Name : Devdeep Shetranjiwala
Email ID : devdeep0702@gmail.com

Task 1. Electron/photon classification

Datasets:</br> https://cernbox.cern.ch/index.php/s/AtBT8y4MiQYFcgc (photons) </br>
https://cernbox.cern.ch/index.php/s/FbXw3V4XNyYB3oA (electrons) </br>

> **Description: </br> 32x32 matrices (two channels - hit energy and time) for two classes of particles electrons and photons impinging on a calorimeter Please use a deep learning method of your choice to achieve the highest possible classification on this dataset.**
>
> **In this task, we will use deep learning to classify two classes of particles: electrons and photons impinging on a calorimeter. We will use two datasets, one for photons and one for electrons, which contains 32x32 matrices (two channels - hit energy and time) for each particle.</br> We will use deep learning framework PyTorch. Our goal is to achieve the highest possible classification accuracy on this dataset with a ROC AUC score of at least 0.80. </br> First, we will load the data and preprocess it.**
> **Data Preprocessing :**
> **We will load the datasets for photons and electrons and preprocess them. We will convert the data into numpy arrays and normalize them by dividing each pixel value by the maximum pixel value.**

In [ ]:

```
import numpy as np
import torch
import torch.nn as nn
import torch.optim as optim
import h5py

from torchvision.transforms import CenterCrop
from torch.utils.data import Dataset, DataLoader, random_split
from sklearn.metrics import roc_auc_score

import warnings
warnings.filterwarnings("ignore")
```

In [ ]:

```
# Setting Seed
torch.manual_seed(42)
torch.cuda.manual_seed_all(42)
torch.cuda.manual_seed(42)
torch.backends.cudnn.benchmark = False
torch.backends.cudnn.deterministic = True
```

In [ ]:

```
# Setting device to GPU
device = "cuda" if torch.cuda.is_available() else "cpu"
device
```

Out[ ]:

```
'cuda'
```

In [ ]:

```
#Getting data
import requests
url = 'https://cernbox.cern.ch/remote.php/dav/public-files/AtBT8y4MiQYFcgc/SinglePhotonPt
50_IMGCROPS_n249k_RHv1.hdf5'
```

```
r = requests.get(url, allow_redirects=True)
open('photons.hdf5', 'wb').write(r.content)
url = 'https://cernbox.cern.ch/remote.php/dav/public-files/FbXw3V4XNyYB3oA/SingleElectron
Pt50_IMGCROPS_n249k_RHv1.hdf5'
r = requests.get(url, allow_redirects=True)
open('electrons.hdf5', 'wb').write(r.content)
```

In [ ]:

```
# Define the neural network
class Net (nn.Module):
    def __init__(self):
        super(Net, self).__init__()
        self.conv1 = nn.Conv2d(2, 32, kernel_size=1)
        self.bn1 = nn.BatchNorm2d(32)
        self.conv2 = nn.Conv2d(32, 32, kernel_size=3, padding=1)
        self.bn2 = nn.BatchNorm2d(32)
        self.pool1 = nn.MaxPool2d(kernel_size=2)
        self.conv3 = nn.Conv2d(32, 64, kernel_size=1)
        self.bn3 = nn.BatchNorm2d(64)
        self.conv4 = nn.Conv2d(64, 64, kernel_size=3, padding=1)
        self.bn4 = nn.BatchNorm2d(64)
        self.pool2 = nn.MaxPool2d(kernel_size=2)
        self.conv5 = nn.Conv2d(64, 128, kernel_size=1)
        self.bn5 = nn.BatchNorm2d(128)
        self.conv6 = nn.Conv2d(128, 128, kernel_size=3, padding=1)
        self.bn6 = nn.BatchNorm2d(128)
        self.pool3 = nn.MaxPool2d(kernel_size=2)
        self.fc1 = nn.Linear(128 * 4 * 4, 64)
        self.dropout = nn.Dropout(0.5)
        self.fc2 = nn.Linear(64, 1)

    def forward(self, x):
        x = x.permute(0, 3, 1, 2)
        x = self.conv1(x)
        x = self.bn1(x)
        x = torch.relu(x)
        x = self.conv2(x)
        x = self.bn2(x)
        x = torch.relu(x)
        x = torch.nn.functional.dropout(x, 0.2)
        x = self.pool1(x)
        x = self.conv3(x)
        x = self.bn3(x)
        x = torch.relu(x)
        x = self.conv4(x)
        x = self.bn4(x)
        x = torch.relu(x)
        x = torch.nn.functional.dropout(x, 0.2)
        x = self.pool2(x)
        x = self.conv5(x)
        x = self.bn5(x)
        x = torch.relu(x)
        x = self.conv6(x)
        x = self.bn6(x)
        x = torch.relu(x)
        x = torch.nn.functional.dropout(x, 0.2)
        x = self.pool3(x)
        x = torch.flatten(x, 1)
        x = self.fc1(x)
        x = torch.relu(x)
        x = self.dropout(x)
        x = self.fc2(x)
        x = torch.sigmoid(x)

        return x
```

