task-3

March 14, 2023

This file contains GNN based solution for specific task of project Graph Neural Networks for End-to-End Particle Identification with the CMS Experiment.

GNN layers been used:

- Graph Convolution Layer
- PointNet Convolution Layer

In both, model architecture is composed of two layers. Latent embedding dimension is set to 300.

Node features: - Channel values - Global Positional encoding (3D coordinates of the nodes) - optional

Edge features: - Euclidean distance between nodes.

Both models are trained for 75 epochs.

[1]:
```
import os
from tqdm import tqdm
import torch
from torch_geometric.nn import MessagePassing
from torch_geometric.nn import global_add_pool, global_mean_pool,␣
 ↪global_max_pool, GlobalAttention, Set2Set
import torch.nn.functional as F
from torch_geometric.utils import degree
from torch.utils.data import random_split
from torch_geometric.loader import DataLoader
import torch.optim as optim
from torchmetrics.classification import MulticlassAUROC, MulticlassAccuracy

from dataset import JetsGraphsDataset
from torch_geometric.nn.conv import GATConv,PointNetConv
```

[2]:
```
### GIN convolution along the graph structure
class GINConv(MessagePassing):
    def __init__(self, emb_dim,input_node_dim,input_edge_dim):

        super(GINConv, self).__init__(aggr = "add")
```

```
        self.mlp = torch.nn.Sequential(torch.nn.Linear(emb_dim, 2*emb_dim),␣
↪torch.nn.BatchNorm1d(2*emb_dim),
                                       torch.nn.ReLU(), torch.nn.
↪Linear(2*emb_dim, emb_dim))
        self.eps = torch.nn.Parameter(torch.Tensor([0]))
        self.linear = torch.nn.Linear(input_node_dim, emb_dim)
        self.edge_encoder = torch.nn.Linear(input_edge_dim, emb_dim)

    def forward(self, x, edge_index, edge_attr):
        x = self.linear(x)
        edge_embedding = self.edge_encoder(edge_attr)
        out = self.mlp((1 + self.eps) *x + self.propagate(edge_index, x=x,␣
↪edge_attr=edge_embedding))

        return out

    def message(self, x_j, edge_attr):
        return F.relu(x_j + edge_attr)

    def update(self, aggr_out):
        return aggr_out

### GCN convolution along the graph structure
class GCNConv(MessagePassing):
    def __init__(self, emb_dim,input_node_dim,input_edge_dim):
        super(GCNConv, self).__init__(aggr='add')

        self.linear = torch.nn.Linear(input_node_dim, emb_dim)
        self.root_emb = torch.nn.Embedding(1, emb_dim)
        self.edge_encoder = torch.nn.Linear(input_edge_dim, emb_dim)

    def forward(self, x, edge_index, edge_attr):
        x = self.linear(x)
        edge_embedding = self.edge_encoder(edge_attr)

        row, col = edge_index

        #edge_weight = torch.ones((edge_index.size(1), ), device=edge_index.
↪device)
        deg = degree(row, x.size(0), dtype = x.dtype) + 1
        deg_inv_sqrt = deg.pow(-0.5)
        deg_inv_sqrt[deg_inv_sqrt == float('inf')] = 0

        norm = deg_inv_sqrt[row] * deg_inv_sqrt[col]

        return self.propagate(edge_index, x=x, edge_attr = edge_embedding,␣
↪norm=norm) + F.relu(x + self.root_emb.weight) * 1./deg.view(-1,1)
```

```
    def message(self, x_j, edge_attr, norm):
        return norm.view(-1, 1) * F.relu(x_j + edge_attr)

    def update(self, aggr_out):
        return aggr_out
```

[3]:
```
class mlp(torch.nn.Module):
    def __init__(self,input_node_dim,emb_dim):
        super(mlp, self).__init__()
        self.mlp = torch.nn.Sequential(torch.nn.Linear(input_node_dim,␣
↪2*emb_dim),
                                       torch.nn.BatchNorm1d(2*emb_dim),
                                       torch.nn.ReLU(),
                                       torch.nn.Linear(2*emb_dim, emb_dim))
    def forward(self,x):
        return self.mlp(x)
```

