Model For Detecting Face Masks in Static Images and Videos

Importing all the necessary libraries

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Dropout, Flatten, Dense
from tensorflow.keras.optimizers import Adam
from tensorflow.keras.preprocessing.image import ImageDataGenerator
import numpy as np
import tensorflow as tf
import matplotlib.pyplot as plt
```

Downloading the dataset directly from Kaggle

```
import os
os.environ['KAGGLE_USERNAME']='taarusshwadhwa'
os.environ['KAGGLE_KEY']='f2ef5c63de2d4d9d7d2f5b93735d0395'
```

```
! kaggle datasets download -d prithwirajmitra/covid-face-mask-detection-dataset
```

```
Downloading covid-face-mask-detection-dataset.zip to /content
 98% 204M/207M [00:10<00:00, 24.4MB/s]
100% 207M/207M [00:10<00:00, 21.4MB/s]
```

Whenever we download a dataset from kaggle directly, it is always in the .zip format. We need to unzip this folder to access all the images of our data.

```
! unzip covid-face-mask-detection-dataset.zip
```

```
Archive:  covid-face-mask-detection-dataset.zip
  inflating: New Masks Dataset/Test/Mask/2070.jpg
```

```
  inflating: New Masks Dataset/Test/Mask/2072.jpg
  inflating: New Masks Dataset/Test/Mask/2083.jpg
  inflating: New Masks Dataset/Test/Mask/2085.jpg
  inflating: New Masks Dataset/Test/Mask/2086.jpg
  inflating: New Masks Dataset/Test/Mask/2110.jpg
  inflating: New Masks Dataset/Test/Mask/2113.jpg
  inflating: New Masks Dataset/Test/Mask/2114.jpeg
  inflating: New Masks Dataset/Test/Mask/2130.jpg
  inflating: New Masks Dataset/Test/Mask/2132.jpg
  inflating: New Masks Dataset/Test/Mask/2135.jpg
  inflating: New Masks Dataset/Test/Mask/2154.png
  inflating: New Masks Dataset/Test/Mask/2158.png
  inflating: New Masks Dataset/Test/Mask/2159.png
  inflating: New Masks Dataset/Test/Mask/2160.png
  inflating: New Masks Dataset/Test/Mask/2170.png
  inflating: New Masks Dataset/Test/Mask/2172.png
  inflating: New Masks Dataset/Test/Mask/2173.png
  inflating: New Masks Dataset/Test/Mask/2174.png
  inflating: New Masks Dataset/Test/Mask/2175.png
  inflating: New Masks Dataset/Test/Mask/2176.png
  inflating: New Masks Dataset/Test/Mask/2178.png
  inflating: New Masks Dataset/Test/Mask/2179.png
  inflating: New Masks Dataset/Test/Mask/2182.png
  inflating: New Masks Dataset/Test/Mask/2187.png
  inflating: New Masks Dataset/Test/Mask/2188.png
  inflating: New Masks Dataset/Test/Mask/2190.png
  inflating: New Masks Dataset/Test/Mask/2191.png
  inflating: New Masks Dataset/Test/Mask/2203.png
  inflating: New Masks Dataset/Test/Mask/2205.png
  inflating: New Masks Dataset/Test/Mask/2212.png
  inflating: New Masks Dataset/Test/Mask/2215.png
  inflating: New Masks Dataset/Test/Mask/2222.png
  inflating: New Masks Dataset/Test/Mask/2225.png
  inflating: New Masks Dataset/Test/Mask/2236.png
  inflating: New Masks Dataset/Test/Mask/2245.png
  inflating: New Masks Dataset/Test/Mask/2250.png
  inflating: New Masks Dataset/Test/Mask/2251.png
  inflating: New Masks Dataset/Test/Mask/2252.png
  inflating: New Masks Dataset/Test/Mask/2254.png
  inflating: New Masks Dataset/Test/Mask/2257.png
  inflating: New Masks Dataset/Test/Mask/2260.png
  inflating: New Masks Dataset/Test/Mask/2263.png
  inflating: New Masks Dataset/Test/Mask/2264.png
  inflating: New Masks Dataset/Test/Mask/2265.png
```

```
  inflating: New Masks Dataset/Test/Mask/2267.png
  inflating: New Masks Dataset/Test/Mask/2268.png
  inflating: New Masks Dataset/Test/Mask/2271.png
  inflating: New Masks Dataset/Test/Mask/2283.png
  inflating: New Masks Dataset/Test/Mask/2300.png
  inflating: New Masks Dataset/Test/Non Mask/real_01032.jpg
  inflating: New Masks Dataset/Test/Non Mask/real_01033.jpg
  inflating: New Masks Dataset/Test/Non Mask/real_01034.jpg
  inflating: New Masks Dataset/Test/Non Mask/real_01035.jpg
  inflating: New Masks Dataset/Test/Non Mask/real_01036.jpg
  inflating: New Masks Dataset/Test/Non Mask/real_01037.jpg
  inflating: New Masks Dataset/Test/Non Mask/real_01038.jpg
```

Declare all the paths for easier access to the respective datasets later on.

```
main_dir = '/content/New Masks Dataset'
train_dir = os.path.join(main_dir,'Train')
test_dir = os.path.join(main_dir,'Test')
validation_dir = os.path.join(main_dir,'Validation')

train_mask_dir = os.path.join(train_dir, 'Mask')
train_nomask_dir = os.path.join(train_dir, 'Non Mask')


train_mask_names = os.listdir(train_mask_dir)
print(train_mask_names[:10])

train_nomask_names = os.listdir(train_nomask_dir)
print(train_nomask_names[:10])
```

```
['0170.jpg', '1558.png', '0725.jpg', '0427.jpg', '0592.jpg', '0536.jpg', '0027.jpg', '0171.jpg', '0901.jpeg', '0286.jpg']
['23.jpg', '332.jpg', '204.jpg', '248.jpg', '106.jpg', '232.jpg', '22.jpg', '49.jpg', '344.jpg', '127.jpg']
```

Visualising Images

We will be visualising 16 images each from the training set:

- 8 images with masks
- 8 images without masks

