

Face Mask Detection using Machine Learning

Milestone 4: Perform your own experiments

Team members: 1. Shashank Reddy Kallem (sk3338@njit.edu)
2. Suvagiya BrijeshKumar Ashokbhai (bas67@njit.edu)
3. Venkata Leela Manohar Morampudi(vm73@njit.edu)

Submission Date: April 25, 2023

Course Name: DS675 Machine Learning
Instructor's Name: Michael Houle

Video Presentation Link:

Google Colab Link: [Google colab notebook](#)

Introduction:

Our project's goal is to create a machine learning model that can determine from an input image whether a person's face is completely covered, partially covered, not covered at all, or not a face at all. This project has a number of practical applications, including in the fields of public health and security, where it's crucial to ensure that rules regarding face coverings are followed.

In order to do this, we have investigated and used a number of machine learning models, such as a convolutional neural network (CNN), VGG-16 and InceptionV3.

In order to help with the enforcement of face covering laws, our goal is to develop a model with high accuracy and robustness.

Dataset Description:

The dataset "Face Mask Usage" is available on Kaggle and is created by James Arnold Nogra. It has data about how people use face masks when there is COVID-19. Some people use face masks right, some use them wrong, and some do not use them at all. The data set can help people who want to make machines that can tell who is using face masks and how. Wearing face masks can stop COVID-19 from spreading. This data set can show us how people wear face masks and if it works in different situations.





The reason why I chose this dataset is because this dataset offers a special chance to create a machine learning model that can determine whether people are wearing face masks in pictures or videos. For many different organizations, including hospitals, airports, and public transportation systems, this can be a useful tool.

In this report, we will look at how other people worked on this data set and what methods they followed for the implementation and share what we think we can do with this dataset.

Link to the dataset:

<https://www.kaggle.com/datasets/jamesnogra/face-mask-usage> Dataset contains

the following folders:

| Name | Date modified | Type | Size |
|---|------------------|-------------|------|
|  fully_covered | 3/4/2023 8:45 PM | File folder | |
|  not_covered | 3/4/2023 8:46 PM | File folder | |
|  not_face | 3/4/2023 8:46 PM | File folder | |
|  partially_covered | 3/4/2023 8:46 PM | File folder | |

- 1) 1,451- fully-covered images.
- 2) 3,664- not covered images.
- 3) 1,114- not face images.
- 4) 392- partially covered face images.

Tasks of Team Members:

1)Shashank Reddy Kallem:

For this project, I will be working on Implementing InceptionV3 and VGG-16. We are working together to implement the given tasks on Milestone2 report.

Apart from that, as per professor's suggestion. We are trying to work on 3 different models individually and compare the results on which model performs better on the given dataset.

We have gone through some research on what models can be more accurate and efficient when dealing with the Image classification. I am using Keras/Tensorflow to train my classification model. It is an ML and deep neural network Library which uses Convolutional Neural Networks.

We trained our model and at present I am trying to change the parameters to see how the accuracy of the model is changing by changing some parameters.

2)Suvagiya Brijeshkumar Ashokbhai :

In this project I'll be working on CNN(convolutional neural network) and video face mask detection implementation for this project. We are collaborating to carry out the tasks listed on the Milestone 2 report. In the Implementation of face mask detection, the input images can be preprocessed to a fixed size and converted to grayscale to reduce computational complexity. A CNN model can then be trained using the preprocessed images and their corresponding labels (with mask or without mask). The model can be evaluated on a separate testing set to measure

its performance. Once the model is trained and evaluated, it can be used to make predictions on new images by passing them through the network and outputting a classification result.

I am trying to train my model with the Activation functions like ReLU, sigmoid, and tanh. The activation functions used in the hidden layers can be specified.

Apart from that I have created some research to get maximum accuracy in this model and also create a confusion matrix that can give the idea about false positive, false negative, true positive and true negative results of prediction.

In the CNN model I am trying to implement Pooling layers like convolutional layer, pooling layer Flatten Layer and Dense Layer which work to help to reduce the size and complexity of the feature maps produced by the convolutional layers. This makes the network more efficient and less prone to overfitting, and can ultimately lead to better performance on the task at hand.

