In [1]:
```
from tensorflow.keras.applications.inception_v3 import InceptionV3
from tensorflow.keras.layers import Conv2D, Flatten, Dense, MaxPool2D, BatchNorma
from tensorflow.keras.applications.resnet50 import ResNet50, preprocess_input, de
from tensorflow.keras.preprocessing import image
from tensorflow.keras.preprocessing.image import ImageDataGenerator, load_img
from tensorflow.keras.models import Sequential, Model
```

In [2]:
```
import matplotlib.pyplot as plt
import numpy as np
import splitfolders
```

In [3]:
```
!pip install split-folders
```

```
Requirement already satisfied: split-folders in c:\users\fahim\anaconda3\envs\t
ensorflow\lib\site-packages (0.5.1)
```

In [4]:
```
!pip install sklearn
```

```
Requirement already satisfied: sklearn in c:\users\fahim\anaconda3\envs\tensorf
low\lib\site-packages (0.0)
Requirement already satisfied: scikit-learn in c:\users\fahim\anaconda3\envs\te
nsorflow\lib\site-packages (from sklearn) (1.1.2)
Requirement already satisfied: numpy>=1.17.3 in c:\users\fahim\anaconda3\envs\t
ensorflow\lib\site-packages (from scikit-learn->sklearn) (1.22.4)
Requirement already satisfied: joblib>=1.0.0 in c:\users\fahim\anaconda3\envs\t
ensorflow\lib\site-packages (from scikit-learn->sklearn) (1.1.0)
Requirement already satisfied: scipy>=1.3.2 in c:\users\fahim\anaconda3\envs\te
nsorflow\lib\site-packages (from scikit-learn->sklearn) (1.7.1)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\fahim\anaconda3
\envs\tensorflow\lib\site-packages (from scikit-learn->sklearn) (3.1.0)
```

In [5]:
```
SEED = 42
```

In [6]:
```
TRAIN_R = 0.6  # Train ratio
VAL_R = 0.1
TEST_R = 0.3
```

In [7]:
```
IMG_HEIGHT, IMG_WIDTH = (128, 128)
BATCH_SIZE = 32
```

In [8]:
```
DATA_DIR_PATH = "I:/mastits/Dataset/"
#I:\mastits\Dataset
#I:\mastits\TrainTest

OUTPUT_DIR = "I:/mastits/TrainTest/"
```

In [9]:

```
splitfolders.ratio(DATA_DIR_PATH, OUTPUT_DIR, seed=SEED, ratio=(TRAIN_R, VAL_R, T

train_data_dir = f"{OUTPUT_DIR}/train"
valid_data_dir = f"{OUTPUT_DIR}/val"
test_data_dir = f"{OUTPUT_DIR}/test"

train_datagen = ImageDataGenerator(
    preprocessing_function=preprocess_input,
    shear_range=0.2,
    zoom_range=0.2,
    horizontal_flip=True)

test_datagen = ImageDataGenerator(
    preprocessing_function=preprocess_input)


train_generator = train_datagen.flow_from_directory(
    train_data_dir,
    target_size=(IMG_HEIGHT, IMG_WIDTH),
    batch_size=BATCH_SIZE,
    class_mode="binary")

valid_generator = train_datagen.flow_from_directory(
    valid_data_dir,
    target_size=(IMG_HEIGHT, IMG_WIDTH),
    batch_size=BATCH_SIZE,
    class_mode="binary")
```

```python

test_generator = test_datagen.flow_from_directory(

    test_data_dir,

    target_size=(IMG_HEIGHT, IMG_WIDTH),

    batch_size=1,

    class_mode="binary")


EPOCHS = 100
```

```
Found 1360 images belonging to 2 classes.
Found 226 images belonging to 2 classes.
Found 682 images belonging to 2 classes.
```