```
import matplotlib.image as mpimg
nrows = 4
ncols = 4
plt.figure(figsize=(12,12))

mask_pic=[]
for i in train_mask_names[0:8]:
  mask_pic.append(os.path.join(train_mask_dir,i))

nomask_pic=[]
for i in train_nomask_names[0:8]:
  nomask_pic.append(os.path.join(train_nomask_dir,i))

print(mask_pic)
print(nomask_pic)

merged_list = mask_pic + nomask_pic

for i in range(0, len(merged_list)):
  data = merged_list[i].split('/', 4)[4]
  sp = plt.subplot(nrows, ncols, i+1)
  sp.axis('Off')
  image = mpimg.imread(merged_list[i])
  sp.set_title(data, fontsize=8)
  plt.imshow(image, cmap='Blues')
```

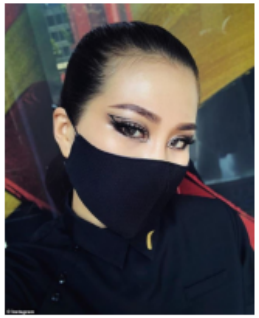
```
['/content/New Masks Dataset/Train/Mask/0170.jpg', '/content/New Masks Dataset/Train/Mask/1558.png', '/
['/content/New Masks Dataset/Train/Non Mask/23.jpg', '/content/New Masks Dataset/Train/Non Mask/332.jpg
```

Mask/0170.jpg

Mask/1558.png

Mask/0725.jpg

Mask/0427.jpg



Mask/0592.jpg

Mask/0536.jpg

Mask/0027.jpg

Mask/0171.jpg

Non Mask/23.jpg

Non Mask/332.jpg

Non Mask/204.jpg

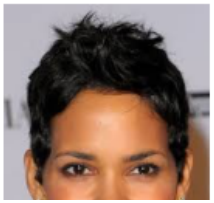
Non Mask/248.jpg

Non Mask/106.jpg

Non Mask/232.jpg

Non Mask/22.jpg

Non Mask/49.jpg



Data Augmentation

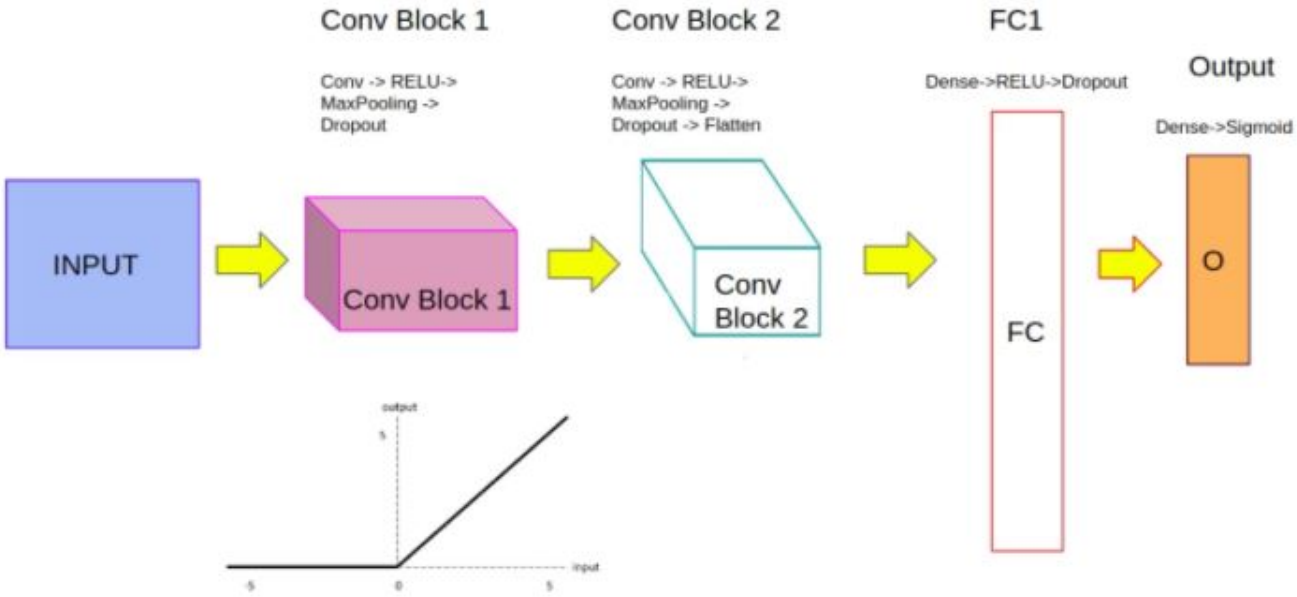
The total number of images in the training set is only 600 images. We want to increase our training set to improve the accuracy of our model.
To do this, we are flipping each of the images in the training dataset horizontally, amongst other operations. We can also, zoom in to a particular picture, blur it out, flip it vertically etc.
Read up more on Data Augmentation to understand it's purpose and impact.

```
train_datagen = ImageDataGenerator(rescale=1./255,
                                   zoom_range = 0.2,
                                   rotation_range = 40,
                                   horizontal_flip = True
                                   )

val_datagen = ImageDataGenerator(rescale = 1./255)
test_datagen = ImageDataGenerator(rescale=1./255)

train_generator = train_datagen.flow_from_directory(train_dir,
                                                    target_size=(150,150),
                                                    batch_size=32,
                                                    class_mode='binary'
                                                    )

val_generator = val_datagen.flow_from_directory(validation_dir,
                                                target_size=(150,150),
                                                batch_size=32,
                                                class_mode='binary'
                                                )

test_generator = test_datagen.flow_from_directory(test_dir,
                                                    target_size=(150,150),
                                                    batch_size=32,
                                                    class_mode='binary'
                                                    )
```

```
Found 600 images belonging to 2 classes.
Found 306 images belonging to 2 classes.
Found 100 images belonging to 2 classes.
```

Building our CNN Model

The model we'll be using has been explained with the help of a flow diagram below.




```
model = Sequential()
model.add(Conv2D(32, (3,3), padding='SAME', activation='relu', input_shape=(150,150,3)))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.5))
model.add(Conv2D(64, (3,3), padding='SAME', activation='relu'))
model.add(MaxPooling2D(pool_size=(2,2)))
model.add(Dropout(0.5))
```

```
model.add(Flatten())
model.add(Dense(256, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(1, activation='sigmoid'))


model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 150, 150, 32)      896

 max_pooling2d (MaxPooling2D  (None, 75, 75, 32)       0
 )

 dropout (Dropout)           (None, 75, 75, 32)        0

 conv2d_1 (Conv2D)           (None, 75, 75, 64)        18496

 max_pooling2d_1 (MaxPooling  (None, 37, 37, 64)       0
 2D)

 dropout_1 (Dropout)         (None, 37, 37, 64)        0

 flatten (Flatten)           (None, 87616)             0

 dense (Dense)               (None, 256)               22429952

 dropout_2 (Dropout)         (None, 256)               0

 dense_1 (Dense)             (None, 1)                 257

=================================================================
Total params: 22,449,601
Trainable params: 22,449,601
Non-trainable params: 0
_________________________________________________________________
```

Compile the built model to configure it for training.

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

```
WARNING:absl:`lr` is deprecated in Keras optimizer, please use `learning_rate` or use the legacy optimizer, e.g.,tf.keras.optimizers.lega
```

Train the CNN Model

```
history = model.fit(train_generator,
                    epochs = 30,
                    validation_data = val_generator)
```

