```
from google.colab import drive
drive.mount("/content/drive")
```

```
Go to this URL in a browser: https://accounts.google.com/o/oauth2/auth?client_id=947318989803-

Enter your authorization code:
··········
Mounted at /content/drive
```

```
!mkdir /root/.kaggle
```

```
!cp /content/drive/'My Drive'/'Colab Notebooks'/kaggle.json /root/.kaggle/
```

```
!pip install -q kaggle
!kaggle datasets download -d iarunava/cell-images-for-detecting-malaria
```

```
Downloading cell-images-for-detecting-malaria.zip to /content
 98% 661M/675M [00:07<00:00, 109MB/s]
100% 675M/675M [00:07<00:00, 90.4MB/s]
```

```
!unzip cell-images-for-detecting-malaria.zip
```

```
!git clone https://github.com/mjkvaak/ImageDataAugmentor.git
```

```
Cloning into 'ImageDataAugmentor'...
remote: Enumerating objects: 141, done.
remote: Counting objects: 100% (141/141), done.
remote: Compressing objects: 100% (111/111), done.
remote: Total 306 (delta 72), reused 70 (delta 23), pack-reused 165
Receiving objects: 100% (306/306), 146.55 KiB | 10.47 MiB/s, done.
Resolving deltas: 100% (171/171), done.
```

```
pip install git+https://github.com/mjkvaak/ImageDataAugmentor
```

```
    Collecting git+https://github.com/mjkvaak/ImageDataAugmentor
      Cloning https://github.com/mjkvaak/ImageDataAugmentor to /tmp/pip-req-build-07_uh1i3
      Running command git clone -q https://github.com/mjkvaak/ImageDataAugmentor /tmp/pip-req-buil
    Collecting opencv-python>=4.2
      Downloading https://files.pythonhosted.org/packages/46/16/e49e5abd27d988e687634f58b0aa329fe7
         |████████████████████████████████| 49.4MB 54kB/s
    Requirement already satisfied: matplotlib in /usr/local/lib/python3.6/dist-packages (from Imag
    Requirement already satisfied: Pillow in /usr/local/lib/python3.6/dist-packages (from ImageDat
```

```
pip install twine
```

```
    Collecting twine
      Downloading https://files.pythonhosted.org/packages/ad/db/b2c65078b783c6694bdfa0911bbbe0e2be
```

```
pip install efficientnet
```

```
Collecting efficientnet
  Downloading https://files.pythonhosted.org/packages/28/91/67848a143b54c331605bfba5fd31cf4e9d
Collecting keras-applications<=1.0.8,>=1.0.7
  Downloading https://files.pythonhosted.org/packages/71/e3/19762fdfc62877ae9102edf6342d71b28f
     |████████████████████████████████| 51kB 3.7MB/s
Requirement already satisfied: scikit-image in /usr/local/lib/python3.6/dist-packages (from ef
Requirement already satisfied: numpy>=1.9.1 in /usr/local/lib/python3.6/dist-packages (from ke
Requirement already satisfied: h5py in /usr/local/lib/python3.6/dist-packages (from keras-appl
Requirement already satisfied: networkx>=2.0 in /usr/local/lib/python3.6/dist-packages (from s
Requirement already satisfied: imageio>=2.3.0 in /usr/local/lib/python3.6/dist-packages (from
Requirement already satisfied: matplotlib!=3.0.0,>=2.0.0 in /usr/local/lib/python3.6/dist-pack
Requirement already satisfied: scipy>=0.19.0 in /usr/local/lib/python3.6/dist-packages (from s
Requirement already satisfied: PyWavelets>=0.4.0 in /usr/local/lib/python3.6/dist-packages (fr
Requirement already satisfied: pillow>=4.3.0 in /usr/local/lib/python3.6/dist-packages (from s
Requirement already satisfied: six in /usr/local/lib/python3.6/dist-packages (from h5py->keras
Requirement already satisfied: decorator>=4.3.0 in /usr/local/lib/python3.6/dist-packages (fro
Requirement already satisfied: python-dateutil>=2.1 in /usr/local/lib/python3.6/dist-packages
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.6/dist-packages (from ma
Requirement already satisfied: pyparsing!=2.0.4,!=2.1.2,!=2.1.6,>=2.0.1 in /usr/local/lib/pyth
Requirement already satisfied: kiwisolver>=1.0.1 in /usr/local/lib/python3.6/dist-packages (fr
Installing collected packages: keras-applications, efficientnet
Successfully installed efficientnet-1.1.0 keras-applications-1.0.8

Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.6/dist-packages (fr
```

```
from ImageDataAugmentor.image_data_augmentor import *
import glob
import pandas as pd
import os
import numpy as np
from sklearn.model_selection import train_test_split

parasitized=os.listdir('cell_images/Parasitized')
uninfected=os.listdir('cell_images/Uninfected')
par=list(zip(parasitized,['parasitized']*(len(parasitized))))
parasitized_df = pd.DataFrame(par,columns=['file','label'])
parasitized_df.head()
```

	file	label
0	C66P27N_ThinF_IMG_20150818_164008_cell_154.png	parasitized
1	C51AP12thinF_IMG_20150724_155046_cell_102.png	parasitized
2	C145P106ThinF_IMG_20151016_154719_cell_151.png	parasitized
3	C184P145ThinF_IMG_20151203_102543_cell_147.png	parasitized
4	C51AP12thinF_IMG_20150724_153313_cell_102.png	parasitized

```
uninf=list(zip(uninfected,['uninfected']*(len(uninfected))))
uninfected_df = pd.DataFrame(uninf,columns=['file','label'])
uninfected_df.head()
```