In [10]:

```
base_model = InceptionV3(include_top=False, weights=None)



x = base_model.output
x = GlobalAveragePooling2D()(x)
x = Dense(1024, activation="relu")(x)



predictions = Dense(1, activation="sigmoid")(x)
model = Model(inputs=base_model.input, outputs=predictions)



for layer in base_model.layers:

    layer.trainable = False



model.compile(

    optimizer="adam",

    loss="binary_crossentropy",

    metrics=["acc"])







history = model.fit(train_generator,

                    validation_data=valid_generator,

                    epochs=EPOCHS)
```

```
9971 - val_loss: 0.0146 - val_acc: 0.9956
Epoch 51/100
43/43 [==============================] - 91s 2s/step - loss: 0.0136 - acc: 0.
9971 - val_loss: 0.0220 - val_acc: 0.9956
Epoch 52/100
43/43 [==============================] - 91s 2s/step - loss: 0.0104 - acc: 0.
9978 - val_loss: 0.0199 - val_acc: 0.9912
Epoch 53/100
43/43 [==============================] - 91s 2s/step - loss: 0.0125 - acc: 0.
9971 - val_loss: 0.0193 - val_acc: 0.9912
Epoch 54/100
43/43 [==============================] - 91s 2s/step - loss: 0.0105 - acc: 0.
9956 - val_loss: 0.0201 - val_acc: 0.9912
Epoch 55/100
43/43 [==============================] - 91s 2s/step - loss: 0.0120 - acc: 0.
9963 - val_loss: 0.0159 - val_acc: 0.9956
Epoch 56/100
```

```
43/43 [==============================] - 91s 2s/step - loss: 0.0091 - acc: 0.
9985 - val_loss: 0.0104 - val_acc: 0.9956
Epoch 57/100
```

In [20]:
```
x_test = []
```

In [21]:
```
from sklearn.metrics import confusion_matrix
```

In [22]:
```
y_true =[]
```

In [23]:
```
for i in range(682):
    x,y=(test_generator.next())
    y_true.append(y.tolist())
    x_test.append(x.tolist()[0])
```

In [24]:
```
y_pred = model.predict(x_test)
y_pred = y_pred>0.5
```

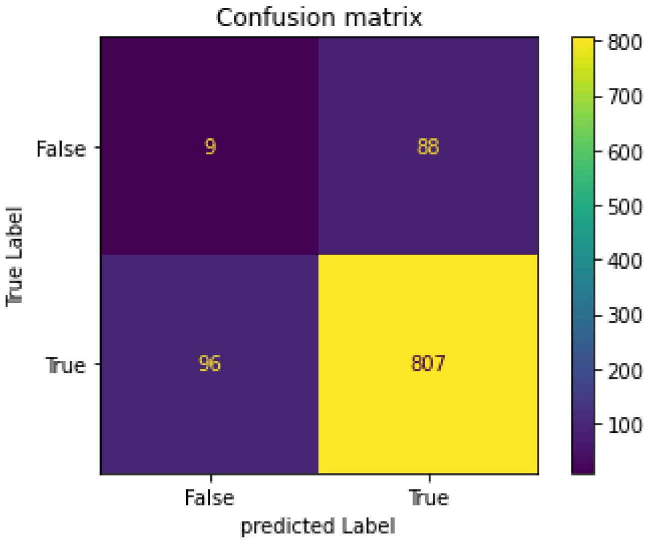
In [25]:
```
cm = confusion_matrix(y_true,y_pred)
```

In [26]:
```
print(cm)
```

```
[[332   2]
 [  2 346]]
```

In [27]:

```
import matplotlib.pyplot as plt
import numpy
from sklearn import metrics
import seaborn as sns

actual = numpy.random.binomial(1,.9,size = 1000)
predicted = numpy.random.binomial(1,.9,size = 1000)

cm = metrics.confusion_matrix(actual, predicted)


cm_display = metrics.ConfusionMatrixDisplay(confusion_matrix = cm, display_labels

cm_display.plot()
plt.title('Confusion matrix')
plt.xlabel('predicted Label',color='black')
plt.ylabel('True Label',color='black')
plt.gcf().axes[0].tick_params(colors='black')
plt.gcf().axes[1].tick_params(colors='black')




plt.show()
```