```
Epoch 1/30
19/19 [==============================] - 23s 715ms/step - loss: 3.2820 - accuracy: 0.5317 - val_loss: 0.6931 - val_accuracy: 0.5000
Epoch 2/30
19/19 [==============================] - 18s 986ms/step - loss: 0.6629 - accuracy: 0.6033 - val_loss: 0.6745 - val_accuracy: 0.7288
Epoch 3/30
19/19 [==============================] - 13s 677ms/step - loss: 0.4764 - accuracy: 0.8017 - val_loss: 0.5064 - val_accuracy: 0.8170
Epoch 4/30
19/19 [==============================] - 13s 684ms/step - loss: 0.3526 - accuracy: 0.8767 - val_loss: 0.5052 - val_accuracy: 0.8137
Epoch 5/30
19/19 [==============================] - 13s 674ms/step - loss: 0.3306 - accuracy: 0.8750 - val_loss: 0.3388 - val_accuracy: 0.8922
Epoch 6/30
19/19 [==============================] - 20s 1s/step - loss: 0.2754 - accuracy: 0.8967 - val_loss: 0.3013 - val_accuracy: 0.9150
Epoch 7/30
19/19 [==============================] - 13s 690ms/step - loss: 0.2767 - accuracy: 0.9050 - val_loss: 0.3156 - val_accuracy: 0.8922
Epoch 8/30
19/19 [==============================] - 13s 689ms/step - loss: 0.2422 - accuracy: 0.9133 - val_loss: 0.2713 - val_accuracy: 0.8987
Epoch 9/30
19/19 [==============================] - 13s 695ms/step - loss: 0.2417 - accuracy: 0.9100 - val_loss: 0.2454 - val_accuracy: 0.9216
Epoch 10/30
19/19 [==============================] - 13s 676ms/step - loss: 0.2027 - accuracy: 0.9350 - val_loss: 0.2707 - val_accuracy: 0.8954
Epoch 11/30
19/19 [==============================] - 12s 649ms/step - loss: 0.2371 - accuracy: 0.9117 - val_loss: 0.2667 - val_accuracy: 0.9183
Epoch 12/30
19/19 [==============================] - 13s 672ms/step - loss: 0.2033 - accuracy: 0.9333 - val_loss: 0.2134 - val_accuracy: 0.9183
Epoch 13/30
19/19 [==============================] - 13s 680ms/step - loss: 0.2137 - accuracy: 0.9267 - val_loss: 0.2428 - val_accuracy: 0.9118
Epoch 14/30
19/19 [==============================] - 13s 689ms/step - loss: 0.2388 - accuracy: 0.9067 - val_loss: 0.3240 - val_accuracy: 0.8725
Epoch 15/30
```

```
19/19 [==============================] - 13s 713ms/step - loss: 0.2401 - accuracy: 0.9183 - val_loss: 0.2359 - val_accuracy: 0.9052
Epoch 16/30
19/19 [==============================] - 13s 688ms/step - loss: 0.2178 - accuracy: 0.9200 - val_loss: 0.2975 - val_accuracy: 0.8693
Epoch 17/30
19/19 [==============================] - 13s 693ms/step - loss: 0.1891 - accuracy: 0.9267 - val_loss: 0.2239 - val_accuracy: 0.9118
Epoch 18/30
19/19 [==============================] - 13s 681ms/step - loss: 0.1862 - accuracy: 0.9383 - val_loss: 0.2377 - val_accuracy: 0.9052
Epoch 19/30
19/19 [==============================] - 13s 682ms/step - loss: 0.1750 - accuracy: 0.9400 - val_loss: 0.2124 - val_accuracy: 0.9183
Epoch 20/30
19/19 [==============================] - 13s 684ms/step - loss: 0.2263 - accuracy: 0.9333 - val_loss: 0.2881 - val_accuracy: 0.9020
Epoch 21/30
19/19 [==============================] - 13s 688ms/step - loss: 0.2253 - accuracy: 0.9100 - val_loss: 0.1988 - val_accuracy: 0.9314
Epoch 22/30
19/19 [==============================] - 13s 683ms/step - loss: 0.2238 - accuracy: 0.9250 - val_loss: 0.2461 - val_accuracy: 0.9150
Epoch 23/30
19/19 [==============================] - 13s 688ms/step - loss: 0.1852 - accuracy: 0.9367 - val_loss: 0.2250 - val_accuracy: 0.9118
Epoch 24/30
19/19 [==============================] - 13s 670ms/step - loss: 0.1853 - accuracy: 0.9317 - val_loss: 0.2447 - val_accuracy: 0.9052
Epoch 25/30
19/19 [==============================] - 13s 693ms/step - loss: 0.1525 - accuracy: 0.9483 - val_loss: 0.2348 - val_accuracy: 0.8954
Epoch 26/30
19/19 [==============================] - 13s 682ms/step - loss: 0.2071 - accuracy: 0.9117 - val_loss: 0.2417 - val_accuracy: 0.8954
Epoch 27/30
19/19 [==============================] - 13s 683ms/step - loss: 0.1832 - accuracy: 0.9217 - val_loss: 0.1961 - val_accuracy: 0.9183
Epoch 28/30
19/19 [==============================] - 13s 680ms/step - loss: 0.1754 - accuracy: 0.9383 - val_loss: 0.2189 - val_accuracy: 0.9150
Epoch 29/30
19/19 [==============================] - 13s 675ms/step - loss: 0.1650 - accuracy: 0.9450 - val_loss: 0.2031 - val_accuracy: 0.9216
```

```
history.history.keys()
```

```
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```

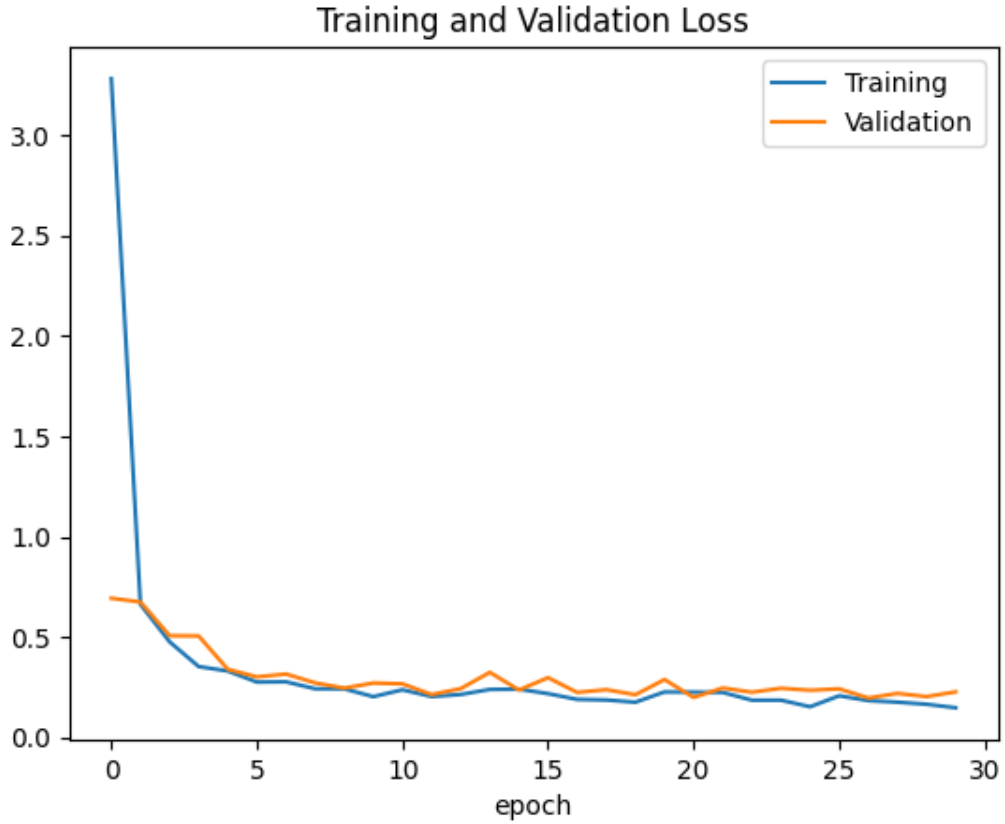
Evaluating the CNN Model

Below, we plot the Training vs Validation Loss and Training vs Validation Accuracy.

