Common Task 2. Deep Learning based Quark-Gluon Classification :

Datasets: https://cernbox.cern.ch/index.php/s/hqz8zE7oxyPjvsL

Description 125x125 matrices (three channel images) for two classes of particles quarks and gluons impinging on a calorimeter. For description of 1st dataset please refer to the link provided for the dataset.

Please use a Convolutional Neural Network (CNN) architecture of your choice to achieve the highest possible classification on this dataset (in your preferred choice offramework for example: Tensorflow/Keras or Pytorch).

Please provide a Jupyter notebook that shows your solution.

Downloading datasets:

```
!wget https://cernbox.cern.ch/index.php/s/hqz8zE7oxyPjvsL/download
!mkdir data
!7z x -o/content/data download
```

```
--2022-04-04 00:09:42--  https://cernbox.cern.ch/index.php/s/hqz8zE7oxyPjvsL/download
Resolving cernbox.cern.ch (cernbox.cern.ch)... 128.142.53.35, 128.142.53.28, 188.184.97
Connecting to cernbox.cern.ch (cernbox.cern.ch)|128.142.53.35|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: unspecified [application/octet-stream]
Saving to: ‘download’

download                [        <=>        ] 690.93M  26.3MB/s    in 33s

2022-04-04 00:10:19 (21.2 MB/s) - ‘download’ saved [724495360]


7-Zip [64] 16.02 : Copyright (c) 1999-2016 Igor Pavlov : 2016-05-21
p7zip Version 16.02 (locale=en_US.UTF-8,Utf16=on,HugeFiles=on,64 bits,2 CPUs Intel(R) Xe

Scanning the drive for archives:
1 file, 724495360 bytes (691 MiB)

Extracting archive: download
--
Path = download
Type = tar
Physical Size = 724495360
Headers Size = 2560
Code Page = UTF-8

Everything is Ok

Files: 3
```

```
Size:       724492307
Compressed: 724495360
```

Setting up imports:

```
import numpy as np
import tensorflow as tf
import pandas as pd
import matplotlib.pyplot as plt
import gc
import os
from pyarrow.parquet import ParquetFile
import pyarrow as pa
import pyarrow.parquet as pq
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report
from sklearn.metrics import roc_auc_score, plot_roc_curve, auc, roc_curve
from itertools import cycle
!pip install fastparquet
```

```
Requirement already satisfied: fastparquet in /usr/local/lib/python3.7/dist-packages (0
Requirement already satisfied: cramjam>=2.3.0 in /usr/local/lib/python3.7/dist-packages
Requirement already satisfied: numpy>=1.18 in /usr/local/lib/python3.7/dist-packages (fr
Requirement already satisfied: fsspec in /usr/local/lib/python3.7/dist-packages (from fa
Requirement already satisfied: pandas>=1.1.0 in /usr/local/lib/python3.7/dist-packages (
Requirement already satisfied: python-dateutil>=2.7.3 in /usr/local/lib/python3.7/dist-p
Requirement already satisfied: pytz>=2017.3 in /usr/local/lib/python3.7/dist-packages (f
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.7/dist-packages (from
```

Double-click (or enter) to edit

```
files=os.listdir("/content/data")
print(files)
```

```
['QCDToGGQQ_IMGjet_RH1all_jet0_run2_n55494.test.snappy.parquet', 'QCDToGGQQ_IMGjet_RH1al
```

```
pf = ParquetFile('/content/data/QCDToGGQQ_IMGjet_RH1all_jet0_run0_n36272.test.snappy.parquet'
first_rows = next(pf.iter_batches(batch_size = 12000, columns=['X_jets', 'y']))
df = pa.Table.from_batches([first_rows]).to_pandas()
del first_rows
```