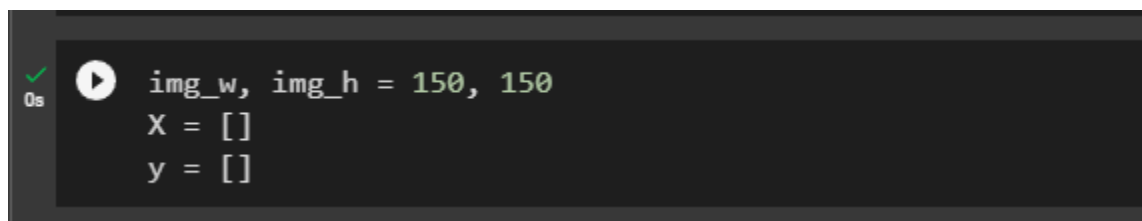
Also I will try to show The Receiver Operating Characteristic (ROC) curve that is a graphical representation of the performance of a binary classifier as the discrimination threshold is varied about the ratio of true positive rate and false positive rate.

3)VENKATA LEELA MANOHAR MORAMPUDI

- Preprocessed the dataset, including resizing, normalization, and data augmentation to improve model performance and generalization.
- Split the dataset into three subsets: training (70%), validation (15%), and testing (15%) to effectively train, validate, and evaluate the model.
- Model selection : ResNet50, Custom CNN (experimenting). Completed training the model on ResNet50.
- We got to know about some other model we are trying to implement Single Shot MultiBox Detector (SSD) and YOLO

InceptionV3 model:

Setting the image size to 150x, 150x



```
img_w, img_h = 150, 150
X = []
y = []
```

Creating two arrays and appending the images X and y

```
✓ [44] X = np.asarray(X)
0s      y = np.asarray(y)
      print(X.shape, y.shape)

      (6621, 150, 150, 3) (6621,)
```

Splitting the data into test, train and validation

```
✓ [47] x_train1, x_test, y_train1, y_test = train_test_split(x, labels, test_size = 0.3, random_state=5)
0s      x_train, x_val, y_train, y_val = train_test_split(x_train1, y_train1, test_size=0.3, random_state=5)

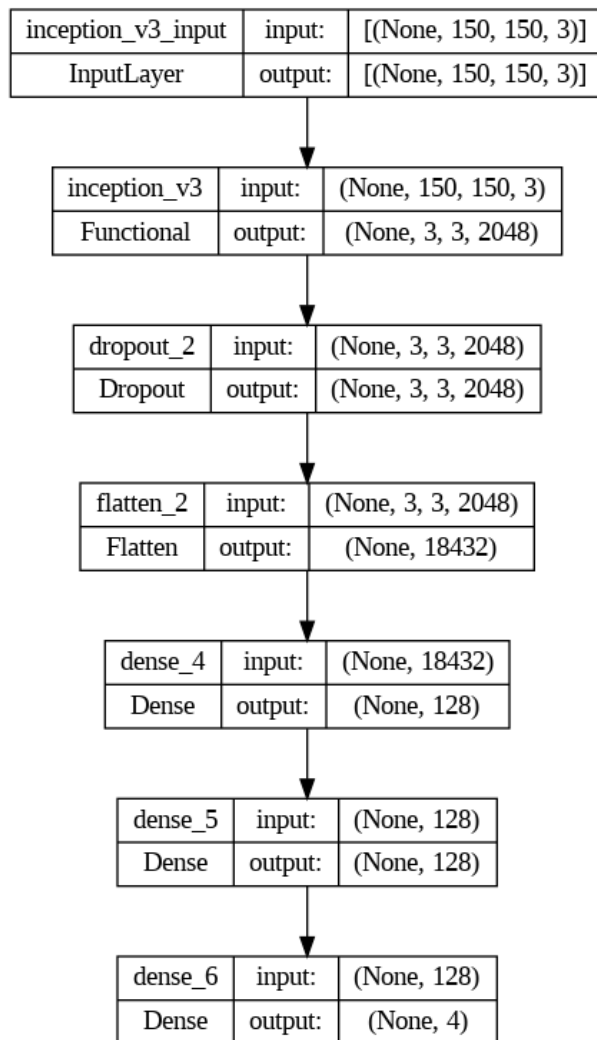
      print('Number of train: {}'.format(len(x_train)))
      print('Number of validation: {}'.format(len(x_val)))
      print('Number of test: {}'.format(len(x_test)))

      Number of train: 3243
      Number of validation: 1391
      Number of test: 1987
```