If the distance between the two curves is substantial, the model is not acceptable.

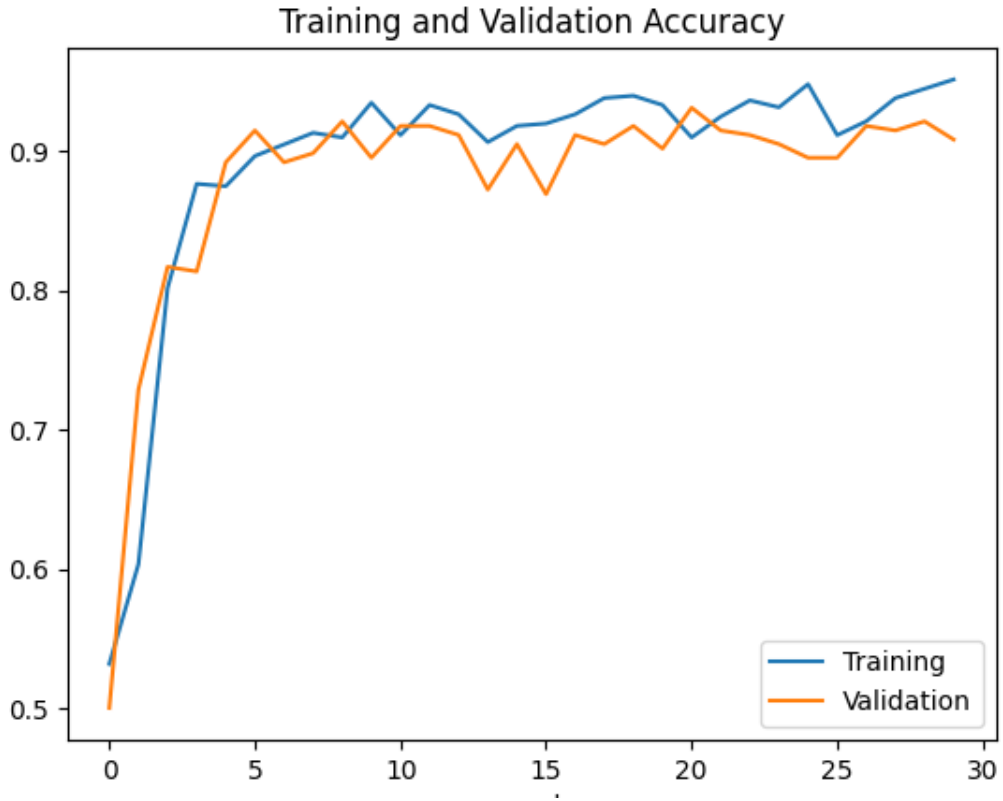
```
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.legend(['Training', 'Validation'])
plt.title('Training and Validation Loss')
plt.xlabel('epoch')
```

```
Text(0.5, 0, 'epoch')
```




```
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.legend(['Training', 'Validation'])
plt.title('Training and Validation Accuracy')
plt.xlabel('epoch')
```

```
Text(0.5, 0, 'epoch')
```



Evaluating our CNN Model

First, we evaluate the CNN Model on our test dataset.

```
test_loss, test_accuracy = model.evaluate(test_generator)
print('test loss:{} test accuracy:{}'.format(test_loss, test_accuracy))
```

```
4/4 [==============================] - 1s 237ms/step - loss: 0.1891 - accuracy: 0.9400
test loss:0.1891394406557083 test accuracy:0.9399999976158142
```

We achieved an accuracy of 83%. However, this can still be improved. Perhaps adding more images to the training set would help, or maybe tuning some hyperparameters may improve the accuracy on the test set as well.

Next, we will test our model on pictures of our own choice.

We can upload the pictures from our device using the colab function.

```
from google.colab import files
from keras.preprocessing import image
uploaded = files.upload()
for fname in uploaded.keys():
  img_path = '/content/'+fname
  img = tf.keras.utils.load_img(img_path, target_size=(150,150))
  images = tf.keras.utils.img_to_array(img)
  images = np.expand_dims(images, axis=0)
  prediction = model.predict(images)
  print(fname)
  if prediction==0:
    print('Mask!')
  else:
    print('No Mask!')
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving real_01033.jpg to real_01033.jpg
1/1 [==============================] - 0s 177ms/step
real_01033.jpg
No Mask!
```

The model was able to predict accurately.

We will use this model in another python function where we will use OpenCV to detect face masks in a video.

As our last step, let's save this model which we can download.

```
model.save('model.h5')
```

```
import cv2
import tensorflow as tf
from tensorflow.keras.models import load_model
#from tf.keras.preprocessing.image import load_img , img_to_array
from google.colab.patches import cv2_imshow
```

```
import numpy as np

model =load_model('/content/model.h5')

img_width , img_height = 150,150

face_cascade = cv2.CascadeClassifier('/content/haarcascade_frontalface_default.xml')

cap = cv2.VideoCapture('/content/video.mp4')

img_count_full = 0

font = cv2.FONT_HERSHEY_SIMPLEX
org = (1,1)
class_label = ''
fontScale = 1
color = (255,0,0)
thickness = 2

while True:
  img_count_full += 1
  response , color_img = cap.read()

  if response == False:
    break


  scale = 50
  width = int(color_img.shape[1]*scale /100)
  height = int(color_img.shape[0]*scale/100)
  dim = (width,height)

  color_img = cv2.resize(color_img, dim ,interpolation= cv2.INTER_AREA)

  gray_img = cv2.cvtColor(color_img,cv2.COLOR_BGR2GRAY)

  faces = face_cascade.detectMultiScale(gray_img, 1.1, 6)

  img_count = 0
  for (x,y,w,h) in faces:
```

```
        org = (x-10,y-10)
        img_count += 1
        color_face = color_img[y:y+h,x:x+w]
        cv2.imwrite('/content/Detected_Faces/%d%dface.jpg'%(img_count_full,img_count),color_face)
        img = tf.keras.utils.load_img('/content/Detected_Faces/%d%dface.jpg'%(img_count_full,img_count),target_size=(img_width,img_height))
        img = tf.keras.utils.img_to_array(img)
        img = np.expand_dims(img,axis=0)
        prediction = model.predict(img)


        if prediction==0:
          class_label = "Mask"
          color = (255,0,0)

        else:
          class_label = "No Mask"
          color = (0,255,0)


        cv2.rectangle(color_img,(x,y),(x+w,y+h),(0,0,255),3)
        cv2.putText(color_img, class_label, org, font ,fontScale, color, thickness,cv2.LINE_AA)

      cv2_imshow(color_img)
      if cv2.waitKey(1) & 0xFF == ord('q'):
        break

    cap.release()
    cv2.destroyAllWindows()
```