	file	label
0	C149P110ThinF_IMG_20151115_114729_cell_8.png	uninfected
1	C78P39ThinF_IMG_20150606_104426_cell_24.png	uninfected
2	C67P28N_ThinF_IMG_20150819_133000_cell_23.png	uninfected
3	C33P1thinF_IMG_20150619_121102a_cell_176.png	uninfected

```
parasitized_df, parasitized_df1 = train_test_split(parasitized_df,
                                                   test_size=0.2,
                                                   random_state=0)

uninfected_df, uninfected_df1 = train_test_split(uninfected_df,
                                                 test_size=0.2,
                                                 random_state=0)


dataframe=pd.concat([parasitized_df, uninfected_df])
dataframe.head()
```

	file	label
309	C144P105ThinF_IMG_20151015_160529_cell_271.png	parasitized
785	C132P93ThinF_IMG_20151004_151733_cell_150.png	parasitized
6887	C179P140ThinF_IMG_20151127_153521_cell_172.png	parasitized
7926	C175P136NThinF_IMG_20151127_141253_cell_249.png	parasitized
13767	C39P4thinF_original_IMG_20150622_110900_cell_1...	parasitized

```
dataframe1=pd.concat([parasitized_df1, uninfected_df1])
dataframe1.head()
```

	file	label
1238	C51AP12thinF_IMG_20150724_153313_cell_99.png	parasitized
9320	C39P4thinF_original_IMG_20150622_111942_cell_1...	parasitized
8919	C179P140ThinF_IMG_20151127_153436_cell_174.png	parasitized
221	C39P4thinF_original_IMG_20150622_114804_cell_1...	parasitized
643	C126P87ThinF_IMG_20151004_104408_cell_168.png	parasitized

```
df = pd.get_dummies(dataframe['label'])
df.head()
```

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from albumentations import *

aug=Compose([RandomRotate90(),
             Flip(),
             Transpose(),
             OneOf([IAAAdditiveGaussianNoise(),
                    GaussNoise(),], p=0.2),
             OneOf([MotionBlur(p=.2),
                    MedianBlur(blur_limit=3, p=.1),
                    Blur(blur_limit=3, p=.1),], p=0.3),
             ShiftScaleRotate(shift_limit=0.0625,
                              scale_limit=0.2,
                              rotate_limit=45, p=.2),
             OneOf([OpticalDistortion(p=0.3),
                    GridDistortion(p=.1),
                    IAAPiecewiseAffine(p=0.3),], p=0.3),
             OneOf([CLAHE(clip_limit=2),
                    IAASharpen(),
                    IAAEmboss(),
                    RandomContrast(),
                    RandomBrightness(),], p=0.3),
             #HueSaturationValue(p=0.3),
             ], p=1)


from ImageDataAugmentor.image_data_augmentor import *

data_gen= ImageDataAugmentor(rescale=1/255, augment=aug)

img_shape=224
batch_size=16

from tensorflow.keras.models import Sequential,Model
from tensorflow.keras.layers import Activation, Dropout, Flatten, Dense, \
                                    Conv2D, MaxPool2D, BatchNormalization, \
                                    Input, MaxPooling2D, GlobalMaxPooling2D, \
                                    concatenate
from tensorflow.keras.layers import GlobalAveragePooling2D
from tensorflow.keras.applications.resnet50 import ResNet50
from tensorflow.keras.callbacks import ModelCheckpoint,ReduceLROnPlateau
from tensorflow.keras.optimizers import Adam
import tensorflow.keras.backend as K
import tensorflow as tf
import random
from sklearn.metrics import confusion_matrix


random.seed(1234)
from sklearn.model_selection import KFold
from sklearn.model_selection import StratifiedKFold
import gc
import seaborn as sns
from sklearn.metrics import roc_curve
from sklearn.metrics import roc_auc_score
```

	parasitized	uninfected

```
df1 = pd.get_dummies(dataframe1['label'])
df1.head()
```

	parasitized	uninfected
1238	1	0
9320	1	0
8919	1	0
221	1	0
643	1	0

```
df = pd.concat([dataframe, df], axis=1)
df.head()
```

	file	label	parasitized	uninfected
309	C144P105ThinF_IMG_20151015_160529_cell_271.png	parasitized	1	0
785	C132P93ThinF_IMG_20151004_151733_cell_150.png	parasitized	1	0
6887	C179P140ThinF_IMG_20151127_153521_cell_172.png	parasitized	1	0
7926	C175P136NThinF_IMG_20151127_141253_cell_249.png	parasitized	1	0
13767	C39P4thinF_original_IMG_20150622_110900_cell_1...	parasitized	1	0

```
df1 = pd.concat([dataframe1, df1], axis=1)
df1.head()
```

	file	label	parasitized	uninfected
1238	C51AP12thinF_IMG_20150724_153313_cell_99.png	parasitized	1	0
9320	C39P4thinF_original_IMG_20150622_111942_cell_1...	parasitized	1	0
8919	C179P140ThinF_IMG_20151127_153436_cell_174.png	parasitized	1	0
221	C39P4thinF_original_IMG_20150622_114804_cell_1...	parasitized	1	0
643	C126P87ThinF_IMG_20151004_104408_cell_168.png	parasitized	1	0

