```
from keras.preprocessing.image import ImageDataGenerator
```

```
train_data_dir='/content/drive/MyDrive/pnemonia/train'
valid_data_dir='/content/drive/MyDrive/pnemonia/test'

train_datagen=ImageDataGenerator(rescale=1.0/255,
                                 rotation_range=45,
                                 shear_range=0.2,
                                 zoom_range=0.2,
                                 height_shift_range=0.2,
                                 width_shift_range=0.2,
                                 horizontal_flip=True,
                                 fill_mode='nearest')
test_datagen=ImageDataGenerator(rescale=1.0/255)

batch_size=32
train_generator=train_datagen.flow_from_directory(
    train_data_dir,
    target_size=(150,150),
    color_mode='rgb',
    class_mode='binary',
    shuffle=True,
    batch_size=batch_size
)
valid_generator=test_datagen.flow_from_directory(
    valid_data_dir,
    target_size=(150,150),
    color_mode='rgb',
    shuffle=False,
    class_mode='binary',
    batch_size=batch_size)
```

```
Found 5211 images belonging to 2 classes.
Found 624 images belonging to 2 classes.
```

```
from keras.models import Sequential
from keras.layers import Dense,Conv2D,MaxPooling2D,LeakyReLU,Flatten,Dropout
```

```
model=Sequential()

#Building a CNN Model
#A CNN model has three layers which includes Convolutional layer,pooling layer and a fully connected layer

#convolutional layer

model.add(Conv2D(32,(3,3),strides=(1,1),padding='same',input_shape=(150,150,3)))
model.add(LeakyReLU(0.1))
#pooling layer
model.add(MaxPooling2D(2,2))

#convolutional layer
model.add(Conv2D(64,(3,3),strides=(1,1),padding='same'))
model.add(LeakyReLU(0.1))
#pooling layer
```