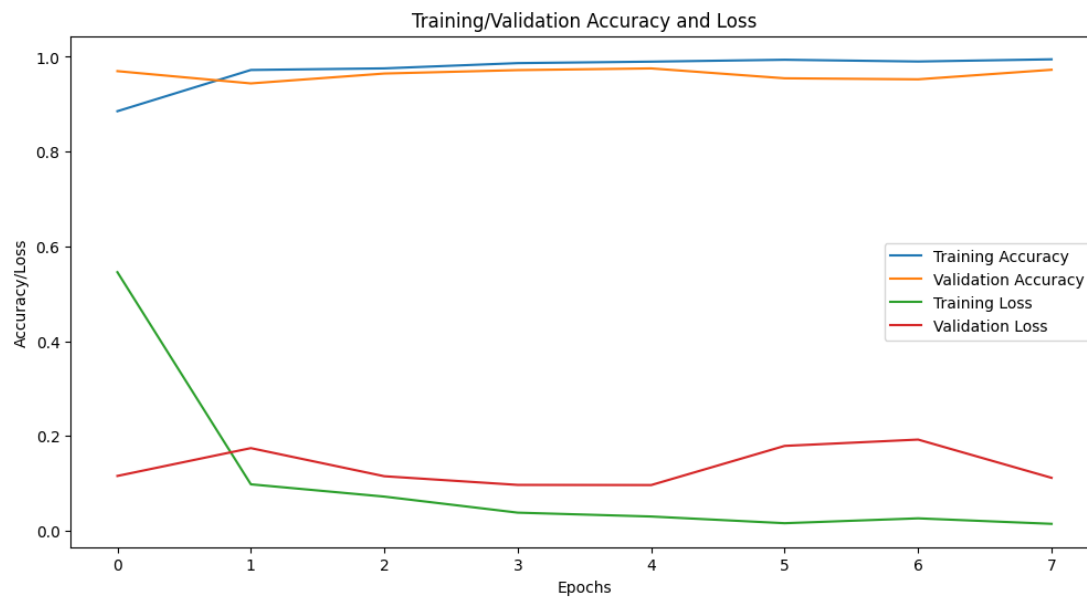
Freezing the last 10 layers of the InceptionV3 model because it is pre trained and the weights are adjusted.

```
✓ [49] #Freezing the layers of the model
0s      for layer in base_model.layers[:-10]:
          layer.trainable=False
```

The below diagram shows the visualization of the neural network.



Result:

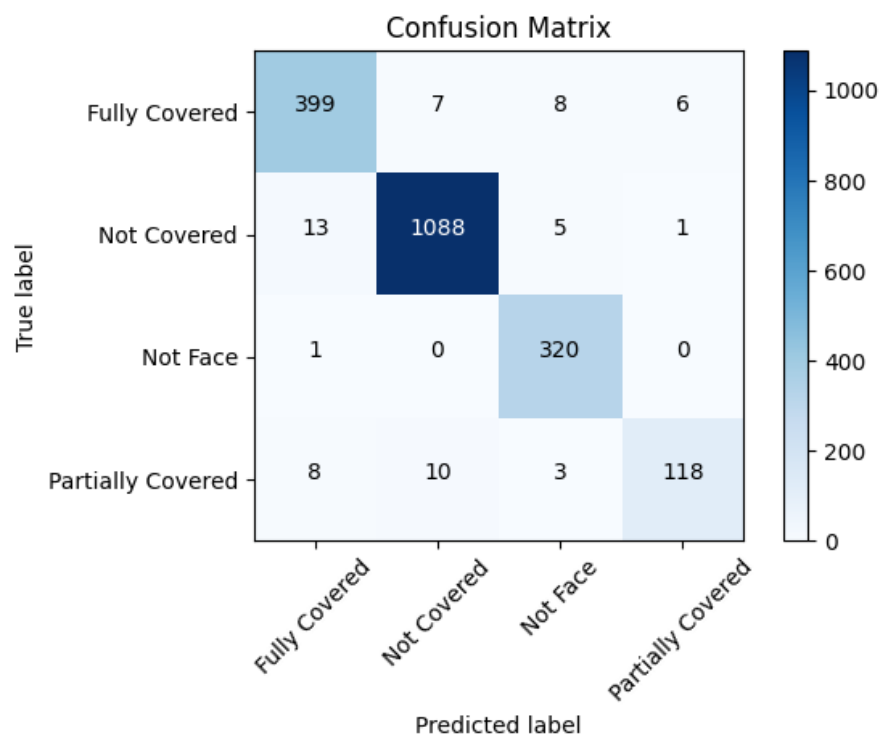


loss: 1.48%

accuracy: 99.48%

precision: 99.51%

recall: 99.48%



Classification Report

| | precision | recall | f1-score | support |
|-------------------|-----------|--------|----------|---------|
| Fully Covered | 0.95 | 0.95 | 0.95 | 420 |
| Not Covered | 0.98 | 0.98 | 0.98 | 1107 |
| Not Face | 0.95 | 1.00 | 0.97 | 321 |
| Partially Covered | 0.94 | 0.85 | 0.89 | 139 |
| accuracy | | | 0.97 | 1987 |
| macro avg | 0.96 | 0.94 | 0.95 | 1987 |
| weighted avg | 0.97 | 0.97 | 0.97 | 1987 |

VGG-16:

