FACIAL EXPRESSION AND FACE MASK DETECTION

A Mini Project Report submitted to

Jawaharlal Nehru Technological University, Hyderabad

In partial fulfillment for the requirement for the award of B.Tech Degree in Computer Science and Engineering

By

CHIRUMALLA KAMAL	**Roll. No 18641A05E2**
EDELLI SRIKANTH	**Roll. No 18641A05D4**
GOURISHETTY CHANDHAN	**Roll. No 18641A05D8**
KUDIKALA ROHITH	**Roll. No 18641A05E7**
KETHAM SAISANTHOSH	**Roll. No 18641A05E4**

Under the Guidance of

Mrs. B. Saritha

Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

VAAGDEVI COLLEGE OF ENGINEERING

(UGC Autonomous, Accredited by NBA, Accredited by NAAC with "A")

Bollikunta, Khila Warangal (Mandal), Warangal Urban-506 005 (T.S)

VAAGDEVI COLLEGE OF ENGINEERING

(UGC Autonomous, Accredited by NBA, Accredited by NAAC with "A")

Bollikunta, Khila Warangal (Mandal), Warangal Urban-506 005 (T.S)

DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING

CERTIFICATE

This is to certify that the mini project entitled **FACIAL EXPRESSION AND FACE MASK DETECTION** is submitted by ***CH.KAMAL, E.SRIKANTH, G.CHANDHAN, K.ROHITH, K.SAISANTHOSH*** bearing **18641A05E2, 18641A05D4, 18641A05D8, 18641A05E7, 18641A05E4** in partial fulfillment of the requirements for the award of the Degree in Bachelor of Technology in Computer Science and Engineering during the academic year 2021-2022.

Guide **Head of the Department** **External Examiner**

Acknowledgement

First and foremost, we sincerely thank our esteemed institution **Vaagdevi College of Engineering, Warangal** for giving us this great opportunity to fulfill our dreams of becoming Software Engineers.

We sincerely thank our Minor project guide **Mrs. B. Saritha**, Assistant Professor of ComputerScience and Engineering for encouraging and supporting us throughout the project from start to end.

We are highly obliged to the Head of the Department of Computer Science and Engineering, **Dr. N. Satyavathi** for her constant support and extensive help in each and every step.

We would like to thank the Head of the Institution and Principal **Dr. K. Prakash** for the extensive support and for helping us in times we need it. Finally, we thank eachand everyone who extended their help directly or indirectly and made this project successful.

CH.KAMAL
(18641A05E2)
E.SRIKANTH
(18641A05D4)
G.CHANDHAN
(18641A05D8)
K.ROHITH
(18641A05E7)
K.ASISANTHOSH
(18641A05E4)

ABSTRACT

In today's world mostly work is going through the machine and their uses are increasing constantly they are helping in manpower and making our work easy and faster. Machine learning is giving the machine a perception, which can lead them to do a variety of tasks such as the variety of detections regarding human face like face mask detection combined with facial emotions. The coronavirus COVID-19 pandemic is causing a global health crisis. One of the effective protection methods is wearing a face mask in public areas according to the World Health Organization (WHO).

During the development of this project, we have used mobilenet using deep and classical machine learning for face mask detection. The proposed model takes images of size 242 X 242, before training standard images are converted into RGB images. We have developed a real-time Convolutional Neural Network model for emotion detection. We have trained our model on 30,000 labelled images. Our model can detect “neutral” (without emotions), “Happy”, “Angry” emotions. We got 92% accuracy for face-mask detection and 70% accuracy for Facial Expression detection. Our model will detect a person is wearing a mask or not if ‘yes’ it will detect the mask and if ‘no’ it will detect emotion in a face.

TABLE OF CONTENTS

1.INTRODUCTION

I. EXISTING SYSTEM

In the age of COVID-19, we have seen that wearing masks can substantially decrease the spread of the virus. However, as we sometimes see online, there are a lot of people that strongly disagree with this statement. Videos online show people becoming very disgruntled when they are asked to follow this protocol. There is no system for Detecting Mask. The shopkeeper or Police or In Offices etc.. has to check manually whether the person is wearing mask or not? This is time taking process for everyone.

II. PROPOSED SYSTEM

In this pandemic, we need a detection system that will give awareness to people about wearing the mask and in case people are not wearing a mask it will detect the emotion of that human face. For the training purpose,we used a powerful deep learning algorithm, convolutional neural network (CNN) and we used the transfer learning method. We trained the face mask model using 'mobile net V2' and we trained our face emotion model using 'sequential' and later we combined both models into a single screen using OpenCV real time detection.

The face emotion system is used for detecting a variety of facial expressions generated from a human face and we trained the model with 3 emotions (angry, happy, neutral) by which a machinewill be able to understand the current 'emotion statuses of a person. The face mask system is used for detecting is a person wearing a mask which is a very important thing to do in this pandemic so the camera can detect live whether a person is wearing it or not. The main aim of this project is for combining the model for combining purpose we again trained the face emotion model with 3 emotions because of limitations of data in the dataset and also for achieving good accuracy in this final model the machine will detect a person 'with mask' or 'no mask' if no it will detect emotion . This project is meant to fix this issue by detecting whether a person is wearing a mask, and if they are not, their facial expression will be read to determine if they are disgruntled.

APPLICATIONS

- Can use as a face mask detector in office.
- Can use as a face mask detector in school.
- Can be used as a mood detecting device.
- Can be used as a face mask detector in public place.

III. SOFTWARE REQUIREMENTS

The Software Requirements in this project include:

a. Python
b. OpenCV framework
c. Visual Studios(IDE)
d. Kaggle to train the system

LIBRARIES USED

The Libraries used in this project are:

- **OpenCV**

Open CV (Open Source Computer Vision Library) is an open-source computer vision software library for machine learning. Open CV was developed to serve the purpose ofcomputer vision applications and to stimulate the usage of machine perception in commercially viable products. OpenCV's role in our project is to invoke the camera capture live video and preprocess the video frames.

- **Numpy**

NumPy is a Python library used for working with arrays. It also has functions for workingin the domain of linear algebra, Fourier transform, and matrices. It is an open-source project and you can use it freely. NumPy stands for Numerical Python. To perform operations on the face encodings array we made use of the Numpy library.

- **Keras**

Keras is an open-source software library that provides a **Python interface for artificial neural networks**. Keras acts as an interface for the TensorFlow library. ... Designed to enable fast experimentation with deep neural networks, it focuses on being user-friendly, modular, and extensible.

- **Tensorflow**

TensorFlow is a Python library for **fast numerical computing** created and released by Google. It is a foundation library that can be used to create Deep Learning models directly or by using wrapper libraries that simplify the process built on top of

TensorFlow.

- **Matplotlib**

Matplotlib is a plotting library for **the Python programming language** and its numerical mathematics extension NumPy. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits like Tkinter, wxPython, Qt, or GTK.

- **Tqdm**

tqdm is a Python library that allows you to output a smart progress bar by wrapping around any iterable. A tqdm progress bar not only shows you how much time has elapsed, but also shows the estimated time remaining for the iterable.