In [ ]:

```
# Defining the Dataset class
class Dataset(Dataset):
```

```
    def __init__(self, electrons_data, photons_data):
        self.electrons_data = electrons_data
        self.photons_data = photons_data
        self.data_key = 'X'
        self.file_electron = h5py.File(self.electrons_data, 'r')
        self.data_electron = self.file_electron[self.data_key]
        self.file_photon = h5py.File(self.photons_data, 'r')
        self.data_photon = self.file_photon[self.data_key]
        self.data = np.concatenate((self.data_electron, self.data_photon), axis=0)
        self.labels = np.concatenate((np.ones(self.data_electron.shape[0]), np.zeros(sel
f.data_photon.shape[0])), axis=0)
        self.labels = np.expand_dims(self.labels, axis=1)
        # wanted to try to normalize data
        #self.data = (self.data - mean)/std


    def __len__(self):
        return self.data.shape[0]

    def __getitem__(self, idx):
        return torch.from_numpy(self.data[idx]), torch.from_numpy(self.labels[idx])

    def close(self):
        self.file_electrons.close()
        self.file_photon.close()
```

In [ ]:

```
# Making training, validation and test datasets
# train data : 80%,
# validation data : 10%,
# testing data : 10%

electrons_data = 'electrons.hdf5'
photons_data = 'photons.hdf5'
dataset = Dataset(electrons_data, photons_data)

# Split the data into training, testing sets and validating sets
train_size = np.int32(0.8 * len(dataset))
test_size = np.int32(0.1 * len(dataset))
val_size = len(dataset) - train_size - test_size

train_data, test_data, val_data = random_split(dataset, [train_size, test_size, val_size
])

# Create the data loaders
train_loader = DataLoader(train_data, batch_size=200, shuffle=True, num_workers=2)
test_loader = DataLoader(test_data, batch_size=200, shuffle=True, num_workers=2)
val_loader = DataLoader(val_data, batch_size=200, shuffle=True, num_workers=2)
```

In [ ]:

```
# Defining the loss function, optimizer
model = Net().to(device)
criterion = nn.BCELoss()
optimizer = optim.SGD(model.parameters(), lr=0.001, momentum=0.8)
```