[4]:
```
class MessagePasssing_Module(torch.nn.Module):
    """
    MessagePasssing_Module contains 1 or more GNN layers stacked.
    Output:
        node representations
    """
    def __init__(self, num_layer, input_node_dim, input_edge_dim, emb_dim,␣
↪hasPos = True,
                 drop_ratio = 0.5, JK = "last", residual = False, gnn_type =␣
↪'gin'):
        '''
            emb_dim (int): node embedding dimensionality
            num_layer (int): number of GNN message passing layers
            hasPos (bool) : whether input node features should contain global␣
↪positioning embeded
                                            ps: global positioning is the␣
↪coordinate of the pixel on 2D grid.

        '''
        super(MessagePasssing_Module, self).__init__()

        self.gnn_type = gnn_type
        self.num_layer = num_layer
        self.drop_ratio = drop_ratio
        self.JK = JK
        self.input_node_dim = input_node_dim
        self.input_edge_dim = input_edge_dim
        ### add residual connection or not
```

```
        self.residual = residual
        self.hasPos = hasPos

        if self.num_layer < 2:
            raise ValueError("Number of GNN layers must be greater than 1.")

        ### List of GNNs
        self.convs = torch.nn.ModuleList()
        self.batch_norms = torch.nn.ModuleList()


        for layer in range(num_layer):
            if layer == 0:
                if gnn_type == 'gin':
                    self.convs.append(GINConv(emb_dim,input_node_dim=self.
↪input_node_dim,input_edge_dim=self.input_edge_dim))
                elif gnn_type == 'gcn':
                    self.convs.append(GCNConv(emb_dim,input_node_dim=self.
↪input_node_dim,input_edge_dim=self.input_edge_dim))
                elif gnn_type == 'gat':
                    self.convs.append(GATConv(in_channels=self.
↪input_node_dim,out_channels=emb_dim,edge_dim=self.input_edge_dim))
                elif gnn_type == "pointnet":
                    local_mlp = mlp(self.input_node_dim+3, emb_dim)
                    global_mlp = None
                    self.convs.append(PointNetConv(local_mlp,global_mlp))
                else:
                    raise ValueError('Undefined GNN type called {}'.
↪format(gnn_type))

            else:
                if gnn_type == 'gin':
                    self.convs.
↪append(GINConv(emb_dim,input_node_dim=emb_dim,input_edge_dim=self.
↪input_edge_dim))
                elif gnn_type == 'gcn':
                    self.convs.
↪append(GCNConv(emb_dim,input_node_dim=emb_dim,input_edge_dim=self.
↪input_edge_dim))
                elif gnn_type == 'gat':
                    self.convs.
↪append(GATConv(in_channels=emb_dim,out_channels=emb_dim,edge_dim=self.
↪input_edge_dim))
                elif gnn_type == "pointnet":
                    local_mlp = mlp(emb_dim+3, emb_dim)
                    global_mlp = None
```

```
                    self.convs.append(PointNetConv(local_mlp,global_mlp))
                else:
                    raise ValueError('Undefined GNN type called {}'.
↪format(gnn_type))

            self.batch_norms.append(torch.nn.BatchNorm1d(emb_dim))

    def forward(self, batched_data):
        x, edge_index, edge_attr, batch = batched_data.x, batched_data.
↪edge_index, batched_data.edge_attr, batched_data.batch

        pos = x[:,3:] # this will keep pos embeddings

        if not self.hasPos:
            x = x[:,:3] # this will drop global positioning/ coords

        h_list = [x]
        for layer in range(self.num_layer):
            if self.gnn_type == 'pointnet':
                h = self.convs[layer](h_list[layer], pos, edge_index)
            else:
                h = self.convs[layer](h_list[layer], edge_index, edge_attr)
            h = self.batch_norms[layer](h)

            if layer == self.num_layer - 1:
                #remove relu for the last layer
                h = F.dropout(h, self.drop_ratio, training = self.training)
            else:
                h = F.dropout(F.relu(h), self.drop_ratio, training = self.
↪training)

            if self.residual:
                h += h_list[layer]

            h_list.append(h)

        ### Different implementations of Jk-concat
        if self.JK == "last":
            node_representation = h_list[-1]
        elif self.JK == "sum":
            node_representation = 0
            for layer in range(self.num_layer + 1):
                node_representation += h_list[layer]

        return node_representation
```

[5]:
```
class GNN(torch.nn.Module):

    def __init__(self, num_classes=2, num_layer = 5,num_pre_fnn_layers=0,num_post_fnn_layers =1,hasPos =True,
                 input_node_dim=3,input_edge_dim = 1, emb_dim = 300, gnn_type ='gcn', residual = False,
                 drop_ratio = 0.5, JK = "last", graph_pooling = "mean"):

        super(GNN, self).__init__()

        self.gnn_type = gnn_type
        self.num_layer = num_layer
        self.drop_ratio = drop_ratio
        self.JK = JK
        self.emb_dim = emb_dim
        self.hasPos = hasPos
        self.num_classes = num_classes
        self.num_pre_fnn_layers = num_pre_fnn_layers
        self.num_post_fnn_layers = num_post_fnn_layers
        self.graph_pooling = graph_pooling
        self.input_node_dim = input_node_dim
        self.input_edge_dim = input_edge_dim

        if self.gnn_type=="pointnet":
            self.hasPos = False

        self.pos_kwd = "hasPos"
        if not self.hasPos:
            self.pos_kwd = "noPos"

        if self.num_layer < 2:
            raise ValueError("Number of GNN layers must be greater than 1.")

        if self.num_post_fnn_layers < 1:
            raise ValueError("Number of GNN layers must be greater than or equal to 1.")

        if self.num_pre_fnn_layers >0:
            self.graph_pred_linear_list.append(torch.nn.Linear(self.input_node_dim, emb_dim))
            for i in range(1,num_pre_fnn_layers):
                self.graph_pred_linear_list.append(torch.nn.Linear(emb_dim,emb_dim))
            self.input_node_dim = emb_dim

        ### GNN to generate node embeddings
```

```
        self.gnn_node = MessagePasssing_Module(num_layer,input_node_dim=self.
↪input_node_dim,
                                               input_edge_dim = self.
↪input_edge_dim, emb_dim=emb_dim,hasPos=self.hasPos,
                                               JK = JK, drop_ratio =␣
↪drop_ratio, residual = residual,
                                               gnn_type = gnn_type)

        ### Pooling function to generate entire-graph embeddings
        if self.graph_pooling == "sum":
            self.pool = global_add_pool
        elif self.graph_pooling == "mean":
            self.pool = global_mean_pool
        elif self.graph_pooling == "max":
            self.pool = global_max_pool
        elif self.graph_pooling == "attention":
            self.pool = GlobalAttention(gate_nn = torch.nn.Sequential(torch.nn.
↪Linear(emb_dim, 2*emb_dim),
                                                                      torch.nn.
↪BatchNorm1d(2*emb_dim), torch.nn.ReLU(),
                                                                      torch.nn.
↪Linear(2*emb_dim, 1)))
        else:
            raise ValueError("Invalid graph pooling type.")

        self.graph_pred_linear_list = torch.nn.ModuleList()

        for i in range(num_post_fnn_layers-1):
            self.graph_pred_linear_list.append(torch.nn.Linear(emb_dim,␣
↪emb_dim))
        self.graph_pred_linear_list.append(torch.nn.Linear(emb_dim, self.
↪num_classes))



    def forward(self, batched_data):

        # input_dim=batched_data.x.size(0)

        h_node = self.gnn_node(batched_data)

        h_graph = self.pool(h_node, batched_data.batch)

        output = h_graph # initial input is set to the output of the GNN
        for fnn_inx in range(self.num_post_fnn_layers):
            output = self.graph_pred_linear_list[fnn_inx](output)
```

```
        return F.softmax(output,dim=1)

    def __str__(self):
        return self.gnn_type+f"-model-{self.pos_kwd}"
```

