+ Code + Text

Graph Neural Networks for Particle Momentum Estimation in the CMS Trigger System Task-1

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

```
Mounted at /content/drive
```

Installing Requirements

```
!pip install pyg_lib torch_scatter torch_sparse -f https://data.pyg.org/whl/torc
!pip install torch-geometric
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/cola
Looking in links: https://data.pyg.org/whl/torch-1.13.1+cu116.html
Collecting pyg_lib
  Downloading https://data.pyg.org/whl/torch-1.13.0%2Bcu116/pyg_lib-0.1.0%2
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.9/1.9 MB 13.1 MB/s eta 0:00
Collecting torch_scatter
  Downloading https://data.pyg.org/whl/torch-1.13.0%2Bcu116/torch_scatter-2
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 9.4/9.4 MB 82.6 MB/s eta 0:00
Collecting torch_sparse
  Downloading https://data.pyg.org/whl/torch-1.13.0%2Bcu116/torch_sparse-0.
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 4.5/4.5 MB 90.2 MB/s eta 0:00
Requirement already satisfied: scipy in /usr/local/lib/python3.8/dist-packa
Requirement already satisfied: numpy<1.27.0,>=1.19.5 in /usr/local/lib/pyth
Installing collected packages: torch_scatter, pyg_lib, torch_sparse
Successfully installed pyg_lib-0.1.0+pt113cu116 torch_scatter-2.1.0+pt113cu
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/cola
Collecting torch-geometric
  Downloading torch_geometric-2.2.0.tar.gz (564 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 565.0/565.0 KB 9.7 MB/s eta 0:
  Preparing metadata (setup.py) ... done
Requirement already satisfied: tqdm in /usr/local/lib/python3.8/dist-packag
Requirement already satisfied: numpy in /usr/local/lib/python3.8/dist-packa
Requirement already satisfied: scipy in /usr/local/lib/python3.8/dist-packa
Requirement already satisfied: jinja2 in /usr/local/lib/python3.8/dist-pack
Requirement already satisfied: requests in /usr/local/lib/python3.8/dist-pa
Requirement already satisfied: pyparsing in /usr/local/lib/python3.8/dist-p
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.8/dis
Collecting psutil>=5.8.0
  Downloading psutil-5.9.4-cp36-abi3-manylinux_2_12_x86_64.manylinux2010_x8
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 280.2/280.2 KB 31.8 MB/s eta 0:
Requirement already satisfied: MarkupSafe>=2.0 in /usr/local/lib/python3.8/
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.8/dis
Requirement already satisfied: chardet<5,>=3.0.2 in /usr/local/lib/python3.
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /usr/local/lib/pyth
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.8/di
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/pytho
Building wheels for collected packages: torch-geometric
  Building wheel for torch-geometric (setup.py) ... done
  Created wheel for torch-geometric: filename=torch_geometric-2.2.0-py3-non
  Stored in directory: /root/.cache/pip/wheels/59/a3/20/198928106d3169865ae
Successfully built torch-geometric
Installing collected packages: psutil, torch-geometric
  Attempting uninstall: psutil
    Found existing installation: psutil 5.4.8
    Uninstalling psutil-5.4.8:
      Successfully uninstalled psutil-5.4.8
Successfully installed psutil-5.9.4 torch-geometric-2.2.0
```

Importing Libraries

```
import random
import numpy as np
import tensorflow as tf
import h5py
from tqdm import tqdm

from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score, roc_curve, auc, f1_score
import matplotlib.pyplot as plt


# Modules for keras model

from keras.models import Model
from keras.layers import Input, Conv2D, BatchNormalization, Activation, Add, Max
from keras.callbacks import LearningRateScheduler


# Modules for pytorch model

import torch
import torch.nn as nn
import torch.nn.functional as F
from torch.utils.data import Dataset, DataLoader
from torchvision import transforms
import torch.optim as optim
from torch.optim.lr_scheduler import StepLR


seed = 3
random.seed(seed)
np.random.seed(seed)
tf.random.set_seed(seed)
```

