**WORD DOCUMENT FOR THE SAMPLE CODE**

**import** os

**import** cv2

**from** PIL **import** Image

**import** numpy **as** np

**import** tensorflow **as** tf

**from** tensorflow.keras.models **import** Sequential

**from** keras.layers **import** Dense, Dropout, Activation, Flatten

**from** keras.layers **import** Conv2D, MaxPooling2D

gpus **=** tf**.**config**.**experimental**.**list\_physical\_devices('GPU')

**for** gpu **in** gpus:

tf**.**config**.**experimental**.**set\_memory\_growth(gpu, **True**)

In [11]:

image\_directory **=** 'chest/train/'

dataset **=** []

labels **=** []

In [12]:

normal\_scan**=**os**.**listdir(image\_directory**+**'NORMAL/')

In [13]:

**for** i,image\_name **in** enumerate(normal\_scan):

image **=** cv2**.**imread(image\_directory **+**'NORMAL/'**+**image\_name)

image **=** Image**.**fromarray(image , 'RGB')

image **=** image**.**resize((300,300))

dataset**.**append(np**.**array(image))

labels**.**append(0)

In [14]:

pneumonia\_scan**=**os**.**listdir(image\_directory**+**'PNEUMONIA/')

**for** i,image\_name **in** enumerate(pneumonia\_scan):

image **=** cv2**.**imread(image\_directory **+**'PNEUMONIA/'**+**image\_name)

image **=** Image**.**fromarray(image , 'RGB')

image **=** image**.**resize((300,300))

dataset**.**append(np**.**array(image))

labels**.**append(1)

In [15]:

dataset **=** np**.**array(dataset)

labels **=** np**.**array(labels)

dataset**.**shape

labels**.**shape

Out[15]:

(2682,)

In [16]:

dataset**=** dataset**/**255

type(dataset)

Out[16]:

numpy.ndarray

In [17]:

**from** sklearn.model\_selection **import** train\_test\_split

In [18]:

X\_train,X\_test,Y\_train,Y\_test **=** train\_test\_split(dataset,labels,test\_size **=**0.2 , random\_state**=**42 ,shuffle**=True**)

In [19]:

print(Y\_train**.**size)

print(Y\_test**.**size)

2145

537

In [11]:

model**=** Sequential()

model**.**add(Conv2D(64,(3,3), input\_shape**=**(300,300,3)))

model**.**add(Activation("relu"))

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

model**.**add(Conv2D(32,(3,3), kernel\_initializer**=**"he\_uniform"))

model**.**add(Activation("relu"))

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

model**.**add(Conv2D(32,(3,3) , kernel\_initializer**=**"he\_uniform"))

model**.**add(Activation("relu"))

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

model**.**add(Flatten())

model**.**add(Dense(32))

model**.**add(Activation("relu"))

model**.**add(Dropout(0.5))

model**.**add(Dense(1))

model**.**add(Activation("sigmoid"))

In [12]:

model**.**compile(loss **=** 'binary\_crossentropy',

optimizer **=** 'adam',

metrics **=** ['accuracy'])

In [13]:

print(model**.**summary())

Model: "sequential"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

conv2d (Conv2D) (None, 298, 298, 64) 1792

activation (Activation) (None, 298, 298, 64) 0

max\_pooling2d (MaxPooling2 (None, 149, 149, 64) 0

D)

conv2d\_1 (Conv2D) (None, 147, 147, 32) 18464

activation\_1 (Activation) (None, 147, 147, 32) 0

max\_pooling2d\_1 (MaxPoolin (None, 73, 73, 32) 0

g2D)

conv2d\_2 (Conv2D) (None, 71, 71, 32) 9248

activation\_2 (Activation) (None, 71, 71, 32) 0

max\_pooling2d\_2 (MaxPoolin (None, 35, 35, 32) 0

g2D)

flatten (Flatten) (None, 39200) 0

dense (Dense) (None, 32) 1254432

activation\_3 (Activation) (None, 32) 0

dropout (Dropout) (None, 32) 0

dense\_1 (Dense) (None, 1) 33

activation\_4 (Activation) (None, 1) 0

=================================================================

Total params: 1283969 (4.90 MB)

Trainable params: 1283969 (4.90 MB)

Non-trainable params: 0 (0.00 Byte)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

None

In [14]:

history **=** model**.**fit(X\_train,

Y\_train,

batch\_size**=**64,

verbose**=**1,

epochs**=**50,

validation\_data**=**(X\_test,Y\_test),

shuffle**=True**

)