[6]:
```
device = torch.device("cuda:0" if torch.cuda.is_available() else torch.
 ↪device("cpu"))
```

[7]:
```
multicls_criterion = torch.nn.CrossEntropyLoss()
epochs = 75
```

[8]:
```
# importing dataset
jets_dataset = JetsGraphsDataset('../dataset/
 ↪',name="QCDToGGQQ_IMGjet_RH1all_jet0_run0_n36272")
```

[9]:
```
# random splitting dataset
train_inx, valid_inx, test_inx = random_split(range(len(jets_dataset)),[0.7,0.
 ↪2,0.1],generator=torch.Generator()
                                              .manual_seed(42))

train_dataloader = DataLoader(jets_dataset[list(train_inx)], batch_size=32,␣
 ↪shuffle=True)
valid_dataloader = DataLoader(jets_dataset[list(valid_inx)], batch_size=32,␣
 ↪shuffle=False)
test_dataloader = DataLoader(jets_dataset[list(test_inx)], batch_size=32,␣
 ↪shuffle=False)
```

[10]:
```
def train(model, device, loader, optimizer):
    model.train()

    loss_accum = 0
    for step, batch in enumerate(tqdm(loader, desc="Iteration")):
        batch=batch.to(device)
        if batch.x.shape[0] == 1:
            pass
        else:
            output = model(batch)
            loss= 0
            optimizer.zero_grad()
            loss += multicls_criterion(output, batch.y.view(-1).to(torch.int64))
            loss.backward()
            optimizer.step()

        loss_accum += loss.item()
```

```
    print('Average training loss: {}'.format(loss_accum / (step + 1)))
```

[11]:
```
def evaluate(model, device, loader,evaluator= "roauc"):
    model.eval()

    preds_list = []
    target_list = []
    for step, batch in enumerate(loader):
        batch = batch.to(device)
        with torch.no_grad():
            output = model(batch)
            preds_list.extend(output.tolist())
        target_list += batch.y.view(-1).tolist()

    if evaluator == "roauc":
        metric = MulticlassAUROC(num_classes=2, average="macro",␣
↪thresholds=None)
    if evaluator == "acc":
        metric = MulticlassAccuracy(num_classes=2, average="macro")
    # print("AUC-ROC metric score : ",metric(torch.Tensor(preds_list),torch.
↪Tensor(target_list)).item())
    return metric(torch.Tensor(preds_list),torch.Tensor(target_list).to(torch.
↪int64)).item()
```

[12]:
```
def train_model(model,optimizer):
    checkpoints_path = "../models"
    checkpoints = os.listdir(checkpoints_path)
    checkpoint_path = list(filter(lambda i : str(model) in i, checkpoints))

    train_curves = []
    valid_curves = []
    starting_epoch = 1
    if len(checkpoint_path)>0:
        checkpoint = torch.load(f"{checkpoints_path}/{checkpoint_path[0]}")
        model.load_state_dict(checkpoint['model_state_dict'])
        optimizer.load_state_dict(checkpoint['optimizer_state_dict'])
        starting_epoch = checkpoint['epoch']+1

    for epoch in range(starting_epoch, epochs + 1):
        print("=====Epoch {}".format(epoch))
        print('Training...')
        train(model, device, train_dataloader, optimizer)

        # save checkpoint of current epoch
        torch.save({
                'epoch': epoch,
                'model_state_dict': model.state_dict(),
```

```
                'optimizer_state_dict': optimizer.state_dict(),
                }, f"{checkpoints_path}/{str(model)}-{epoch}.pt")

        # delete checkpoint of previous epoch
        if epoch>1:
            os.remove(f"{checkpoints_path}/{str(model)}-{epoch-1}.pt")

        print("Evaluating...")
        train_perf_roauc = evaluate(model,device,train_dataloader)
        valid_perf_roauc = evaluate(model,device,valid_dataloader)
#         test_perf_roauc = evaluate(model,device,test_dataloader)
#         train_perf_acc = evaluate(model,device,train_dataloader, evaluator =␣
 ↪"acc")
#         valid_perf_acc = evaluate(model,device,valid_dataloader,evaluator =␣
 ↪"acc")
#         test_perf_acc = evaluate(model,device,test_dataloader,evaluator =␣
 ↪"acc")

#         train_curves.append([train_perf_acc,train_perf_roauc])
#         valid_curves.append([valid_perf_acc,valid_perf_roauc])

#         print('ROAUC scores: ',{'Train': train_perf_roauc, 'Validation':␣
 ↪valid_perf_roauc, "Test": test_perf_roauc}, '\nAccuracy scores: ',
#               {'Train': train_perf_acc, 'Validation': valid_perf_acc, "Test":␣
 ↪test_perf_acc})

        print('ROAUC scores: ',{'Train': train_perf_roauc, 'Validation':␣
 ↪valid_perf_roauc})



    print('\nFinished training!')
    print('\nROAUC Test score: {}'.
 ↪format(evaluate(model,device,test_dataloader)))

    return train_curves,valid_curves
```

