Importing Necessary Libraries and Packages

```
import os
import pathlib
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import matplotlib.image as mpimg
import random
import cv2
import tensorflow as tf
from sklearn.model_selection import train_test_split
from tensorflow.keras.models import Sequential
from tensorflow.keras.models import Model
from tensorflow.keras.layers import Dense, Activation, Conv2D, MaxPool2D, Flatten, Dropout, BatchNormalization
from tensorflow.keras.callbacks import EarlyStopping

from google.colab import files
from sklearn.metrics import classification_report,confusion_matrix
```

Uploading files from local device

```
files.upload()
```

```
Choose Files kaggle.json
- kaggle.json(application/json) - 76 bytes, last modified: 9/6/2022 - 100% done
Saving kaggle.json to kaggle.json
{'kaggle.json':
b'{"username":"muralikrishnamaganti","key":"5ee391fe4eb789a61af4587c8f2b25e5"}'}
```

```
os.environ["KAGGLE_CONFIG_DIR"] = "/content"
```

Downloading Dataset from Kaggle

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

```
Warning: Your Kaggle API key is readable by other users on this system! To fix this, you can run 'chmod 600 /content/kaggle.json'
Downloading covid-face-mask-detection-dataset.zip to /content
 93% 193M/207M [00:03<00:00, 92.5MB/s]
100% 207M/207M [00:03<00:00, 59.9MB/s]
```

Unzipping the Dataset

```
!unzip \*.zip
```

```
 inflating: New Masks Dataset/Validation/Non Mask/real_01005.jpg
 inflating: New Masks Dataset/Validation/Non Mask/real_01006.jpg
 inflating: New Masks Dataset/Validation/Non Mask/real_01007.jpg
 inflating: New Masks Dataset/Validation/Non Mask/real_01008.jpg
 inflating: New Masks Dataset/Validation/Non Mask/real_01009.jpg
 inflating: New Masks Dataset/Validation/Non Mask/real_01010.jpg
 inflating: New Masks Dataset/Validation/Non Mask/real_01011.jpg
 inflating: New Masks Dataset/Validation/Non Mask/real_01012.jpg
 inflating: New Masks Dataset/Validation/Non Mask/real_01013.jpg
 inflating: New Masks Dataset/Validation/Non Mask/real_01014.jpg
 inflating: New Masks Dataset/Validation/Non Mask/real_01015.jpg
 inflating: New Masks Dataset/Validation/Non Mask/real_01016.jpg
 inflating: New Masks Dataset/Validation/Non Mask/real_01017.jpg
 inflating: New Masks Dataset/Validation/Non Mask/real_01018.jpg
 inflating: New Masks Dataset/Validation/Non Mask/real_01019.jpg
 inflating: New Masks Dataset/Validation/Non Mask/real_01020.jpg
 inflating: New Masks Dataset/Validation/Non Mask/real_01021.jpg
 inflating: New Masks Dataset/Validation/Non Mask/real_01022.jpg
 inflating: New Masks Dataset/Validation/Non Mask/real_01023.jpg
 inflating: New Masks Dataset/Validation/Non Mask/real_01024.jpg
 inflating: New Masks Dataset/Validation/Non Mask/real_01025.jpg
 inflating: New Masks Dataset/Validation/Non Mask/real_01026.jpg
 inflating: New Masks Dataset/Validation/Non Mask/real_01027.jpg
 inflating: New Masks Dataset/Validation/Non Mask/real_01028.jpg
 inflating: New Masks Dataset/Validation/Non Mask/real_01029.jpg
 inflating: New Masks Dataset/Validation/Non Mask/real_01030.jpg
 inflating: New Masks Dataset/Validation/Non Mask/real_01031.jpg
```

```
for dirpath,dirnames,filenames in os.walk("/content/New Masks Dataset"):
    print(f"there are {len(dirnames)} directories and {len(filenames)} images in '{dirpath}'.")
```

```
 there are 3 directories and 0 images in '/content/New Masks Dataset'.
 there are 2 directories and 0 images in '/content/New Masks Dataset/Validation'.
 there are 0 directories and 153 images in '/content/New Masks Dataset/Validation/Mask'.
 there are 0 directories and 153 images in '/content/New Masks Dataset/Validation/Non Mask'.
 there are 2 directories and 0 images in '/content/New Masks Dataset/Test'.
 there are 0 directories and 50 images in '/content/New Masks Dataset/Test/Mask'.
 there are 0 directories and 50 images in '/content/New Masks Dataset/Test/Non Mask'.
 there are 2 directories and 0 images in '/content/New Masks Dataset/Train'.
 there are 0 directories and 300 images in '/content/New Masks Dataset/Train/Mask'.
 there are 0 directories and 300 images in '/content/New Masks Dataset/Train/Non Mask'.
```