In [ ]:

```
# Training

epochs = 200 # takes time but gives good result
min_val_loss = np.inf

train_losses = []
val_losses = []
train_aucs = []
val_aucs = []
counter = 0

for e in range(epochs):
```

```
    train_loss = 0.0
    train_auc = 0.0
    i = 0
    for data in train_loader: # Training Loop
        inputs, labels = data
        inputs, labels = inputs.to(device), labels.to(device)
        optimizer.zero_grad()
        #print(inputs.shape)
        outputs = model(inputs)
        labels , outputs = labels.type(torch.FloatTensor),outputs.type(torch.FloatTensor
)
        loss = criterion(outputs, labels)
        loss.backward()
        optimizer.step()
        train_loss += loss.item()
        with torch.no_grad():
            train_auc += roc_auc_score(labels.numpy(), outputs.numpy())

        if i % 100 == 99:    # save every 100 mini-batches
            train_losses.append(train_loss / 100)
            train_aucs.append(train_auc / 100)
            train_loss = 0.0
            train_auc = 0.0

        i += 1

    with torch.no_grad():
        val_loss = 0.0
        val_auc = 0.0
        model.eval()
        for data in val_loader: # Validation Loop
            inputs, labels = data
            inputs, labels = inputs.to(device), labels.to(device)
            outputs = model(inputs)
            labels , outputs = labels.type(torch.FloatTensor),outputs.type(torch.FloatTe
nsor)
            loss = criterion(outputs,labels)
            val_loss += loss.item()
            val_auc += roc_auc_score(labels.numpy(), outputs.numpy())

    print(f'Epoch {e+1} Val Loss: {val_loss / len(val_loader)} \t\t Val Accuracy: {val_au
c / len(val_loader)}')

    val_losses.append(val_loss / len(val_loader))
    val_aucs.append(val_auc / len(val_loader))

    # basic need for task - 1 AUC ROC SCORE
#     if (val_auc / len(val_loader)>=0.85) :
#         break

    if min_val_loss > val_loss:
        print(f'Validation Loss Decreased({min_val_loss:.6f}--->{val_loss:.6f}) \t Savin
g The Model')
        min_val_loss = val_loss
        counter = 0

        # Saving the model
        torch.save(model.state_dict(), 'saved_model.pth')

    else:
        # Early Stopping
        counter += 1
        if counter >= 20:
            print('Training Stopped.')
            break
```

```
Epoch 1 Val Loss: 0.6072167335265133    Val Accuracy: 0.7369851546367092
Validation Loss Decreased(inf--->151.196967)   Saving The Model
Epoch 2 Val Loss: 0.6081583854185051    Val Accuracy: 0.740356061553586
Epoch 3 Val Loss: 0.5914301632877335    Val Accuracy: 0.7503892535118144
Validation Loss Decreased(151.196967--->147.266111)   Saving The Model
Epoch 4 Val Loss: 0.600578661424568    Val Accuracy: 0.7381515964134102
```

```
Epoch 5 Val Loss: 0.5842341525727007    Val Accuracy: 0.7579935904893578
Validation Loss Decreased(147.266111--->145.474304)   Saving The Model
Epoch 6 Val Loss: 0.5934161859822561    Val Accuracy: 0.7548254557609606
Epoch 7 Val Loss: 0.5819701387221554    Val Accuracy: 0.7637875211259614
Validation Loss Decreased(145.474304--->144.910565)   Saving The Model
Epoch 8 Val Loss: 0.5812393273694448    Val Accuracy: 0.7645595698981348
Validation Loss Decreased(144.910565--->144.728593)   Saving The Model