Training PointNet Conv based GNN model

[ ]:
```
pointnet_model = GNN(num_classes = 2, num_layer = 2,num_post_fnn_layers=2,␣
 ↪input_node_dim=3,input_edge_dim = 1,
            gnn_type = 'pointnet', emb_dim = 300, drop_ratio = 0.3).to(device)
optimizer = optim.Adam(pointnet_model.parameters(), lr=1e-3)

train_model(pointnet_model,optimizer)
```

```
=====Epoch 1
```

```
Training…

Iteration: 100%|      | 794/794 [01:37<00:00,  8.12it/s]

Average training loss: 0.5968443244245251
Evaluating…
ROAUC scores:  {'Train': 0.7725400328636169, 'Validation': 0.7705419063568115}
=====Epoch 2
Training…

Iteration: 100%|      | 794/794 [02:44<00:00,  4.84it/s]

Average training loss: 0.5875780677765383
Evaluating…
ROAUC scores:  {'Train': 0.7801496982574463, 'Validation': 0.7786107063293457}
=====Epoch 3
Training…

Iteration: 100%|      | 794/794 [03:17<00:00,  4.02it/s]

Average training loss: 0.5844302855300663
Evaluating…
ROAUC scores:  {'Train': 0.7826152443885803, 'Validation': 0.7812164425849915}
=====Epoch 4
Training…

Iteration: 100%|      | 794/794 [03:18<00:00,  4.00it/s]

Average training loss: 0.5846906998370697
Evaluating…
ROAUC scores:  {'Train': 0.7792961597442627, 'Validation': 0.7805557250976562}
=====Epoch 5
Training…

Iteration: 100%|      | 794/794 [03:12<00:00,  4.13it/s]

Average training loss: 0.5830133735983438
Evaluating…
ROAUC scores:  {'Train': 0.7796788215637207, 'Validation': 0.7771973013877869}
=====Epoch 6
Training…

Iteration: 100%|      | 794/794 [03:12<00:00,  4.12it/s]

Average training loss: 0.5814722128569029
Evaluating…
ROAUC scores:  {'Train': 0.7812932729721069, 'Validation': 0.781397819519043}
=====Epoch 7
Training…

Iteration: 100%|      | 794/794 [03:11<00:00,  4.15it/s]

Average training loss: 0.5789326649694059
Evaluating…
ROAUC scores:  {'Train': 0.7868567705154419, 'Validation': 0.7841024398803711}
```

```
=====Epoch 8
Training…

Iteration: 100%|      | 794/794 [03:18<00:00,  3.99it/s]

Average training loss: 0.5804146431780582
Evaluating…
ROAUC scores:  {'Train': 0.7854431867599487, 'Validation': 0.7837200164794922}
=====Epoch 9
Training…

Iteration: 100%|      | 794/794 [03:12<00:00,  4.13it/s]

Average training loss: 0.5800728388787517
Evaluating…
ROAUC scores:  {'Train': 0.7872354984283447, 'Validation': 0.7860591411590576}
=====Epoch 10
Training…

Iteration: 100%|      | 794/794 [03:16<00:00,  4.04it/s]

Average training loss: 0.5798492744677614
Evaluating…
ROAUC scores:  {'Train': 0.7855093479156494, 'Validation': 0.7855976819992065}
=====Epoch 11
Training…

Iteration: 100%|      | 794/794 [03:17<00:00,  4.01it/s]

Average training loss: 0.5787262726640822
Evaluating…
ROAUC scores:  {'Train': 0.7860662341117859, 'Validation': 0.7840490341186523}
=====Epoch 12
Training…

Iteration: 100%|      | 794/794 [03:11<00:00,  4.14it/s]

Average training loss: 0.5791401911142191
Evaluating…
ROAUC scores:  {'Train': 0.7866134643554688, 'Validation': 0.7852720022201538}
=====Epoch 13
Training…

Iteration: 100%|      | 794/794 [03:16<00:00,  4.03it/s]

Average training loss: 0.579151221576806
Evaluating…
ROAUC scores:  {'Train': 0.7879475951194763, 'Validation': 0.785601019859314}
=====Epoch 14
Training…

Iteration: 100%|      | 794/794 [03:14<00:00,  4.09it/s]

Average training loss: 0.5781001861134464
Evaluating…
```

```
ROAUC scores:  {'Train': 0.7771030068397522, 'Validation': 0.7736259698867798}
=====Epoch 15
Training…

Iteration: 100%|      | 794/794 [08:41<00:00,  1.52it/s]

Average training loss: 0.577440016553444
Evaluating…
ROAUC scores:  {'Train': 0.7870498299598694, 'Validation': 0.7850008010864258}
=====Epoch 16
Training…

Iteration: 100%|      | 794/794 [03:03<00:00,  4.33it/s]

Average training loss: 0.5777557659659638
Evaluating…
ROAUC scores:  {'Train': 0.762190580368042, 'Validation': 0.7602964639663696}
=====Epoch 17
Training…

Iteration: 100%|      | 794/794 [02:57<00:00,  4.47it/s]