Reading the dataset

```
with h5py.File('/content/drive/MyDrive/GSOC/task1/SingleElectronPt50_IMGCROPS_n2
  X_electron = f.get('X')[:]
  y_electron = f.get('y')[:]

with h5py.File('/content/drive/MyDrive/GSOC/task1/SinglePhotonPt50_IMGCROPS_n249
  X_photon = f.get('X')[:]
  y_photon = f.get('y')[:]
```

```
print(X_electron.shape, y_electron.shape)
print(X_photon.shape, y_photon.shape)
```

```
(249000, 32, 32, 2) (249000,)
(249000, 32, 32, 2) (249000,)
```

```
X = np.concatenate((X_electron, X_photon), axis = 0)
print(X.shape)
```

```
(498000, 32, 32, 2)
```

```
y = np.concatenate((y_electron, y_photon), axis = 0)
y = y.reshape(-1,1)
print(y.shape)
```

```
(498000, 1)
```

```
del(X_photon, X_electron, y_electron, y_photon)
```

```
# Using only the energy channel for classification

X = X.T[0].T
X.reshape(-1,32,32,1)
X.shape
```

```
(498000, 32, 32)
```

Train-Val-Test split for the dataset

Default ratio is set as 80/10/10 train-val-test

```
X_train, X_rem, Y_train, Y_rem = train_test_split(X, y, test_size = 0.20, strati
                                                  shuffle = True, random_state =
del(X, y)

X_val, X_test, Y_val, Y_test = train_test_split(X_rem, Y_rem, test_size = 0.50,
                                                shuffle = True, random_state = 1
del(X_rem, Y_rem)
```

```
print(X_train.shape, X_test.shape, X_val.shape)
```

```
(398400, 32, 32) (49800, 32, 32) (49800, 32, 32)
```

```
np.unique(Y_train, return_counts = True)
```

```
(array([0., 1.], dtype=float32), array([199200, 199200]))
```

Model - Keras CNN

Built ResNet-15 architecture after going through the paper *'End-to-End Physics Event Classification with CMS Open Data'*.