```
df.shape
```

```
(22048, 4)
```

```
df1.shape
```

```
(5512, 4)
```

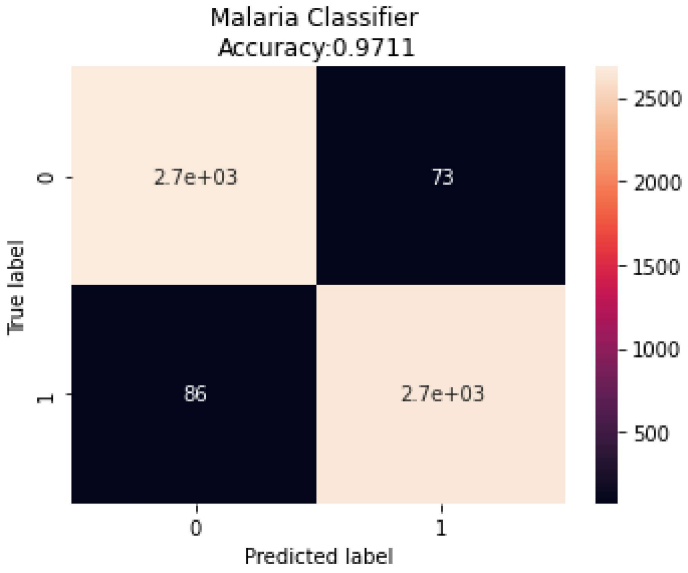
```
!mkdir data
!cp -r cell_images/Parasitized/* data
!cp -r cell_images/Uninfected/* data
```

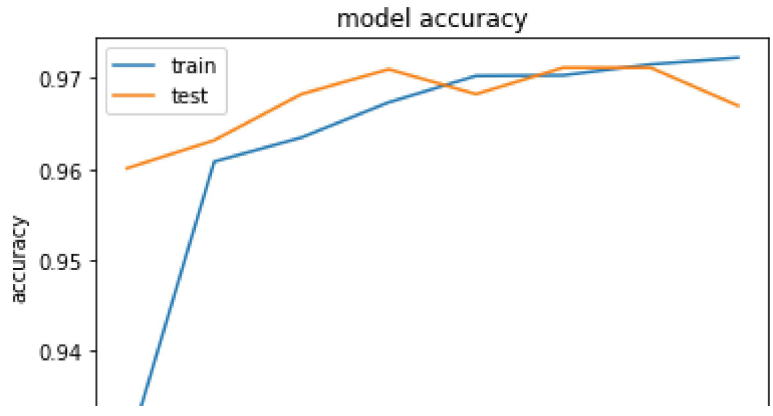
```
from sklearn.metrics import roc_auc_score
from sklearn.metrics import auc
from sklearn.metrics import precision_recall_curve
from sklearn.metrics import classification_report, accuracy_score, f1_score
import efficientnet.tfkeras as efn

def custom_loss(y_true, y_pred):
  return tf.keras.losses.categorical_crossentropy(y_true, y_pred,
                                                  label_smoothing=0.1)

classification_reports=[]
i=1
kf = StratifiedKFold(4,shuffle=True,random_state=50)  # Reduced to 4. Faster! (just a test)
history=[]
accuracy=[]
f1_scores=[]
fold=0
for train_index, test_index in kf.split(df, df['label']):
  train=df.iloc[train_index,:]
  val=df.iloc[test_index,:]
  print('-----------fold {}--------------'.format(fold))
  train_generator=data_gen.flow_from_dataframe(train, directory='data',
                                               target_size=(img_shape,
                                                            img_shape),
                                               x_col="file",
                                               y_col=['parasitized','uninfected'],
                                               class_mode='raw',
                                               shuffle=True,
                                               batch_size=batch_size)
  val_generator=data_gen.flow_from_dataframe(val,directory='data',
                                             target_size=(img_shape,img_shape),
                                             x_col="file",
                                             y_col=['parasitized','uninfected'],
                                             class_mode='raw',
                                             shuffle=False,
                                             batch_size=batch_size)
  model =efn.EfficientNetB4(weights ='noisy-student', include_top=False,
                            input_shape = (img_shape,img_shape,3))
  x = model.output
  x = GlobalAveragePooling2D()(x)
  x = Dense(128, activation="relu")(x)
  x = Dropout(0.3)(x)
  x = Dense(64, activation="relu")(x)
  x = Dropout(0.3)(x)
  x = Dense(32, activation="relu")(x)
  x = Dropout(0.3)(x)

  predictions = Dense(2, activation="softmax")(x)

  model = Model(inputs=model.input, outputs=predictions)
  model.compile(optimizer=Adam(0.0001), loss=custom_loss, metrics=['accuracy'])
  results = model.fit(train_generator, epochs =8, #epochs=11,
                      steps_per_epoch=train_generator.n/batch_size,
                      validation_data=val_generator,
                      validation_steps=val_generator.n/batch_size,
                      callbacks=[ReduceLROnPlateau(monitor='val_loss',
                                                   factor=0.5, patience=3,
                                                   min_lr=0.000001)
```

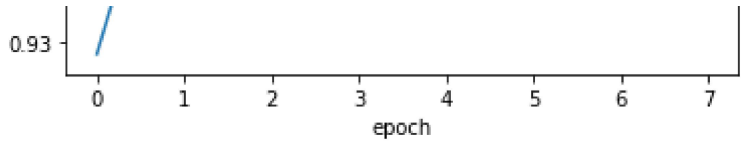
```
                                        min_lr=0.000001),
                                ModelCheckpoint('model_{}.hdf5'.format(fold),
                                                save_best_only=True,
                                                monitor='val_loss',
                                                mode='min')
                                ])
model.load_weights('model_{}.hdf5'.format(fold))
model.save('final_model_{}'.format(fold))
val_generator.reset()
y_pred=model.predict(val_generator,
                     steps=val_generator.n/batch_size,
                     verbose=1)
y_pred=y_pred.round().astype(int)
y_true=val.iloc[:,2::]
y_true = y_true[0:len(y_pred)]
classification_reports.append(classification_report(y_true,
                                                    y_pred,
                                                    target_names=['parasitized',
                                                                  'uninfected'],
                                                    digits=6))
accuracy.append(accuracy_score(y_true,y_pred))
f1=f1_score(y_true,y_pred,average='macro')
print('f1 score is ', f1)
f1_scores.append(f1)
fold+=1
cm=confusion_matrix(y_true.values.argmax(axis=1), y_pred.argmax(axis=1))
plt.figure(figsize=(5.5,4))
sns.heatmap(cm, annot=True)
plt.title('Malaria Classifier \nAccuracy:{0:.4f}'.format( \
                                 accuracy_score(y_true, y_pred)))
plt.ylabel('True label')
plt.xlabel('Predicted label')
plt.show()
plt.plot(results.history['accuracy'])
plt.plot(results.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
plt.plot(results.history['loss'])
plt.plot(results.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
from itertools import cycle
plt.style.use('ggplot')
fpr = dict()
tpr = dict()
roc_auc = dict()
n_classes=2