Retrieve details of the dataset and then split the data:

```
X_dataset = np.array(np.array(np.array(df['X_jets'].tolist()).tolist()).tolist())
y_dataset = df['y'].to_numpy()
print(X_dataset.shape, y_dataset.shape)
del df
```

```
(12000, 3, 125, 125) (12000,)
```

```
X_train, X_test, y_train, y_test = train_test_split(X_dataset, y_dataset, test_size = 0.2, ra
X_train = np.moveaxis(X_train, 1, -1)
X_test = np.moveaxis(X_test, 1, -1)
gc.collect()
del X_dataset, y_dataset
X_train.shape, y_train.shape, X_test.shape, y_test.shape
```

```
((9600, 125, 125, 3), (9600,), (2400, 125, 125, 3), (2400,))
```

Define the CNN model:

```
num_classes = 1
input_shape = (125, 125, 3)
model = tf.keras.applications.ResNet101V2(
          include_top=True,
          weights=None,
          input_shape=input_shape,
          classes=1,
          classifier_activation='sigmoid'
)
model.compile(optimizer=tf.keras.optimizers.Adam(learning_rate=0.001),loss='binary_crossentro
model.summary()
```

```
Model: "resnet101v2"
__________________________________________________________________________________________________
 Layer (type)                   Output Shape         Param #     Connected to
==================================================================================================
 input_1 (InputLayer)           [(None, 125, 125, 3  0           []
                                )]

 conv1_pad (ZeroPadding2D)      (None, 131, 131, 3)  0           ['input_1[0][0]']

 conv1_conv (Conv2D)            (None, 63, 63, 64)   9472        ['conv1_pad[0][0]']

 pool1_pad (ZeroPadding2D)      (None, 65, 65, 64)   0           ['conv1_conv[0][0]']

 pool1_pool (MaxPooling2D)      (None, 32, 32, 64)   0           ['pool1_pad[0][0]']

 conv2_block1_preact_bn (BatchN  (None, 32, 32, 64)  256         ['pool1_pool[0][0]']
 ormalization)

 conv2_block1_preact_relu (Acti  (None, 32, 32, 64)  0           ['conv2_block1_preac
 vation)
```

```
conv2_block1_1_conv (Conv2D)   (None, 32, 32, 64)   4096        ['conv2_block1_preac
                                                                 ]

conv2_block1_1_bn (BatchNormal  (None, 32, 32, 64)  256         ['conv2_block1_1_con
ization)

conv2_block1_1_relu (Activatio  (None, 32, 32, 64)  0           ['conv2_block1_1_bn[
n)

conv2_block1_2_pad (ZeroPaddin  (None, 34, 34, 64)  0           ['conv2_block1_1_rel
g2D)

conv2_block1_2_conv (Conv2D)   (None, 32, 32, 64)   36864       ['conv2_block1_2_pad

conv2_block1_2_bn (BatchNormal  (None, 32, 32, 64)  256         ['conv2_block1_2_con
ization)

conv2_block1_2_relu (Activatio  (None, 32, 32, 64)  0           ['conv2_block1_2_bn[
n)

conv2_block1_0_conv (Conv2D)   (None, 32, 32, 256)  16640       ['conv2_block1_preac
                                                                 ]

conv2_block1_3_conv (Conv2D)   (None, 32, 32, 256)  16640       ['conv2_block1_2_rel

conv2_block1_out (Add)         (None, 32, 32, 256)  0           ['conv2_block1_0_con
                                                                  'conv2_block1_3_con

conv2_block2_preact_bn (BatchN  (None, 32, 32, 256)  1024       ['conv2_block1_out[0
ormalization)

conv2_block2_preact_relu (Acti  (None, 32, 32, 256)  0          ['conv2_block2_preac
vation)

conv2_block2_1_conv (Conv2D)   (None, 32, 32, 64)   16384       ['conv2_block2_preac
```

Defining callback:

```
filepath="classifier_weights2-improvement-{epoch:02d}-{val_accuracy:.2f}.hdf5"
checkpoint1 = tf.keras.callbacks.ModelCheckpoint(filepath, monitor='val_accuracy', verbose=1,
callbacks_list = [checkpoint1]
```

Compiling and Fitting the model with training data:

```
history = model.fit(X_train,
                    y_train,
                    epochs=15,
                    callbacks=callbacks_list,
```