Average training loss: 0.5766549351008172
Evaluating…
ROAUC scores:  {'Train': 0.7872689366340637, 'Validation': 0.7858028411865234}
=====Epoch 18
Training…

Iteration: 100%|      | 794/794 [02:54<00:00,  4.55it/s]

Average training loss: 0.5769129636080499
Evaluating…
ROAUC scores:  {'Train': 0.788252055644989, 'Validation': 0.7873647212982178}
=====Epoch 19
Training…

Iteration: 100%|      | 794/794 [02:49<00:00,  4.67it/s]

Average training loss: 0.576685763201125
Evaluating…
ROAUC scores:  {'Train': 0.7860899567604065, 'Validation': 0.7834397554397583}
=====Epoch 20
Training…

Iteration: 100%|      | 794/794 [02:50<00:00,  4.67it/s]

Average training loss: 0.5760915947500945
Evaluating…
ROAUC scores:  {'Train': 0.7894119024276733, 'Validation': 0.7862321734428406}
=====Epoch 21
Training…

Iteration: 100%|      | 794/794 [02:46<00:00,  4.77it/s]
```

```
Average training loss: 0.5769186928605553
Evaluating…
ROAUC scores:  {'Train': 0.7865864038467407, 'Validation': 0.7861353158950806}
=====Epoch 22
Training…

Iteration: 100%|      | 794/794 [02:49<00:00,  4.68it/s]

Average training loss: 0.5753102543522188
Evaluating…
ROAUC scores:  {'Train': 0.7883642911911011, 'Validation': 0.7849191427230835}
=====Epoch 23
Training…

Iteration: 100%|      | 794/794 [02:51<00:00,  4.64it/s]

Average training loss: 0.5745986026690649
Evaluating…
ROAUC scores:  {'Train': 0.7894608974456787, 'Validation': 0.7863061428070068}
=====Epoch 24
Training…

Iteration: 100%|      | 794/794 [02:59<00:00,  4.42it/s]

Average training loss: 0.5776910598692425
Evaluating…
ROAUC scores:  {'Train': 0.7852466106414795, 'Validation': 0.7835105061531067}
=====Epoch 25
Training…

Iteration: 100%|      | 794/794 [02:57<00:00,  4.48it/s]

Average training loss: 0.5764218710681954
Evaluating…
ROAUC scores:  {'Train': 0.7644175887107849, 'Validation': 0.7614254951477051}
=====Epoch 26
Training…

Iteration: 100%|      | 794/794 [02:53<00:00,  4.57it/s]

Average training loss: 0.5773236190416952
Evaluating…
ROAUC scores:  {'Train': 0.7885024547576904, 'Validation': 0.7852720022201538}
=====Epoch 27
Training…

Iteration: 100%|      | 794/794 [02:54<00:00,  4.54it/s]

Average training loss: 0.5759505648306695
Evaluating…
ROAUC scores:  {'Train': 0.7902500629425049, 'Validation': 0.7867677211761475}
=====Epoch 28
Training…
```

```
Iteration: 100%|      | 794/794 [02:53<00:00,  4.57it/s]

Average training loss: 0.5751700830699815
Evaluating…
ROAUC scores:  {'Train': 0.7886245250701904, 'Validation': 0.7871099710464478}
=====Epoch 29
Training…

Iteration: 100%|      | 794/794 [03:03<00:00,  4.34it/s]

Average training loss: 0.5751021978461772
Evaluating…
ROAUC scores:  {'Train': 0.773712158203125, 'Validation': 0.7691928148269653}
=====Epoch 30
Training…

Iteration: 100%|      | 794/794 [02:57<00:00,  4.49it/s]

Average training loss: 0.5749188440122293
Evaluating…
ROAUC scores:  {'Train': 0.787826418876648, 'Validation': 0.7838537096977234}
=====Epoch 31
Training…

Iteration: 100%|      | 794/794 [02:56<00:00,  4.51it/s]

Average training loss: 0.5751555557139875
Evaluating…
ROAUC scores:  {'Train': 0.7896990776062012, 'Validation': 0.7879261374473572}
=====Epoch 32
Training…

Iteration: 100%|      | 794/794 [02:56<00:00,  4.50it/s]

Average training loss: 0.5739112086124925
Evaluating…
ROAUC scores:  {'Train': 0.789286732673645, 'Validation': 0.7848456501960754}
=====Epoch 33
Training…

Iteration: 100%|      | 794/794 [02:57<00:00,  4.47it/s]

Average training loss: 0.5746987097209286
Evaluating…
ROAUC scores:  {'Train': 0.7911396026611328, 'Validation': 0.7875732183456421}
=====Epoch 34
Training…

Iteration: 100%|      | 794/794 [02:53<00:00,  4.59it/s]

Average training loss: 0.5755641829381962
Evaluating…
ROAUC scores:  {'Train': 0.7922297120094299, 'Validation': 0.7895095348358154}
```

```
=====Epoch 35
Training…

Iteration:  43%|        | 343/794 [01:13<01:36,  4.66it/s]
```

