FACE MASK DETECTION SYSTEMFINAL REPORT

PROJECT BY: GADWALA VENKATA VISHNU VARDHAN

ABSTRACT:

In the introduction, we mentioned the significance of taking precautionary measuresto prevent the spread of the COVID-19 virus. Wearing masks is one of the most effective ways to prevent the virus's spread, and the mask detection system aims to ensure that people wear masks in public places.

PROBLEM STATEMENT:

- The objective of this project is to develop a face mask detection system using OpenCV and Keras.
- The system should be able to detect faces in real-time video streams and determine whether the person is wearing a face mask or not.
- The face mask detection system will be trained using a dataset of images of people wearing face masks and not wearing face masks.
- The system will use Deep Neural Networks (DNN) to classify the images and determine whether a face mask is present or not.

DATASET:

We used a our own dataset that consisted of 7553 images of people wearing and not wearing masks. The dataset was split into two equal parts, with 3776images of people wearing masks and 3777 images of people not wearing masks. Thedataset was balanced to ensure that the model is not biased towards one class.

METHODODLOGY:

We used a deep Neural Networks approach to train our mask detection model.

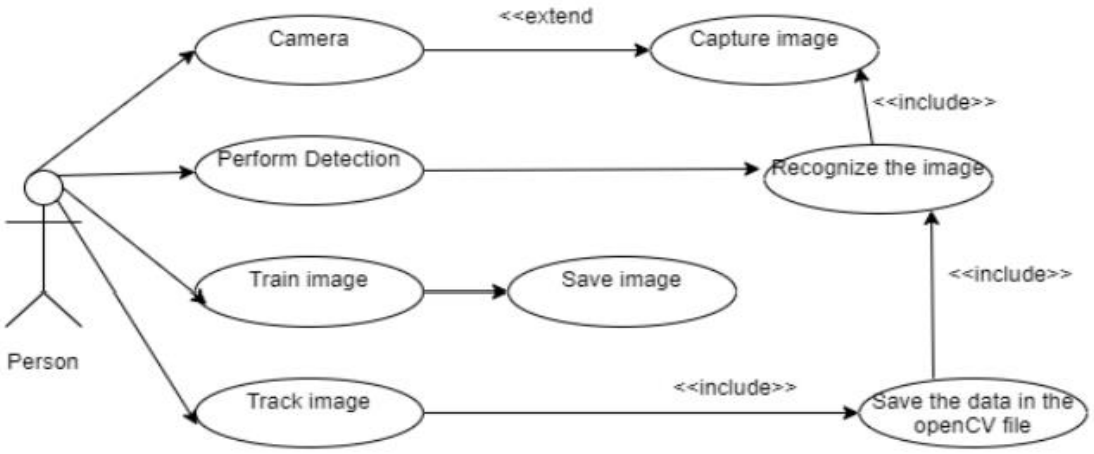
We trained our model on a cloud based GPU for 30 epochs, which means that the model was trained on the dataset 30 times. We used the Adam optimizer with a learning rate of 0.001, which is a commonly used optimizer for training deep learning models. We used the binary cross-entropy loss function to optimize the model, which is a commonly used loss function for binary classification problems.

EVALUATION:

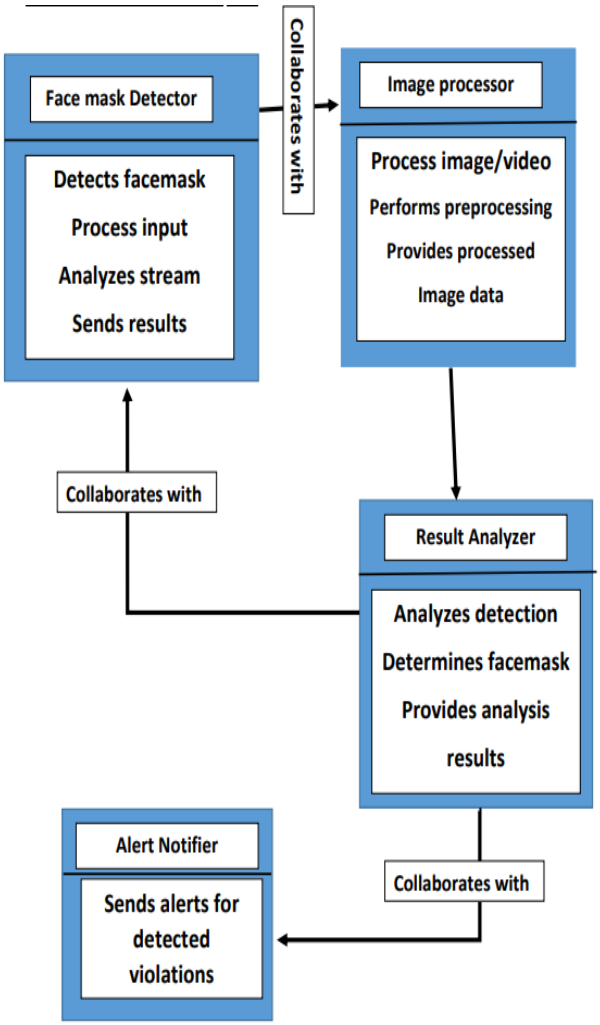
We evaluated our model on a test set that consisted of 1,511 images, which were notused during the training phase. We used accuracy as our primary metric to evaluate the model's performance. Our model achieved an accuracy of 98%, which means thatthe model predicted the correct class for 98% of the test set images.

We also calculated the precision and recall values for our model. Precision measuresthe proportion of true positives among the total positive predictions, while recall measures the proportion of true positives among the total actual positive instances.Our model achieved a precision value of 98%, which means that 98% of the predictions made by the model were true positives. The recall value was 98%, whichmeans that the model correctly identified 98% of the actual positive instances in thetest set.

USE CASE DIAGRAM:

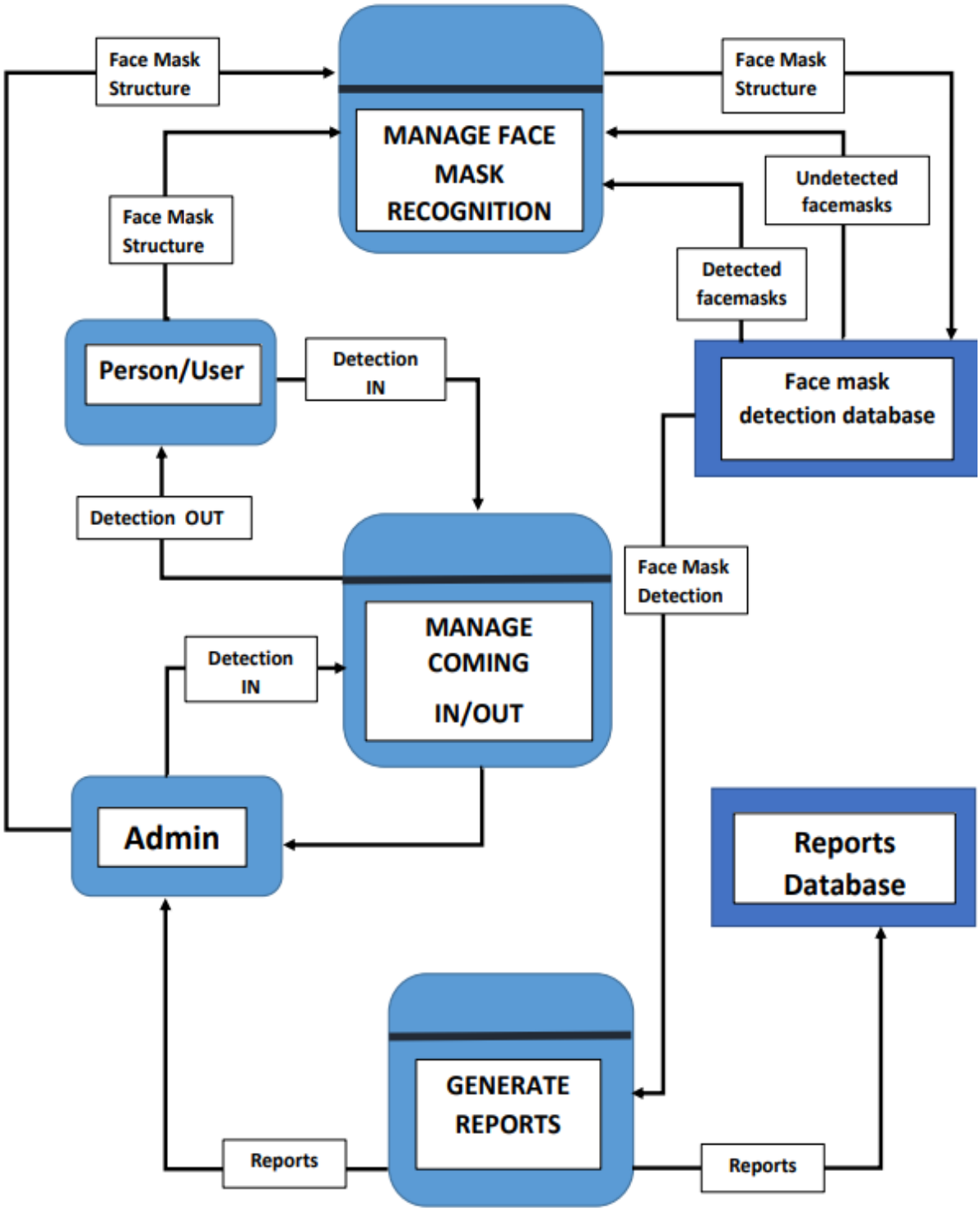


Class Diagram:



DATA FLOW DIAGRAM

DATA FLOW DIGRAM



CODE:

Building Deep Neural network

8.3 Build instance of Network

In [56]:

```
def build_model():
    input_layer = Input(shape=(120,120,3))

    vgg = VGG16(include_top=False)(input_layer)

    # Classification Model
    f1 = GlobalMaxPooling2D()(vgg)
    class1 = Dense(2048, activation='relu')(f1)
    class2 = Dense(1, activation='sigmoid')(class1)

    # Bounding box model
    f2 = GlobalMaxPooling2D()(vgg)
    regress1 = Dense(2048, activation='relu')(f2)
    regress2 = Dense(4, activation='sigmoid')(regress1)

    facetracker = Model(inputs=input_layer, outputs=[class2, regress2])
    return facetracker
```

Architecture of DNN

8.4 Test out Neural Network

[57]:

```
facetracker = build_model()
```

[58]:

```
facetracker.summary()
```

```
Model: "model"
_________________________________________________________________________________________________
 Layer (type)                Output Shape              Param #   Connected to
=================================================================================================
 input_2 (InputLayer)        [(None, 120, 120, 3       0         []
                             )]

 vgg16 (Functional)          (None, None, None,        14714688  ['input_2[0][0]']
                             512)

 global_max_pooling2d (GlobalMa  (None, 512)           0         ['vgg16[0][0]']
 xPooling2D)

 global_max_pooling2d_1 (Global  (None, 512)           0         ['vgg16[0][0]']
 MaxPooling2D)

 dense (Dense)               (None, 2048)              1050624   ['global_max_pooling2d[0][0]']

 dense_2 (Dense)             (None, 2048)              1050624   ['global_max_pooling2d_1[0][0]']

 dense_1 (Dense)             (None, 1)                 2049      ['dense[0][0]']

 dense_3 (Dense)             (None, 4)                 8196      ['dense_2[0][0]']

=================================================================================================
Total params: 16,826,181
Trainable params: 16,826,181
Non-trainable params: 0
_________________________________________________________________________________________________
```