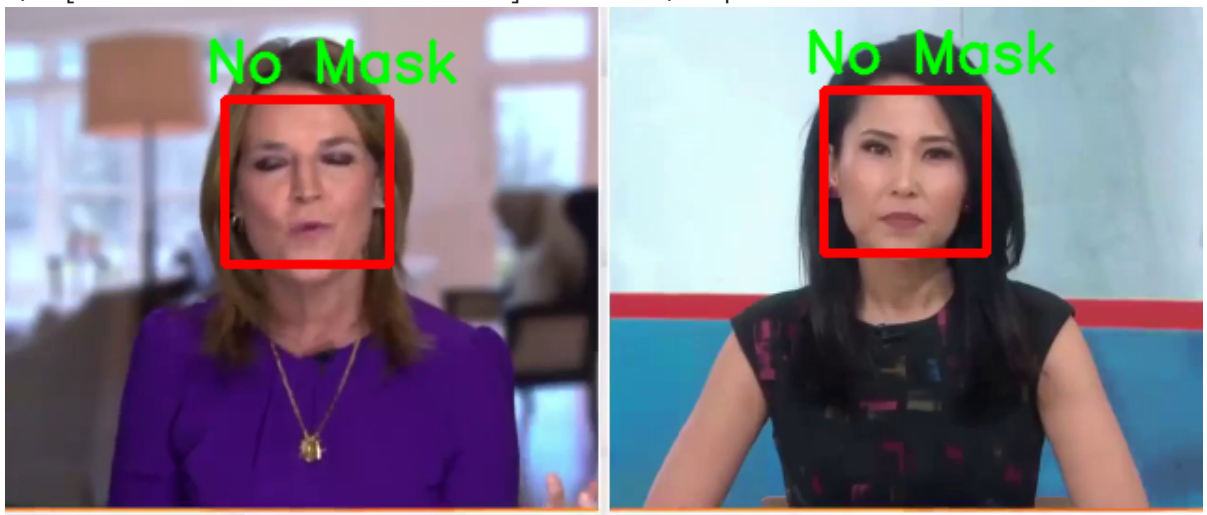
```
1/1 [==============================] - 2s 2s/step
1/1 [==============================] - 0s 20ms/step
```


```
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 22ms/step
```

```
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 28ms/step
```

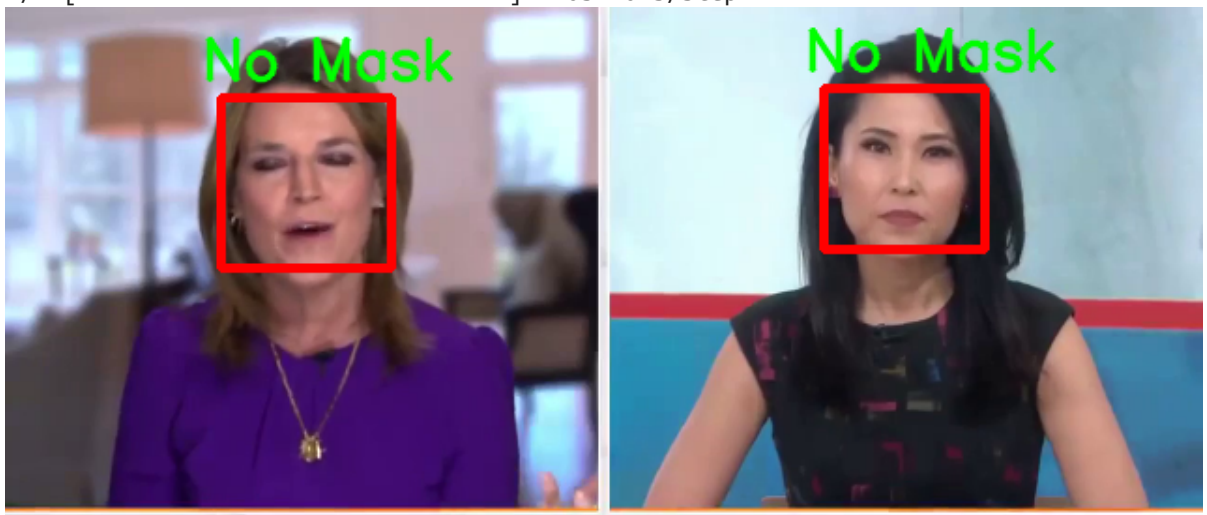

```
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 18ms/step
```


```
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 19ms/step
```

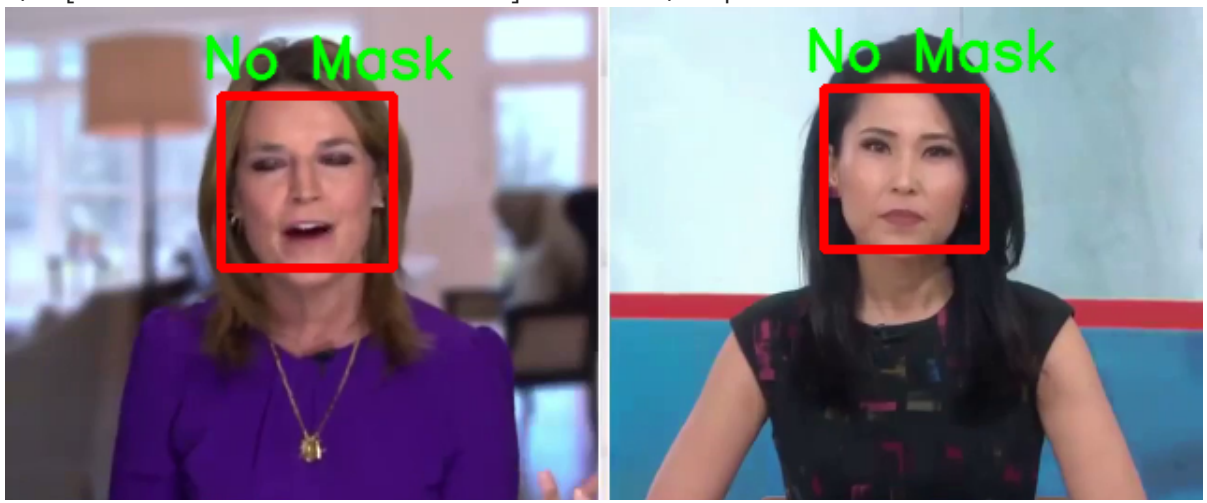



```
1/1 [==============================] - 0s 18ms/step
1/1 [==============================] - 0s 22ms/step
```

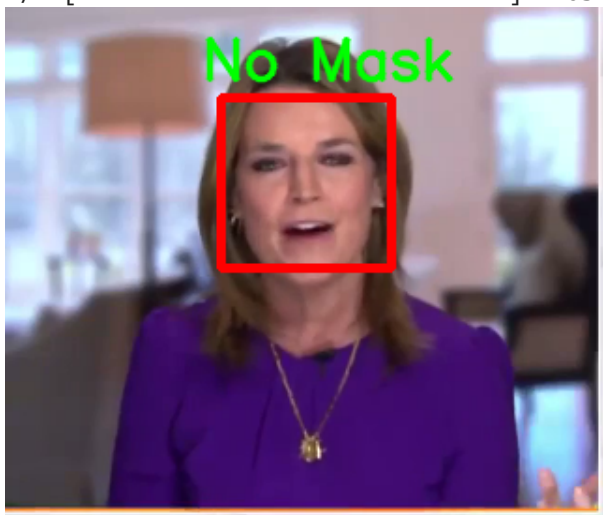

```
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 20ms/step
```