[ ]:
```
pointnet_model = GNN(num_classes = 2, num_layer = 2,num_post_fnn_layers=2,␣
  ↪input_node_dim=3,input_edge_dim = 1,
            gnn_type = 'pointnet', emb_dim = 300, drop_ratio = 0.3).to(device)
optimizer = optim.Adam(pointnet_model.parameters(), lr=1e-3)

train_model(pointnet_model,optimizer)
```

```
=====Epoch 35
Training…

Iteration: 100%|      | 794/794 [01:39<00:00,  7.98it/s]

Average training loss: 0.5739827171456003
Evaluating…
ROAUC scores:  {'Train': 0.7900856733322144, 'Validation': 0.7864100933074951}
=====Epoch 36
Training…

Iteration: 100%|      | 794/794 [02:26<00:00,  5.40it/s]

Average training loss: 0.5746025304575111
Evaluating…
ROAUC scores:  {'Train': 0.7909984588623047, 'Validation': 0.7868366837501526}
=====Epoch 37
Training…

Iteration: 100%|      | 794/794 [02:40<00:00,  4.96it/s]

Average training loss: 0.5739615860603918
Evaluating…
ROAUC scores:  {'Train': 0.7911142110824585, 'Validation': 0.7861701250076294}
=====Epoch 38
Training…

Iteration: 100%|      | 794/794 [02:29<00:00,  5.30it/s]

Average training loss: 0.5743755280070701
Evaluating…
ROAUC scores:  {'Train': 0.7898068428039551, 'Validation': 0.7860455513000488}
=====Epoch 39
Training…

Iteration: 100%|      | 794/794 [02:33<00:00,  5.16it/s]

Average training loss: 0.5730699439267968
Evaluating…
ROAUC scores:  {'Train': 0.7923632264137268, 'Validation': 0.7872792482376099}
```

```
=====Epoch 40
Training…

Iteration: 100%|      | 794/794 [02:32<00:00,  5.22it/s]

Average training loss: 0.5726659616610266
Evaluating…
ROAUC scores:  {'Train': 0.7820241451263428, 'Validation': 0.7792727947235107}
=====Epoch 41
Training…

Iteration: 100%|      | 794/794 [02:35<00:00,  5.11it/s]

Average training loss: 0.5738611338796183
Evaluating…
ROAUC scores:  {'Train': 0.7933529615402222, 'Validation': 0.7894002199172974}
=====Epoch 42
Training…

Iteration: 100%|      | 794/794 [02:34<00:00,  5.15it/s]

Average training loss: 0.5728686149759917
Evaluating…
ROAUC scores:  {'Train': 0.7913365364074707, 'Validation': 0.7864618301391602}
=====Epoch 43
Training…

Iteration: 100%|      | 794/794 [02:36<00:00,  5.08it/s]

Average training loss: 0.5727560592238189
Evaluating…
ROAUC scores:  {'Train': 0.7900986671447754, 'Validation': 0.7850940227508545}
=====Epoch 44
Training…

Iteration: 100%|      | 794/794 [02:38<00:00,  5.00it/s]

Average training loss: 0.5730623385167543
Evaluating…
ROAUC scores:  {'Train': 0.7675602436065674, 'Validation': 0.7637292742729187}
=====Epoch 45
Training…

Iteration: 100%|      | 794/794 [02:35<00:00,  5.10it/s]

Average training loss: 0.5731236220382022
Evaluating…
ROAUC scores:  {'Train': 0.7934221029281616, 'Validation': 0.7886366844177246}
=====Epoch 46
Training…

Iteration: 100%|      | 794/794 [02:37<00:00,  5.05it/s]

Average training loss: 0.5723259001219603
Evaluating…
```

```
ROAUC scores:  {'Train': 0.7934394478797913, 'Validation': 0.7887582182884216}
=====Epoch 47
Training…

Iteration: 100%|      | 794/794 [02:37<00:00,  5.05it/s]

Average training loss: 0.5739216277145919
Evaluating…
ROAUC scores:  {'Train': 0.7764313220977783, 'Validation': 0.7731903195381165}
=====Epoch 48
Training…

Iteration: 100%|      | 794/794 [02:37<00:00,  5.05it/s]

Average training loss: 0.5730520148646622
Evaluating…
ROAUC scores:  {'Train': 0.7866554856300354, 'Validation': 0.7817510962486267}
=====Epoch 49
Training…

Iteration: 100%|      | 794/794 [02:37<00:00,  5.05it/s]

Average training loss: 0.5717012625872638
Evaluating…
ROAUC scores:  {'Train': 0.7875276803970337, 'Validation': 0.7825453281402588}
=====Epoch 50
Training…

Iteration: 100%|      | 794/794 [02:43<00:00,  4.87it/s]

Average training loss: 0.5716628171785052
Evaluating…
ROAUC scores:  {'Train': 0.7603659629821777, 'Validation': 0.7561715841293335}
=====Epoch 51
Training…

Iteration: 100%|      | 794/794 [02:39<00:00,  4.98it/s]

Average training loss: 0.571504502062233
Evaluating…
ROAUC scores:  {'Train': 0.7938394546508789, 'Validation': 0.7886555790901184}
=====Epoch 52
Training…

Iteration: 100%|      | 794/794 [02:38<00:00,  5.02it/s]

Average training loss: 0.5719038715740896
Evaluating…
ROAUC scores:  {'Train': 0.794448733329773, 'Validation': 0.7890756130218506}
=====Epoch 53
Training…

Iteration: 100%|      | 794/794 [02:42<00:00,  4.89it/s]
```

```
Average training loss: 0.5718246735538584
Evaluating…
ROAUC scores:  {'Train': 0.7927464246749878, 'Validation': 0.7879352569580078}
=====Epoch 54
Training…

Iteration: 100%|      | 794/794 [02:32<00:00,  5.20it/s]

Average training loss: 0.5709853943259049
Evaluating…
ROAUC scores:  {'Train': 0.7714006900787354, 'Validation': 0.7676116228103638}
=====Epoch 55
Training…

Iteration: 100%|      | 794/794 [02:37<00:00,  5.04it/s]

Average training loss: 0.5721219574249061
Evaluating…
ROAUC scores:  {'Train': 0.7951651215553284, 'Validation': 0.7891930341720581}
=====Epoch 56
Training…

Iteration: 100%|      | 794/794 [02:37<00:00,  5.04it/s]

Average training loss: 0.5709663217404027
Evaluating…
ROAUC scores:  {'Train': 0.7937808036804199, 'Validation': 0.788083553314209}
=====Epoch 57
Training…

Iteration: 100%|      | 794/794 [02:35<00:00,  5.10it/s]

Average training loss: 0.5717437531170376
Evaluating…
```