- **Sklearn**

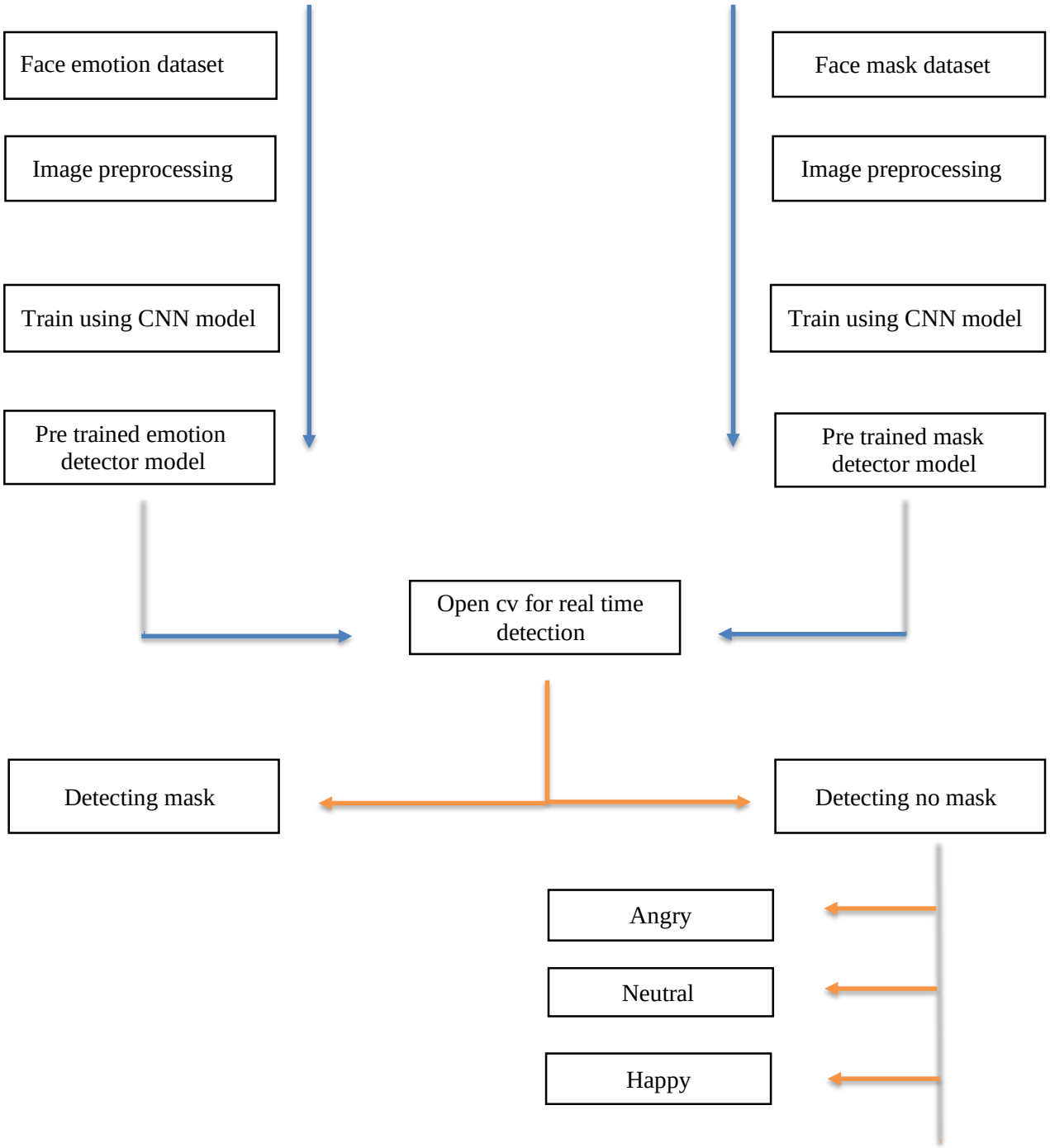
Scikit-learn is probably the most useful library for machine learning in Python. The sklearn library contains a lot of efficient tools for machine learning and statistical modeling including classification, regression, clustering and dimensionality reduction.

IV. HARDWARE REQUIREMENTS

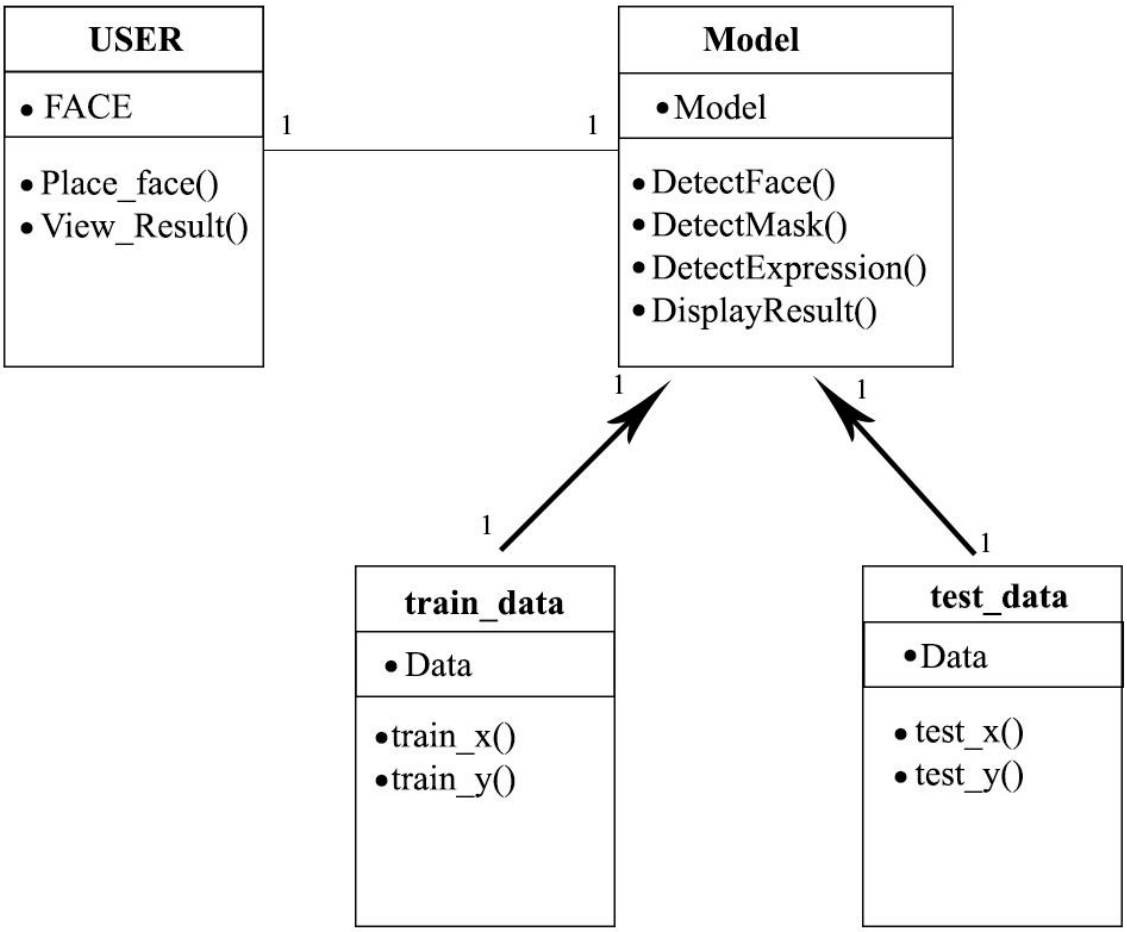
Components	**Minimum**	**Recommended**
Processor	Intel Core i5	Intel Core i7 10th GEN
RAM	8GB	16GB
Camera	HD 720p Webcam	Full HD 1080p Webcam
Disk	512Gb	1000Gb