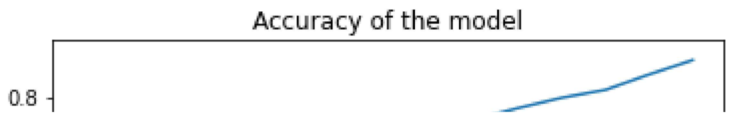
```
                    validation_data = (X_test, y_test)
    )
```

```
Epoch 1/15
300/300 [==============================] - ETA: 0s - loss: 0.6341 - accuracy: 0.6789
Epoch 1: val_accuracy improved from -inf to 0.49750, saving model to classifier_weigh
300/300 [==============================] - 95s 241ms/step - loss: 0.6341 - accuracy:
Epoch 2/15
300/300 [==============================] - ETA: 0s - loss: 0.5845 - accuracy: 0.7097
Epoch 2: val_accuracy did not improve from 0.49750
300/300 [==============================] - 68s 228ms/step - loss: 0.5845 - accuracy:
Epoch 3/15
300/300 [==============================] - ETA: 0s - loss: 0.5684 - accuracy: 0.7222
Epoch 3: val_accuracy improved from 0.49750 to 0.58250, saving model to classifier_we
300/300 [==============================] - 72s 242ms/step - loss: 0.5684 - accuracy:
Epoch 4/15
300/300 [==============================] - ETA: 0s - loss: 0.5594 - accuracy: 0.7255
Epoch 4: val_accuracy improved from 0.58250 to 0.60792, saving model to classifier_we
300/300 [==============================] - 73s 244ms/step - loss: 0.5594 - accuracy:
Epoch 5/15
300/300 [==============================] - ETA: 0s - loss: 0.5569 - accuracy: 0.7353
Epoch 5: val_accuracy improved from 0.60792 to 0.68292, saving model to classifier_we
300/300 [==============================] - 74s 246ms/step - loss: 0.5569 - accuracy:
Epoch 6/15
300/300 [==============================] - ETA: 0s - loss: 0.5429 - accuracy: 0.7377
Epoch 6: val_accuracy improved from 0.68292 to 0.69125, saving model to classifier_we
300/300 [==============================] - 74s 245ms/step - loss: 0.5429 - accuracy:
Epoch 7/15
300/300 [==============================] - ETA: 0s - loss: 0.5358 - accuracy: 0.7469
Epoch 7: val_accuracy improved from 0.69125 to 0.72083, saving model to classifier_we
300/300 [==============================] - 74s 246ms/step - loss: 0.5358 - accuracy:
Epoch 8/15
300/300 [==============================] - ETA: 0s - loss: 0.5240 - accuracy: 0.7522
Epoch 8: val_accuracy improved from 0.72083 to 0.72250, saving model to classifier_we
300/300 [==============================] - 74s 246ms/step - loss: 0.5240 - accuracy:
Epoch 9/15
300/300 [==============================] - ETA: 0s - loss: 0.5079 - accuracy: 0.7625
Epoch 9: val_accuracy did not improve from 0.72250
300/300 [==============================] - 71s 238ms/step - loss: 0.5079 - accuracy:
Epoch 10/15
300/300 [==============================] - ETA: 0s - loss: 0.4974 - accuracy: 0.7736
Epoch 10: val_accuracy did not improve from 0.72250
300/300 [==============================] - 71s 238ms/step - loss: 0.4974 - accuracy:
Epoch 11/15
300/300 [==============================] - ETA: 0s - loss: 0.4726 - accuracy: 0.7869
Epoch 11: val_accuracy did not improve from 0.72250
300/300 [==============================] - 71s 238ms/step - loss: 0.4726 - accuracy:
Epoch 12/15
300/300 [==============================] - ETA: 0s - loss: 0.4535 - accuracy: 0.7989
Epoch 12: val_accuracy did not improve from 0.72250
300/300 [==============================] - 71s 237ms/step - loss: 0.4535 - accuracy:
Epoch 13/15
300/300 [==============================] - ETA: 0s - loss: 0.4324 - accuracy: 0.8080
Epoch 13: val_accuracy did not improve from 0.72250
300/300 [==============================] - 71s 238ms/step - loss: 0.4324 - accuracy:
Epoch 14/15
```

```
300/300 [==============================] - ETA: 0s - loss: 0.4042 - accuracy: 0.8259
Epoch 14: val_accuracy did not improve from 0.72250
300/300 [==============================] - 71s 237ms/step - loss: 0.4042 - accuracy:
Epoch 15/15
```