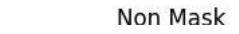
```
def view_image(target_dir, target_class):
    target_folder = target_dir+target_class
    random_image = random.sample(os.listdir(target_folder),1)
    print(random_image)
    img = mpimg.imread(target_folder+"/"+ random_image[0])
    plt.imshow(img)
    plt.title(target_class)
    plt.axis("off")
    print(f"image shape {img.shape}")

    return img
```

Testing the model

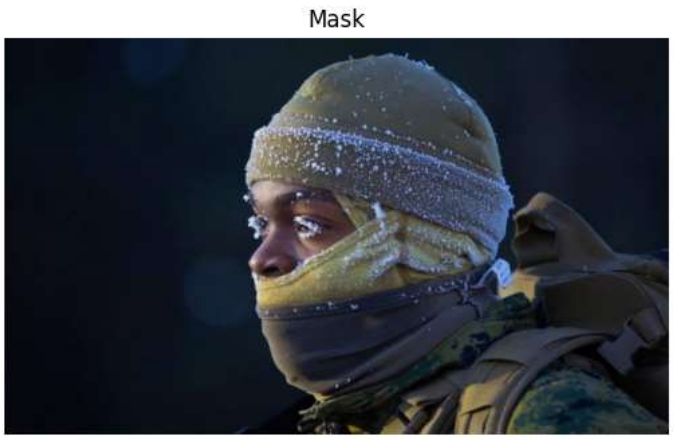
```
img = view_image("/content/New Masks Dataset/Train/","Non Mask")
```

```
['9.jpg']
image shape (275, 183, 3)
```




```
img = view_image("/content/New Masks Dataset/Train/","Mask")
```

```
['1251.jpg']
image shape (428, 720, 3)
```




```
data=[]
labels=[]
no_mask=os.listdir("/content/New Masks Dataset/Train/Non Mask/")
for a in no_mask:

    image = cv2.imread("/content/New Masks Dataset/Train/Non Mask/"+a,)
    image = cv2.resize(image, (224, 224))


    data.append(image)
    labels.append(0)

no_mask=os.listdir("/content/New Masks Dataset/Test/Non Mask/")
for a in no_mask:

    image = cv2.imread("/content/New Masks Dataset/Test/Non Mask/"+a,)
    image = cv2.resize(image, (224, 224))


    data.append(image)
    labels.append(0)


mask=os.listdir("/content/New Masks Dataset/Train/Mask/")
for a in mask:

    image = cv2.imread("/content/New Masks Dataset/Train/Mask/"+a,)
    image = cv2.resize(image, (224, 224))


    data.append(image)
    labels.append(1)


mask=os.listdir("/content/New Masks Dataset/Test/Mask/")
for a in mask:

    image = cv2.imread("/content/New Masks Dataset/Test/Mask/"+a,)
    image = cv2.resize(image, (224, 224))


    data.append(image)
    labels.append(1)


data = np.array(data) / 255.0
labels = np.array(labels)



data.shape
```

```
(700, 224, 224, 3)
```

```
X_train, X_test, y_train, y_test = train_test_split(data, labels, test_size=0.1, random_state=42,shuffle=True,
                                                    stratify = labels)



base_model = tf.keras.applications.MobileNet(input_shape=[224,224,3], weights = "imagenet", include_top=False)

base_model.trainable = False

# for layer in base_model.layers[30:]:
#   layer.trainable = False


model = Flatten()(base_model.output)
model = Dense(units=256, activation="relu")(model)
model = Dense(units=64, activation="relu")(model)
prediction_layer = Dense(units=1, activation="sigmoid")(model)

model = Model(inputs = base_model.input, outputs = prediction_layer)
model.compile(optimizer='SGD',loss='binary_crossentropy',metrics=['accuracy'])
```

```
Downloading data from https://storage.googleapis.com/tensorflow/keras-applications/mobilenet/mobilenet_1_0_224_tf_no_top.h5
17225924/17225924 [==============================] - 0s 0us/step
```