2. DESIGN OF THE PROJECT

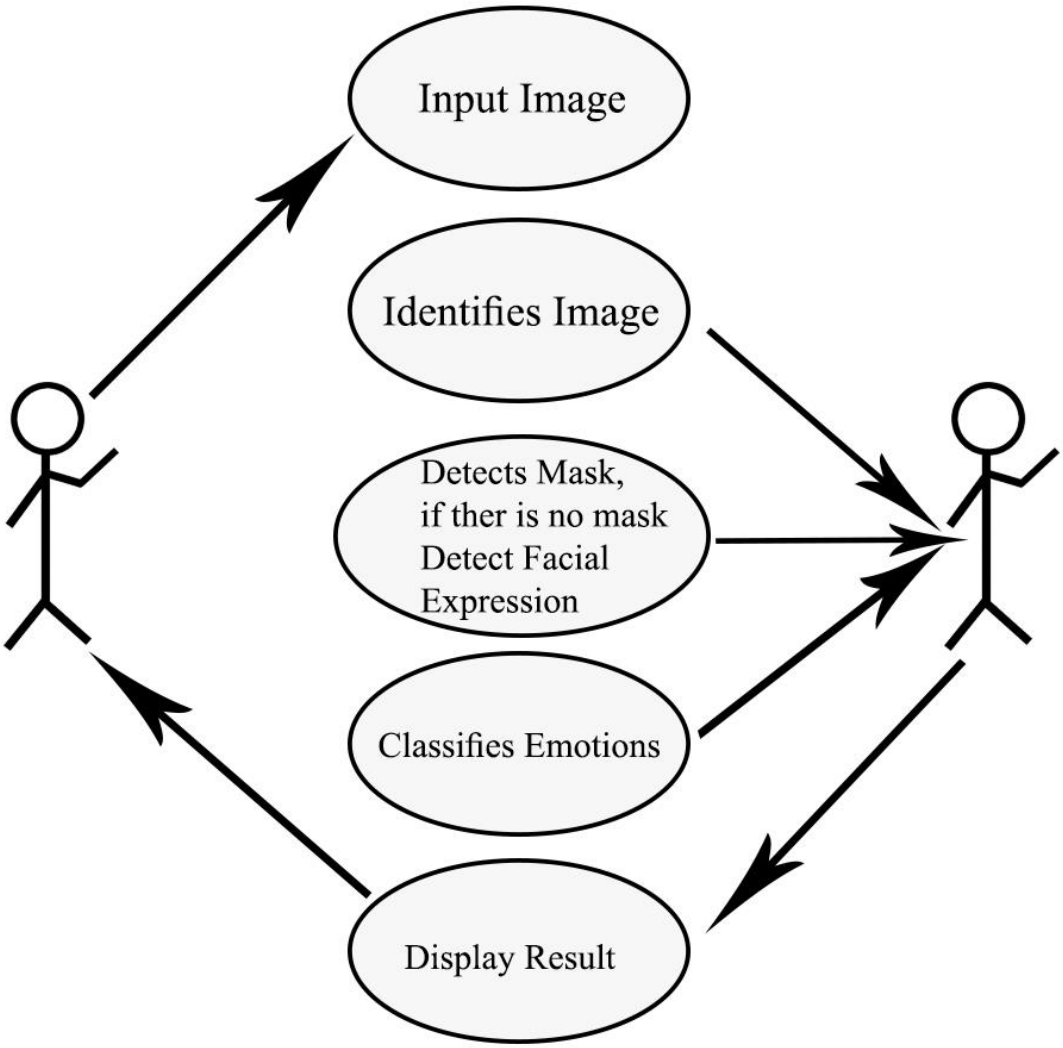
METHODOLOGY



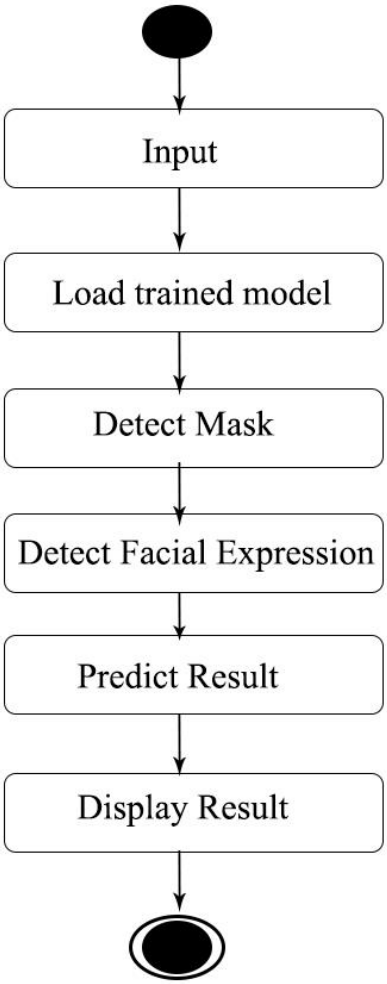
CLASS DIAGRAM



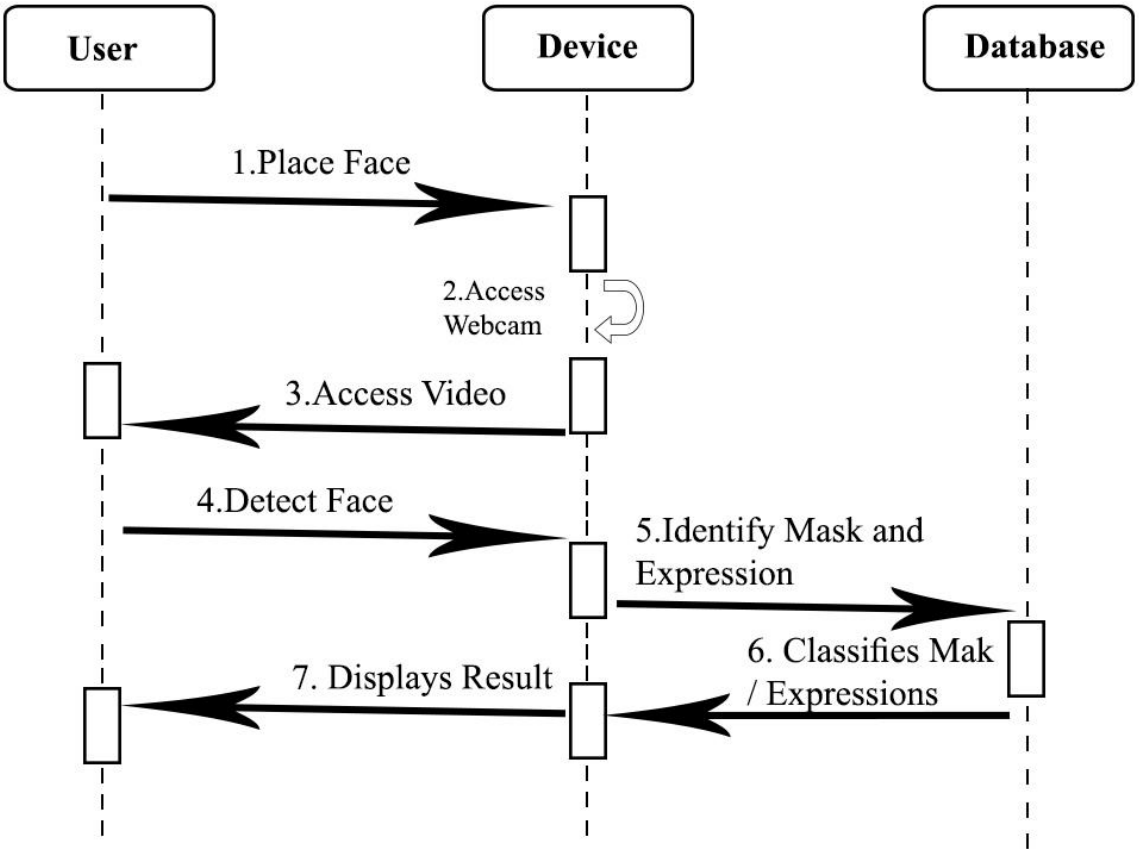
USE CASE DIAGRAM



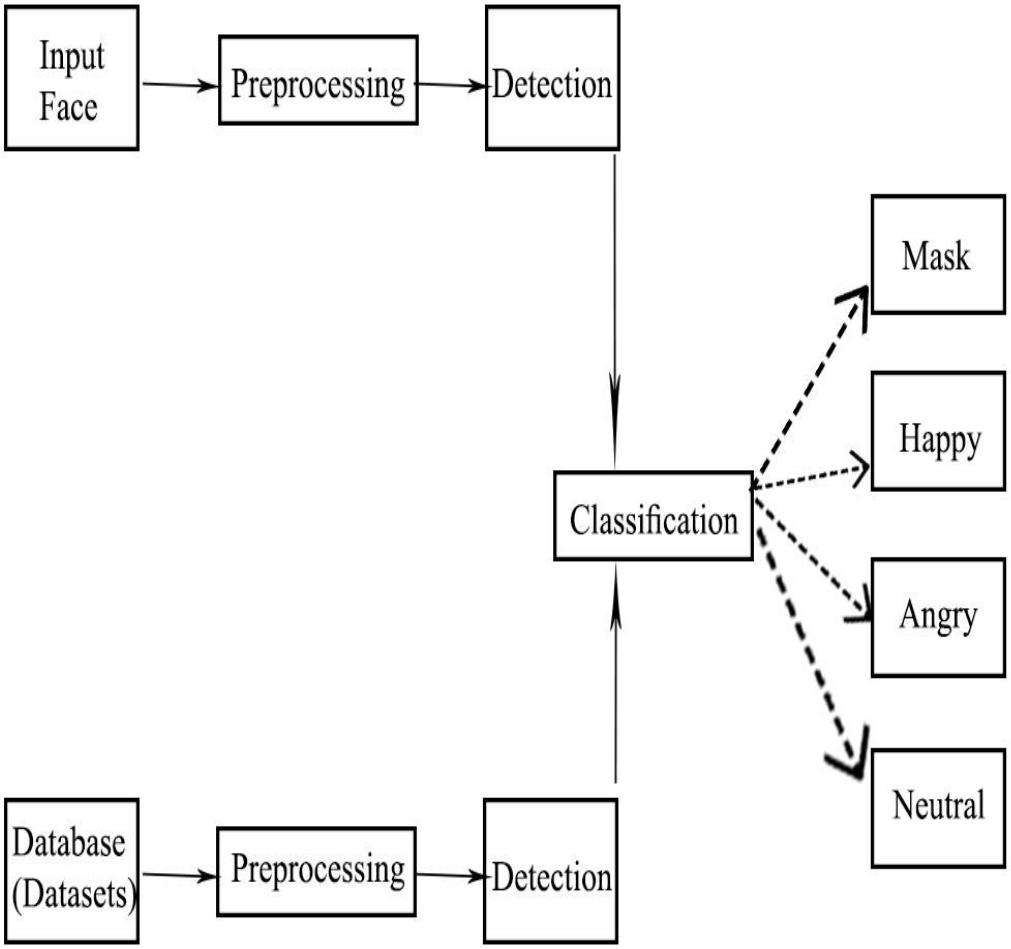
ACTIVITY DIAGRAM



SEQUENCE DIAGRAM



DATA FLOW DIAGRAM



3. IMPLEMENTATIONS

Dataset

1. Face Mask: ~12K Images Dataset (Kaggle)

2. Emotion: ~30K Images Dataset (Kaggle)

Mask Detection

The first step to this project is to build a neural network that can detect whether a person is wearing a mask. For this portion, we will use Mobilenet.