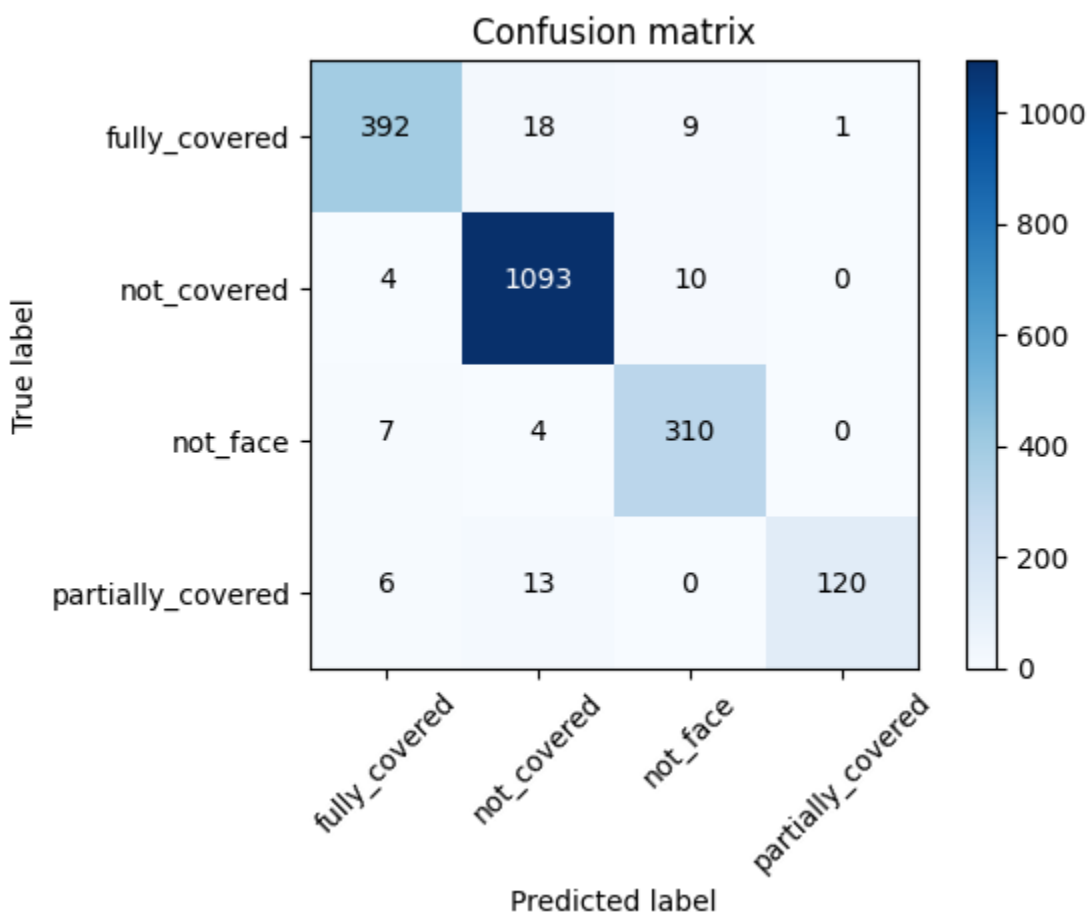
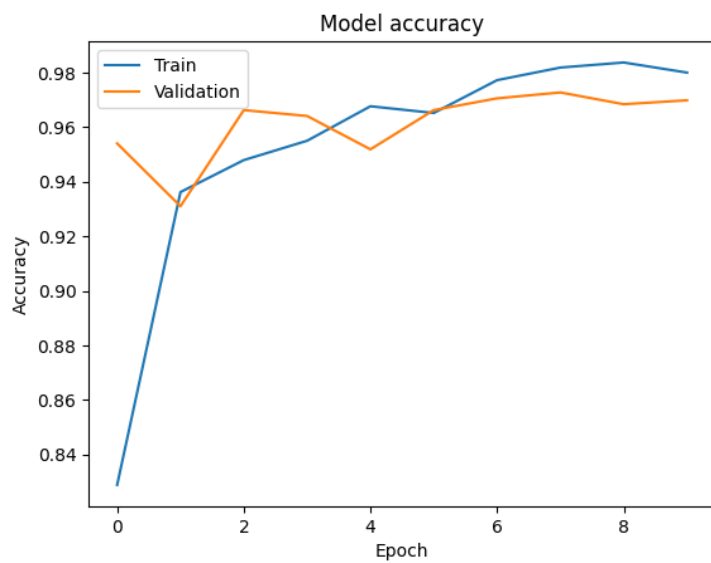
```
[12] # Train the model
history = model.fit(x_train, y_train, batch_size=32, epochs=10, validation_data=(x_val, y_val))

Epoch 1/10
102/102 [=====] - 26s 134ms/step - loss: 0.4838 - accuracy: 0.8289 - val_loss: 0.1526 - val_accuracy: 0.9540
Epoch 2/10
102/102 [=====] - 12s 116ms/step - loss: 0.1813 - accuracy: 0.9362 - val_loss: 0.1899 - val_accuracy: 0.9310
Epoch 3/10
102/102 [=====] - 12s 117ms/step - loss: 0.1544 - accuracy: 0.9479 - val_loss: 0.0980 - val_accuracy: 0.9662
Epoch 4/10
102/102 [=====] - 10s 95ms/step - loss: 0.1283 - accuracy: 0.9550 - val_loss: 0.1048 - val_accuracy: 0.9641
Epoch 5/10
102/102 [=====] - 12s 119ms/step - loss: 0.1008 - accuracy: 0.9676 - val_loss: 0.1421 - val_accuracy: 0.9518
Epoch 6/10
102/102 [=====] - 12s 118ms/step - loss: 0.0965 - accuracy: 0.9652 - val_loss: 0.1018 - val_accuracy: 0.9662
Epoch 7/10
102/102 [=====] - 10s 96ms/step - loss: 0.0705 - accuracy: 0.9772 - val_loss: 0.0885 - val_accuracy: 0.9705
Epoch 8/10
102/102 [=====] - 9s 93ms/step - loss: 0.0646 - accuracy: 0.9818 - val_loss: 0.0808 - val_accuracy: 0.9727
Epoch 9/10
102/102 [=====] - 9s 93ms/step - loss: 0.0525 - accuracy: 0.9837 - val_loss: 0.0908 - val_accuracy: 0.9684
Epoch 10/10
102/102 [=====] - 9s 93ms/step - loss: 0.0607 - accuracy: 0.9800 - val_loss: 0.0849 - val_accuracy: 0.9698
```

```
# Evaluate the model
scores = model.evaluate(x_test, y_test, verbose=1)
print('Test loss:', scores[0])
print('Test accuracy:', scores[1])

63/63 [=====] - 5s 74ms/step - loss: 0.1267 - accuracy: 0.9638
Test loss: 0.1267203986644745
Test accuracy: 0.963764488697052
```