Measuring Model Performance

```
model.fit(X_train, y_train, epochs=15,validation_split= 0.1, batch_size=32)
```

```
Epoch 1/15
18/18 [==============================] - 33s 2s/step - loss: 0.9121 - accuracy: 0.8430 - val_loss: 0.0059 - val_accuracy: 1.0000
Epoch 2/15
18/18 [==============================] - 27s 1s/step - loss: 0.0109 - accuracy: 0.9965 - val_loss: 0.0036 - val_accuracy: 1.0000
Epoch 3/15
18/18 [==============================] - 30s 2s/step - loss: 0.0039 - accuracy: 1.0000 - val_loss: 0.0027 - val_accuracy: 1.0000
Epoch 4/15
18/18 [==============================] - 27s 2s/step - loss: 0.0027 - accuracy: 1.0000 - val_loss: 0.0020 - val_accuracy: 1.0000
Epoch 5/15
18/18 [==============================] - 27s 1s/step - loss: 0.0021 - accuracy: 1.0000 - val_loss: 0.0017 - val_accuracy: 1.0000
Epoch 6/15
18/18 [==============================] - 29s 2s/step - loss: 0.0016 - accuracy: 1.0000 - val_loss: 0.0015 - val_accuracy: 1.0000
Epoch 7/15
18/18 [==============================] - 36s 2s/step - loss: 0.0014 - accuracy: 1.0000 - val_loss: 0.0013 - val_accuracy: 1.0000
Epoch 8/15
18/18 [==============================] - 29s 2s/step - loss: 0.0012 - accuracy: 1.0000 - val_loss: 0.0012 - val_accuracy: 1.0000
Epoch 9/15
18/18 [==============================] - 30s 2s/step - loss: 0.0010 - accuracy: 1.0000 - val_loss: 0.0011 - val_accuracy: 1.0000
Epoch 10/15
18/18 [==============================] - 27s 2s/step - loss: 8.7856e-04 - accuracy: 1.0000 - val_loss: 9.9145e-04 - val_accuracy: 1
Epoch 11/15
18/18 [==============================] - 28s 2s/step - loss: 7.8521e-04 - accuracy: 1.0000 - val_loss: 9.1027e-04 - val_accuracy: 1
Epoch 12/15
18/18 [==============================] - 27s 2s/step - loss: 7.0491e-04 - accuracy: 1.0000 - val_loss: 8.5597e-04 - val_accuracy: 1
Epoch 13/15
18/18 [==============================] - 30s 2s/step - loss: 6.4034e-04 - accuracy: 1.0000 - val_loss: 7.9494e-04 - val_accuracy: 1
Epoch 14/15
18/18 [==============================] - 27s 2s/step - loss: 5.8750e-04 - accuracy: 1.0000 - val_loss: 7.5129e-04 - val_accuracy: 1
Epoch 15/15
18/18 [==============================] - 28s 2s/step - loss: 5.4186e-04 - accuracy: 1.0000 - val_loss: 7.1055e-04 - val_accuracy: 1
<keras.callbacks.History at 0x7f26d23cbbb0>
```

```
predictions = model.predict(X_test)

predict=[]

for i in range(len(predictions)):
    if predictions[i][0]>0.5:
        predict.append(1)
    else:
        predict.append(0)
```

```
3/3 [==============================] - 4s 684ms/step
```

```
pd.DataFrame(confusion_matrix(y_test, predict), columns= ["No Mask", "Mask"], index = ["No Mask", "Mask"])
```

	No Mask	Mask
No Mask	33	2
Mask	0	35

Saving the model

```
model_name = "/content/mask_detection_best.h5"
tf.keras.models.save_model(model, model_name)
```