Before we build the model, we need to extract each image and preprocess it so that it can be fed into Mobilenet. In the dataset above, we see that there are three directories in the outermost layer: (1) Train, (2) Test, and (3) Validation. Below is the code for how to extract each image, preprocess them for mobilenet, and save them to a NumPy array.

```
From tqdm import tqdm
import cv2
import os
from keras.preprocessing import image
import numpy as np
import pandas as pd
from keras.utils import to_categorical
from sklearn.utils import shuffle as sk_shuffle
from sklearn.model_selection import train_test_split
from keras.applications.mobilenet_v2 import preprocess_input

def get_emotion_classes(class_type):
      angry_paths = [f'../EmotionDataset/{class_type}/angry/{i}' for i in
os.listdir(f'../EmotionDataset/{class_type}/angry')]
   angry_labels = [0 for i in range(len(angry_paths))]

   happy_paths = [f'../EmotionDataset/{class_type}/happy/{i}' for i in
os.listdir(f'../EmotionDataset/{class_type}/happy')]
```

```
    happy_labels = [1 for i in range(len(happy_paths))]

    neutral_paths = [f'../EmotionDataset/{class_type}/neutral/{i}' for i in
os.listdir(f'../EmotionDataset/{class_type}/neutral')]
    neutral_labels = [2 for i in range(len(neutral_paths))]
labels = np.array(angry_labels +  happy_labels + \
                  neutral_labels + sad_labels )

    print(f'{class_type.upper()} Value Count')
    print(pd.Series(labels).value_counts())
    print('~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~')
    labels = to_categorical(labels)
    paths = np.array(angry_paths + happy_paths + neutral_paths)
    paths, labels = sk_shuffle(paths, labels)
    return paths, labels
#0: angry  || 1: happy || 2: neutral

def get_emotion_splits(dim, pick_name, model_type = 'Mobilenet', bw = False):

    #Train

    train_paths, train_labels = get_emotion_classes('train')
    test_paths, test_labels = get_emotion_classes('test')

    train_images = np.array([get_image_value(i, dim, bw, model_type) for i in
tqdm(train_paths, desc = 'Getting Emotion Train Images')])
    test_images = np.array([get_image_value(i, dim, bw, model_type) for i in
tqdm(test_paths, desc = 'Getting Emotion Test Images')])

    if model_type == 'Mobilenet' and bw == True:
        train_images = np.stack((train_images,)*3, axis =-1)
        test_images = np.stack((test_images,)*3, axis = -1)

    tts = (train_images, test_images, train_labels, test_labels)

    pickle.dump(tts, open(f'../Pickles/TTSEmotion_{pick_name}.p', 'wb'), protocol = 4)
    print('Finished Pickling Emotions')

    return tts
dim = (224,224)
tts = get_emotion_splits(dim, pick_name = 'Mobilenet', bw = False)
```

After running the cell above, you should see a window like this.

```
Getting Train Images:   0%|
| 15/10000 [00:00<01:11, 138.86it/s]

TRAIN Value Counts
1    5000
0    5000
dtype: int64
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

Getting Train Images:   2%|
| 185/10000 [00:01<01:06, 147.20it/s]
```

In the later sections, we will reuse the function get_image_value, which is why we pass a parameter stating which model type it should retrieve the image for.

Now that we have the image values, it is time to build the neural network that will be used to detect the mask. Make sure you have a folder called ModelWeights inside the current directory. In this folder, we will save the weights for each model that we train.

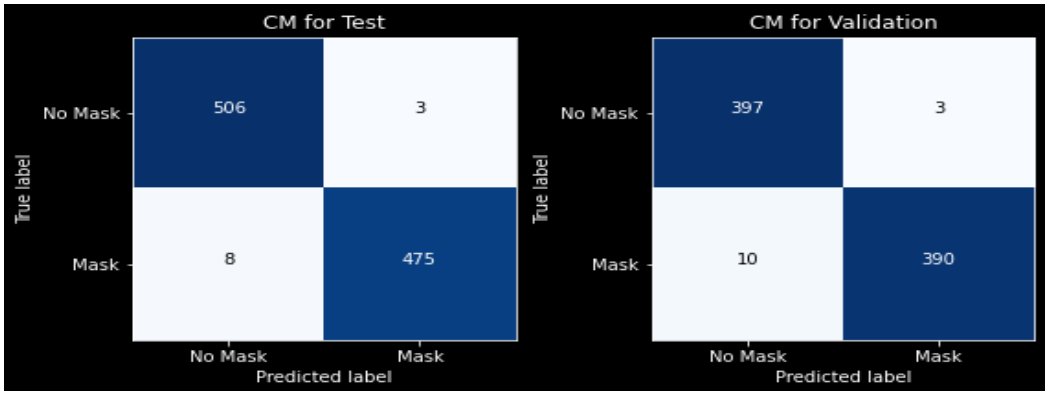
```
from keras.preprocessing.image import ImageDataGenerator
from keras.applications import MobileNetV2
from keras.layers import AveragePooling2D, Dropout, Flatten, Dense, Input,
GlobalAveragePooling2D, Conv2D, MaxPooling2D
from keras import regularizersfrom keras.models import Model, Sequential
from keras.optimizers import Adam
from keras.applications.mobilenet_v2 import preprocess_input
from sklearn.metrics import classification_report, confusion_matrix
import matplotlib.pyplot as plt
import numpy as np
from keras.callbacks import ModelCheckpoint, EarlyStopping, ReduceLROnPlateau
import cv2
from keras.models import load_model
def get_mobilenet(dim):
 model = Sequential()
optimizer = Adam(lr = .0005)
baseModel = MobileNetV2(weights="imagenet", include_top=False,
input_tensor=Input(shape=dim))
 model.add(baseModel)
model.add(AveragePooling2D(pool_size=(7, 7)))
model.add(Flatten(name="flatten"))
model.add(Dense(256, activation="relu"))
 model.add(Dropout(0.6))
 model.add(Dense(2, activation="sigmoid", name = 'Output'))
 for layer in baseModel.layers:
 layer.trainable = False
model.compile(loss = 'binary_crossentropy', optimizer = optimizer, metrics = ['accuracy'])
 return model
 early_stopping = EarlyStopping(monitor='val_loss', verbose = 1, patience=5, min_delta = .00075)
```

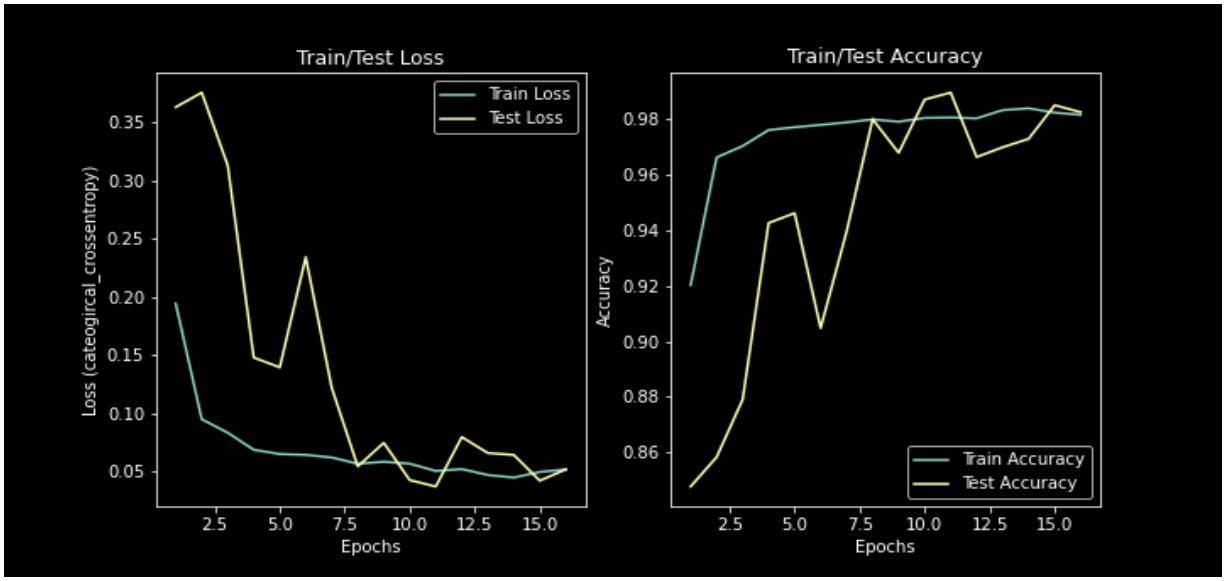
```
model_checkpoint = ModelCheckpoint(f'ModelWeights/Mobilenet_Masks.h5', verbose = 1,
save_best_only=True, monitor = 'val_loss')
lr_plat = ReduceLROnPlateau(patience = 5, mode = 'min')
epochs = 2000
batch_size = 64
dim = (x_train.shape[1], x_train.shape[2], x_train.shape[3])
mobilenet = get_mobilenet(dim =dim)
augmentation =ImageDataGenerator(rotation_range = 20, width_shift_range = .2,
height_shift_range = .2, horizontal_flip = True, shear_range = .15, fill_mode = 'nearest',
zoom_range = .15)
augmentation.fit(x_train)
mobilenet_history = mobilenet.fit_generator(augmentation.flow(x_train, y_train, batch_size =
batch_size),
 epochs = epochs,
callbacks = [early_stopping, model_checkpoint, lr_plat], validation_data = (x_test, y_test),
verbose= 1)
```