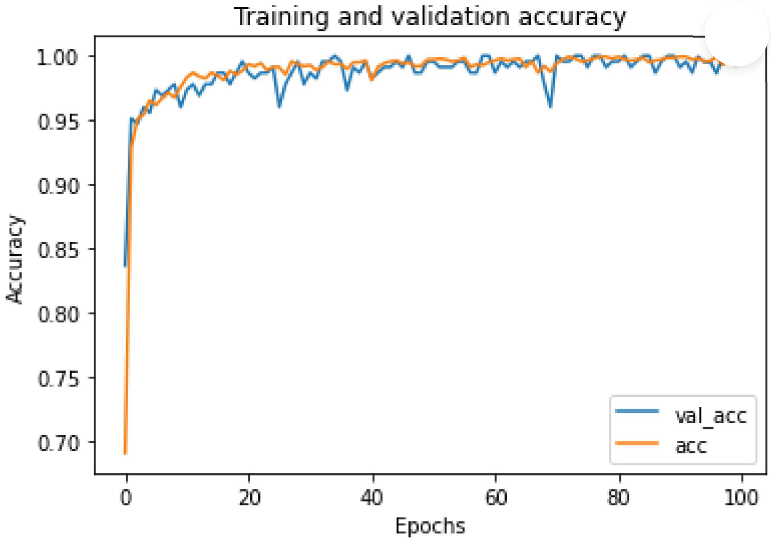
In [28]:

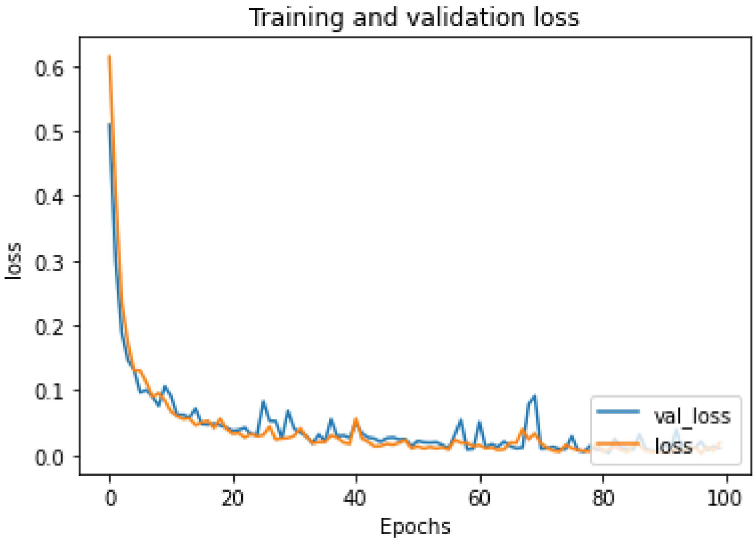
```
from sklearn.metrics import classification_report
```

```
In [29]: print(classification_report(y_true, y_pred, target_names=['abnormal','normal']))
```

```
              precision    recall  f1-score   support

    abnormal       0.99      0.99      0.99       334
      normal       0.99      0.99      0.99       348

    accuracy                           0.99       682
   macro avg       0.99      0.99      0.99       682
weighted avg       0.99      0.99      0.99       682
```

```
In [31]: plt.plot(np.arange(EPOCHS),history.history['val_acc'],label='val_acc')
         plt.plot(np.arange(EPOCHS),history.history['acc'],label='acc')

         plt.title('Training and validation accuracy')
         plt.xlabel('Epochs')
         plt.ylabel('Accuracy')
         plt.legend(loc='lower right')

         plt.show()

         plt.plot(np.arange(EPOCHS),history.history['val_loss'],label='val_loss')
         plt.plot(np.arange(EPOCHS),history.history['loss'],label='loss')



         plt.ylabel('loss')



         plt.title('Training and validation loss')
         plt.xlabel('Epochs')

         plt.legend(loc='lower right')
         plt.show()
```




Training and validation loss
loss
0.0
0.1
0.2
0.3
0.4
0.5
0.6
Epochs
0
20
40
60
80
100
val_loss
loss

```
In [ ]:
```