[19]:
```
# epochs 50-75
pointnet_model = GNN(num_classes = 2, num_layer = 2,num_post_fnn_layers=2,␣
 ↪input_node_dim=3,input_edge_dim = 1,
            gnn_type = 'pointnet', emb_dim = 300, drop_ratio = 0.3).to(device)
optimizer = optim.Adam(pointnet_model.parameters(), lr=1e-3)

train_model(pointnet_model,optimizer)
```

```
=====Epoch 58
Training…

Iteration: 100%|      | 794/794 [01:32<00:00,  8.63it/s]

Average training loss: 0.5710191993554233
Evaluating…
ROAUC scores:  {'Train': 0.7960894107818604, 'Validation': 0.7901884913444519}
=====Epoch 59
Training…
```

```
Iteration: 100%|     | 794/794 [02:51<00:00,  4.63it/s]

Average training loss: 0.5708773756582731
Evaluating…
ROAUC scores:  {'Train': 0.7905668020248413, 'Validation': 0.7861651182174683}
=====Epoch 60
Training…

Iteration: 100%|     | 794/794 [03:20<00:00,  3.96it/s]

Average training loss: 0.5698322208236988
Evaluating…
ROAUC scores:  {'Train': 0.7800576686859131, 'Validation': 0.7751147747039795}
=====Epoch 61
Training…

Iteration: 100%|     | 794/794 [03:19<00:00,  3.99it/s]

Average training loss: 0.5710302815404287
Evaluating…
ROAUC scores:  {'Train': 0.7954208254814148, 'Validation': 0.790581226348877}
=====Epoch 62
Training…

Iteration: 100%|     | 794/794 [03:19<00:00,  3.97it/s]

Average training loss: 0.5712567357257271
Evaluating…
ROAUC scores:  {'Train': 0.7953957319259644, 'Validation': 0.7892595529556274}
=====Epoch 63
Training…

Iteration: 100%|     | 794/794 [03:22<00:00,  3.93it/s]

Average training loss: 0.5708156433003375
Evaluating…
ROAUC scores:  {'Train': 0.7958442568778992, 'Validation': 0.7875226140022278}
=====Epoch 64
Training…

Iteration: 100%|     | 794/794 [03:18<00:00,  4.01it/s]

Average training loss: 0.5698565028101131
Evaluating…
ROAUC scores:  {'Train': 0.7940566539764404, 'Validation': 0.7897814512252808}
=====Epoch 65
Training…

Iteration: 100%|     | 794/794 [03:19<00:00,  3.97it/s]

Average training loss: 0.5705780312231266
Evaluating…
ROAUC scores:  {'Train': 0.796878457069397, 'Validation': 0.7902677059173584}
```

```
=====Epoch 66
Training…

Iteration: 100%|      | 794/794 [03:18<00:00,  4.01it/s]

Average training loss: 0.5694479918675098
Evaluating…
ROAUC scores:  {'Train': 0.7934308052062988, 'Validation': 0.7879504561424255}
=====Epoch 67
Training…

Iteration: 100%|      | 794/794 [03:17<00:00,  4.02it/s]

Average training loss: 0.56918925542375
Evaluating…
ROAUC scores:  {'Train': 0.7101675271987915, 'Validation': 0.7067291736602783}
=====Epoch 68
Training…

Iteration: 100%|      | 794/794 [03:18<00:00,  3.99it/s]

Average training loss: 0.5690981335468797
Evaluating…
ROAUC scores:  {'Train': 0.7929885387420654, 'Validation': 0.7869628667831421}
=====Epoch 69
Training…

Iteration: 100%|      | 794/794 [03:28<00:00,  3.81it/s]

Average training loss: 0.569651496913331
Evaluating…
ROAUC scores:  {'Train': 0.7912898063659668, 'Validation': 0.7845079898834229}
=====Epoch 70
Training…

Iteration: 100%|      | 794/794 [03:22<00:00,  3.92it/s]

Average training loss: 0.5689726767672099
Evaluating…
ROAUC scores:  {'Train': 0.7971503734588623, 'Validation': 0.7909290194511414}
=====Epoch 71
Training…

Iteration: 100%|      | 794/794 [03:22<00:00,  3.93it/s]

Average training loss: 0.5689667861542713
Evaluating…
ROAUC scores:  {'Train': 0.7955853939056396, 'Validation': 0.7889811992645264}
=====Epoch 72
Training…

Iteration: 100%|      | 794/794 [03:20<00:00,  3.96it/s]

Average training loss: 0.5694600624086275
Evaluating…
```

```
ROAUC scores:  {'Train': 0.7941993474960327, 'Validation': 0.7878971099853516}
=====Epoch 73
Training…

Iteration: 100%|      | 794/794 [03:16<00:00,  4.04it/s]

Average training loss: 0.568854150191062
Evaluating…
ROAUC scores:  {'Train': 0.7977845072746277, 'Validation': 0.7904917001724243}
=====Epoch 74
Training…

Iteration: 100%|      | 794/794 [03:22<00:00,  3.92it/s]

Average training loss: 0.56788597794564
Evaluating…
ROAUC scores:  {'Train': 0.7942689657211304, 'Validation': 0.7867396473884583}
=====Epoch 75
Training…

Iteration: 100%|      | 794/794 [03:23<00:00,  3.91it/s]

Average training loss: 0.5683751221492849
Evaluating…
ROAUC scores:  {'Train': 0.7932374477386475, 'Validation': 0.785496175289154}

Finished training!

ROAUC Test score: 0.7726539969444275
```