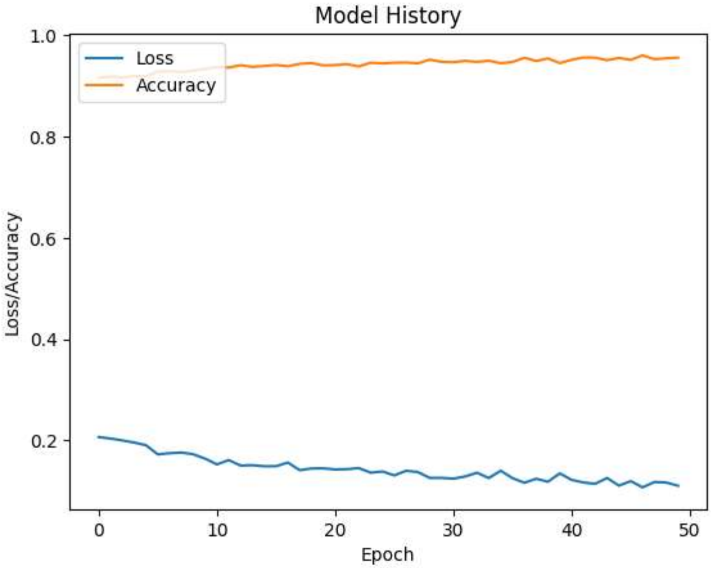
```
model.add(MaxPooling2D(2,2))

#convolutional layer
model.add(Conv2D(128,(3,3),strides=(1,1),padding='same'))
model.add(LeakyReLU(0.1))
#pooling layer
model.add(MaxPooling2D(2,2))

model.add(Flatten())
#fully connected layer
model.add(Dense(128,activation='relu'))
model.add(Dense(1,activation='sigmoid'))


#compiling a model
model.compile(optimizer='adam',loss='binary_crossentropy',metrics=['accuracy'])

#model summary
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_3 (Conv2D)           (None, 150, 150, 32)      896

 leaky_re_lu_3 (LeakyReLU)   (None, 150, 150, 32)      0

 max_pooling2d_3 (MaxPooling  (None, 75, 75, 32)       0
 2D)

 conv2d_4 (Conv2D)           (None, 75, 75, 64)        18496

 leaky_re_lu_4 (LeakyReLU)   (None, 75, 75, 64)        0

 max_pooling2d_4 (MaxPooling  (None, 37, 37, 64)       0
 2D)

 conv2d_5 (Conv2D)           (None, 37, 37, 128)       73856

 leaky_re_lu_5 (LeakyReLU)   (None, 37, 37, 128)       0

 max_pooling2d_5 (MaxPooling  (None, 18, 18, 128)      0
 2D)

 flatten_1 (Flatten)         (None, 41472)             0

 dense_2 (Dense)             (None, 128)               5308544

 dense_3 (Dense)             (None, 1)                 129

=================================================================
Total params: 5,401,921
Trainable params: 5,401,921
Non-trainable params: 0
_________________________________________________________________
```

```
history=model.fit(train_generator,epochs=50,steps_per_epoch=5211/batch_size,validation_data=valid_generator,validation_steps=624/batch_size)
```

```
Epoch 23/50
162/162 [==============================] - 99s 610ms/step - loss: 0.1454 - accuracy: 0.9384 - val_loss: 0.2959 - val_accuracy: 0.8926
Epoch 24/50
162/162 [==============================] - 105s 648ms/step - loss: 0.1364 - accuracy: 0.9459 - val_loss: 0.2631 - val_accuracy: 0.8830
Epoch 25/50
162/162 [==============================] - 102s 629ms/step - loss: 0.1387 - accuracy: 0.9443 - val_loss: 0.2694 - val_accuracy: 0.8830
Epoch 26/50
162/162 [==============================] - 107s 657ms/step - loss: 0.1311 - accuracy: 0.9459 - val_loss: 0.2968 - val_accuracy: 0.8862
Epoch 27/50
162/162 [==============================] - 98s 602ms/step - loss: 0.1401 - accuracy: 0.9463 - val_loss: 0.5629 - val_accuracy: 0.7756
Epoch 28/50
162/162 [==============================] - 103s 634ms/step - loss: 0.1375 - accuracy: 0.9445 - val_loss: 0.2810 - val_accuracy: 0.9006
Epoch 29/50
162/162 [==============================] - 99s 606ms/step - loss: 0.1260 - accuracy: 0.9520 - val_loss: 0.2638 - val_accuracy: 0.8990
Epoch 30/50
162/162 [==============================] - 99s 607ms/step - loss: 0.1261 - accuracy: 0.9474 - val_loss: 0.3012 - val_accuracy: 0.8958
Epoch 31/50
162/162 [==============================] - 97s 594ms/step - loss: 0.1246 - accuracy: 0.9468 - val_loss: 0.3888 - val_accuracy: 0.8718
Epoch 32/50
162/162 [==============================] - 96s 593ms/step - loss: 0.1289 - accuracy: 0.9495 - val_loss: 0.2647 - val_accuracy: 0.8766
Epoch 33/50
162/162 [==============================] - 95s 587ms/step - loss: 0.1363 - accuracy: 0.9474 - val_loss: 0.2729 - val_accuracy: 0.8926
Epoch 34/50
162/162 [==============================] - 96s 590ms/step - loss: 0.1258 - accuracy: 0.9497 - val_loss: 0.2887 - val_accuracy: 0.8942
Epoch 35/50
162/162 [==============================] - 99s 607ms/step - loss: 0.1403 - accuracy: 0.9449 - val_loss: 0.3193 - val_accuracy: 0.8782
Epoch 36/50
162/162 [==============================] - 99s 606ms/step - loss: 0.1255 - accuracy: 0.9472 - val_loss: 0.3093 - val_accuracy: 0.8926
Epoch 37/50
162/162 [==============================] - 96s 588ms/step - loss: 0.1165 - accuracy: 0.9557 - val_loss: 0.2840 - val_accuracy: 0.8830
Epoch 38/50
162/162 [==============================] - 99s 606ms/step - loss: 0.1244 - accuracy: 0.9491 - val_loss: 0.3199 - val_accuracy: 0.8990
Epoch 39/50
162/162 [==============================] - 98s 605ms/step - loss: 0.1185 - accuracy: 0.9541 - val_loss: 0.2843 - val_accuracy: 0.8750
Epoch 40/50
162/162 [==============================] - 99s 605ms/step - loss: 0.1349 - accuracy: 0.9451 - val_loss: 0.3077 - val_accuracy: 0.8990
Epoch 41/50
162/162 [==============================] - 97s 592ms/step - loss: 0.1221 - accuracy: 0.9516 - val_loss: 0.3211 - val_accuracy: 0.9006
Epoch 42/50
162/162 [==============================] - 93s 572ms/step - loss: 0.1171 - accuracy: 0.9561 - val_loss: 0.2892 - val_accuracy: 0.9054
Epoch 43/50
162/162 [==============================] - 90s 550ms/step - loss: 0.1145 - accuracy: 0.9559 - val_loss: 0.2899 - val_accuracy: 0.9054
Epoch 44/50
162/162 [==============================] - 98s 599ms/step - loss: 0.1259 - accuracy: 0.9507 - val_loss: 0.3398 - val_accuracy: 0.8862
Epoch 45/50
162/162 [==============================] - 103s 633ms/step - loss: 0.1108 - accuracy: 0.9553 - val_loss: 0.3135 - val_accuracy: 0.8782
Epoch 46/50
162/162 [==============================] - 100s 616ms/step - loss: 0.1197 - accuracy: 0.9514 - val_loss: 0.3105 - val_accuracy: 0.8574
Epoch 47/50
162/162 [==============================] - 105s 646ms/step - loss: 0.1073 - accuracy: 0.9603 - val_loss: 0.3280 - val_accuracy: 0.8910
Epoch 48/50
162/162 [==============================] - 103s 634ms/step - loss: 0.1179 - accuracy: 0.9528 - val_loss: 0.3048 - val_accuracy: 0.8878
Epoch 49/50
162/162 [==============================] - 105s 645ms/step - loss: 0.1169 - accuracy: 0.9545 - val_loss: 0.3270 - val_accuracy: 0.8429
Epoch 50/50
162/162 [==============================] - 103s 635ms/step - loss: 0.1106 - accuracy: 0.9557 - val_loss: 0.2598 - val_accuracy: 0.9006
```

```
plt.plot(history.history['loss'])
plt.plot(history.history['accuracy'])
plt.title('Model History')
plt.ylabel('Loss/Accuracy')
plt.xlabel('Epoch')
plt.legend(['Loss', 'Accuracy'], loc='upper left')
plt.show()
```



```
loss,accuracy=model.evaluate(valid_generator)
print('loss:',loss)
print('accuracy:',accuracy*100)
```

```
20/20 [==============================] - 8s 394ms/step - loss: 0.2598 - accuracy: 0.9006
loss: 0.2598098814487457
accuracy: 90.06410241127014
```

```
true_labels=valid_generator.classes
```