Above, we specified that the model should train for 2000 epochs. Because the model likely does not need 2000 epochs, we included an early stopping parameter to prevent the model from overfitting after 5 consecutive iterations with no change in the loss. After running the cell above, you should see a window like this:

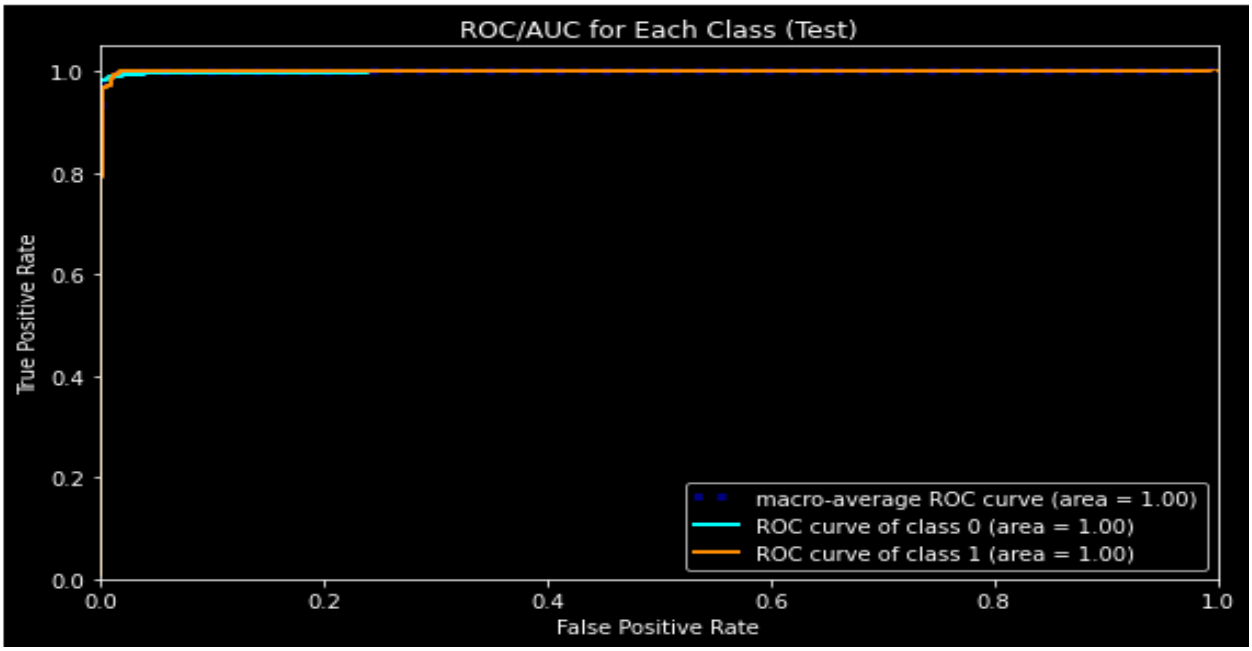
```
Epoch 1/2000
INFO:plaidml:Analyzing Ops: 1758 of 2052 operations complete
 35/157 [=====>........................] - ETA: 1:55 - loss: 0.3850 - acc: 0.8252
```

After the training is complete, you should see a .h5 file within the ModelWeights folder named Mobilenet_Masks.h5. In this file, the weights for the model are stored and will be used later on when we apply this neural network to live video.






```
F1 Scores Test
~~~~~~~~~~~~~~~~~~~~~~~~~
No Mask(0): 0.9838909541511772
Mask(1): 0.9836065573770492
```



Above, we see the ROC score, confusion matrix, and loss/accuracy for the mobilenet training. Considering the amount of data we used, these metrics are pretty good. For the validation set, only 13 images were incorrectly classified. As for the loss and accuracy, the loss was able to go below .1 and the accuracy was well above 94%. Finally, the ROC score shows great success as each class had a perfect score of 1.0, while F1 scores for each class were greater than .98.

Facial Expression Detection

Like the last model, we must first start by extracting the image values and placing them into a NumPy array. Like I mentioned earlier, we will reuse the get_image_value function within a new function designed to extract only the emotion images. The dataset contains 7 classes: angry, happy, neutral, sad, disgust, fear, and surprise. For this project, we will only be focusing on the first 3 classes: angry, happy, and neutral. Also, the model that we will use for training will take an input image with size (48,48,3), which is much smaller than mobilenet dimensions of (224,224,3).

```
def get_emotion_classes(class_type, max_values):
angry_paths = [f'../EmotionDataset/{class_type}/angry/{i}' for i in
os.listdir(f'../EmotionDataset/{class_type}/angry')][:max_values]
angry_labels = [0 for i in range(len(angry_paths))]
happy_paths = [f'../EmotionDataset/{class_type}/happy/{i}' for i in
os.listdir(f'../EmotionDataset/{class_type}/happy')][:max_values]
happy_labels = [1 for i in range(len(happy_paths))]
neutral_paths = [f'../EmotionDataset/{class_type}/neutral/{i}' for i in
os.listdir(f'../EmotionDataset/{class_type}/neutral')][:max_values]
neutral_labels = [2 for i in range(len(neutral_paths))]
labels = np.array(angry_labels + happy_labels + neutral_labels) print(f'{class_type.upper()} Value
Count')
print(pd.Series(labels).value_counts())
print('~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~')
labels = to_categorical(labels)
paths = np.array(angry_paths + happy_paths + neutral_paths)
paths, labels = sk_shuffle(paths, labels)
 return paths, labels

#0: angry 1: happy 2: neutral
def get_emotion_splits(dim, model_type = 'mobilenet', max_values = 6000):

train_paths, train_labels = get_emotion_classes('train', max_values)
test_paths, test_labels = get_emotion_classes('test', max_values)
train_images = np.array([get_image_value(i, dim, model_type) for i in tqdm(train_paths, desc =
```

```
'Getting Train Images')])
test_images = np.array([get_image_value(i, dim, model_type) for i in tqdm(test_paths, desc =
'Getting Test Images')])

return train_images, test_images, train_labels, test_labels
x_train, x_test, y_train, y_test = get_emotion_splits(dim = (48,48), model_type = 'normal',
max_values = 4000)
```

As you can see above, we limited each class to include only a maximum of 4000 images. This was performed so that training would be faster and so that we could properly track the performance with equal class balance.

After running the code above, you should see a similar window as before that looks like this:

```
TRAIN Value Count
2    4000
1    4000
0    3995
dtype: int64
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
TEST Value Count
1    1774
2    1233
0     958
dtype: int64
~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
Getting Train Images:  14%|
| 1634/11995 [00:10<01:07, 152.86it/s]
```

Now that we have the train test split arrays, we will build the neural network that will detect the emotion on a person's face. Below is the code for how to build this neural network. Unlike mobilenet, we will not apply augmentation to the dataset.

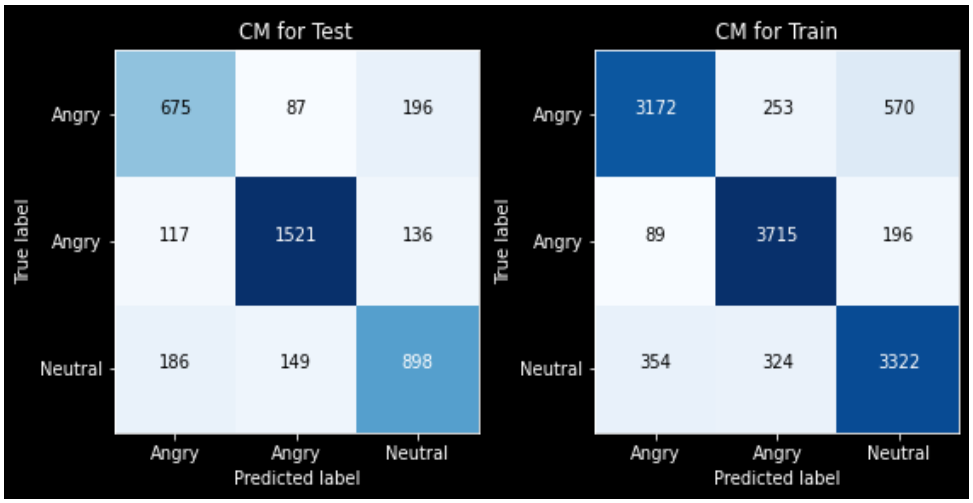
```
def get_conv_model(dim):
model = Sequential()
model.add(Conv2D(32, kernel_size = (3,3), activation = 'relu', input_shape = dim))
model.add(Conv2D(64, kernel_size = (3,3), activation = 'relu'))
model.add(MaxPooling2D(pool_size = (2,2)))
model.add(Dropout(.35))