RESULT:



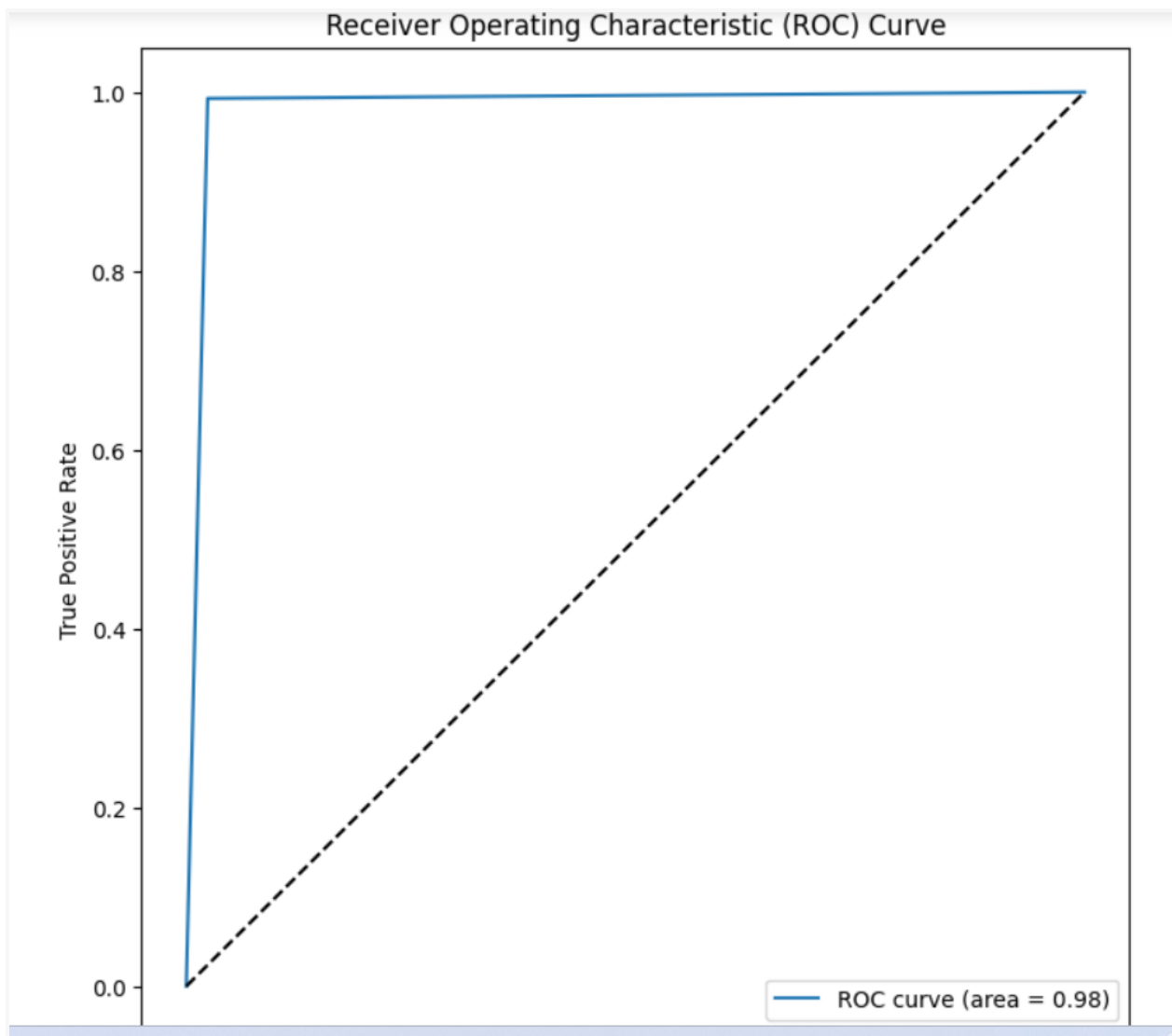
CNN :- CNNs typically consist of multiple layers, including convolutional layers, pooling layers, and fully connected layers. Convolutional layers perform the feature extraction by applying a series of filters to the input data. Pooling layers reduce the dimensionality of the output from the convolutional layers. Finally, fully connected layers process the output from the convolutional and pooling layers to produce the final predictions.

```
In [5]: # Evaluate model
test_loss, test_acc = model.evaluate(test_data, test_labels)
print('Test accuracy:', test_acc)

12/12 [=====] - 0s 23ms/step - loss: 0.0401 - accuracy: 0.9892
Test accuracy: 0.989159882068634
```