Epoch 9 Val Loss: 0.5788110372531845    Val Accuracy: 0.7652007781786321
Validation Loss Decreased(144.728593--->144.123948)   Saving The Model
Epoch 10 Val Loss: 0.579700713416l8    Val Accuracy: 0.7666604812819116
Epoch 11 Val Loss: 0.5755691753333831    Val Accuracy: 0.7679217511111103
Validation Loss Decreased(144.123948--->143.316725)   Saving The Model
Epoch 12 Val Loss: 0.5708940789163353    Val Accuracy: 0.772955152918326
Validation Loss Decreased(143.316725--->142.152626)   Saving The Model
Epoch 13 Val Loss: 0.5771601701596655    Val Accuracy: 0.7721761164428127
Epoch 14 Val Loss: 0.5742615105876003    Val Accuracy: 0.7717848550599461
Epoch 15 Val Loss: 0.5729399172656507    Val Accuracy: 0.7720169070709292
Epoch 16 Val Loss: 0.5681727051016796    Val Accuracy: 0.7758303530510955
Validation Loss Decreased(142.152626--->141.475004)   Saving The Model
Epoch 17 Val Loss: 0.5692711361201412    Val Accuracy: 0.775288075208234
Epoch 18 Val Loss: 0.5632378049643643    Val Accuracy: 0.7802353137509261
Validation Loss Decreased(141.475004--->140.246213)   Saving The Model
Epoch 19 Val Loss: 0.5658256288273746    Val Accuracy: 0.7778272929789806
Epoch 20 Val Loss: 0.5722388421675287    Val Accuracy: 0.7759157862184276
Epoch 21 Val Loss: 0.5653194252506317    Val Accuracy: 0.7784003965847642
Epoch 22 Val Loss: 0.5635452829450968    Val Accuracy: 0.7800464385111475
Epoch 23 Val Loss: 0.5628212647265699    Val Accuracy: 0.7810200951661862
Validation Loss Decreased(140.246213--->140.142495)   Saving The Model
Epoch 24 Val Loss: 0.5665067813722006    Val Accuracy: 0.7772758021496893
Epoch 25 Val Loss: 0.5604754581030114    Val Accuracy: 0.7829288498297073
Validation Loss Decreased(140.142495--->139.558389)   Saving The Model
Epoch 26 Val Loss: 0.5662742034498468    Val Accuracy: 0.7811252167960143
Epoch 27 Val Loss: 0.5661287434608582    Val Accuracy: 0.776744856420648
Epoch 28 Val Loss: 0.563333258810771    Val Accuracy: 0.7808372050480276
Epoch 29 Val Loss: 0.5618171416612036    Val Accuracy: 0.7834433635004471
Epoch 30 Val Loss: 0.5601869692285377    Val Accuracy: 0.7843426015591506
Validation Loss Decreased(139.558389--->139.486555)   Saving The Model
Epoch 31 Val Loss: 0.5614525851715042    Val Accuracy: 0.7830695119366444
Epoch 32 Val Loss: 0.56464904343268    Val Accuracy: 0.7806838786327736
Epoch 33 Val Loss: 0.5560201278412678    Val Accuracy: 0.7871157355751278
Validation Loss Decreased(139.486555--->138.449012)   Saving The Model
Epoch 34 Val Loss: 0.5574087534084856    Val Accuracy: 0.7872656919887278
Epoch 35 Val Loss: 0.5598642514891414    Val Accuracy: 0.7851876514740206
Epoch 36 Val Loss: 0.5596227503445254    Val Accuracy: 0.7865980354471361
Epoch 37 Val Loss: 0.5594058689104027    Val Accuracy: 0.7849425328545718
Epoch 38 Val Loss: 0.5602379547783649    Val Accuracy: 0.7873709362934163
Epoch 39 Val Loss: 0.5606001039824812    Val Accuracy: 0.7842565084936277
Epoch 40 Val Loss: 0.5544772454533711    Val Accuracy: 0.7893949849190912
Validation Loss Decreased(138.449012--->138.064834)   Saving The Model
Epoch 41 Val Loss: 0.5615286855812532    Val Accuracy: 0.7857096780097315