model.add(Conv2D(128, kernel_size=(3, 3), activation='relu'))
```

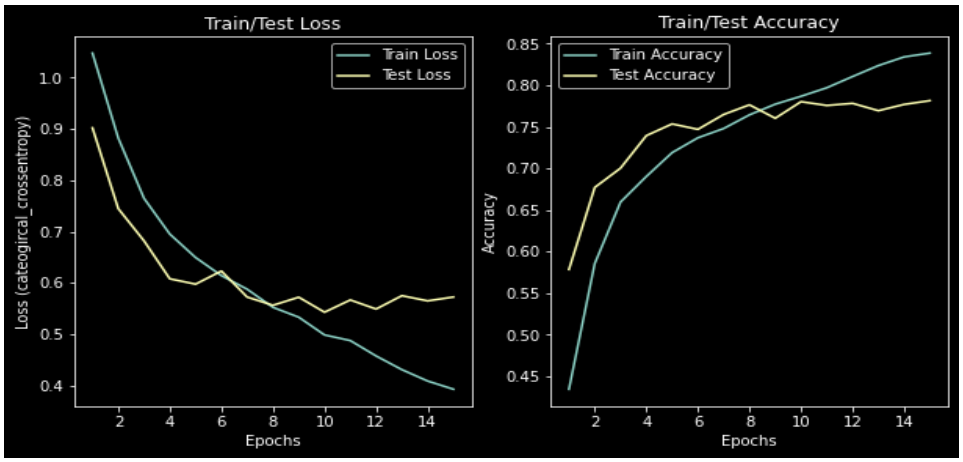
```
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Conv2D(128, kernel_size=(3, 3), activation='relu'))
model.add(MaxPooling2D(pool_size=(2, 2)))
model.add(Dropout(0.35))

model.add(Flatten())
model.add(Dense(1024, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(3, activation = 'softmax', name = 'Output'))
model.compile(optimizer = 'adam', loss = 'categorical_crossentropy', metrics = ['accuracy'])
return model

early_stopping = EarlyStopping(monitor='val_loss', verbose = 1, patience=5, min_delta = .00075)
model_checkpoint = ModelCheckpoint(f'ModelWeights/Normal_Emotions.h5', verbose = 1,
save_best_only=True, monitor = 'val_loss')

lr_plat = ReduceLROnPlateau(patience = 5, mode = 'min')
epochs = 2000
batch_size = 32
augment = False
dim = (x_train.shape[1], x_train.shape[2], x_train.shape[3])
cnn = get_conv_model(dim =dim)

cnn_history = cnn.fit(x_train, y_train, batch_size = batch_size,
epochs = epochs,
callbacks = [early_stopping, model_checkpoint, lr_plat], validation_data = (x_test, y_test),
verbose= 1)
```

After running the code above, you should see a window like this:

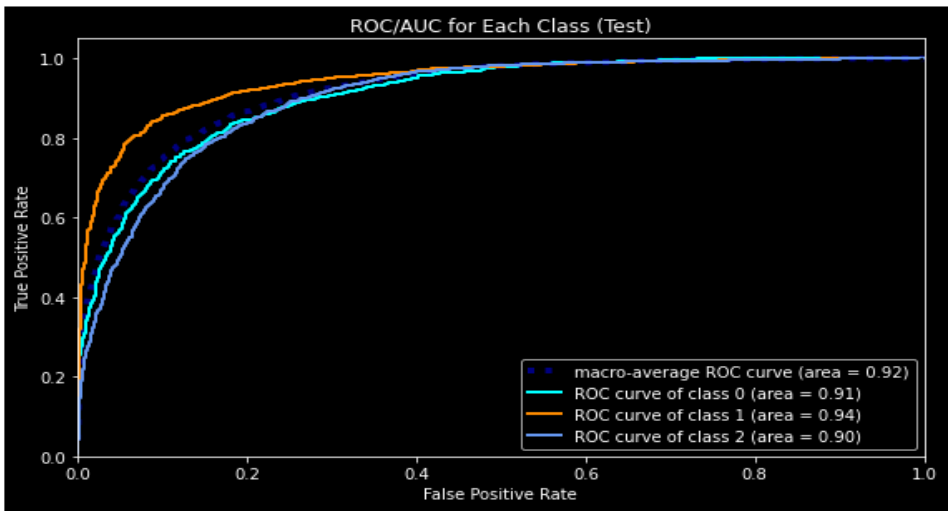
```
Train on 11995 samples, validate on 3965 samples
Epoch 1/2000
  544/11995 [>.............................] - ETA: 2:31 - loss: 1.1146 - acc: 0.3566
```

Once the training is complete, you should find a .h5 file called Normal_Emotions.h5 which will contain the weights for the model. Like the previous model we trained, this will be used for the live video portion below.






```
F1 Scores Test
~~~~~~~~~~~~~~~~~~~~~~~~
Angry(0): 0.6973140495867769
Happy(1): 0.8615123194562448
Neutral(2): 0.729192042224929
```



Above, we see the ROC score, confusion matrix, and loss/accuracy for the emotion model. Considering the amount of data we used, these metrics are pretty good, but not as good as mobilenet. For the training set, only 1,786 images were incorrectly classified out of the ~12,000. As for the loss and accuracy, the loss was able to go below .7 and the accuracy stayed between 70–75%. Finally, the ROC score shows pretty good success as each class maintained a score greater than .9, while F1 scores for each class were between .7 and .9.

Deployment

Now that we have each model trained, we will apply them both to live video using OpenCV. For this portion, you must have a webcam set up on you local machine.

Before we get into the code, let me first explain the steps for how this work.

1. Using haar cascade classifier, we ill find the coordinates of a face within a video frame
2. After finding the coordinates of the face, we will extract that portion, which is also called our region of interest (ROI).
3. Because we want to figure out if there is a mask present, we will first reshape that ROI to (224,224) so that it can be fed into our mobilenet model.
4. If the mobilenet model predicts there is a mask, then it will continue without using the emotion model.
5. However, if the mobilenet model predicts there is no mask, then it will use the ROI again and input it into the emotion model.

Below is the code for how it would work. At first glance, it seems like there are a lot of conditions within the for loop, but just read it slowly and it will make sense.

```
#livevide
o

            classifier=cv2.CascadeClassifier(f'ModelWeights/haarcascade_frontalface_default.x
            ml')
            mask_model = load_model(f'ModelWeights/Mobilenet_Masks.h5')
            emotion_model = load_model(f'ModelWeights/Normal_Emotions.h5')

            emotion_dim = (48,48)
            mask_dim = (224,224)

            mask_dict = {0: 'No Mask', 1: 'Mask'}
            mask_dict_color = {0: (0,0,255), 1: (0,255,0)} #colors for bounding boxes
            emotion_dict = {0: 'angry', 1: 'happy', 2: 'neutral'}

            vid_frames = []
            cap = cv2.VideoCapture(0)

            while True:
                ret, frame = cap.read()
                frame = cv2.flip(frame, 1,1)
                clone = frame.copy()
                bboxes = classifier.detectMultiScale(clone)
                for i in bboxes:
                    x, y, width, height = i[0], i[1], i[2], i[3]
                    x2, y2 = x+ width, y+height
                    mask_roi = clone[y:y2, x:x2]
                    emotion_roi = mask_roi.copy()
                    emotion_roi = cv2.resize(emotion_roi, emotion_dim, interpolation =
            cv2.INTER_CUBIC)
                    mask_roi = cv2.resize(mask_roi, mask_dim, interpolation =
            cv2.INTER_CUBIC)

                    #preprocess mask_input
                    mask_roi= preprocess_input(mask_roi)

                    #preprocess emotion input
                    emotion_roi = emotion_roi/255

                    #resize emotion and mask to feed into nn
                    mask_roi = mask_roi.reshape(1, mask_roi.shape[0], mask_roi.shape[1],
            mask_roi.shape[2])
```

```
        emotion_roi = emotion_roi.reshape(1, emotion_roi.shape[0],
emotion_roi.shape[1], emotion_roi.shape[2])
        #mask predictions
        mask_predict = mask_model.predict(mask_roi)[0]
        mask_idx = np.argmax(mask_predict)
        mask_conf = f'{round(np.max(mask_predict)*100)}%'
        mask_cat = mask_dict[mask_idx]
        mask_color = mask_dict_color[mask_idx]
        if mask_idx == 0: #if there is no mask detected --> move onto emotions
            #emotion predictions
            emotion_predict = emotion_model.predict(emotion_roi)[0]
            emotion_idx = np.argmax(emotion_predict)
            emotion_cat = emotion_dict[emotion_idx]
            emotion_conf = f'{round(np.max(emotion_predict)*100)}%'
            cv2.putText(clone, f'{mask_cat}: {mask_conf} || {emotion_cat}:
{emotion_conf}', (x, y+15), cv2.FONT_HERSHEY_SIMPLEX, .5, mask_color, 2)
            cv2.rectangle(clone, (x,y), (x2,y2), mask_color, 1)
            continue
        cv2.putText(clone, f'{mask_cat}: {mask_conf}', (x, y+15),
cv2.FONT_HERSHEY_SIMPLEX, .5, mask_color, 2)
        cv2.rectangle(clone, (x,y), (x2,y2), mask_color, 1)

    cv2.imshow('LIVE', clone)
    vid_frames.append(clone)

    if cv2.waitKey(1) & 0xFF ==ord('q'):
        break
cap.release()
cv2.destroyAllWindows()
```

Once you run the code above, a window should pop up . To Stop this process, press the ‘q’ button on your keyboard.