Training DNN

In [74]:

```
class FaceTracker(Model):
    def __init__(self, eyetracker,  **kwargs):
        super().__init__(**kwargs)
        self.model = eyetracker

    def compile(self, opt, classloss, localizationloss, **kwargs):
        super().compile(**kwargs)
        self.closs = classloss
        self.lloss = localizationloss
        self.opt = opt

    def train_step(self, batch, **kwargs):

        X, y = batch

        with tf.GradientTape() as tape:
            classes, coords = self.model(X, training=True)

            batch_classloss = self.closs(y[0], classes)
            batch_localizationloss = self.lloss(tf.cast(y[1], tf.float32), coords)

            total_loss = batch_localizationloss+0.5*batch_classloss

            grad = tape.gradient(total_loss, self.model.trainable_variables)

        opt.apply_gradients(zip(grad, self.model.trainable_variables))

        return {"total_loss":total_loss, "class_loss":batch_classloss, "regress_loss":batch_localizationlos

    def test_step(self, batch, **kwargs):
        X, y = batch

        classes, coords = self.model(X, training=False)

        batch_classloss = self.closs(y[0], classes)
        batch_localizationloss = self.lloss(tf.cast(y[1], tf.float32), coords)
        total_loss = batch_localizationloss+0.5*batch_classloss

        return {"total_loss":total_loss, "class_loss":batch_classloss, "regress_loss":batch_localizationlos
```

REAL TIME DETECTION WITH VIDEO

11.3 Real Time Detection

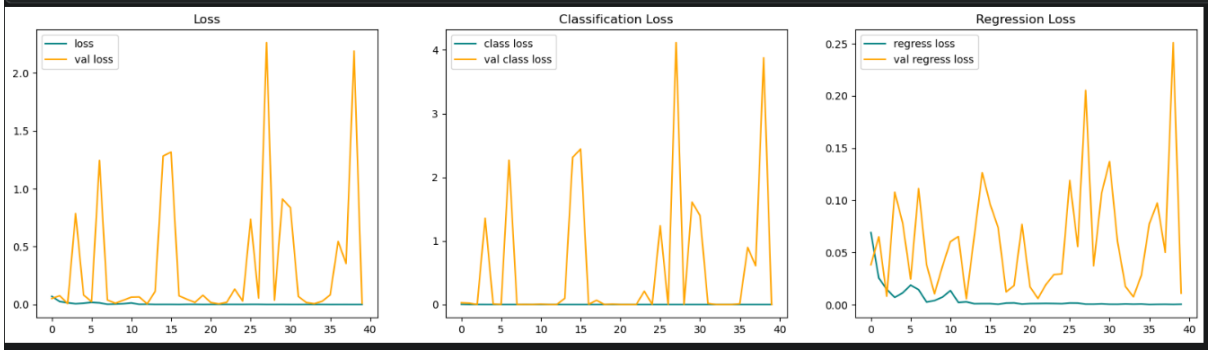
In [3]:
```
import cv2
import tensorflow as tf
import numpy as np
```

In [4]:
```
cap = cv2.VideoCapture(0)
while cap.isOpened():
    _ , frame = cap.read()
    frame = frame[50:500, 50:500,:]
    rgb = cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
    resized = tf.image.resize(rgb, (120,120))
    yhat = facetracker.predict(np.expand_dims(resized/255,0))
    sample_coords = yhat[1][0]
    if (yhat[0] >= 0.5 and yhat[0] <= 1):
        # Controls the main rectangle
        cv2.rectangle(frame,
                      tuple(np.multiply(sample_coords[:2], [450,450]).astype(int)),
                      tuple(np.multiply(sample_coords[2:], [450,450]).astype(int)),
                            (0,255,0), 2)
        # Controls the label rectangle
        cv2.rectangle(frame,
                      tuple(np.add(np.multiply(sample_coords[:2], [450,450]).astype(int),
                                    [0,-30])),
                      tuple(np.add(np.multiply(sample_coords[:2], [450,450]).astype(int),
                                    [80,0])),
                            (0,255,0), -1)
        # Controls the text rendered
        cv2.putText(frame,'Mask',tuple(np.add(np.multiply(sample_coords[:2], [450,450]).astype(int),
                                              [0,-5])),
                    cv2.FONT_HERSHEY_SIMPLEX, 1, (255,255,255), 1, cv2.LINE_AA)
        print(yhat[0])
```