Accuracy=98.91%

ROC Curve:

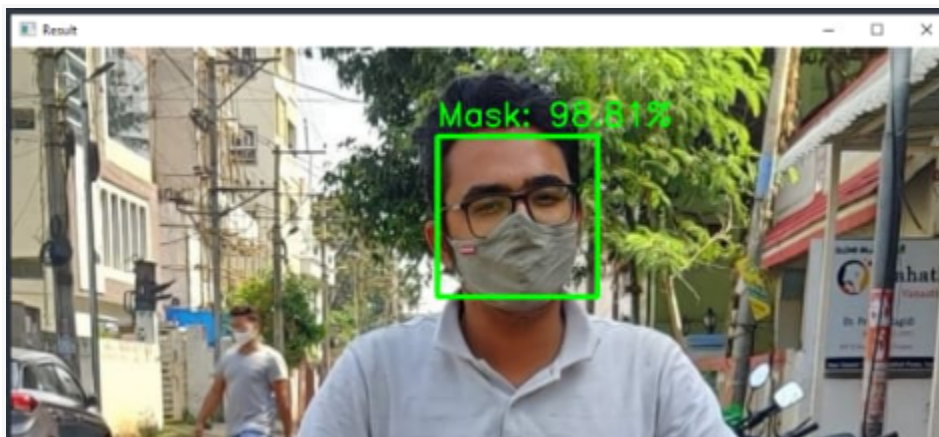


Perform experiment on video face mask detection:-

- Now we create the live video face mask detection in this case and we use default OpenCV's haar cascade classifier for square box in face and load the trained model. using the cv2 start the camera and scan the face and give the output in terms of "Mask" and "No mask".
- someTimes this model predicts false predictions and gives the wrong output which is shown below. I have tried many times but it gives wrong outputs many times.

```
In [2]:  
  
# take the trained model  
model = load_model('face_mask.h5')  
  
# create the class label  
class_labels = {1: 'Mask', 0: 'No mask'}  
  
# start the camera  
cap = cv2.VideoCapture(0)  
  
# Set the box's color and the font used to show the label  
font = cv2.FONT_HERSHEY_SIMPLEX  
font_scale = 0.5  
thickness = 1  
box_color = (0, 255, 0)  
  
while True:  
    # Take a picture with the camera  
    ret, frame = cap.read()  
  
    # Preprocess the image  
    image = cv2.resize(frame, (64, 64))  
    image = image / 255.0  
    image = np.expand_dims(image, axis=0)  
  
    # Predict the label of the image  
    prediction = model.predict(image)  
    label = np.argmax(prediction, axis=1)[0]  
    label_text = class_labels[label]  
  
    # Detect faces in the image using OpenCV's haarcascade classifier  
    face_cascade = cv2.CascadeClassifier('haarcascade_frontalface_default.xml')  
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)  
    faces = face_cascade.detectMultiScale(gray, scaleFactor=1.1, minNeighbors=5, minSize=(30, 30), flags=cv2.CASCADE_SCALE_IMAGE)
```

OUTPUT:



With mask



Without mask

Summary :- The main characteristic of CNNs is that they apply a convolutional operation on the input data to extract features, which are then used to make predictions. This allows the network to learn patterns in the input data that are relevant to the task at hand, such as object recognition or classification. A Convolutional Neural Network (CNN) is a type of neural network used in deep learning for image and video processing. CNNs extract features from input data using convolutional layers, which apply a series of filters to the input. These features are then processed by pooling layers to reduce dimensionality and fully connected layers to produce predictions.