Epoch 42 Val Loss: 0.5547835898686604    Val Accuracy: 0.7891461732074977
Epoch 43 Val Loss: 0.5579593055937664    Val Accuracy: 0.7875759980505621
Epoch 44 Val Loss: 0.5536967803196735    Val Accuracy: 0.7905988124290759
Validation Loss Decreased(138.064834--->137.870498)   Saving The Model
Epoch 45 Val Loss: 0.5548423807544401    Val Accuracy: 0.7890263193817489
Epoch 46 Val Loss: 0.5532090175582702    Val Accuracy: 0.7906870422088842
Validation Loss Decreased(137.870498--->137.749045)   Saving The Model
Epoch 47 Val Loss: 0.5518236157884456    Val Accuracy: 0.7913949235727069
Validation Loss Decreased(137.749045--->137.404080)   Saving The Model
Epoch 48 Val Loss: 0.5552902398817989    Val Accuracy: 0.7889332470742472
Epoch 49 Val Loss: 0.5509715440522236    Val Accuracy: 0.7929410748884489
Validation Loss Decreased(137.404080--->137.191914)   Saving The Model
Epoch 50 Val Loss: 0.5599430213012848    Val Accuracy: 0.7862640177677204
Epoch 51 Val Loss: 0.5553564818269278    Val Accuracy: 0.7879640720303269
Epoch 52 Val Loss: 0.5533078248242298    Val Accuracy: 0.790961877167009
Epoch 53 Val Loss: 0.5510819677607601    Val Accuracy: 0.7938194375269 73
Epoch 54 Val Loss: 0.5491095867501684    Val Accuracy: 0.7948023969771705
Validation Loss Decreased(137.191914--->136.728287)   Saving The Model
Epoch 55 Val Loss: 0.5542238871735262    Val Accuracy: 0.7920002021426027
Epoch 56 Val Loss: 0.5540607851671885    Val Accuracy: 0.7926925465637594
Epoch 57 Val Loss: 0.5586805237105572    Val Accuracy: 0.7915490703203594
Epoch 58 Val Loss: 0.5522291061868629    Val Accuracy: 0.7909825558336883
```

```
Epoch 59 Val Loss: 0.5536317725976309    Val Accuracy: 0.791813017332504
Epoch 60 Val Loss: 0.5509639117851793    Val Accuracy: 0.7932795228280101
Epoch 61 Val Loss: 0.5488814898284085    Val Accuracy: 0.7943622201409652
Validation Loss Decreased(136.728287--->136.671491)   Saving The Model
Epoch 62 Val Loss: 0.5539373347318794    Val Accuracy: 0.7934323168531511
Epoch 63 Val Loss: 0.5554737137503414    Val Accuracy: 0.7890231694629167
Epoch 64 Val Loss: 0.5503094555383705    Val Accuracy: 0.7949597994877232
Epoch 65 Val Loss: 0.54909282479899    Val Accuracy: 0.796323277858574
Epoch 66 Val Loss: 0.5488084126189052    Val Accuracy: 0.7946651847460203
Validation Loss Decreased(136.671491--->136.653295)   Saving The Model
Epoch 67 Val Loss: 0.552268603958758    Val Accuracy: 0.7918372287297116
Epoch 68 Val Loss: 0.5471031270831465    Val Accuracy: 0.7963583213446038
Validation Loss Decreased(136.653295--->136.228679)   Saving The Model
Epoch 69 Val Loss: 0.5486813665154469    Val Accuracy: 0.7941641678449053
Epoch 70 Val Loss: 0.5534806477736278    Val Accuracy: 0.7932121882230542
Epoch 71 Val Loss: 0.5492884612466437    Val Accuracy: 0.7957704804938063
Epoch 72 Val Loss: 0.5500988726874432    Val Accuracy: 0.7951950553971937
Epoch 73 Val Loss: 0.5463436036464201    Val Accuracy: 0.7966609653803394
Validation Loss Decreased(136.228679--->136.039557)   Saving The Model
Epoch 74 Val Loss: 0.5502901565597718    Val Accuracy: 0.7956635674809804
Epoch 75 Val Loss: 0.5463940573025898    Val Accuracy: 0.7976926218882002
Epoch 76 Val Loss: 0.5498145846238577    Val Accuracy: 0.7939793907418021