for i in range(n_classes):
  fpr[i], tpr[i], _ = roc_curve(y_true.values[:, i], y_pred[:, i])
  roc_auc[i] = auc(fpr[i], tpr[i])
```

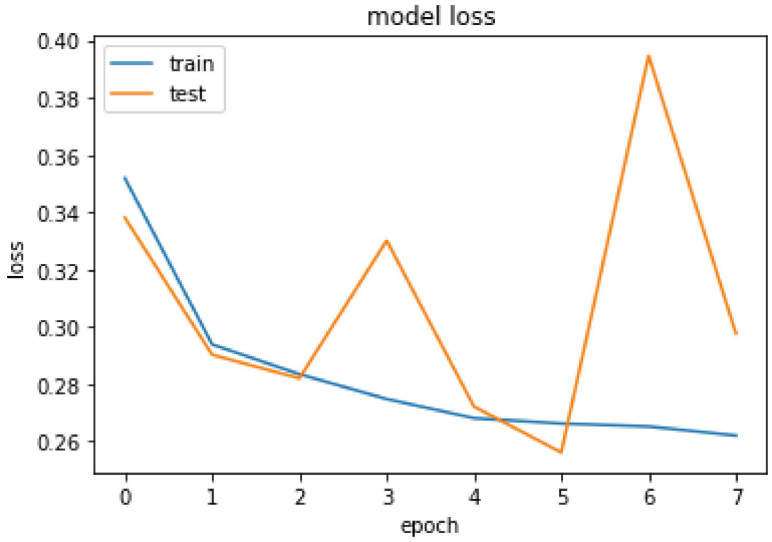
```
colors = cycle(['blue', 'red'])
for i, color in zip(range(n_classes), colors):
  plt.plot(fpr[i], tpr[i], color=color, lw=1.5,
           label='ROC curve of class {0} (area = {1:0.2f})' ''.format(i+1,
                                                                      roc_auc[i]))
plt.plot([0, 1], [0, 1], 'k-', lw=1.5)
plt.xlim([-0.05, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver operating characteristic for binary-class data')
plt.legend(loc="lower right")
plt.show()
del model
tf.keras.backend.clear_session()
gc.collect()
```

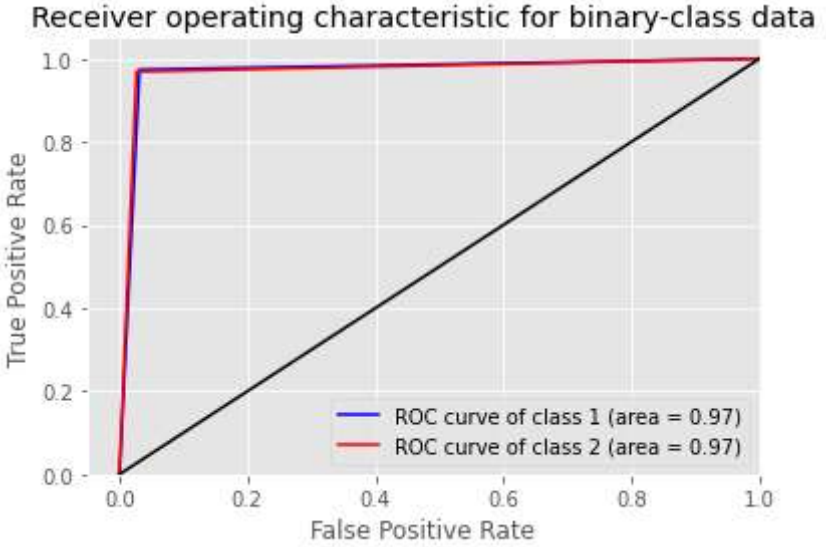
```
/usr/local/lib/python3.6/dist-packages/statsmodels/tools/_testing.py:19: FutureWarning: pandas
  import pandas.util.testing as tm
/usr/local/lib/python3.6/dist-packages/ImageDataAugmentor/dataframe_iterator.py:276: UserWarni
  .format(n_invalid, x_col)
-----------fold 0--------------
Found 16535 validated image filenames.
Found 5511 validated image filenames.
Downloading data from https://github.com/qubvel/efficientnet/releases/download/v0.0.1/efficien
71680000/71678424 [==============================] - 2s 0us/step
Epoch 1/8
1034/1033 [==============================] - 541s 523ms/step - loss: 0.3519 - accuracy: 0.9286
Epoch 2/8