Removed BatchNorm and replaced MeanPooling with MaxPooling.

```
def residual_block(x, filters, strides=(1, 1), kernel_size=(3, 3)):
    """Residual block for ResNet"""

    shortcut = x
    x = Conv2D(filters, kernel_size=kernel_size, strides=strides, padding='same'
    x = Activation('relu')(x)

    x = Conv2D(filters, kernel_size=kernel_size, strides=(1, 1), padding='same')

    if strides != (1, 1) or shortcut.shape[-1] != filters:
        shortcut = Conv2D(filters, kernel_size=(1, 1), strides=strides, padding=

    x = Add()([shortcut, x])
    x = Activation('relu')(x)

    return x

def resnet15(input_shape, num_classes):
    """ResNet15 architecture"""

    input_tensor = Input(shape=input_shape)

    x = Conv2D(16, kernel_size=(3, 3), strides=(1, 1), padding='same')(input_ten
    x = Activation('relu')(x)
    x = MaxPooling2D(pool_size=(2, 2), strides=(2, 2), padding='same')(x)

    for i in range(3):
        filters = 16 * (2 ** i)
        strides = (2, 2) if i != 0 else (1, 1)
        x = residual_block(x, filters=filters, strides=strides)

    x = Flatten()(x)
    x = Dense(256, activation='relu')(x)
    x = Dense(64, activation='relu')(x)
    x = Dense(num_classes, activation='sigmoid')(x)

    model = Model(inputs=input_tensor, outputs=x)
    return model


model_keras = resnet15((32,32,1), 1)
model_keras.summary()
```

```
Model: "model_2"
________________________________________________________________________________
 Layer (type)                   Output Shape         Param #     Connected
================================================================================
 input_3 (InputLayer)           [(None, 32, 32, 1)]  0           []
```

```
 conv2d_18 (Conv2D)             (None, 32, 32, 16)   160         ['input_3[

 activation_14 (Activation)     (None, 32, 32, 16)   0           ['conv2d_1

 max_pooling2d_2 (MaxPooling2D)  (None, 16, 16, 16)  0           ['activati

 conv2d_19 (Conv2D)             (None, 16, 16, 16)   2320        ['max_pool

 activation_15 (Activation)     (None, 16, 16, 16)   0           ['conv2d_1

 conv2d_20 (Conv2D)             (None, 16, 16, 16)   2320        ['activati

 add_6 (Add)                    (None, 16, 16, 16)   0           ['max_pool
                                                                  'conv2d_2

 activation_16 (Activation)     (None, 16, 16, 16)   0           ['add_6[0]

 conv2d_21 (Conv2D)             (None, 8, 8, 32)     4640        ['activati

 activation_17 (Activation)     (None, 8, 8, 32)     0           ['conv2d_2

 conv2d_23 (Conv2D)             (None, 8, 8, 32)     544         ['activati

 conv2d_22 (Conv2D)             (None, 8, 8, 32)     9248        ['activati

 add_7 (Add)                    (None, 8, 8, 32)     0           ['conv2d_2
                                                                  'conv2d_2

 activation_18 (Activation)     (None, 8, 8, 32)     0           ['add_7[0]

 conv2d_24 (Conv2D)             (None, 4, 4, 64)     18496       ['activati

 activation_19 (Activation)     (None, 4, 4, 64)     0           ['conv2d_2

 conv2d_26 (Conv2D)             (None, 4, 4, 64)     2112        ['activati

 conv2d_25 (Conv2D)             (None, 4, 4, 64)     36928       ['activati

 add_8 (Add)                    (None, 4, 4, 64)     0           ['conv2d_2
                                                                  'conv2d_2

 activation_20 (Activation)     (None, 4, 4, 64)     0           ['add_8[0]

 flatten_2 (Flatten)            (None, 1024)         0           ['activati

 dense_6 (Dense)                (None, 256)          262400      ['flatten_

 dense_7 (Dense)                (None, 64)           16448       ['dense_6[

 dense_8 (Dense)                (None, 1)            65          ['dense_7[

==========================================================================
Total params: 355,681
Trainable params: 355,681
```

Adding a learning rate scheduler which makes the learning rate half after every 10 epochs.

```
def scheduler(epoch, lr):
    if epoch % 10 == 0 and epoch != 0:
        lr = lr / 2
    return lr

lr_schedule = LearningRateScheduler(scheduler)
```