Epoch 77 Val Loss: 0.547870142153468    Val Accuracy: 0.7954288318352085
Epoch 78 Val Loss: 0.5486669311801113    Val Accuracy: 0.7951280335239711
Epoch 79 Val Loss: 0.5579696082207094    Val Accuracy: 0.7916047021364552
Epoch 80 Val Loss: 0.5435890096976576    Val Accuracy: 0.7990652135810219
Validation Loss Decreased(136.039557--->135.353663)   Saving The Model
Epoch 81 Val Loss: 0.5471631761296207    Val Accuracy: 0.7965416909685112
Epoch 82 Val Loss: 0.5465099015628477    Val Accuracy: 0.7974240666673633
Epoch 83 Val Loss: 0.5490827853660507    Val Accuracy: 0.7971313369292102
Epoch 84 Val Loss: 0.552619075679396    Val Accuracy: 0.7924731653017879
Epoch 85 Val Loss: 0.5459773471077762    Val Accuracy: 0.7974210618893576
Epoch 86 Val Loss: 0.5459741634058665    Val Accuracy: 0.7979919688288144
Epoch 87 Val Loss: 0.547325289871798    Val Accuracy: 0.7956925068785714
Epoch 88 Val Loss: 0.5486206241161469    Val Accuracy: 0.7963201817506046
Epoch 89 Val Loss: 0.5441504472709564    Val Accuracy: 0.7988325699428362
Epoch 90 Val Loss: 0.5432127673463171    Val Accuracy: 0.7997780787156075
Validation Loss Decreased(135.353663--->135.259979)   Saving The Model
Epoch 91 Val Loss: 0.5507725414980846    Val Accuracy: 0.7975458469629182
Epoch 92 Val Loss: 0.5451186544684521    Val Accuracy: 0.7983876271184818
Epoch 93 Val Loss: 0.5435707248358362    Val Accuracy: 0.7999792324613776
Epoch 94 Val Loss: 0.5440631664421663    Val Accuracy: 0.7989747040582595
Epoch 95 Val Loss: 0.5506150795513367    Val Accuracy: 0.793918734119664
Epoch 96 Val Loss: 0.5447543131778518    Val Accuracy: 0.798523968757935
Epoch 97 Val Loss: 0.5444216291349097    Val Accuracy: 0.7987665003514922
Epoch 98 Val Loss: 0.5429947424126438    Val Accuracy: 0.8003196392163822
Validation Loss Decreased(135.259979--->135.205691)   Saving The Model
Epoch 99 Val Loss: 0.5434152144026086    Val Accuracy: 0.7998955599857424
Epoch 100 Val Loss: 0.5479928755377191    Val Accuracy: 0.7987572783264381
Epoch 101 Val Loss: 0.5434252987185635    Val Accuracy: 0.7995108796509299
Epoch 102 Val Loss: 0.5436309410865048    Val Accuracy: 0.8005066888760752
Epoch 103 Val Loss: 0.5456557466562493    Val Accuracy: 0.7993690255128586
Epoch 104 Val Loss: 0.5442182991399344    Val Accuracy: 0.799690623427624
Epoch 105 Val Loss: 0.5437312808381506    Val Accuracy: 0.7996385198059367
Epoch 106 Val Loss: 0.5438162975282554    Val Accuracy: 0.8006004636720778
Epoch 107 Val Loss: 0.5446193883696713    Val Accuracy: 0.7996048067121134
Epoch 108 Val Loss: 0.5449816524743076    Val Accuracy: 0.8004485400622667
Epoch 109 Val Loss: 0.5425583464074805    Val Accuracy: 0.8008658828947387
Validation Loss Decreased(135.205691--->135.097028)   Saving The Model
Epoch 110 Val Loss: 0.5418022716619882    Val Accuracy: 0.8010110653428814
Validation Loss Decreased(135.097028--->134.908766)   Saving The Model
Epoch 111 Val Loss: 0.5481426007297623    Val Accuracy: 0.796777538330923
Epoch 112 Val Loss: 0.5422392495904103    Val Accuracy: 0.8005136268851132
Epoch 113 Val Loss: 0.5471664896930557    Val Accuracy: 0.7988482250751664
Epoch 114 Val Loss: 0.5419716951119373    Val Accuracy: 0.8008111687969681
Epoch 115 Val Loss: 0.5421704361477051    Val Accuracy: 0.8008548722283753