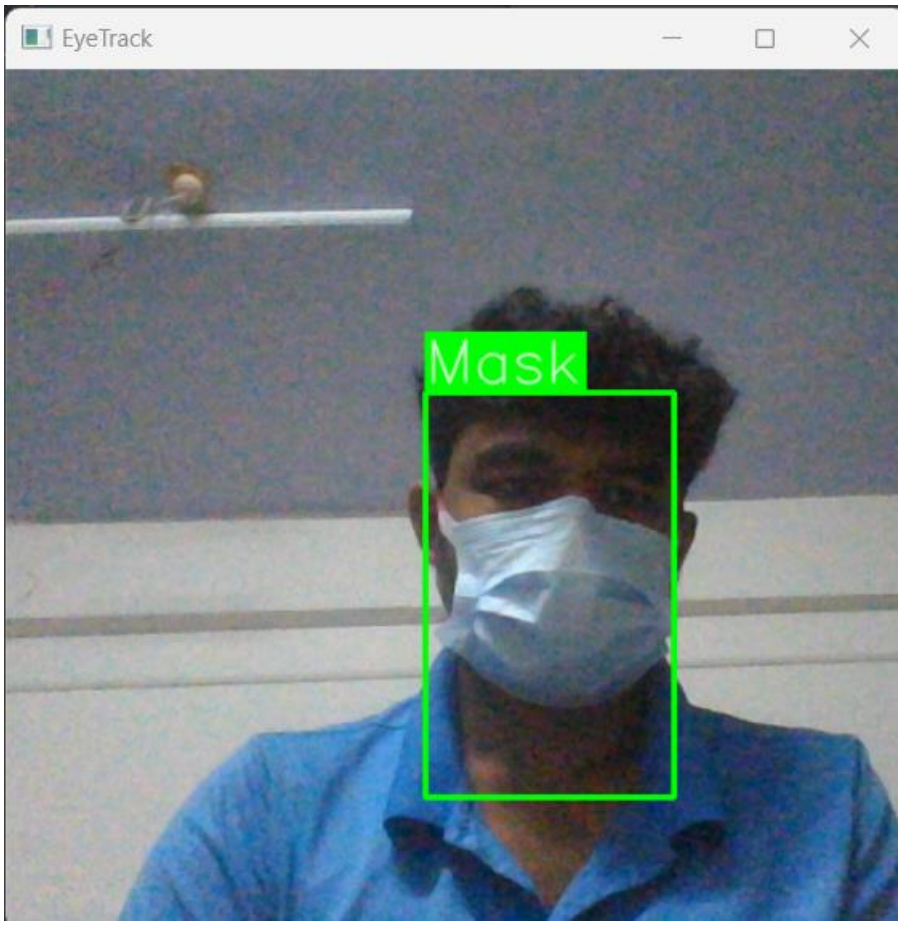
```
    elif (yhat[0] < 0.5 and yhat[0] > 0):
        # Controls the main rectangle
        cv2.rectangle(frame,
                      tuple(np.multiply(sample_coords[:2], [450,450]).astype(int)),
                      tuple(np.multiply(sample_coords[2:], [450,450]).astype(int)),
                            (0,0,255), 2)
        # Controls the label rectangle
        cv2.rectangle(frame,
                      tuple(np.add(np.multiply(sample_coords[:2], [450,450]).astype(int),
                                    [0,-30])),
                      tuple(np.add(np.multiply(sample_coords[:2], [450,450]).astype(int),
                                    [140,0])),
                            (0,0,255), -1)
    # else:
        cv2.rectangle(frame,
                      tuple(np.add(np.multiply(sample_coords[:2], [450,450]).astype(int),
                                    [0,-30])),
                      tuple(np.add(np.multiply(sample_coords[:2], [450,450]).astype(int),
                                    [140,0])),
                            (0,0,255), -1)
        print(yhat[0])


        # Controls the text rendered
        cv2.putText(frame,'No Mask',tuple(np.add(np.multiply(sample_coords[:2], [450,450]).astype(int),
                                                 [0,-5])),
                    cv2.FONT_HERSHEY_SIMPLEX, 1, (255,255,255), 1, cv2.LINE_AA)

    cv2.imshow('EyeTrack', frame)

    if cv2.waitKey(1) & 0xFF == 27:
        break
cap.release()
cv2.destroyAllWindows()
```