1034/1033 [==============================] - 537s 520ms/step - loss: 0.2939 - accuracy: 0.9608
Epoch 3/8
1034/1033 [==============================] - 518s 501ms/step - loss: 0.2835 - accuracy: 0.9635
Epoch 4/8
1034/1033 [==============================] - 520s 503ms/step - loss: 0.2748 - accuracy: 0.9673
Epoch 5/8
1034/1033 [==============================] - 512s 495ms/step - loss: 0.2681 - accuracy: 0.9702
Epoch 6/8
1034/1033 [==============================] - 515s 498ms/step - loss: 0.2663 - accuracy: 0.9703
Epoch 7/8
1034/1033 [==============================] - 509s 492ms/step - loss: 0.2652 - accuracy: 0.9715
Epoch 8/8
1034/1033 [==============================] - 510s 493ms/step - loss: 0.2621 - accuracy: 0.9722
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tensorflow/python/training/trac
Instructions for updating:
This property should not be used in TensorFlow 2.0, as updates are applied automatically.
WARNING:tensorflow:From /usr/local/lib/python3.6/dist-packages/tensorflow/python/training/trac
Instructions for updating:
This property should not be used in TensorFlow 2.0, as updates are applied automatically.
INFO:tensorflow:Assets written to: final_model_0/assets
345/344 [==============================] - 88s 255ms/step
f1 score is  0.9711484256735678
```



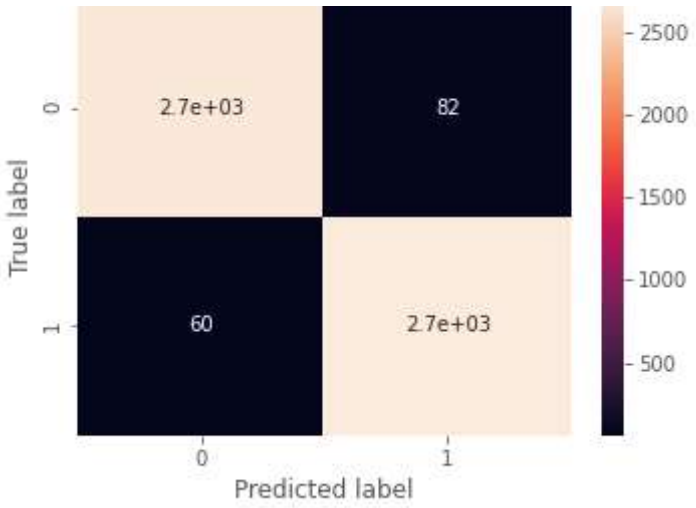


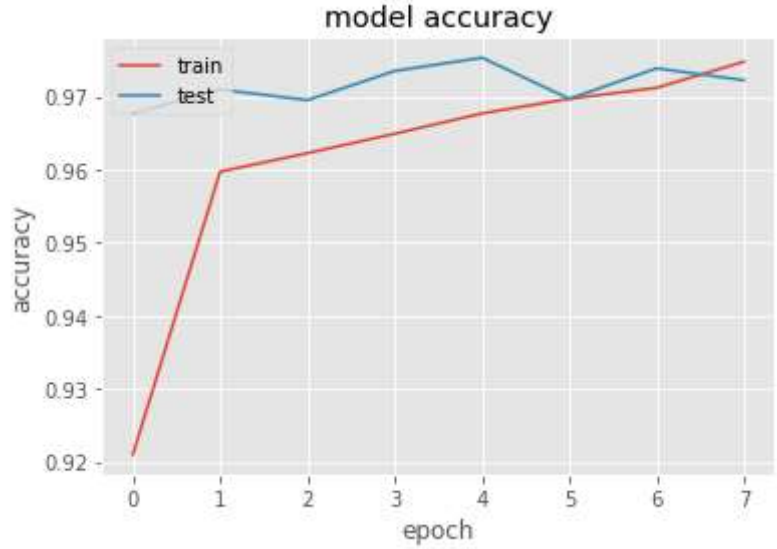


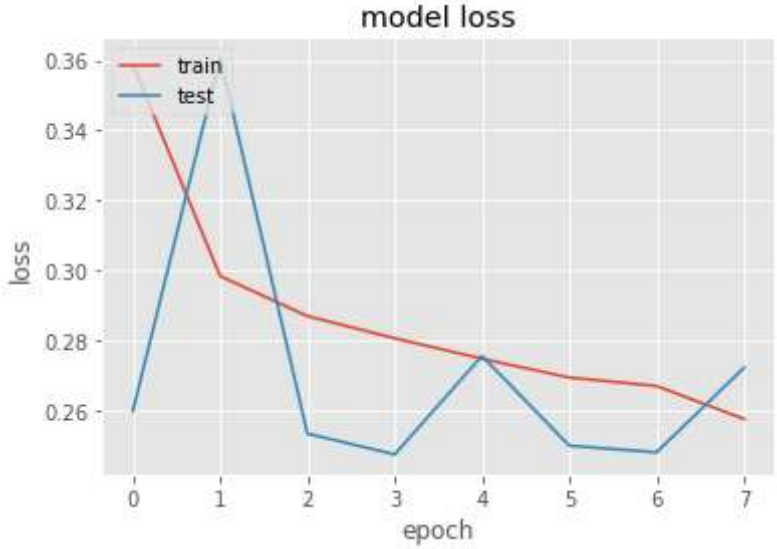


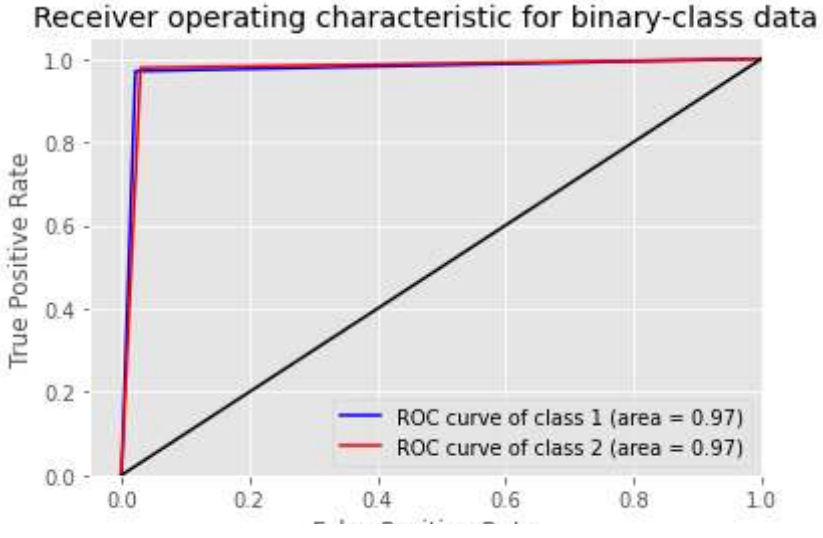



