In [ ]:
```
```

In [1]:
```
!unzip '/content/drive/My Drive/Covid19Pred/Dataset_kaggle.zip'
```

```
Archive:  /content/drive/My Drive/Covid19Pred/Dataset_kaggle.zip
replace Dataset_kaggle/COVID/Covid (1).png? [y]es, [n]o, [A]ll, [N]one, [r]enam
e: N
```

In [2]:
```
import os
import cv2
import numpy as np
from PIL import Image
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
import tensorflow as tf
from tensorflow import keras
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D,MaxPooling2D,Dense,Flatten,Dropout
from tensorflow.keras.layers import BatchNormalization
from tensorflow.keras.applications.mobilenet import MobileNet
from tensorflow.keras.applications.resnet50 import ResNet50
from tensorflow.keras import optimizers
```

In [3]:
```
yes=os.listdir('/content/Dataset_kaggle/COVID')
no=os.listdir('/content/Dataset_kaggle/non-COVID')
```

In [4]:
```
data=np.concatenate([yes,no])
len(data)==len(yes)+len(no)
```

Out[4]:
```
True
```

In [5]:
```
target_x=np.full(len(yes),1)
target_y=np.full(len(no),0)
data_target=np.concatenate([target_x,target_y])
```

In [6]:
```
yes_values=os.listdir('/content/Dataset_kaggle/COVID')
no_values=os.listdir('/content/Dataset_kaggle/non-COVID')
```

In [7]:
```
X_data =[]
for file in yes_values:
    img = cv2.imread('/content/Dataset_kaggle/COVID/'+file)
    face = cv2.resize(img, (224, 224) )
    (b, g, r)=cv2.split(face)
    img=cv2.merge([r,g,b])
    X_data.append(img)
```

In [8]:
```
for file in no_values:
    img = cv2.imread('/content/Dataset_kaggle/non-COVID/'+file)
    face = cv2.resize(img, (224, 224) )
    (b, g, r)=cv2.split(face)
    img=cv2.merge([r,g,b])
    X_data.append(img)
```

In [9]:
```
X = np.squeeze(X_data)
```

In [10]:
```
X = X.astype('float32')
X /= 255
```

In [11]:
```
train_size = None # number of samples for training
test_size = None # number of samples for testing
```

In [12]:
```
x_train,x_test,y_train,y_test=train_test_split(X, data_target, test_size=0.2, ra
if train_size:
    x_train = x_train[:train_size]
    y_train = y_train[:train_size]
if test_size:
    x_test = x_test[:test_size]
    y_test = y_test[:test_size]

print(x_train.shape, x_test.shape)
```

```
(1984, 224, 224, 3) (497, 224, 224, 3)
```

In [13]:
```
mobile_net_model= MobileNet(weights='imagenet', include_top=False, input_shape=(
mobile_net_model.trainable = False

model =tf.keras.Sequential()
model.add(mobile_net_model)
model.add(Flatten())
model.add(Dense(128, activation='relu'))
model.add(Dropout(0.2))
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.1))
model.add(Dense(2, activation='softmax'))
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
mobilenet_1.00_224 (Model)   (None, 7, 7, 1024)        3228864
_________________________________________________________________
flatten (Flatten)            (None, 50176)             0
_________________________________________________________________
dense (Dense)                (None, 128)               6422656
_________________________________________________________________
dropout (Dropout)            (None, 128)               0
_________________________________________________________________
dense_1 (Dense)              (None, 64)                8256
_________________________________________________________________
dropout_1 (Dropout)          (None, 64)                0
_________________________________________________________________
dense_2 (Dense)              (None, 2)                 130
=================================================================
Total params: 9,659,906
Trainable params: 6,431,042
Non-trainable params: 3,228,864
_________________________________________________________________
```

In [19]:
```
model.compile(loss='sparse_categorical_crossentropy',
              optimizer=optimizers.Adam(),
              metrics=['acc'])
```