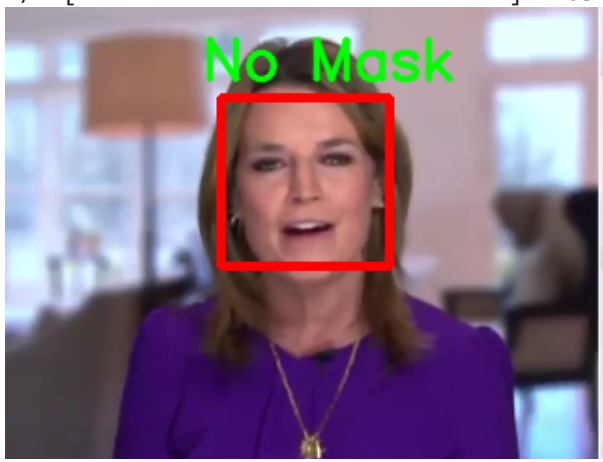

```
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 19ms/step
```

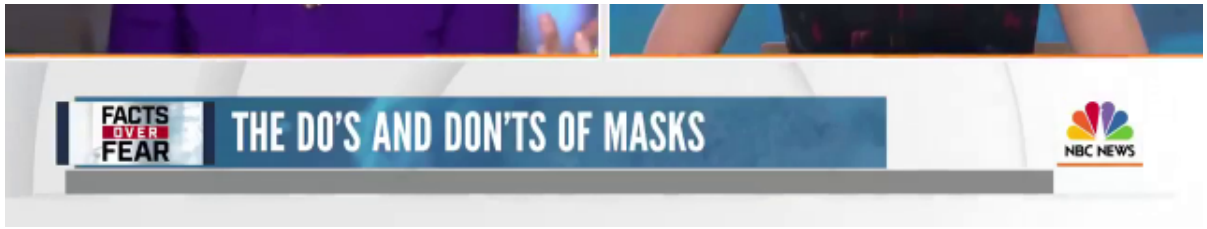



```
1/1 [==============================] - 0s 27ms/step
1/1 [==============================] - 0s 25ms/step
```




```
1/1 [==============================] - 0s 23ms/step
1/1 [==============================] - 0s 18ms/step
```



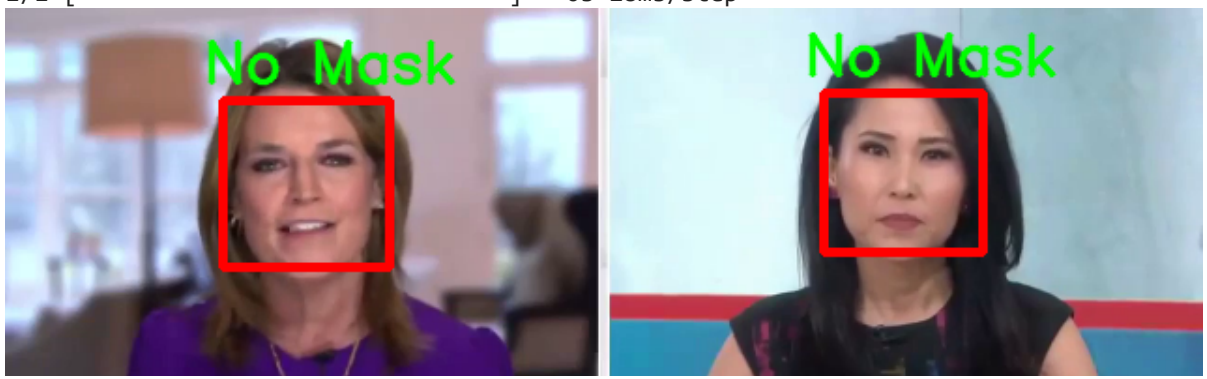



```
1/1 [==============================] - 0s 21ms/step
1/1 [==============================] - 0s 24ms/step
```


```
1/1 [==============================] - 0s 18ms/step
1/1 [==============================] - 0s 21ms/step
```


```
1/1 [==============================] - 0s 22ms/step
1/1 [==============================] - 0s 27ms/step
```


```
1/1 [==============================] - 0s 27ms/step
1/1 [==============================] - 0s 28ms/step
```



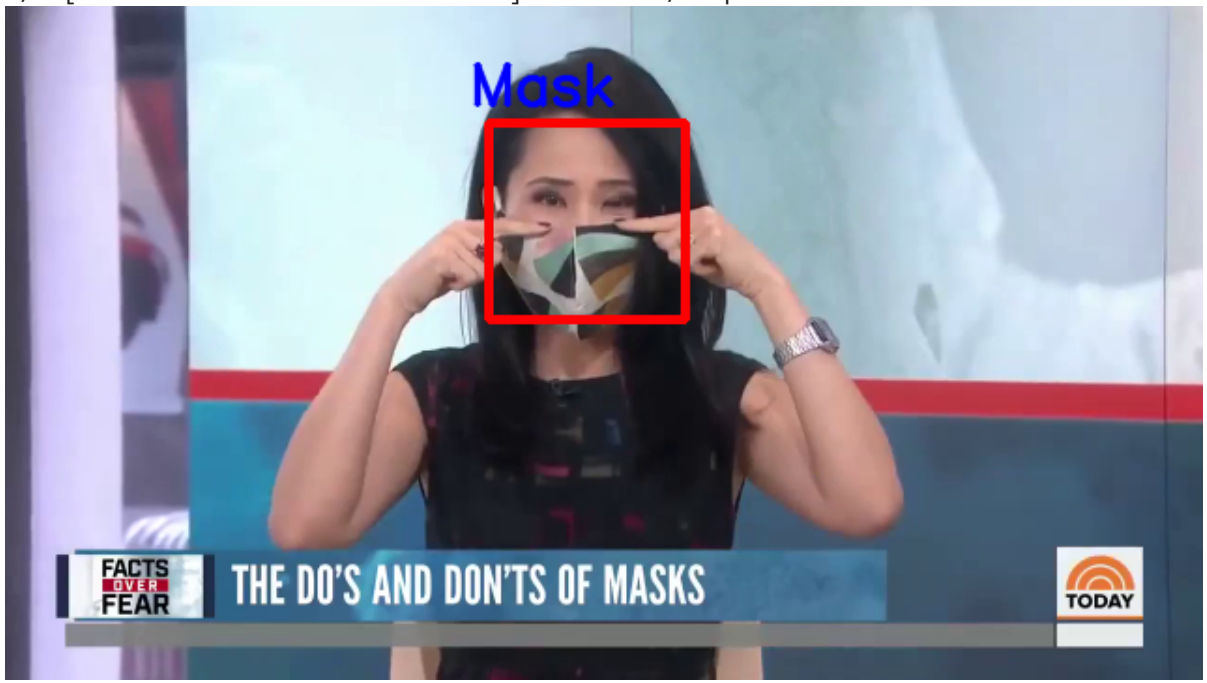

```
1/1 [==============================] - 0s 19ms/step
1/1 [==============================] - 0s 29ms/step
```


```
1/1 [==============================] - 0s 19ms/step
```


```
1/1 [==============================] - 0s 18ms/step
```


```
1/1 [==============================] - 0s 19ms/step
```


```
1/1 [==============================] - 0s 25ms/step
```




```
1/1 [==============================] - 0s 36ms/step
```


```
1/1 [==============================] - 0s 38ms/step
```


```
1/1 [==============================] - 0s 41ms/step
```


```
1/1 [==============================] - 0s 33ms/step
```


```
1/1 [==============================] - 0s 36ms/step
```


```
1/1 [==============================] - 0s 58ms/step
```


```
1/1 [==============================] - 0s 56ms/step
```


```
1/1 [==============================] - 0s 31ms/step
```


```
1/1 [==============================] - 0s 32ms/step
```


```
1/1 [==============================] - 0s 42ms/step
```