Epoch 1/50

34/34 [==============================] - 191s 6s/step - loss: 0.7304 - accuracy: 0.5413 - val\_loss: 0.6471 - val\_accuracy: 0.4935

Epoch 2/50

34/34 [==============================] - 178s 5s/step - loss: 0.6051 - accuracy: 0.6900 - val\_loss: 0.4961 - val\_accuracy: 0.8976

Epoch 3/50

34/34 [==============================] - 171s 5s/step - loss: 0.5246 - accuracy: 0.7939 - val\_loss: 0.4365 - val\_accuracy: 0.9032

Epoch 4/50

34/34 [==============================] - 171s 5s/step - loss: 0.4975 - accuracy: 0.8145 - val\_loss: 0.4671 - val\_accuracy: 0.7970

Epoch 5/50

34/34 [==============================] - 169s 5s/step - loss: 0.4969 - accuracy: 0.8042 - val\_loss: 0.4227 - val\_accuracy: 0.8771

Epoch 6/50

34/34 [==============================] - 168s 5s/step - loss: 0.4772 - accuracy: 0.8242 - val\_loss: 0.4042 - val\_accuracy: 0.9255

Epoch 7/50

34/34 [==============================] - 169s 5s/step - loss: 0.4459 - accuracy: 0.8452 - val\_loss: 0.4157 - val\_accuracy: 0.9274

Epoch 8/50

34/34 [==============================] - 168s 5s/step - loss: 0.4286 - accuracy: 0.8769 - val\_loss: 0.3913 - val\_accuracy: 0.9348

Epoch 9/50

34/34 [==============================] - 168s 5s/step - loss: 0.4122 - accuracy: 0.8848 - val\_loss: 0.3748 - val\_accuracy: 0.9460

Epoch 10/50

34/34 [==============================] - 168s 5s/step - loss: 0.4080 - accuracy: 0.8816 - val\_loss: 0.3611 - val\_accuracy: 0.9460

Epoch 11/50

34/34 [==============================] - 168s 5s/step - loss: 0.3836 - accuracy: 0.8960 - val\_loss: 0.3582 - val\_accuracy: 0.9534

Epoch 12/50

34/34 [==============================] - 169s 5s/step - loss: 0.3716 - accuracy: 0.8946 - val\_loss: 0.3747 - val\_accuracy: 0.9404

Epoch 13/50

34/34 [==============================] - 168s 5s/step - loss: 0.3617 - accuracy: 0.9012 - val\_loss: 0.3465 - val\_accuracy: 0.9590

Epoch 14/50

34/34 [==============================] - 168s 5s/step - loss: 0.3457 - accuracy: 0.9133 - val\_loss: 0.3449 - val\_accuracy: 0.9665

Epoch 15/50

34/34 [==============================] - 168s 5s/step - loss: 0.3363 - accuracy: 0.9226 - val\_loss: 0.3422 - val\_accuracy: 0.9553

Epoch 16/50

34/34 [==============================] - 168s 5s/step - loss: 0.3158 - accuracy: 0.9361 - val\_loss: 0.3316 - val\_accuracy: 0.9628

Epoch 17/50

34/34 [==============================] - 168s 5s/step - loss: 0.3054 - accuracy: 0.9389 - val\_loss: 0.3771 - val\_accuracy: 0.9516

Epoch 18/50

34/34 [==============================] - 168s 5s/step - loss: 0.2859 - accuracy: 0.9459 - val\_loss: 0.3260 - val\_accuracy: 0.9609

Epoch 19/50

34/34 [==============================] - 168s 5s/step - loss: 0.2811 - accuracy: 0.9483 - val\_loss: 0.3738 - val\_accuracy: 0.9460

Epoch 20/50

34/34 [==============================] - 168s 5s/step - loss: 0.2689 - accuracy: 0.9557 - val\_loss: 0.2921 - val\_accuracy: 0.9553

Epoch 21/50

34/34 [==============================] - 168s 5s/step - loss: 0.2677 - accuracy: 0.9524 - val\_loss: 0.3314 - val\_accuracy: 0.9590

Epoch 22/50

34/34 [==============================] - 168s 5s/step - loss: 0.2621 - accuracy: 0.9515 - val\_loss: 0.3171 - val\_accuracy: 0.9572

Epoch 23/50

34/34 [==============================] - 169s 5s/step - loss: 0.2554 - accuracy: 0.9529 - val\_loss: 0.2891 - val\_accuracy: 0.9609