Epoch 116 Val Loss: 0.5440301757500353    Val Accuracy: 0.8012869699931994
Epoch 117 Val Loss: 0.5419068623738117    Val Accuracy: 0.8011014252145348
Epoch 118 Val Loss: 0.5403775511735893    Val Accuracy: 0.8015702623705115
Validation Loss Decreased(134.908766--->134.554010)   Saving The Model
Epoch 119 Val Loss: 0.5445032192760682    Val Accuracy: 0.798902460855946
Epoch 120 Val Loss: 0.5418781277884441    Val Accuracy: 0.8019421454174165
```

```
Epoch 121 Val Loss: 0.5461475211214349    Val Accuracy: 0.8008584215404588
Epoch 122 Val Loss: 0.5446651758678467    Val Accuracy: 0.7987802875450409
Epoch 123 Val Loss: 0.542019715510219    Val Accuracy: 0.8019373856528804
Epoch 124 Val Loss: 0.5422930437398245    Val Accuracy: 0.8011130578682756
Epoch 125 Val Loss: 0.5425454135161327    Val Accuracy: 0.8002590976443041
Epoch 126 Val Loss: 0.5425348413517197    Val Accuracy: 0.8008960365165013
Epoch 127 Val Loss: 0.5419042321094069    Val Accuracy: 0.8016338316719579
Epoch 128 Val Loss: 0.5412611666932163    Val Accuracy: 0.8014828641114063
Epoch 129 Val Loss: 0.5408807967800692    Val Accuracy: 0.8026792388251707
Epoch 130 Val Loss: 0.5479325045304126    Val Accuracy: 0.7959223178314737
Epoch 131 Val Loss: 0.540656698396407    Val Accuracy: 0.8019421958012173
Epoch 132 Val Loss: 0.5419903063630483    Val Accuracy: 0.8007079053818065
Epoch 133 Val Loss: 0.5411662948897564    Val Accuracy: 0.8020616996100395
Epoch 134 Val Loss: 0.5415161696064424    Val Accuracy: 0.8017890334493841
Epoch 135 Val Loss: 0.5418746952550957    Val Accuracy: 0.8017071168050216
Epoch 136 Val Loss: 0.5401937995091021    Val Accuracy: 0.8019535849055458
Validation Loss Decreased(134.554010--->134.508256)   Saving The Model
Epoch 137 Val Loss: 0.5410854903809038    Val Accuracy: 0.8016166115176938
Epoch 138 Val Loss: 0.5404421925305363    Val Accuracy: 0.8025713305898755
Epoch 139 Val Loss: 0.539483884730971    Val Accuracy: 0.8029643490146431
Validation Loss Decreased(134.508256--->134.331487)   Saving The Model
Epoch 140 Val Loss: 0.5409936641593535    Val Accuracy: 0.8031312776366267
Epoch 141 Val Loss: 0.539796689188624    Val Accuracy: 0.8032585010611509
Epoch 142 Val Loss: 0.5433084883364329    Val Accuracy: 0.8003947186449506
Epoch 143 Val Loss: 0.5401424138421514    Val Accuracy: 0.8025408459124364
Epoch 144 Val Loss: 0.5412057852409929    Val Accuracy: 0.8013703334315806
Epoch 145 Val Loss: 0.5391651543986845    Val Accuracy: 0.8033985036546101
Validation Loss Decreased(134.331487--->134.252123)   Saving The Model
Epoch 146 Val Loss: 0.5411269231493693    Val Accuracy: 0.8029710581286199
Epoch 147 Val Loss: 0.5389605515213856    Val Accuracy: 0.8037414160758705
Validation Loss Decreased(134.252123--->134.201177)   Saving The Model
Epoch 148 Val Loss: 0.5385785791050478    Val Accuracy: 0.8039301387250876
Validation Loss Decreased(134.201177--->134.106066)   Saving The Model
Epoch 149 Val Loss: 0.5386352916079832    Val Accuracy: 0.8042345842758178
Epoch 150 Val Loss: 0.5393852559198816    Val Accuracy: 0.8033927363106489
Epoch 151 Val Loss: 0.5401578314572453    Val Accuracy: 0.8029361446459983
Epoch 152 Val Loss: 0.5394743629488121    Val Accuracy: 0.8032579291621142