Training the Model

Set the optimiser and loss function. Evalutation metric is the ROC-AUC.

```
optimizer = tf.keras.optimizers.Adam(learning_rate = 5e-4)
model_keras.compile(loss='binary_crossentropy', optimizer=optimizer, metrics=[tf
```

```
# Chose the epochs and batch_size according to the paper

model_keras.fit(X_train, Y_train, validation_data = (X_val, Y_val), epochs=60, b
```

```
1107/1107 [==============================] - 9s 8ms/step - loss: 0.5289 - a
Epoch 29/60
1107/1107 [==============================] - 9s 8ms/step - loss: 0.5283 - a
Epoch 30/60
1107/1107 [==============================] - 9s 8ms/step - loss: 0.5282 - a
Epoch 31/60
1107/1107 [==============================] - 9s 8ms/step - loss: 0.5258 - a
Epoch 32/60
1107/1107 [==============================] - 9s 8ms/step - loss: 0.5253 - a
Epoch 33/60
1107/1107 [==============================] - 9s 8ms/step - loss: 0.5251 - a
Epoch 34/60
1107/1107 [==============================] - 9s 8ms/step - loss: 0.5247 - a
Epoch 35/60
1107/1107 [==============================] - 9s 8ms/step - loss: 0.5245 - a
Epoch 36/60
1107/1107 [==============================] - 9s 8ms/step - loss: 0.5241 - a
Epoch 37/60
1107/1107 [==============================] - 9s 8ms/step - loss: 0.5239 - a
Epoch 38/60
1107/1107 [==============================] - 9s 8ms/step - loss: 0.5238 - a
Epoch 39/60
1107/1107 [==============================] - 9s 8ms/step - loss: 0.5235 - a
Epoch 40/60
1107/1107 [==============================] - 9s 8ms/step - loss: 0.5234 - a
Epoch 41/60
```

```
Epoch 41/60
1107/1107 [==============================] - 9s 8ms/step - loss: 0.5218 - a
Epoch 42/60
1107/1107 [==============================] - 9s 8ms/step - loss: 0.5215 - a
Epoch 43/60
1107/1107 [==============================] - 9s 8ms/step - loss: 0.5214 - a
Epoch 44/60
1107/1107 [==============================] - 9s 8ms/step - loss: 0.5212 - a
Epoch 45/60
1107/1107 [==============================] - 9s 8ms/step - loss: 0.5211 - a
Epoch 46/60
1107/1107 [==============================] - 9s 8ms/step - loss: 0.5210 - a
Epoch 47/60
1107/1107 [==============================] - 9s 8ms/step - loss: 0.5209 - a
Epoch 48/60
1107/1107 [==============================] - 9s 8ms/step - loss: 0.5206 - a
Epoch 49/60
1107/1107 [==============================] - 9s 8ms/step - loss: 0.5205 - a
Epoch 50/60
1107/1107 [==============================] - 9s 8ms/step - loss: 0.5204 - a
Epoch 51/60
1107/1107 [==============================] - 9s 8ms/step - loss: 0.5195 - a
Epoch 52/60
1107/1107 [==============================] - 9s 8ms/step - loss: 0.5194 - a
Epoch 53/60
1107/1107 [==============================] - 9s 8ms/step - loss: 0.5192 - a
Epoch 54/60
1107/1107 [==============================] - 9s 8ms/step - loss: 0.5192 - a
Epoch 55/60
1107/1107 [==============================] - 9s 8ms/step - loss: 0.5191 - a
Epoch 56/60
1107/1107 [==============================] - 9s 8ms/step - loss: 0.5190 - a
Epoch 57/60
1107/1107 [==============================] - 9s 8ms/step - loss: 0.5189 - a
Epoch 58/60
```

Testing the Model

```
Y_pred = model_keras.predict(X_test)

Y_probas = Y_pred.ravel()
Y_abs = (Y_pred>0.5)
```

```
1557/1557 [==============================] - 4s 2ms/step
```

Calculating few metrics

```
accuracy_keras = accuracy_score(Y_test, Y_abs)
f1_keras = accuracy_score(Y_test, Y_abs)
fpr_keras, tpr_keras, thresholds_keras = roc_curve(Y_test, Y_probas)
auc_keras = auc(fpr_keras, tpr_keras)


print('RESULTS\n')

print(f'Testing Accuracy : {accuracy_keras:.3f}')
print(f'F1 score : {f1_keras:.3f}')
print(f'ROC-AUC : {auc_keras:.4f}')
```

```
RESULTS

Testing Accuracy : 0.737
F1 score : 0.737
ROC-AUC : 0.8078
```

Model - PyTorch CNN

Creating Custom Dataset

```
class MyDataset(Dataset):

    """
    Custom dataset for Image dataset
    """

    def __init__(self, X, y):

      X = torch.from_numpy(X)
      X = X.unsqueeze(-1)
      self.X = X.permute(0, 3, 1, 2)
      self.y = torch.from_numpy(y).squeeze().long()

    def __len__(self):
        return len(self.y)

    def __getitem__(self, idx):
      return self.X[idx], self.y[idx]
```

```
train_dataset = MyDataset(X_train, Y_train)
val_dataset = MyDataset(X_val, Y_val)
test_dataset = MyDataset(X_test, Y_test)
```

Data Loaders

```
def get_data_loaders(train_dataset, val_dataset, test_dataset, batch_size=32):

    """
    Function to create the DataLoaders for train-val-test data.
    Can specify batch size. Default value is set to 32.
    """

    # Shuffle=True for training data to get diversity in batches at each trainin
    train_loader = DataLoader(train_dataset, batch_size=batch_size, shuffle=True
    val_loader = DataLoader(val_dataset, batch_size=batch_size, shuffle=False)
    test_loader = DataLoader(test_dataset, batch_size=batch_size, shuffle=False)

    return train_loader, val_loader, test_loader

train_loader, val_loader, test_loader = get_data_loaders(train_dataset, val_data
```

