B P, “RECOGNITION of STRESS USING FACE IMAGE and FACIAL LANDMARKS” International Advanced Research Journal in Science, Engineering and Technology Vol. 8, Issue 5, May 2021 DOI: 10.17148/IARJSET.2021.85106

[5] Ms. Dhanashree V. Udgirkar, “Tracking and Detecting Depression Level using Facial Recognition and PEN&IQ Test,” IJARSCT, pp. 97–100, Dec. 2023, doi: 10.48175/IJARSCT-14011.

[6] A. I. Siam, N. F. Soliman, A. D. Algarni, F. E. Abd El-Samie, and A. Sedik, “Deploying Machine Learning Techniques for Human Emotion Detection,” Computational Intelligence and Neuroscience, vol. 2022, p. e8032673, Feb. 2022, doi: 10.1155/2022/8032673.

[7]N. V. Kimmatkar and B. V. Babu, “Novel Approach for Emotion Detection and Stabilizing Mental State by Using Machine Learning Techniques,” Computers, vol. 10, no. 3, p. 37, Mar. 2021, doi: 10.3390/computers10030037.

[8] Y. Khaireddin and Z. Chen, “Facial Emotion Recognition: State of the Art Performance on FER2013.” arXiv, May 08, 2021. doi: 10.48550/arXiv.2105.03588.

[9] P. R. Naidu, S. P. Sagar, K. Praveen, K. Kiran, and K. Khalandar, “Stress Recognition Using Facial Landmarks and Cnn (Alexnet),” J. Phys.: Conf. Ser., vol. 2089, no. 1, p. 012039, Nov. 2021, doi: 10.1088/1742-6596/2089/1/012039

[10] J. Almeida and F. Rodrigues, “Facial Expression Recognition System for Stress Detection with Deep Learning:,” in Proceedings of the 23rd International Conference on Enterprise Information Systems, Online Streaming: SCITEPRESS - Science and Technology Publications, 2021, pp. 256–263. doi: 10.5220/0010474202560263.

[11] G. Giannakakis et al., “Stress and anxiety detection using facial cues from videos,” Biomedical Signal Processing and Control, vol. 31, pp. 89–101, Jan. 2017, doi: 10.1016/j.bspc.2016.06.020.

[12] Zhang, J., Mei, X., Liu, H., Yuan, S., & Qian, T. (2019). Detecting Negative Emotional Stress Based on Facial Expression in Real Time. 2019 IEEE 4th International Conference on Signal and Image Processing (ICSIP). doi:10.1109/siprocess.2019.886873

[13] L. Vaiani, M. La Quatra, L. Cagliero, and P. Garza, “ViPER: Video-based Perceiver for Emotion Recognition,” in Proceedings of the 3rd International on Multimodal Sentiment Analysis Workshop and Challenge, in MuSe’ 22. New York, NY, USA: Association for Computing Machinery, Oct. 2022, pp. 67–73. doi: 10.1145/3551876.3554806.

[14] *Keras 3 API documentation*. (n.d.). Keras. Retrieved from https://keras.io/api/

[15] *OpenCV: OpenCV modules*. (n.d.). OpenCV Documentation. Retrieved from https://docs.opencv.org/4.x/index.html

[16] M. Pediaditis et al., “Extraction of facial features as indicators of stress and anxiety,” in 2015 37th Annual International Conference of the IEEE Engineering in Medicine and Biology Society (EMBC), Aug. 2015, pp. 3711–3714. doi: 10.1109/EMBC.2015.7319199.

[17] J. A. Harrigan and D. M. OConnell, "How do you look when feeling anxious? Facial displays of anxiety", Personality and Individual Differences, vol. 21, no. 2, pp. 205-212, 1996.

[18] M. Hamilton, "The assessment of anxiety-states by rating", British Journal of Medical Psychology, vol. 32, no. 1, pp. 50-55, 1959.