Loss plots

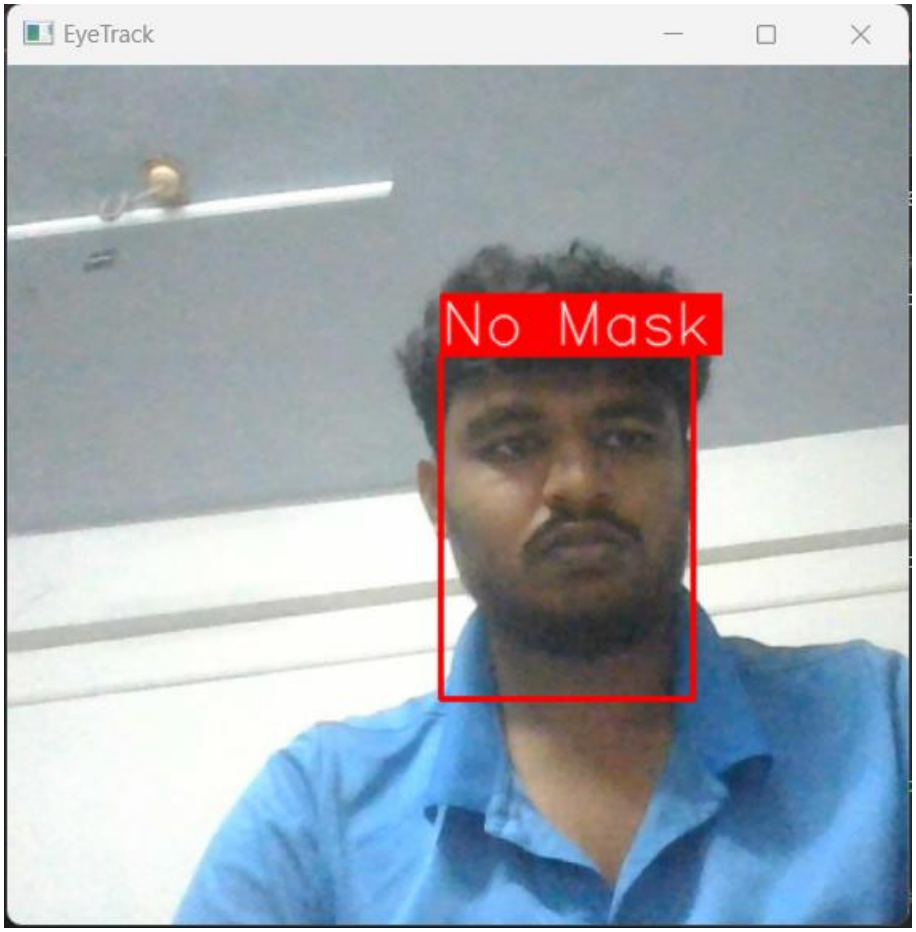
```python
fig, ax = plt.subplots(ncols=3, figsize=(20,5))

ax[0].plot(hist.history['total_loss'], color='teal', label='loss')
ax[0].plot(hist.history['val_total_loss'], color='orange', label='val loss')
ax[0].title.set_text('Loss')
ax[0].legend()

ax[1].plot(hist.history['class_loss'], color='teal', label='class loss')
ax[1].plot(hist.history['val_class_loss'], color='orange', label='val class loss')
ax[1].title.set_text('Classification Loss')
ax[1].legend()

ax[2].plot(hist.history['regress_loss'], color='teal', label='regress loss')
ax[2].plot(hist.history['val_regress_loss'], color='orange', label='val regress loss')
ax[2].title.set_text('Regression Loss')
ax[2].legend()

plt.show()
```



WITH MASK DETECTION



WITHOUT MASK DETECTION



CONCLUSION:

In conclusion, we developed a mask detection system that can detect whether people are wearing masks or not. We used a deep learning-based approach and achieved an accuracy of 98% on the test set. Our model can be an effective tool forenforcing mask-wearing in public places. We also deployed our model on a low- power device like the Raspberry Pi, which makes it easily accessible and affordable.Overall, our mask detection system has the potential to contribute to the ongoing efforts to prevent the spread of the COVID-19 virus.