Set Device

```
def get_device():
    return torch.device('cuda' if torch.cuda.is_available() else 'cpu')


device = get_device()
print(device)
```

```
cuda
```

Model Architecture

```
class ResidualBlock(nn.Module):

    """Residual block for ResNet"""

    def __init__(self, in_channels, out_channels, stride=(1, 1), kernel_size=(3,
        super(ResidualBlock, self).__init__()
        self.shortcut = nn.Identity()
```

```
        if stride != (1, 1) or in_channels != out_channels:
            self.shortcut = nn.Conv2d(in_channels, out_channels, kernel_size=(1,

        self.conv1 = nn.Conv2d(in_channels, out_channels, kernel_size=kernel_siz
        self.conv2 = nn.Conv2d(out_channels, out_channels, kernel_size=kernel_si

    def forward(self, x):

        shortcut = self.shortcut(x)

        x = F.relu(self.conv1(x))
        x = self.conv2(x)

        x += shortcut
        x = F.relu(x)

        return x

class ResNet15(nn.Module):

    """ResNet15 architecture"""

    def __init__(self, in_channels = 1, num_classes = 2):
        super(ResNet15, self).__init__()

        self.conv1 = nn.Conv2d(in_channels, 16, kernel_size=(3, 3), stride=(1, 1
        self.pool1 = nn.MaxPool2d(kernel_size=(2, 2), stride=(2, 2))

        self.layer1 = nn.Sequential(
            ResidualBlock(16, 16),
            ResidualBlock(16, 32, stride=(2, 2)),
            ResidualBlock(32, 64, stride=(2, 2))
        )

        self.fc1 = nn.Linear(64 * 4 * 4, 256)
        self.fc2 = nn.Linear(256, 64)
        self.fc3 = nn.Linear(64, num_classes)

    def forward(self, x):

        x = F.relu(self.conv1(x))
        x = self.pool1(x)
        x = self.layer1(x)

        x = x.view(x.size(0), -1)

        x = F.relu(self.fc1(x))
```

```
        x = F.relu(self.fc2(x))

        x = self.fc3(x)

        return x
```

Setup -

1. Model Creation
2. Setting Adam optimizer with learning rate scheduler
3. Defining CrossEntropyLoss function

```
# Model
model_pytorch = ResNet15()
model_pytorch = model_pytorch.to(device)

# Optimizer and Learning Rate Scheduler

optimizer = optim.Adam(model_pytorch.parameters(), lr=5e-4)
scheduler = StepLR(optimizer, step_size=10, gamma=0.5)

# Loss Function

criterion = torch.nn.CrossEntropyLoss()
```

```
print(model_pytorch)
```

```
ResNet15(
  (conv1): Conv2d(1, 16, kernel_size=(3, 3), stride=(1, 1), padding=(1, 1),
  (pool1): MaxPool2d(kernel_size=(2, 2), stride=(2, 2), padding=0, dilation
  (layer1): Sequential(
    (0): ResidualBlock(
      (shortcut): Identity()
      (conv1): Conv2d(16, 16, kernel_size=(3, 3), stride=(1, 1), padding=(1
      (conv2): Conv2d(16, 16, kernel_size=(3, 3), stride=(1, 1), padding=(1
    )
    (1): ResidualBlock(
      (shortcut): Conv2d(16, 32, kernel_size=(1, 1), stride=(2, 2), bias=Fa
      (conv1): Conv2d(16, 32, kernel_size=(3, 3), stride=(2, 2), padding=(1
      (conv2): Conv2d(32, 32, kernel_size=(3, 3), stride=(1, 1), padding=(1
    )
    (2): ResidualBlock(
      (shortcut): Conv2d(32, 64, kernel_size=(1, 1), stride=(2, 2), bias=Fa
      (conv1): Conv2d(32, 64, kernel_size=(3, 3), stride=(2, 2), padding=(1
      (conv2): Conv2d(64, 64, kernel_size=(3, 3), stride=(1, 1), padding=(1
    )
  )
  (fc1): Linear(in_features=1024, out_features=256, bias=True)
  (fc2): Linear(in_features=256, out_features=64, bias=True)
  (fc3): Linear(in_features=64, out_features=2, bias=True)
)
```