```
----------fold 1-------------
Found 16534 validated image filenames.
Found 5512 validated image filenames.
/usr/local/lib/python3.6/dist-packages/ImageDataAugmentor/dataframe_iterator.py:276: UserWarni
  .format(n_invalid, x_col)
Epoch 1/8
1034/1033 [==============================] - 519s 501ms/step - loss: 0.3588 - accuracy: 0.9210
Epoch 2/8
1034/1033 [==============================] - 510s 493ms/step - loss: 0.2985 - accuracy: 0.9597
Epoch 3/8
1034/1033 [==============================] - 512s 495ms/step - loss: 0.2871 - accuracy: 0.9623
Epoch 4/8
1034/1033 [==============================] - 514s 497ms/step - loss: 0.2808 - accuracy: 0.9649
Epoch 5/8
1034/1033 [==============================] - 515s 498ms/step - loss: 0.2749 - accuracy: 0.9677
Epoch 6/8
1034/1033 [==============================] - 513s 496ms/step - loss: 0.2696 - accuracy: 0.9697
Epoch 7/8
1034/1033 [==============================] - 510s 493ms/step - loss: 0.2672 - accuracy: 0.9712
Epoch 8/8
1034/1033 [==============================] - 510s 493ms/step - loss: 0.2578 - accuracy: 0.9748
INFO:tensorflow:Assets written to: final_model_1/assets
345/344 [==============================] - 92s 266ms/step
f1 score is  0.9742376157194834
```


2.7e+03
82
60
2.7e+03
True label
Predicted label
0
1
2500
2000
1500
1000
500

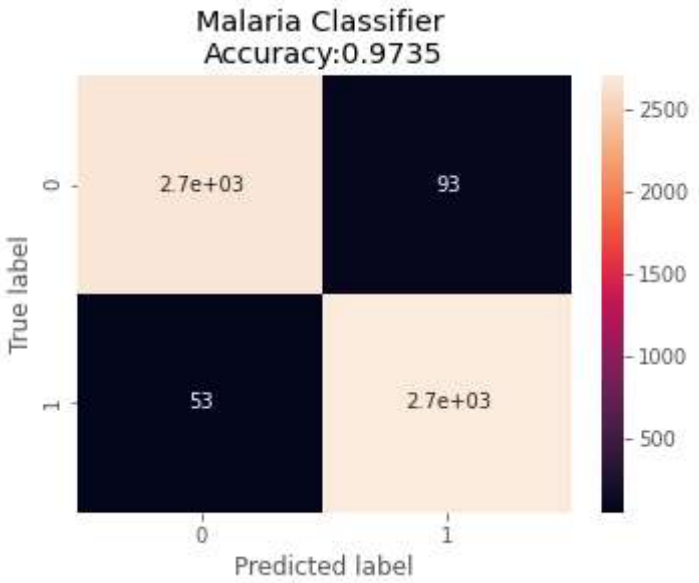

model accuracy
train
test
accuracy
epoch

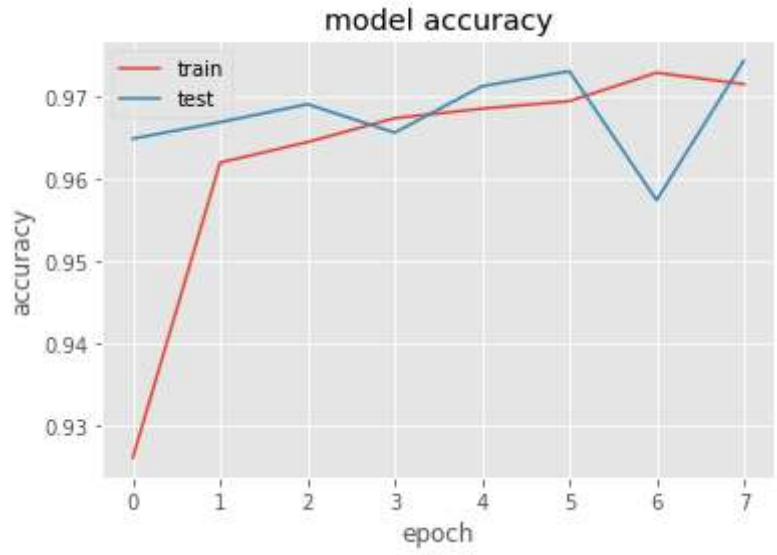

model loss
train
test
loss
epoch

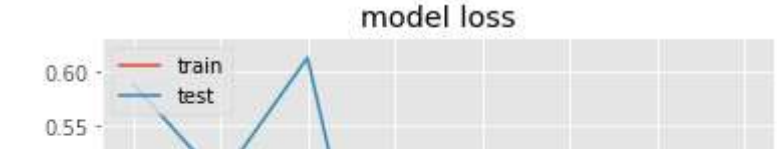

Receiver operating characteristic for binary-class data
True Positive Rate
ROC curve of class 1 (area = 0.97)
ROC curve of class 2 (area = 0.97)

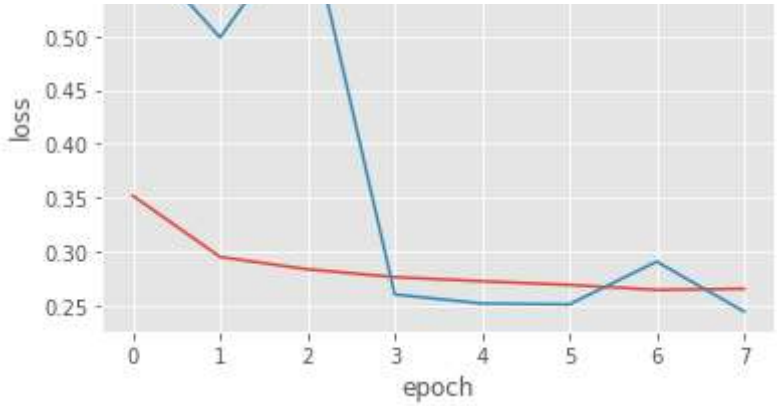
False Positive Rate

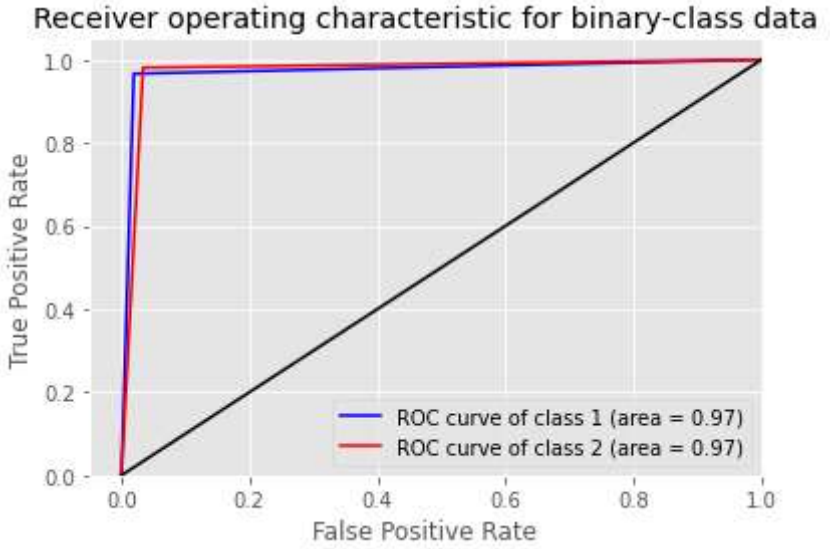
```
----------fold 2-------------
Found 16535 validated image filenames.
Found 5511 validated image filenames.
/usr/local/lib/python3.6/dist-packages/ImageDataAugmentor/dataframe_iterator.py:276: UserWarni
  .format(n_invalid, x_col)
Epoch 1/8
1034/1033 [==============================] - 518s 501ms/step - loss: 0.3516 - accuracy: 0.9262
Epoch 2/8
1034/1033 [==============================] - 513s 496ms/step - loss: 0.2948 - accuracy: 0.9619
Epoch 3/8
1034/1033 [==============================] - 507s 490ms/step - loss: 0.2833 - accuracy: 0.9644
Epoch 4/8
1034/1033 [==============================] - 514s 497ms/step - loss: 0.2760 - accuracy: 0.9673
Epoch 5/8
1034/1033 [==============================] - 515s 498ms/step - loss: 0.2722 - accuracy: 0.9684
Epoch 6/8
1034/1033 [==============================] - 524s 507ms/step - loss: 0.2688 - accuracy: 0.9693
Epoch 7/8
1034/1033 [==============================] - 515s 498ms/step - loss: 0.2641 - accuracy: 0.9728
Epoch 8/8
1034/1033 [==============================] - 525s 507ms/step - loss: 0.2652 - accuracy: 0.9714
INFO:tensorflow:Assets written to: final_model_2/assets
345/344 [==============================] - 93s 271ms/step
f1 score is  0.973506203571993
```



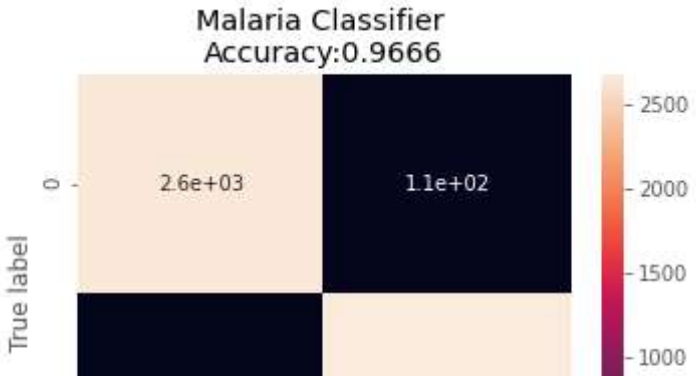