Epoch 153 Val Loss: 0.5389294193451664    Val Accuracy: 0.8029245585821578
Epoch 154 Val Loss: 0.5388284866589619    Val Accuracy: 0.803953710084886
Epoch 155 Val Loss: 0.5394690258196558    Val Accuracy: 0.802977102594302
Epoch 156 Val Loss: 0.5391070788883301    Val Accuracy: 0.803405083201045
Epoch 157 Val Loss: 0.5395317518088713    Val Accuracy: 0.8030079367605725
Epoch 158 Val Loss: 0.5409737165912567    Val Accuracy: 0.8027854759267059
Epoch 159 Val Loss: 0.5413098846334051    Val Accuracy: 0.8019477851859811
Epoch 160 Val Loss: 0.5464323723172567    Val Accuracy: 0.7995118602896795
Epoch 161 Val Loss: 0.5381207226749405    Val Accuracy: 0.8044785374359396
Validation Loss Decreased(134.106066--->133.992060)   Saving The Model
Epoch 162 Val Loss: 0.5409473373468621    Val Accuracy: 0.8022029597780724
Epoch 163 Val Loss: 0.5390406970278805    Val Accuracy: 0.8040416822331017
Epoch 164 Val Loss: 0.5397855160705536    Val Accuracy: 0.8031397950390577
Epoch 165 Val Loss: 0.5422663562987224    Val Accuracy: 0.8012166130611782
Epoch 166 Val Loss: 0.5389690734296438    Val Accuracy: 0.8038128338075441
Epoch 167 Val Loss: 0.5405272846241074    Val Accuracy: 0.803631937965966
Epoch 168 Val Loss: 0.5403668824209267    Val Accuracy: 0.8032395928838816
Epoch 169 Val Loss: 0.5382595525448581    Val Accuracy: 0.8040823176775677
Epoch 170 Val Loss: 0.5403155205718964    Val Accuracy: 0.8036169962006562
Epoch 171 Val Loss: 0.5381770964607178    Val Accuracy: 0.8039606861984997
Epoch 172 Val Loss: 0.5395261715933022    Val Accuracy: 0.8033569778577089
Epoch 173 Val Loss: 0.5392816391096537    Val Accuracy: 0.8050212978607125
Epoch 174 Val Loss: 0.5412740672687929    Val Accuracy: 0.8044511703401473
Epoch 175 Val Loss: 0.5399958820467493    Val Accuracy: 0.8038542720628452
Epoch 176 Val Loss: 0.5386090101487186    Val Accuracy: 0.8049632324604314
Epoch 177 Val Loss: 0.5383178019619371    Val Accuracy: 0.8047474655726518
Epoch 178 Val Loss: 0.5381267623729017    Val Accuracy: 0.8043545799264975
Epoch 179 Val Loss: 0.539576028484896    Val Accuracy: 0.8041004291718947
Epoch 180 Val Loss: 0.5395186196369339    Val Accuracy: 0.8037332392317453
Epoch 181 Val Loss: 0.5388431030823045    Val Accuracy: 0.8038766250458411
Training Stopped.
```

In [ ]:

```
# Performance on test set
```

```
trained_model = Net().to(device)

trained_model.load_state_dict(torch.load('saved_model.pth'))
trained_model.eval()

with torch.no_grad():
    test_loss = 0.0
    test_auc = 0.0
    for data in test_loader:
        inputs, labels = data
        inputs, labels = inputs.to(device), labels.to(device)
        outputs = trained_model(inputs)
        labels , outputs = labels.type(torch.FloatTensor),outputs.type(torch.FloatTensor
)
        loss = criterion(outputs,labels)
        test_loss += loss.item()
        test_auc += roc_auc_score(labels.numpy(), outputs.numpy())

print(f"The loss on testing data is {test_loss/len(test_loader)} and the ROC-AUC is {test
_auc/len(test_loader)}")
```

```
The loss on testing data is 0.5423456724867763 and the ROC-AUC is 0.8003556647834663
```

Best ROC AUC score (validate) : 0.8083 </br> Best ROC AUC score (test) : 0.8004 </br>