4. TESTING

Test Scenario ID	Test Case ID	Test Case	Expected Output	Actual Output	Test Result
TS001	TC001	Verify the Face with Mask in Lighting Area	Should display Mask 100%	Mask 100%	Pass
TS002	TC002	Verify the Smiley Face without Mask in Lighting area	Should display No Mask Happy 100%	No Mask Happy 100%	Pass
TS003	TC003	Verify the no expression Face without Mask in Lighting area	Should display No Mask Neutral 100%	No Mask Neutral 100%	Pass
TS004	TC004	Verify the Angry expression Face without Mask in Lighting area	Should display No Mask Angry 100%	No Mask Angry 100%	Pass
TS005	TC005	Verify the Face with Mask in Low Lighting Area	Should display Mask 100%	No Mask Expression %	Fail
TS006	TC006	Verify the Smiley Face without Mask in Low Lighting area	Should display No Mask Happy 100%	Mask 100%	Fail
TS007	TC007	Verify the no expression Face without Mask in Low Lighting area	Should display No Mask Neutral 100%	Mask100%	Fail
TS008	TC008	Verify the Angry Face without Mask in Low Lighting area	Should display No Mask Angry 100%	Mask 100%	Fail
TS009	TC009	Verify the Angry Face Foreigner picture without Mask	Should display No Mask Angry 100%	No Mask Sad 100%	Pass
TS010	TC010	Verify the Angry Face of African People without Mask	Should display No Mask Angry 100%	Mask 100%	Fail