In [21]:
```
history = model.fit(
    x_train,
    y_train,
    epochs=20,
    batch_size=100,
    validation_data=(x_test, y_test),
    verbose=1)
```

```
Epoch 1/20
20/20 [==============================] - 5s 256ms/step - loss: 0.7145 - acc: 0.
8725 - val_loss: 0.3426 - val_acc: 0.9296
Epoch 2/20
20/20 [==============================] - 5s 257ms/step - loss: 0.3725 - acc: 0.
9254 - val_loss: 0.2068 - val_acc: 0.9396
Epoch 3/20
20/20 [==============================] - 5s 257ms/step - loss: 0.1416 - acc: 0.
9556 - val_loss: 0.1630 - val_acc: 0.9477
Epoch 4/20
20/20 [==============================] - 5s 256ms/step - loss: 0.0691 - acc: 0.
9723 - val_loss: 0.1241 - val_acc: 0.9557
Epoch 5/20
20/20 [==============================] - 5s 257ms/step - loss: 0.0516 - acc: 0.
9803 - val_loss: 0.1119 - val_acc: 0.9557
Epoch 6/20
20/20 [==============================] - 5s 256ms/step - loss: 0.0351 - acc: 0.
9854 - val_loss: 0.1038 - val_acc: 0.9537
Epoch 7/20
20/20 [==============================] - 5s 257ms/step - loss: 0.0286 - acc: 0.
9899 - val_loss: 0.1061 - val_acc: 0.9557
Epoch 8/20
20/20 [==============================] - 5s 258ms/step - loss: 0.0260 - acc: 0.
9894 - val_loss: 0.1072 - val_acc: 0.9577
Epoch 9/20
20/20 [==============================] - 5s 257ms/step - loss: 0.0221 - acc: 0.
9924 - val_loss: 0.1088 - val_acc: 0.9678
Epoch 10/20
20/20 [==============================] - 5s 258ms/step - loss: 0.0185 - acc: 0.
9945 - val_loss: 0.1135 - val_acc: 0.9638
Epoch 11/20
20/20 [==============================] - 5s 257ms/step - loss: 0.0188 - acc: 0.
9904 - val_loss: 0.1354 - val_acc: 0.9638
Epoch 12/20
20/20 [==============================] - 5s 256ms/step - loss: 0.0176 - acc: 0.
9940 - val_loss: 0.1084 - val_acc: 0.9658
Epoch 13/20
20/20 [==============================] - 5s 257ms/step - loss: 0.0121 - acc: 0.
9960 - val_loss: 0.1291 - val_acc: 0.9638
Epoch 14/20
20/20 [==============================] - 5s 259ms/step - loss: 0.0189 - acc: 0.
9940 - val_loss: 0.1583 - val_acc: 0.9618
Epoch 15/20
20/20 [==============================] - 5s 255ms/step - loss: 0.0134 - acc: 0.
9950 - val_loss: 0.1118 - val_acc: 0.9658
Epoch 16/20
20/20 [==============================] - 5s 258ms/step - loss: 0.0077 - acc: 0.
9975 - val_loss: 0.1125 - val_acc: 0.9678
```

```
Epoch 17/20
20/20 [==============================] - 5s 257ms/step - loss: 0.0090 - acc: 0.
9980 - val_loss: 0.1154 - val_acc: 0.9658
Epoch 18/20
20/20 [==============================] - 5s 258ms/step - loss: 0.0050 - acc: 0.
9995 - val_loss: 0.1394 - val_acc: 0.9658
Epoch 19/20
20/20 [==============================] - 5s 255ms/step - loss: 0.0090 - acc: 0.
9980 - val_loss: 0.1287 - val_acc: 0.9678
Epoch 20/20
20/20 [==============================] - 5s 255ms/step - loss: 0.0076 - acc: 0.
9980 - val_loss: 0.1655 - val_acc: 0.9638
```

In [22]:
```
final_loss, final_acc = model.evaluate(x_test, y_test, verbose=0)
print('The final accuracy is ',final_acc)
```

```
The final accuracy is  0.9637826681137085
```

In [ ]:
```

```