```
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 matplotlib.pyplot as plt
```

```
import os
os.environ['KAGGLE_USERNAME']='salandriniru28'
os.environ['KAGGLE_KEY']='e41a03d6483801cd3a17b9fa3c194210'
```

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

```
Downloading covid-face-mask-detection-dataset.zip to /content
 95% 197M/207M [00:00<00:00, 214MB/s]
100% 207M/207M [00:01<00:00, 214MB/s]
```

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

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

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

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

```
['1513.jpg', '0715.jpg', '0488.jpg', '1555.jpg', '0972.jpg', '0746.jpg', '0003.jpg',
```

```
train_nonmask_names = os.listdir(train_nonmask_dir)
print(train_nonmask_names[:10])
```

```
['259.jpg', '308.jpg', '311.jpg', '164.jpg', '231.jpg', '136.jpg', '351.jpg', '108.jp
```

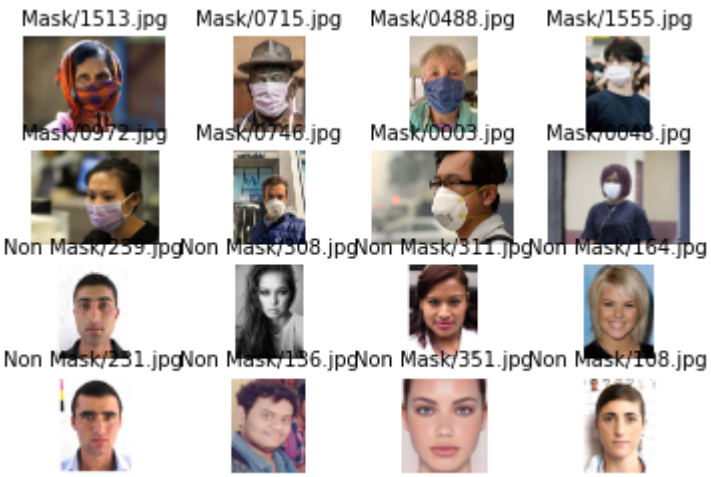
```
#Image Visualization
```

```
import matplotlib.image as mpimg #for loading,rescaling and displaying the image
nrows=4
ncols=4
plt.figure(figsize=(12,12))
```

```
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_nonmask_names[0:8]:
    nomask_pic.append(os.path.join(train_nonmask_dir,i))

print(mask_pic)
print(nomask_pic)
```

```
['/content/New Masks Dataset/Train/Mask/1513.jpg', '/content/New Masks Dataset/Train/
['/content/New Masks Dataset/Train/Non Mask/259.jpg', '/content/New Masks Dataset/Tra
<Figure size 864x864 with 0 Axes>
```

```
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=10)
  plt.imshow(image,cmap='gray')

plt.show()
```




```
#Data Augmentation


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

```
train_genertor = train_datagen.flow_from_directory(train_dir,
                                                   target_size=(150,150),
                                                   batch_size=32,
                                                   class_mode='binary')
test_genertor = test_datagen.flow_from_directory(test_dir,
                                                   target_size=(150,150),
                                                   batch_size=32,
                                                   class_mode='binary')
validation_genertor = validation_datagen.flow_from_directory(valid_dir,
                                                   target_size=(150,150),
                                                   batch_size=32,
                                                   class_mode='binary')
```

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

```
train_genertor.class_indices
```

```
{'Mask': 0, 'Non Mask': 1}
```

```
train_genertor.image_shape
```

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

```
# Building the CNN

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(0.50))
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 (MaxPooling2 (None, 37, 37, 64)        0
_________________________________________________________________
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
_________________________________________________________________
```

```
#Train & Evaluation Performace of Model


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

history = model.fit(train_genertor,
                    epochs=30,
                    validation_data=validation_genertor)
```