Epoch 24/50

34/34 [==============================] - 168s 5s/step - loss: 0.2457 - accuracy: 0.9562 - val\_loss: 0.3088 - val\_accuracy: 0.9572

Epoch 25/50

34/34 [==============================] - 168s 5s/step - loss: 0.2385 - accuracy: 0.9585 - val\_loss: 0.3873 - val\_accuracy: 0.9348

Epoch 26/50

34/34 [==============================] - 168s 5s/step - loss: 0.2442 - accuracy: 0.9534 - val\_loss: 0.3470 - val\_accuracy: 0.9590

Epoch 27/50

34/34 [==============================] - 170s 5s/step - loss: 0.2357 - accuracy: 0.9552 - val\_loss: 0.3376 - val\_accuracy: 0.9665

Epoch 28/50

34/34 [==============================] - 168s 5s/step - loss: 0.2337 - accuracy: 0.9534 - val\_loss: 0.3037 - val\_accuracy: 0.9683

Epoch 29/50

34/34 [==============================] - 169s 5s/step - loss: 0.2218 - accuracy: 0.9622 - val\_loss: 0.2793 - val\_accuracy: 0.9683

Epoch 30/50

34/34 [==============================] - 168s 5s/step - loss: 0.2196 - accuracy: 0.9590 - val\_loss: 0.2992 - val\_accuracy: 0.9628

Epoch 31/50

34/34 [==============================] - 168s 5s/step - loss: 0.2160 - accuracy: 0.9604 - val\_loss: 0.3256 - val\_accuracy: 0.9628

Epoch 32/50

34/34 [==============================] - 168s 5s/step - loss: 0.2122 - accuracy: 0.9604 - val\_loss: 0.2873 - val\_accuracy: 0.9683

Epoch 33/50

34/34 [==============================] - 168s 5s/step - loss: 0.2002 - accuracy: 0.9660 - val\_loss: 0.2954 - val\_accuracy: 0.9721

Epoch 34/50

34/34 [==============================] - 168s 5s/step - loss: 0.1964 - accuracy: 0.9669 - val\_loss: 0.3007 - val\_accuracy: 0.9534

Epoch 35/50

34/34 [==============================] - 168s 5s/step - loss: 0.1983 - accuracy: 0.9627 - val\_loss: 0.2701 - val\_accuracy: 0.9665

Epoch 36/50

34/34 [==============================] - 168s 5s/step - loss: 0.2145 - accuracy: 0.9510 - val\_loss: 0.3306 - val\_accuracy: 0.9553

Epoch 37/50

34/34 [==============================] - 168s 5s/step - loss: 0.1895 - accuracy: 0.9664 - val\_loss: 0.3081 - val\_accuracy: 0.9609

Epoch 38/50

34/34 [==============================] - 168s 5s/step - loss: 0.1960 - accuracy: 0.9604 - val\_loss: 0.2735 - val\_accuracy: 0.9665

Epoch 39/50

34/34 [==============================] - 168s 5s/step - loss: 0.1841 - accuracy: 0.9548 - val\_loss: 0.2243 - val\_accuracy: 0.9702

Epoch 40/50

34/34 [==============================] - 168s 5s/step - loss: 0.1775 - accuracy: 0.9538 - val\_loss: 0.1847 - val\_accuracy: 0.9590

Epoch 41/50

34/34 [==============================] - 168s 5s/step - loss: 0.1453 - accuracy: 0.9632 - val\_loss: 0.2021 - val\_accuracy: 0.9609

Epoch 42/50

34/34 [==============================] - 168s 5s/step - loss: 0.1777 - accuracy: 0.9487 - val\_loss: 0.1464 - val\_accuracy: 0.9665

Epoch 43/50

34/34 [==============================] - 167s 5s/step - loss: 0.1535 - accuracy: 0.9483 - val\_loss: 0.2074 - val\_accuracy: 0.9646

Epoch 44/50

34/34 [==============================] - 168s 5s/step - loss: 0.1404 - accuracy: 0.9576 - val\_loss: 0.1902 - val\_accuracy: 0.9628

Epoch 45/50

34/34 [==============================] - 168s 5s/step - loss: 0.1396 - accuracy: 0.9566 - val\_loss: 0.3050 - val\_accuracy: 0.9534

Epoch 46/50

34/34 [==============================] - 167s 5s/step - loss: 0.1265 - accuracy: 0.9627 - val\_loss: 0.2594 - val\_accuracy: 0.9628

Epoch 47/50

34/34 [==============================] - 167s 5s/step - loss: 0.1390 - accuracy: 0.9538 - val\_loss: 0.1303 - val\_accuracy: 0.9609