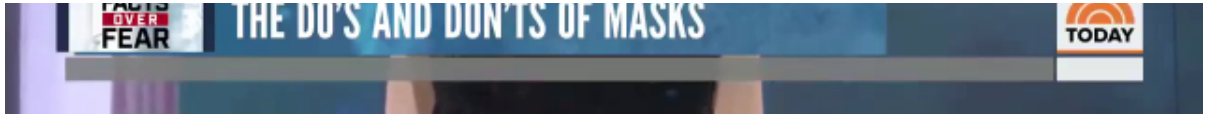

```
1/1 [==============================] - 0s 26ms/step
```


```
1/1 [==============================] - 0s 29ms/step
```


```
1/1 [==============================] - 0s 26ms/step
```

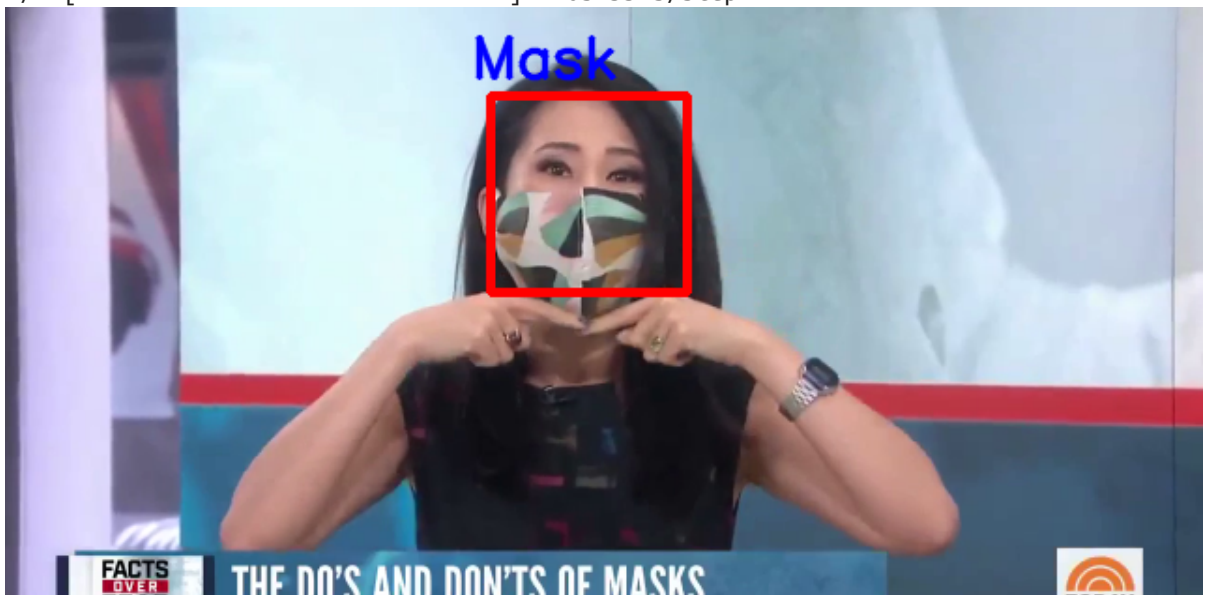

```
1/1 [==============================] - 0s 26ms/step
```

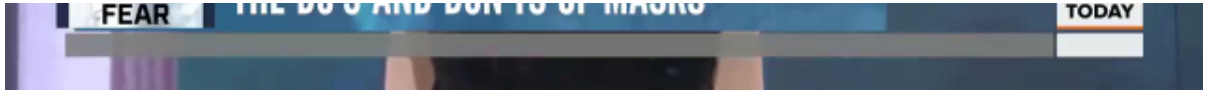

```
1/1 [==============================] - 0s 27ms/step
```




```
1/1 [==============================] - 0s 25ms/step
```

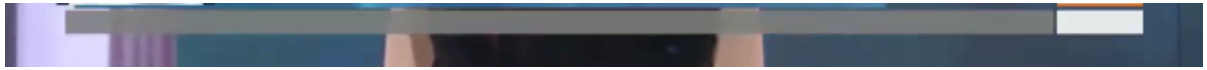

```
1/1 [==============================] - 0s 33ms/step
```






```
1/1 [==============================] - 0s 25ms/step
```


```
1/1 [==============================] - 0s 33ms/step
```



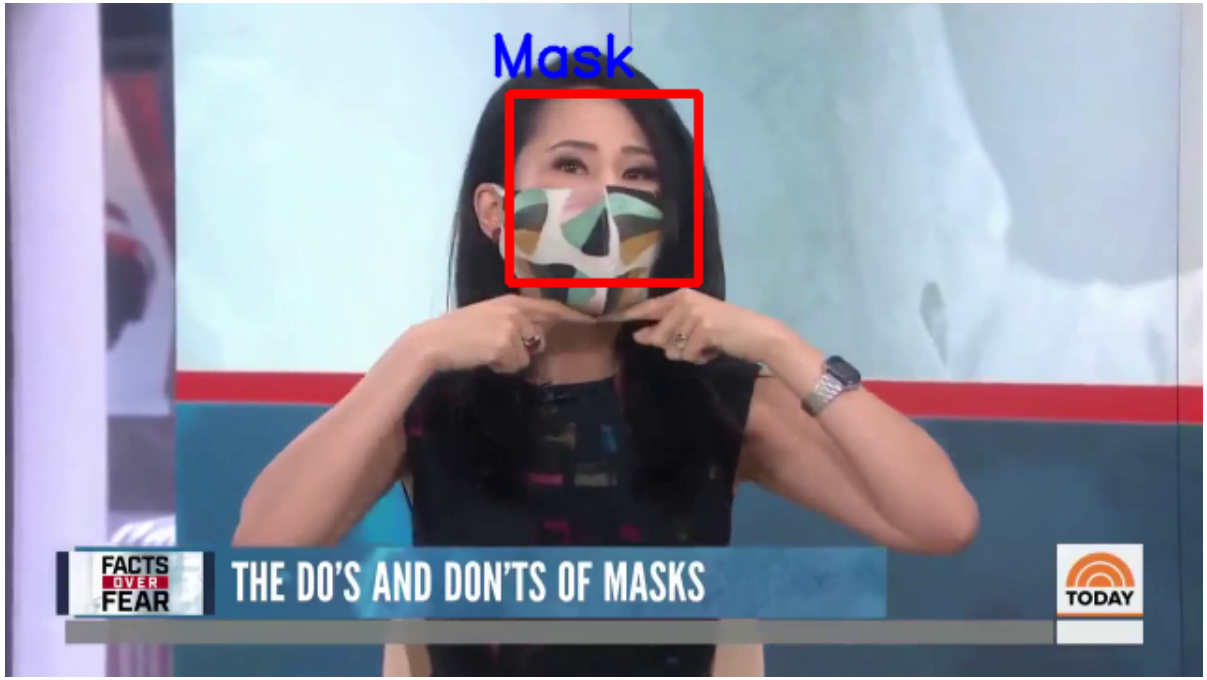
```
1/1 [==============================] - 0s 28ms/step
```


```
1/1 [==============================] - 0s 25ms/step
```