```
Epoch 1/30
19/19 [==============================] - 11s 584ms/step - loss: 5.2082 - accuracy:
Epoch 2/30
19/19 [==============================] - 11s 586ms/step - loss: 0.6765 - accuracy:
Epoch 3/30
19/19 [==============================] - 11s 584ms/step - loss: 0.6467 - accuracy:
Epoch 4/30
19/19 [==============================] - 11s 577ms/step - loss: 0.4861 - accuracy:
Epoch 5/30
19/19 [==============================] - 11s 579ms/step - loss: 0.3923 - accuracy:
Epoch 6/30
19/19 [==============================] - 11s 578ms/step - loss: 0.3448 - accuracy:
Epoch 7/30
19/19 [==============================] - 11s 592ms/step - loss: 0.3124 - accuracy:
Epoch 8/30
19/19 [==============================] - 11s 588ms/step - loss: 0.2754 - accuracy:
Epoch 9/30
19/19 [==============================] - 11s 579ms/step - loss: 0.2739 - accuracy:
Epoch 10/30
19/19 [==============================] - 11s 579ms/step - loss: 0.2372 - accuracy:
Epoch 11/30
19/19 [==============================] - 11s 580ms/step - loss: 0.2430 - accuracy:
Epoch 12/30
```

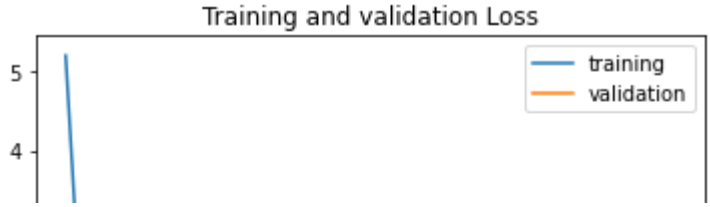
```
19/19 [==============================] - 11s 596ms/step - loss: 0.2625 - accuracy:
Epoch 13/30
19/19 [==============================] - 11s 589ms/step - loss: 0.2468 - accuracy:
Epoch 14/30
19/19 [==============================] - 11s 590ms/step - loss: 0.2200 - accuracy:
Epoch 15/30
19/19 [==============================] - 11s 580ms/step - loss: 0.2232 - accuracy:
Epoch 16/30
19/19 [==============================] - 11s 574ms/step - loss: 0.1928 - accuracy:
Epoch 17/30
19/19 [==============================] - 11s 581ms/step - loss: 0.1756 - accuracy:
Epoch 18/30
19/19 [==============================] - 11s 585ms/step - loss: 0.2111 - accuracy:
Epoch 19/30
19/19 [==============================] - 11s 581ms/step - loss: 0.1976 - accuracy:
Epoch 20/30
19/19 [==============================] - 11s 579ms/step - loss: 0.1832 - accuracy:
Epoch 21/30
19/19 [==============================] - 11s 584ms/step - loss: 0.1917 - accuracy:
Epoch 22/30
19/19 [==============================] - 11s 575ms/step - loss: 0.1960 - accuracy:
Epoch 23/30
19/19 [==============================] - 11s 579ms/step - loss: 0.1620 - accuracy:
Epoch 24/30
19/19 [==============================] - 11s 575ms/step - loss: 0.2011 - accuracy:
Epoch 25/30
19/19 [==============================] - 11s 575ms/step - loss: 0.2073 - accuracy:
Epoch 26/30
19/19 [==============================] - 11s 578ms/step - loss: 0.1943 - accuracy:
Epoch 27/30
19/19 [==============================] - 11s 578ms/step - loss: 0.1894 - accuracy:
Epoch 28/30
19/19 [==============================] - 11s 581ms/step - loss: 0.1938 - accuracy:
Epoch 29/30
19/19 [==============================] - 11s 579ms/step - loss: 0.1795 - accuracy:
```

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

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

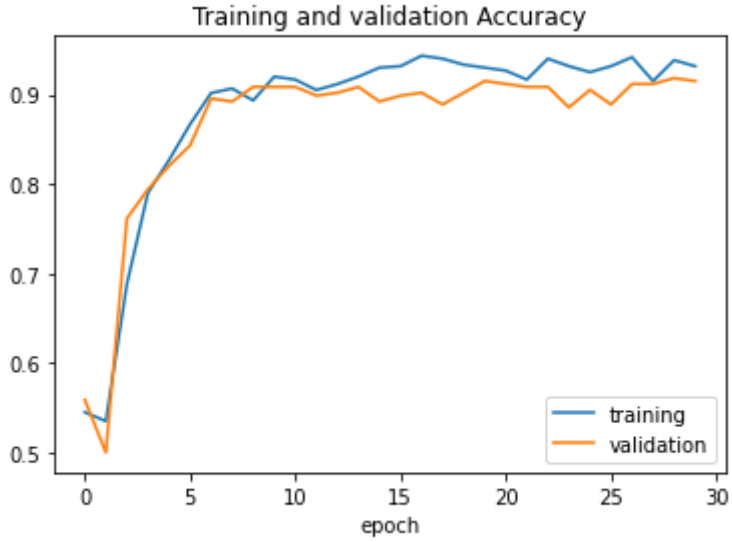
```
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.legend(['training','validation'])
plt.title('Training and validation Loss')
plt.xlabel('epoch')
```

```
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')
```




```
test_loss , test_acc=model.evaluate(test_genertor)
print('test loss:{} test acc:{}'.format(test_loss,test_acc))
```

```
4/4 [==============================] - 1s 206ms/step - loss: 0.1817 - accuracy: 0.946
test loss:0.18174588680267334 test acc:0.9399999976158142
```

```
#use the train model to detect the face mask on the static image


from google.colab import files
from keras_preprocessing import image
uploaded = files.upload()
for fname in uploaded.keys():
  img_path='/content/'+fname
  img=image.load_img(img_path,target_size=(150,150))
  images=image.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 PassPort.jpeg to PassPort (1).jpeg
PassPort.jpeg
NO Mask
```

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