```
-----------fold 3-------------
Found 16534 validated image filenames.
Found 5512 validated image filenames.
/usr/local/lib/python3.6/dist-packages/ImageDataAugmentor/dataframe_iterator.py:276: UserWarni
  .format(n_invalid, x_col)
Epoch 1/8
1034/1033 [==============================] - 524s 507ms/step - loss: 0.3627 - accuracy: 0.9158
Epoch 2/8
1034/1033 [==============================] - 530s 513ms/step - loss: 0.2951 - accuracy: 0.9600
Epoch 3/8
1034/1033 [==============================] - 532s 514ms/step - loss: 0.2807 - accuracy: 0.9664
Epoch 4/8
1034/1033 [==============================] - 533s 516ms/step - loss: 0.2754 - accuracy: 0.9676
Epoch 5/8
1034/1033 [==============================] - 538s 521ms/step - loss: 0.2708 - accuracy: 0.9691
Epoch 6/8
1034/1033 [==============================] - 543s 525ms/step - loss: 0.2640 - accuracy: 0.9719
Epoch 7/8
1034/1033 [==============================] - 543s 525ms/step - loss: 0.2639 - accuracy: 0.9716
Epoch 8/8
1034/1033 [==============================] - 546s 528ms/step - loss: 0.2582 - accuracy: 0.9734
INFO:tensorflow:Assets written to: final_model_3/assets
345/344 [==============================] - 99s 286ms/step
f1 score is  0.9666172984885016
```

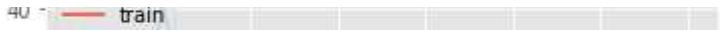