Epoch 48/50

34/34 [==============================] - 167s 5s/step - loss: 0.1294 - accuracy: 0.9576 - val\_loss: 0.1956 - val\_accuracy: 0.9628

Epoch 49/50

34/34 [==============================] - 167s 5s/step - loss: 0.1104 - accuracy: 0.9618 - val\_loss: 0.2105 - val\_accuracy: 0.9646

Epoch 50/50

34/34 [==============================] - 167s 5s/step - loss: 0.1188 - accuracy: 0.9594 - val\_loss: 0.1306 - val\_accuracy: 0.9683

In [15]:

tf**.**keras**.**models**.**save\_model(

model,

"CNN\_bal.model",

overwrite**=True**,

include\_optimizer**=True**

)

INFO:tensorflow:Assets written to: CNN\_bal.model\assets

INFO:tensorflow:Assets written to: CNN\_bal.model\assets

In [20]:

**from** tensorflow.keras.models **import** load\_model

resnet\_bal **=** load\_model("resnet\_b\_model.model")

In [21]:

**from** tensorflow.keras.models **import** load\_model

CNN\_model **=** load\_model("CNN\_bal.model")

In [22]:

CNN\_model**.**evaluate(X\_test,Y\_test)

17/17 [==============================] - 10s 560ms/step - loss: 0.1306 - accuracy: 0.9683

Out[22]:

[0.13062962889671326, 0.968342661857605]

In [23]:

test\_values1 **=** CNN\_model**.**predict(X\_test)

y\_preds1 **=** test\_values1**.**ravel()

17/17 [==============================] - 8s 447ms/step

In [24]:

test\_values2 **=** resnet\_bal**.**predict(X\_test)

y\_preds2 **=** test\_values2**.**ravel()

17/17 [==============================] - 48s 3s/step

In [25]:

**from** sklearn.metrics **import** confusion\_matrix

y\_preds1 **=** (test\_values1**>=** 0.5)**.**astype(int)

cm**=**confusion\_matrix(Y\_test,y\_preds1)

print(cm)

[[258 7]

[ 10 262]]

In [26]:

y\_preds1 **=** test\_values1**.**ravel()

In [23]:

**import** matplotlib.pyplot **as** plt

**%matplotlib** inline

**from** sklearn.metrics **import** roc\_curve

fpr1,tpr1,thresholds1 **=** roc\_curve(Y\_test,y\_preds1)

fpr2,tpr2,thresholds2 **=** roc\_curve(Y\_test,y\_preds2)

plt**.**figure(2)

plt**.**plot([0,1],[0,1],'y--')

plt**.**plot(fpr1,tpr1,marker**=**'.')

plt**.**plot(fpr2,tpr2,marker**=**'\*')

plt**.**xlabel("False Positive Rate")

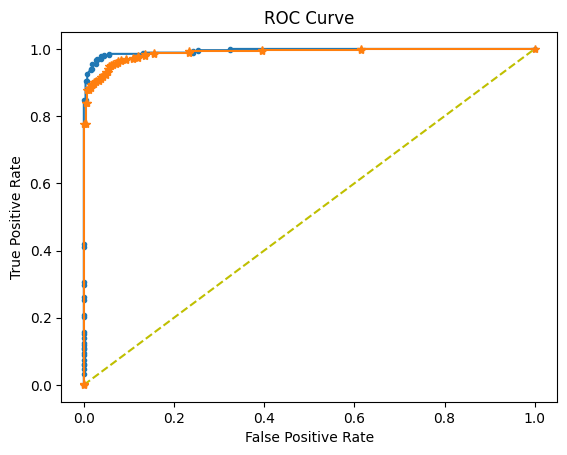
plt**.**ylabel("True Positive Rate")

plt**.**title("ROC Curve")

plt**.**show

Out[23]:

<function matplotlib.pyplot.show(close=None, block=None)>



In [24]:

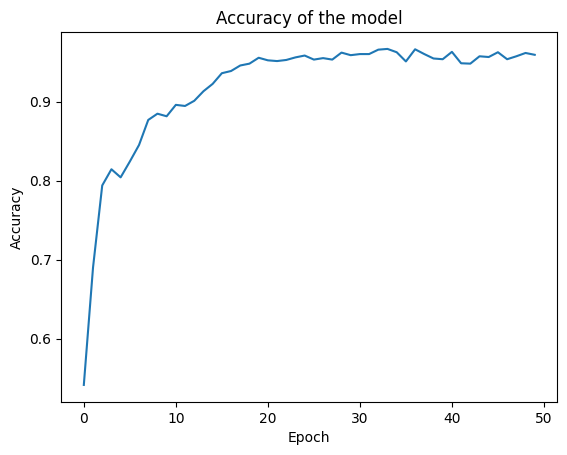
plt**.**plot(model**.**history**.**history['accuracy'])

plt**.**xlabel('Epoch')

plt**.**ylabel('Accuracy')

plt**.**title('Accuracy of the model')

plt**.**show()