Training

```
def train(model, device, loader, optimizer, criterion, scheduler):

    model.train()
    for data in tqdm(loader):  # Iterate in batches over the training dataset.

        X, y = data
        X = X.to(device)
        y = y.to(device)

        out = model(X)  # Perform a single forward pass.

        loss = criterion(out, y)  # Compute the loss.
        loss.backward()  # Derive gradients.
        optimizer.step()  # Update parameters based on gradients.
        optimizer.zero_grad()  # Clear gradients.

    scheduler.step()

    lr = optimizer.param_groups[0]['lr']
```

```
    print(f"learning rate: {lr:.6f}")

    return model


def evaluate(model, device, loader):

    model.eval()

    y_true = []
    y_probas = []

    with torch.no_grad():

        for data in tqdm(loader):

            X, y = data

            X = X.to(device)
            y = y.to(device)

            out = model(X)

            y_true += y.cpu().numpy().tolist()
            y_probas += out[:, 1].cpu().numpy().tolist()  # probability of class

    # Calculating few metrics

        fpr, tpr, thresholds = roc_curve(y_true, y_probas)
        roc_auc = auc(fpr, tpr)

        print(f'Val AUC : {roc_auc:.3f}\n')



# Training Loop

epochs = 60

for epoch in range(epochs):

  print(f'Epoch : {epoch+1} \n')

  model_pytorch = train(model_pytorch, device, train_loader, optimizer, criterio
  evaluate(model_pytorch, device, val_loader)
```

```
Epoch : 18

100%|██████████| 1107/1107 [00:10<00:00, 102.50it/s]
```

```
learning rate: 0.000250
100%|██████████| 139/139 [00:00<00:00, 204.26it/s]
Val AUC : 0.806

Epoch : 19

100%|██████████| 1107/1107 [00:10<00:00, 101.63it/s]
learning rate: 0.000250
100%|██████████| 139/139 [00:00<00:00, 205.32it/s]
Val AUC : 0.806

Epoch : 20

100%|██████████| 1107/1107 [00:10<00:00, 100.71it/s]
learning rate: 0.000125
100%|██████████| 139/139 [00:00<00:00, 206.76it/s]
Val AUC : 0.807

Epoch : 21

100%|██████████| 1107/1107 [00:10<00:00, 101.02it/s]
learning rate: 0.000125
100%|██████████| 139/139 [00:00<00:00, 207.26it/s]
Val AUC : 0.808

Epoch : 22

100%|██████████| 1107/1107 [00:10<00:00, 102.89it/s]
learning rate: 0.000125
100%|██████████| 139/139 [00:00<00:00, 154.47it/s]
Val AUC : 0.808

Epoch : 23

100%|██████████| 1107/1107 [00:10<00:00, 102.35it/s]
learning rate: 0.000125
100%|██████████| 139/139 [00:00<00:00, 204.28it/s]
Val AUC : 0.808

Epoch : 24

100%|██████████| 1107/1107 [00:11<00:00, 100.62it/s]
learning rate: 0.000125
100%|██████████| 139/139 [00:00<00:00, 205.96it/s]
Val AUC : 0.808

Epoch : 25

100%|██████████| 1107/1107 [00:11<00:00, 99.88it/s]
learning rate: 0.000125
100%|██████████| 139/139 [00:00<00:00, 200.85it/s]
Val AUC : 0.807

Epoch : 26
```

```
100%|██████████| 1107/1107 [00:10<00:00, 101.02it/s]
learning rate: 0.000125
100%|██████████| 139/139 [00:00<00:00, 206.90it/s]
```