```
accuracy
```

```
[0.9711486118671747,
 0.9742380261248186,
 0.9735075303937579,
 0.9666182873730044]
```

```
print(np.mean(accuracy))
```

```
0.971378113939689
```




```
f1_scores
```

```
[0.9711484256735678, 0.9742376157194834, 0.973506203571993, 0.9666172984885016]
```

```
np.mean(f1_scores)
```

```
0.9713773858633865
```

```
for i in range(4):
  print('--------------FOLD {}-----------------'.format(i))
  print(classification_reports[i])
```

```
     --------------FOLD 0------------------
                   precision    recall  f1-score   support

      parasitized   0.968942  0.973512  0.971222      2756
       uninfected   0.973377  0.968784  0.971075      2755

        micro avg   0.971149  0.971149  0.971149      5511
        macro avg   0.971159  0.971148  0.971148      5511
     weighted avg   0.971159  0.971149  0.971148      5511
      samples avg   0.971149  0.971149  0.971149      5511

     --------------FOLD 1------------------
                   precision    recall  f1-score   support

      parasitized   0.978054  0.970247  0.974135      2756
       uninfected   0.970482  0.978229  0.974340      2756

        micro avg   0.974238  0.974238  0.974238      5512
        macro avg   0.974268  0.974238  0.974238      5512
     weighted avg   0.974268  0.974238  0.974238      5512
      samples avg   0.974238  0.974238  0.974238      5512
```

```
val=df1.iloc[:,:]
va_generator=data_gen.flow_from_dataframe(val,directory='data',
                                          target_size=(img_shape,img_shape),
                                          x_col="file",
                                          y_col=['parasitized','uninfected'],
                                          class_mode='raw',
                                          shuffle=False,
                                          batch_size=batch_size)
```

```
Found 5512 validated image filenames.
--------------FOLD 3------------------
```

```
import tensorflow as tf

model=tf.keras.models.load_model("final_model_3",compile=False)

y_pred=model.predict(val_generator,
                     steps=val_generator.n/batch_size,
                     verbose=1)

y_pred=y_pred.round().astype(int)
y_true=val.iloc[:,2::]
y_true = y_true[0:len(y_pred)]
print(classification_report(y_true, y_pred,
                            target_names=['parasitized','uninfected'], digits=6))
accuracy.append(accuracy_score(y_true,y_pred))
f1=f1_score(y_true,y_pred,average='macro')
print('f1 score is ', f1)
cm=confusion_matrix(y_true.values.argmax(axis=1), y_pred.argmax(axis=1))
plt.figure(figsize=(5.5,4))
sns.heatmap(cm, annot=True)
plt.title('Malaria Classifier \nAccuracy:{0:.4f}'.format(accuracy_score(y_true,
                                                                       y_pred)))
plt.ylabel('True label')
plt.xlabel('Predicted label')
plt.show()
from itertools import cycle
plt.style.use('ggplot')
```

```
plt.style.use('ggplot')
fpr = dict()
tpr = dict()
roc_auc = dict()
n_classes=2
for i in range(n_classes):
  fpr[i], tpr[i], _ = roc_curve(y_true.values[:, i], y_pred[:, i])
  roc_auc[i] = auc(fpr[i], tpr[i])
colors = cycle(['blue', 'red'])
for i, color in zip(range(n_classes), colors):
  plt.plot(fpr[i], tpr[i], color=color, lw=1.5,
           label='ROC curve of class {0} (area = {1:0.2f})' ''.format(i+1,
                                                                      roc_auc[i]))
plt.plot([0, 1], [0, 1], 'k-', lw=1.5)
plt.xlim([-0.05, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver operating characteristic for binary-class data')
plt.legend(loc="lower right")
plt.show()
del model
tf.keras.backend.clear_session()
gc.collect()
```

```
345/344 [==============================] - 100s 291ms/step
              precision    recall  f1-score   support

 parasitized   0.643826  0.637518  0.640656      2756
  uninfected   0.641035  0.647315  0.644160      2756

   micro avg   0.642417  0.642417  0.642417      5512
   macro avg   0.642430  0.642417  0.642408      5512
weighted avg   0.642430  0.642417  0.642408      5512
 samples avg   0.642417  0.642417  0.642417      5512

f1 score is  0.6424079655276096
```