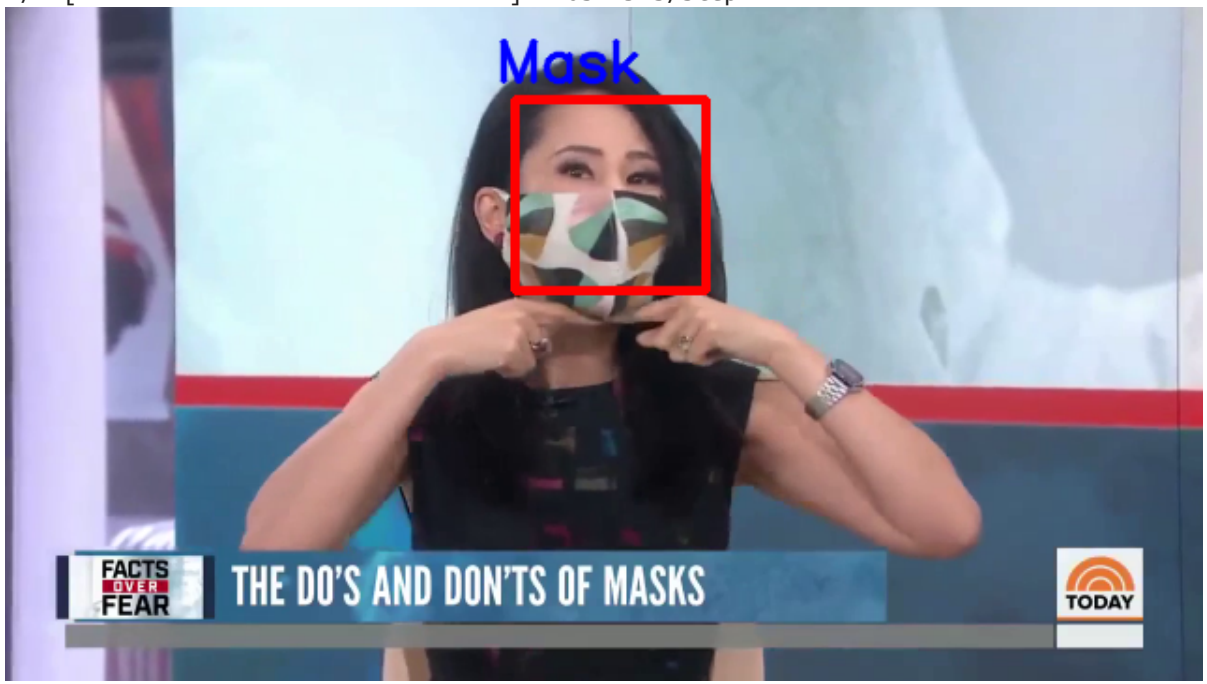
```
1/1 [==============================] - 0s 26ms/step
```


```
1/1 [==============================] - 0s 25ms/step
```

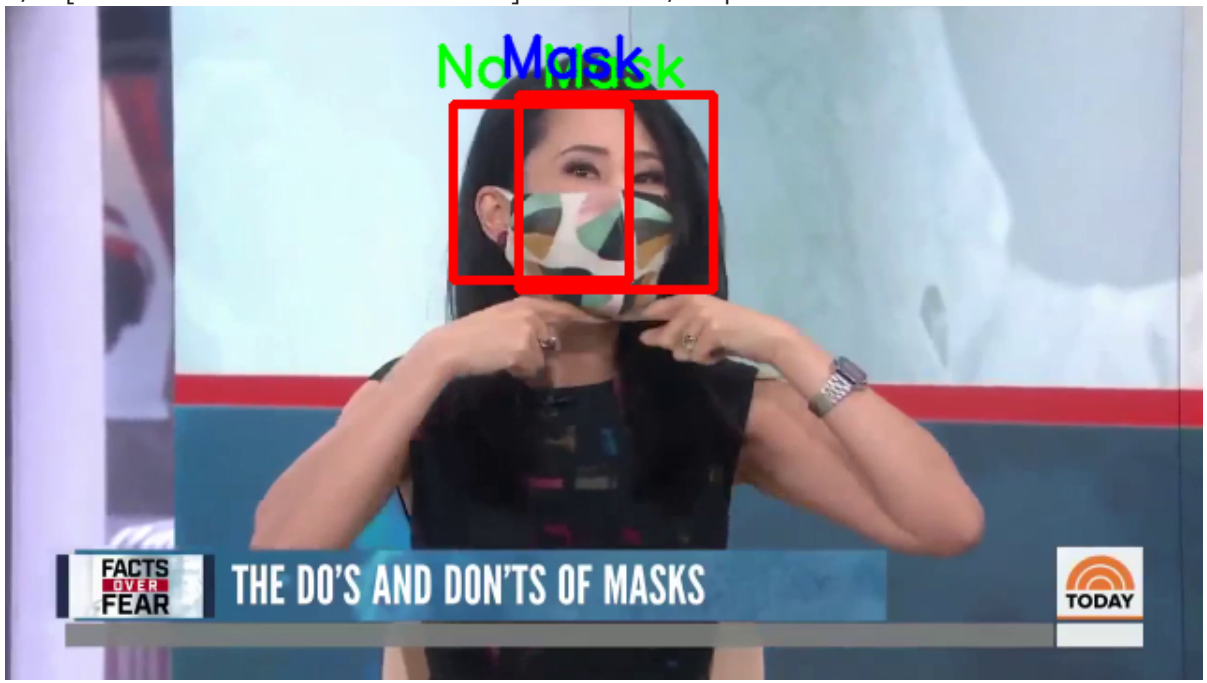
```
1/1 [==============================] - 0s 28ms/step
```




```
1/1 [==============================] - 0s 25ms/step
```



```
1/1 [==============================] - 0s 32ms/step
1/1 [==============================] - 0s 32ms/step
```




```
1/1 [==============================] - 0s 27ms/step
```