In [25]:

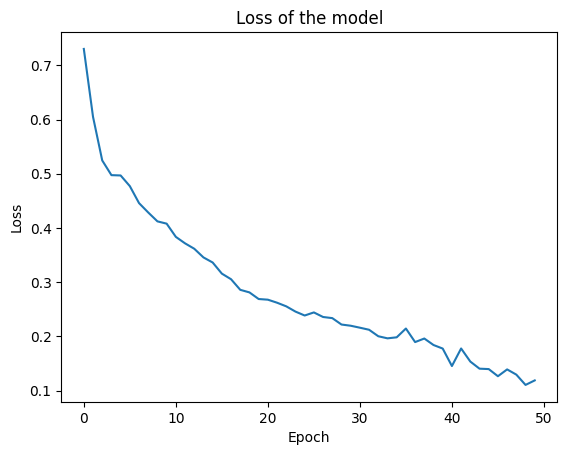
plt**.**plot(model**.**history**.**history['loss'])

plt**.**xlabel('Epoch')

plt**.**ylabel('Loss')

plt**.**title('Loss of the model')

plt**.**show()



In [27]:

tests**=**[]

sample**=**os**.**listdir('pam/')

**for** i,image\_name **in** enumerate(sample):

image **=** cv2**.**imread('pam/'**+**image\_name)

image **=** Image**.**fromarray(image , 'RGB')

image **=** image**.**resize((300,300))

tests**.**append(np**.**array(image))

tests**=**np**.**array(tests)

In [29]:

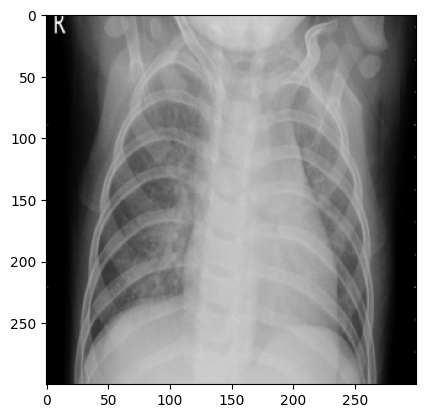
**import** matplotlib.pyplot **as** plt

**%matplotlib** inline

plt**.**imshow(image)

Out[29]:

<matplotlib.image.AxesImage at 0x21ee2018890>



In [29]:

CNN\_model**.**summary()

Model: "sequential"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

=================================================================

conv2d (Conv2D) (None, 298, 298, 64) 1792

activation (Activation) (None, 298, 298, 64) 0

max\_pooling2d (MaxPooling2 (None, 149, 149, 64) 0

D)

conv2d\_1 (Conv2D) (None, 147, 147, 32) 18464

activation\_1 (Activation) (None, 147, 147, 32) 0

max\_pooling2d\_1 (MaxPoolin (None, 73, 73, 32) 0

g2D)

conv2d\_2 (Conv2D) (None, 71, 71, 32) 9248

activation\_2 (Activation) (None, 71, 71, 32) 0

max\_pooling2d\_2 (MaxPoolin (None, 35, 35, 32) 0

g2D)

flatten (Flatten) (None, 39200) 0

dense (Dense) (None, 32) 1254432

activation\_3 (Activation) (None, 32) 0

dropout (Dropout) (None, 32) 0

dense\_1 (Dense) (None, 1) 33

activation\_4 (Activation) (None, 1) 0

=================================================================

Total params: 1283969 (4.90 MB)

Trainable params: 1283969 (4.90 MB)

Non-trainable params: 0 (0.00 Byte)

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

In [30]:

difference**=** []

In [31]:

**for** i,j **in** enumerate(Y\_test):

difference**.**append((j**-**y\_preds1[i])**\*\***2)

In [32]:

diff **=** np**.**array(difference)

In [33]:

mse **=** diff**.**mean()

In [34]:

**import** math

rmse **=** math**.**sqrt(mse)

In [35]:

print("root mean square error = ",rmse)

print("mean square error =",mse)

root mean square error = 0.15616943805591527

mean square error = 0.024388893382700356