5. RESULTS

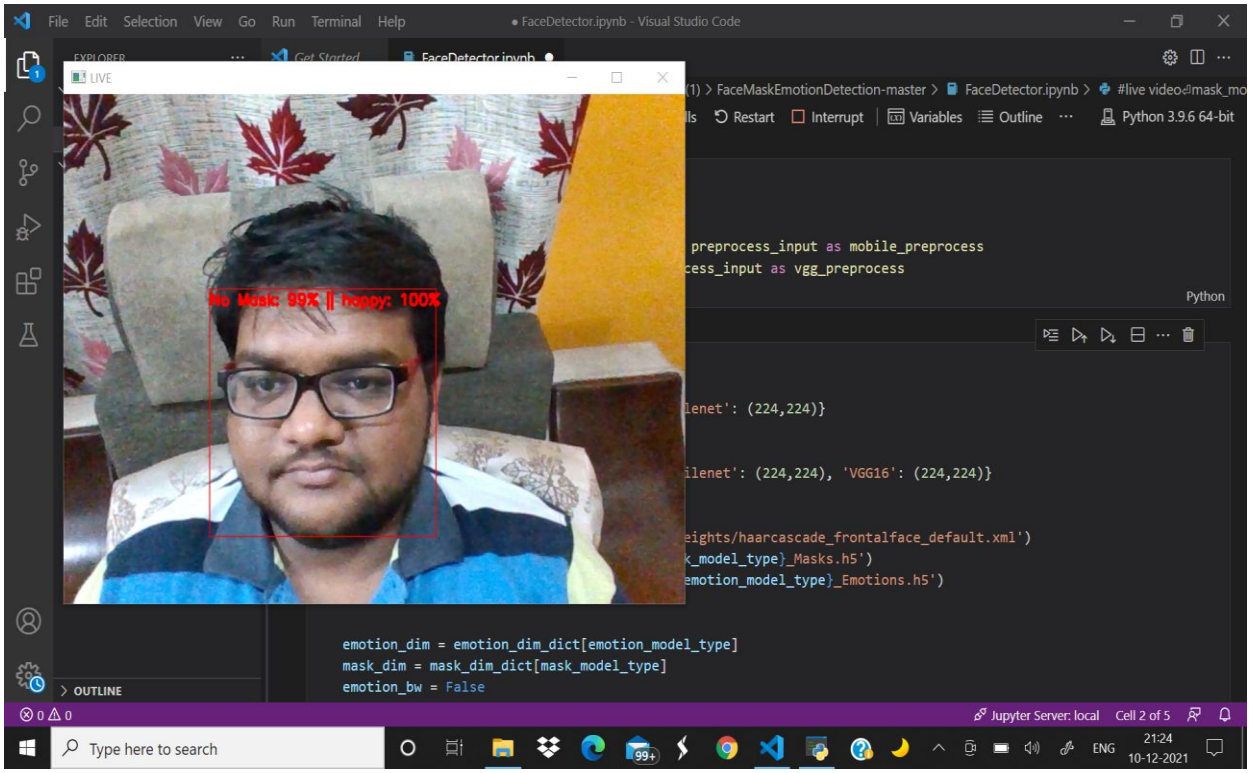


Figure 1: Happy Face Detected

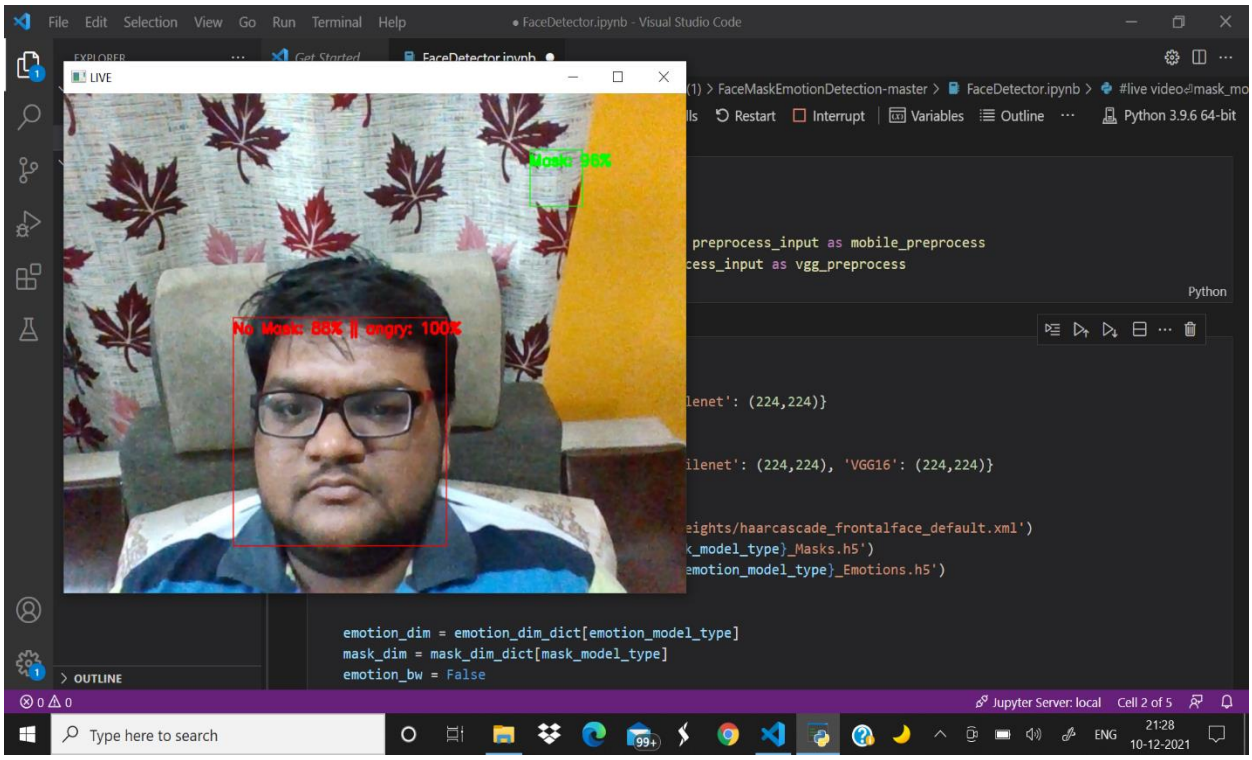


Figure 2: Angry Face Detected

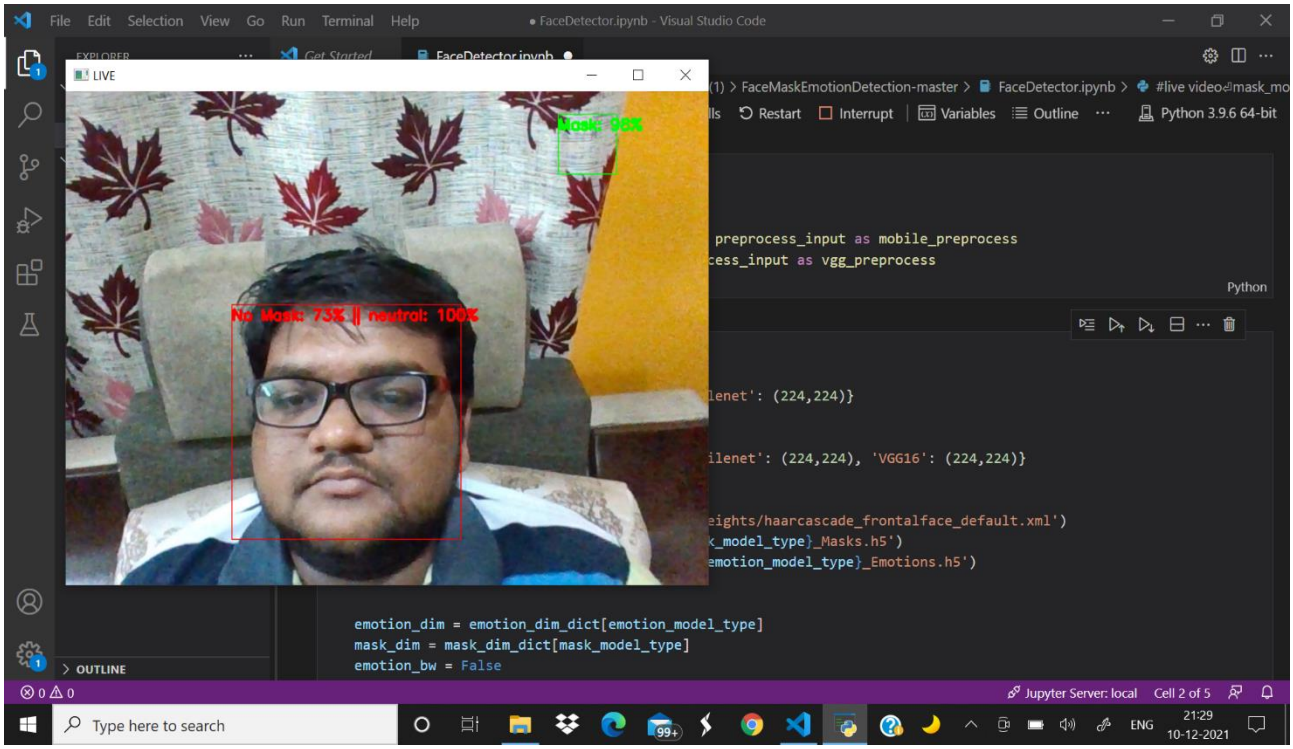


Figure 3 : Neutral Face Detected

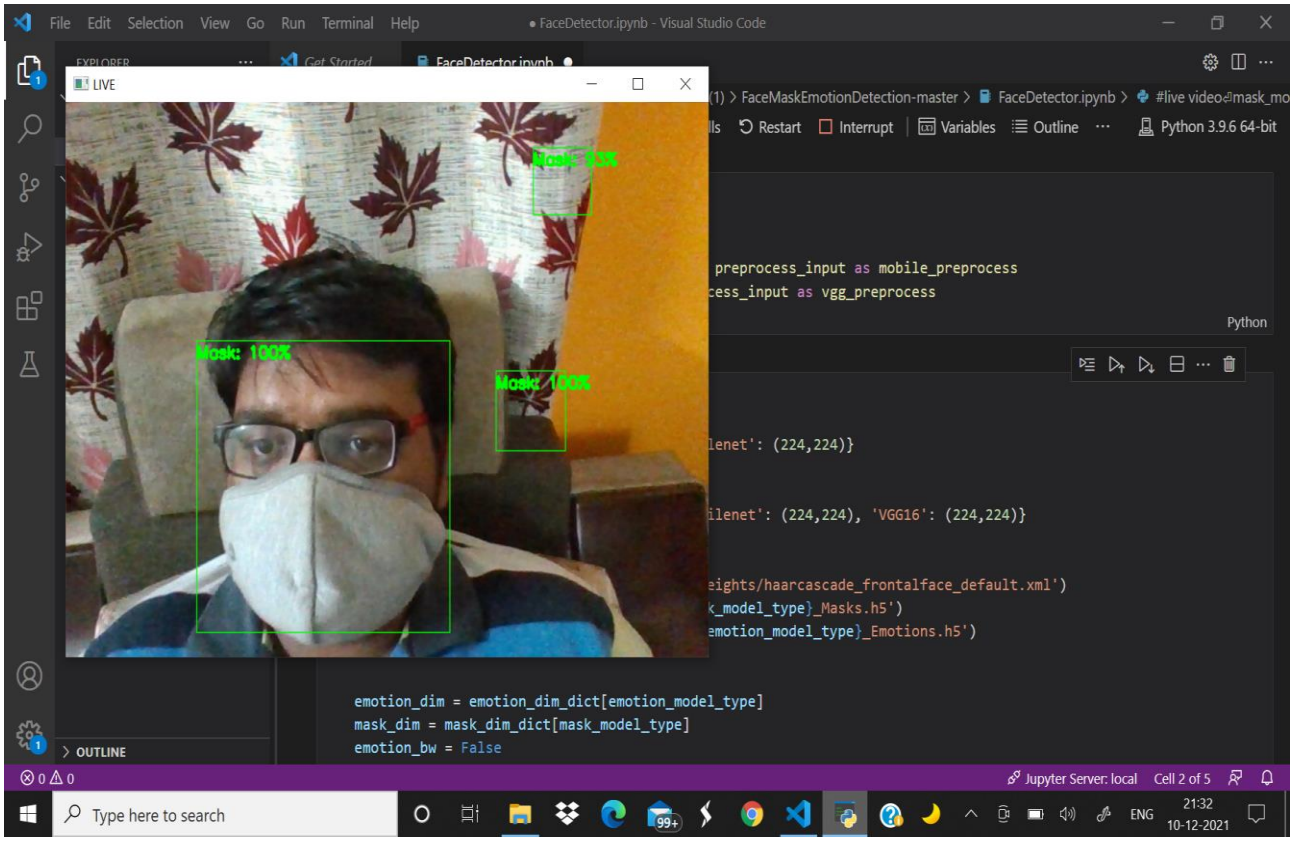


Figure 4: Mask Detected

- **Advantages**

1. It saves time.
2. Reduce man power.
3. Give awareness for self-protection.
4. Easy to use.
5. Efficient and reliable.
6. Easy to manage

- **Disadvantages**

1. Less emotions.
2. Works based on Datasets Trained.
3. Unable to detect at Low Lighting areas.
4. Some people can easily escape by cheating.
5. Inner emotions of person cannot be detected.
6. Cannot able to detect if a person wearing other than mask (Hand Kerchief).

6. CONCLUSIONS AND FUTURE SCOPE

The experiment is implemented successfully and hence shows the proposed system can be successfully exploited for face mask violation detection with emotions. It is able to process on real-time images and videostreams which makes it applicable in the real world as it can work on limited computational capability devicesalso. We all know technology is emerging with trends do our project may possibly contribute to the public health care in this pandemic.

There are few limitations for this model. Because of the dataset the model is unable to distinguish whether the mask is present if the person in the frame is wearing glasses, Detection is taking place under good lighting, As the datasets contains completely foreigners images, our Project is accurate with only those pictures. We can improve the model's performance by adding more images of persons.

Future Scope

We are planning to improve our face mask with an emotion tool that can be released as an open-source project,and our software can be equated to an existing USB, IP cameras, and CCTV for further progressed detectionwhich can make a drastic change in the public health care.

7.BIBLIOGRAPHY

1) Amil Khanzada , charles Bai ,Ferhat Turker Celepcikay , (2004) Stanford university *“* Facial Expression Recognition with Deep Learning: In this paper they were able to achieve the highest accuracy and to applyFER models to the real world.”
2) Ravichandra Ginne , Krupa Jariwala ,(2018) *“*Facial Expression Recognition Using Cnn: In this paper theyproposed a model consisting of many subnets that are structure differently.”,march 2018. International Journal of Advances in Electronics and Computer Science, ISSN: 2393-2835
3) Vinitha.V , Velantina.V (2020) International Research Journal of Engineering and Technology (IRJET) *“*COVID-19 Face mask Detection with Deep Learning and Computer Vision : In this paper they used architecture that consists of MobileNet as the backbone it can be used for high and low computation scenarios.”
4) Raghuvanshi, Arushi, and Vivek Choksi. "Facial Expression Recognition with Convolutional Neural Networks." Stanford University, 2016