Plotting the results:

```
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Accuracy of the model')
plt.ylabel('Accuracy')
plt.xlabel('Epochs')
plt.legend(['Train', 'Test'], loc='lower right')
plt.show()

plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Loss metrics of the model')
plt.ylabel('Loss')
plt.xlabel('Epochs')
plt.legend(['train', 'test'], loc='upper right')
plt.show()
```



Checking the performance of the model on training data and predicitons:

```
best_epoch=np.argmax(history.history['val_accuracy'])
best_acc=np.max(history.history['val_accuracy'])
model.load_weights('/content/classifier_weights2-improvement-08-0.72.hdf5')
```

Classification Report and ROC AUC score on test data:

```
predictions = model.predict(X_test)
bin =[0 if p<0.5 else 1 for p in predictions]
print(classification_report(y_test,bin))
```

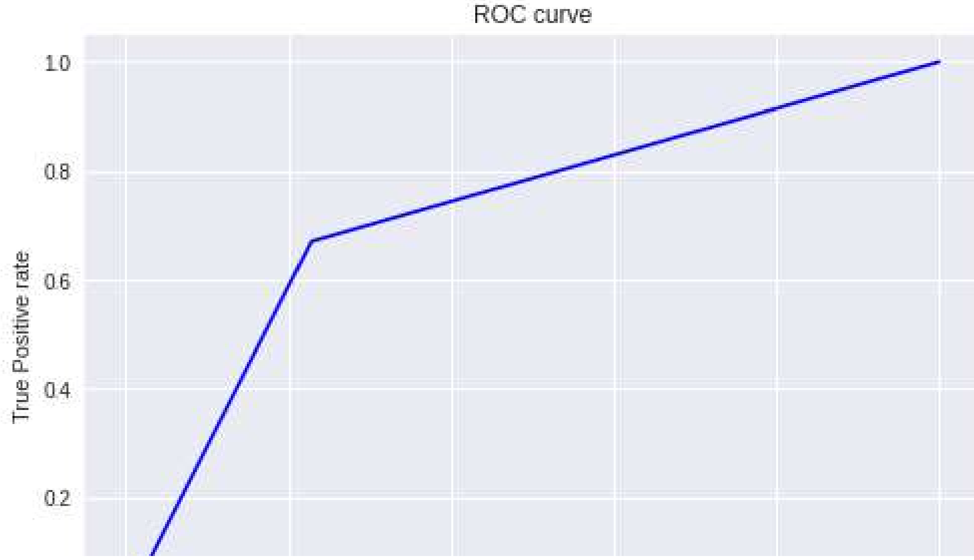
```
              precision    recall  f1-score   support

         0.0       0.71      0.77      0.74      1231
         1.0       0.74      0.67      0.70      1169

    accuracy                           0.72      2400
   macro avg       0.72      0.72      0.72      2400
weighted avg       0.72      0.72      0.72      2400
```

```
print("ROC AUC:")
roc_auc_score(y_test, bin)
```

```
ROC AUC:
0.7211944916016868
```

```
fpr, tpr, thresh = roc_curve(y_test, bin, pos_label=1)
plt.style.use('seaborn')
plt.plot(fpr, tpr,color='blue')
plt.title('ROC curve')
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive rate')
plt.legend(loc='best')
plt.savefig('ROC',dpi=300)
plt.show();
```

```
No handles with labels found to put in legend.
```



References:

- Quark-Gluon Jet Discrimination Using Convolutional Neural Networks
- Using Deep Learning to Discriminate Between Quark and Gluon Jets
- Discriminating quark/gluon jets with deep learning
- End-to-end jet classification of quarks and gluons with the CMS Open Data