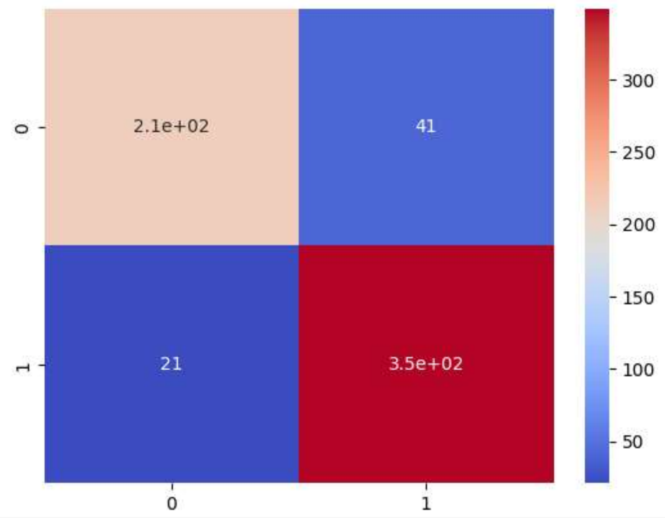
```
import numpy as np
y_probs=model.predict(valid_generator)
y_predict=np.round(y_probs)
```

```
20/20 [==============================] - 6s 287ms/step
```

```
from sklearn.metrics import classification_report , confusion_matrix
import seaborn as sns
```

```
cm=confusion_matrix(y_predict,true_labels)
sns.heatmap(cm,annot=True,cmap='coolwarm')
```

```
<Axes: >
```



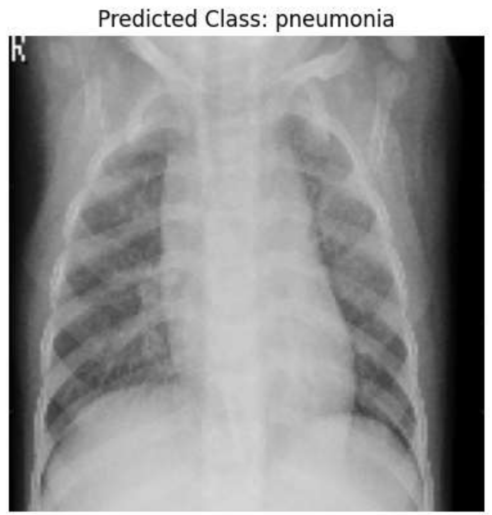
```
cr=classification_report(y_predict,true_labels)
print('Classification Report')
print(cr)
```

```
Classification Report
              precision    recall  f1-score   support

         0.0       0.91      0.84      0.87       254
         1.0       0.89      0.94      0.92       370

    accuracy                           0.90       624
   macro avg       0.90      0.89      0.90       624
weighted avg       0.90      0.90      0.90       624
```

Testing model with unseen pics

```
import cv2 as cv
import matplotlib.pyplot as plt

path = '/content/drive/MyDrive/pnemonia/val/pneumonia/person1947_bacteria_4876.jpeg' #pneumonia x-ray image
input_pic = cv.imread(path)
resized_pic = cv.resize(input_pic, (150, 150))
normalized_pic = resized_pic / 255.0
reshaped_pic = normalized_pic.reshape(1, 150, 150, 3)
prediction = model.predict(reshaped_pic)
predicted_class = "pneumonia" if prediction[0][0]> 0.5 else "normal"

plt.imshow(normalized_pic)
plt.title(f"Predicted Class: {predicted_class}")
plt.axis('off')
plt.show()
```

```
1/1 [==============================] - 0s 80ms/step
```



Predicted Class: pneumonia

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

✓ 1s completed at 5:28 AM

Could not connect to the reCAPTCHA service. Please check your internet connection and reload to get a reCAPTCHA challenge.