Testing

```
def test(model, device, loader):

    model.eval()
    y_true = []
    y_probas = []
    y_pred = []

    with torch.no_grad():

        for data in tqdm(loader):

            X, y = data

            X = X.to(device)
            y = y.to(device)

            out = model(X)

            y_true += y.cpu().numpy().tolist()
            y_pred += out.argmax(dim=1).cpu().numpy().tolist()  # absoulte predi
            y_probas += out[:, 1].cpu().numpy().tolist()  # probability of class

    # Calculating few metrics

    acc = accuracy_score(y_true, y_pred)
    f1 = f1_score(y_true, y_pred)
    fpr, tpr, thresholds = roc_curve(y_true, y_probas)
    roc_auc = auc(fpr, tpr)

    print('\nResults\n')
    print(f'Testing Accuracy {acc:.3f}')
    print(f'F1 score: {f1:.3f}')
    print(f'ROC-AUC: {roc_auc:.4f}\n')

    return acc, f1, fpr, tpr, roc_auc
```

```
acc_pytorch, f1_pytorch, fpr_pytorch, tpr_pytorch, auc_pytorch = test(model_pyto
```

```
100%|██████████| 139/139 [00:00<00:00, 200.01it/s]

Results

Testing Accuracy 0.736
F1 score: 0.738
ROC-AUC: 0.8058
```

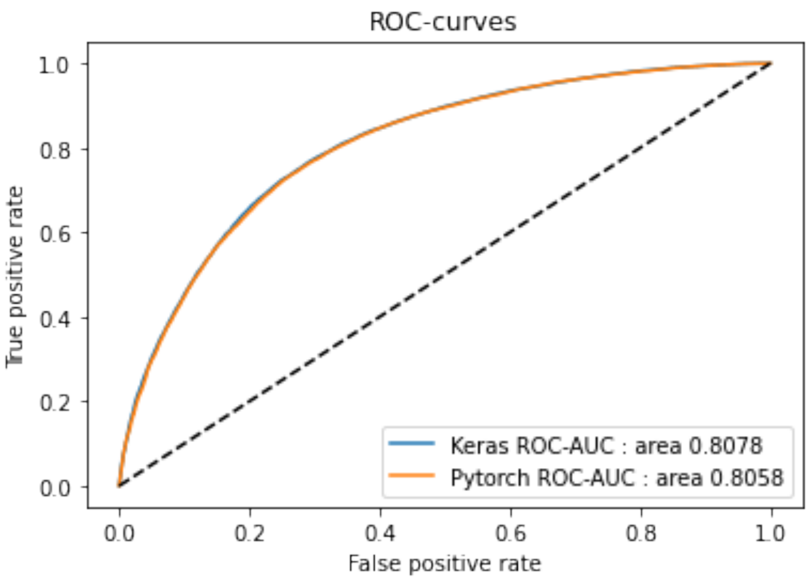
Results

1. Plots

```
plt.plot(fpr_keras, tpr_keras, label=f'Keras ROC-AUC : area {auc_keras:.4f}')
plt.plot(fpr_pytorch, tpr_pytorch, label=f'Pytorch ROC-AUC : area {auc_pytorch:.

plt.plot([0, 1], [0, 1], 'k--')
plt.xlabel('False positive rate')
plt.ylabel('True positive rate')
plt.title('ROC-curves')

plt.legend()
plt.savefig('roc-auc.png')
plt.show()
```



2. Metrics

Keras Implementation

```
print(f'Testing Accuracy {accuracy_keras:.3f}')
print(f'F1 score: {f1_keras:.3f}')
print(f'ROC-AUC: {auc_keras:.4f}')
```

```
Testing Accuracy 0.737
F1 score: 0.737
ROC-AUC: 0.8078
```

Pytorch Implementation

```
print(f'Testing Accuracy {acc_pytorch:.3f}')
print(f'F1 score: {f1_pytorch:.3f}')
print(f'ROC-AUC: {auc_pytorch:.4f}')
```

```
Testing Accuracy 0.736
F1 score: 0.738
ROC-AUC: 0.8058
```

0s completed at 12:51 AM