[19] *Understanding Deep Convolutional Neural Networks*. (n.d.). Retrieved from run ai: https://www.run.ai/guides/deep-learning-for-computer-vision/deep-convolutional-neural-networks

[21] Webb, S. (2020, December 18). *k-fold cross-validation explained in plain English*. Towards Data Science. Retrieved from https://towards datascience.com/k-fold-cross-validation-explained-in-plain-english-659e33c0bc0

[22] *Callbacks in Keras*. (n.d.). Scaler. Retrieved from https://www.scaler.com/topics/keras/callbacks-in-keras/

[23] *pandas documentation — pandas 2.1.3 documentation*. (n.d.). Pandas. Retrieved from https://pandas.pydata.org/docs/

[24] *FER-2013*. (n.d.). Kaggle. Retrieved from https://www.kaggle.com/datasets/msambare/fer2013

[25] *RAVDESS Emotional speech audio*. (n.d.). Kaggle. Retrieved from https://www.kaggle.com/datasets/uwrfkaggler/ravdess-emotional-speech-audio

[26] *Optimizers in Deep Learning*. (n.d.). Scaler. Retrieved from https://www.scaler.com/topics/deep-learning/optimizers-in-deep-learning/

JAYPEE UNIVERSITY OF INFORMATION TECHNOLOGY, WAKNAGHAT

PLAGIARISM VERIFICATION REPORT

Date: ……………………….

Type of Document (Tick): **PhD Thesis** **M.Tech Dissertation/ Report** **B.Tech Project Report** **Paper**

Name: ______________________________**Department:** _________________**Enrolment No** _________

Contact No. ______________________________**E-mail.** _______________________________________

Name of the Supervisor: ____

Title of the Thesis/Dissertation/Project Report/Paper (In Capital letters): _________________________

____

____

UNDERTAKING

I undertake that I am aware of the plagiarism related norms/ regulations, if I found guilty of any plagiarism and copyright violations in the above thesis/report even after award of degree, the University reserves the rights to withdraw/revoke my degree/report. Kindly allow me to avail Plagiarism verification report for the document mentioned above.

Complete Thesis/Report Pages Detail:

– Total No. of Pages =
– Total No. of Preliminary pages =
– Total No. of pages accommodate bibliography/references =

(Signature of Student)

FOR DEPARTMENT USE

We have checked the thesis/report as per norms and found **Similarity Index** at.................. (%). Therefore, we
are forwarding the complete thesis/report for final plagiarism check. The plagiarism verification report may be handed over to the candidate.

(Signature of Guide/Supervisor) **Signature of HOD**

FOR LRC USE

The above document was scanned for plagiarism check. The outcome of the same is reported below:

Copy Received on	**Excluded**	**Similarity Index (%)**	**Generated Plagiarism Report Details** (Title, Abstract & Chapters)	
	• All Preliminary Pages • Bibliography/Images/Quotes • 14 Words String		Word Counts	
Report Generated on			Character Counts	
		Submission ID	Total Pages Scanned	
			File Size	

Checked by
Name & Signature **Librarian**

………

Please send your complete thesis/report in (PDF) with Title Page, Abstract and Chapters in (Word File) through the supervisor at plagcheck.juit@gmail.com

Mritunjay

ORIGINALITY REPORT

14%	11%	8%	8%
SIMILARITY INDEX	INTERNET SOURCES	PUBLICATIONS	STUDENT PAPERS

PRIMARY SOURCES

1	Submitted to University of Strathclyde Student Paper	2%
2	dspace.library.uvic.ca Internet Source	1%
3	Submitted to Liverpool John Moores University Student Paper	1%
4	joiv.org Internet Source	1%
5	www.kdnuggets.com Internet Source	1%
6	Submitted to National Institute of Technology Jamshedpur Student Paper	1%
7	www.iaeng.org Internet Source	1%
8	www.nature.com Internet Source	1%