[19]:
```
([], [])
```

Training of GCN based model

[20]:
```
gcn_model = GNN(num_classes = 2, num_layer =␣
 ↪2,num_post_fnn_layers=2,hasPos=False, input_node_dim=3,input_edge_dim = 1,
            gnn_type = 'gcn', emb_dim = 300, drop_ratio = 0.3).to(device)
optimizer = optim.Adam(gcn_model.parameters(), lr=1e-3)

train_model(gcn_model,optimizer)
```

```
=====Epoch 43
Training…

Iteration: 100%|      | 794/794 [00:39<00:00, 20.08it/s]

Average training loss: 0.5746043768877948
Evaluating…
ROAUC scores:  {'Train': 0.7906577587127686, 'Validation': 0.7882513999938965}
=====Epoch 44
Training…

Iteration: 100%|      | 794/794 [00:43<00:00, 18.38it/s]
```

```
Average training loss: 0.5759014831081746
Evaluating…
ROAUC scores:  {'Train': 0.7816154956817627, 'Validation': 0.7795676589012146}
=====Epoch 45
Training…

Iteration: 100%|      | 794/794 [00:43<00:00, 18.12it/s]

Average training loss: 0.5736135304799909
Evaluating…
ROAUC scores:  {'Train': 0.791023850440979, 'Validation': 0.7881148457527161}
=====Epoch 46
Training…

Iteration: 100%|      | 794/794 [00:38<00:00, 20.39it/s]

Average training loss: 0.5759049139410183
Evaluating…
ROAUC scores:  {'Train': 0.7914109826087952, 'Validation': 0.7874666452407837}
=====Epoch 47
Training…

Iteration: 100%|      | 794/794 [00:39<00:00, 19.97it/s]

Average training loss: 0.5750361867155176
Evaluating…
ROAUC scores:  {'Train': 0.7898668050765991, 'Validation': 0.7889251708984375}
=====Epoch 48
Training…

Iteration: 100%|      | 794/794 [00:39<00:00, 19.89it/s]

Average training loss: 0.5750056571639155
Evaluating…
ROAUC scores:  {'Train': 0.7915517687797546, 'Validation': 0.7874159812927246}
=====Epoch 49
Training…

Iteration: 100%|      | 794/794 [00:43<00:00, 18.38it/s]

Average training loss: 0.5752063297880087
Evaluating…
ROAUC scores:  {'Train': 0.7891422510147095, 'Validation': 0.78626549243927}
=====Epoch 50
Training…

Iteration: 100%|      | 794/794 [00:41<00:00, 18.99it/s]

Average training loss: 0.5739271611125403
Evaluating…
ROAUC scores:  {'Train': 0.7896944284439087, 'Validation': 0.7870839834213257}
=====Epoch 51
Training…
```

```
Iteration: 100%|      | 794/794 [00:42<00:00, 18.76it/s]

Average training loss: 0.5742622155236357
Evaluating…
ROAUC scores:  {'Train': 0.7914423942565918, 'Validation': 0.78886878490448}
=====Epoch 52
Training…

Iteration: 100%|      | 794/794 [00:40<00:00, 19.81it/s]

Average training loss: 0.5745668791478467
Evaluating…
ROAUC scores:  {'Train': 0.7913110256195068, 'Validation': 0.7879801988601685}
=====Epoch 53
Training…

Iteration: 100%|      | 794/794 [00:42<00:00, 18.47it/s]

Average training loss: 0.5746284820120641
Evaluating…
ROAUC scores:  {'Train': 0.7903238534927368, 'Validation': 0.7874116897583008}
=====Epoch 54
Training…

Iteration: 100%|      | 794/794 [00:41<00:00, 18.97it/s]

Average training loss: 0.5747975259268614
Evaluating…
ROAUC scores:  {'Train': 0.7913990020751953, 'Validation': 0.7879424095153809}
=====Epoch 55
Training…

Iteration: 100%|      | 794/794 [00:38<00:00, 20.75it/s]

Average training loss: 0.5740465172367072
Evaluating…
ROAUC scores:  {'Train': 0.7895991206169128, 'Validation': 0.7875176668167114}
=====Epoch 56
Training…

Iteration: 100%|      | 794/794 [00:41<00:00, 18.98it/s]

Average training loss: 0.5753059338412296
Evaluating…
ROAUC scores:  {'Train': 0.7914857864379883, 'Validation': 0.7878527641296387}
=====Epoch 57
Training…

Iteration: 100%|      | 794/794 [00:42<00:00, 18.74it/s]

Average training loss: 0.5752448492146259
Evaluating…
ROAUC scores:  {'Train': 0.791583776473999, 'Validation': 0.7883838415145874}
```

```
=====Epoch 58
Training…

Iteration: 100%|      | 794/794 [00:42<00:00, 18.73it/s]

Average training loss: 0.5732090090324056
Evaluating…
ROAUC scores:  {'Train': 0.7915147542953491, 'Validation': 0.7868059873580933}
=====Epoch 59
Training…

Iteration: 100%|      | 794/794 [00:41<00:00, 19.11it/s]

Average training loss: 0.5734008358467736
Evaluating…
ROAUC scores:  {'Train': 0.7920928001403809, 'Validation': 0.7882958054542542}
=====Epoch 60
Training…

Iteration: 100%|      | 794/794 [00:40<00:00, 19.48it/s]

Average training loss: 0.5740976874113684
Evaluating…
ROAUC scores:  {'Train': 0.7914626598358154, 'Validation': 0.7887256741523743}
=====Epoch 61
Training…

Iteration: 100%|      | 794/794 [00:43<00:00, 18.41it/s]

Average training loss: 0.5745489175809121
Evaluating…
ROAUC scores:  {'Train': 0.7921646237373352, 'Validation': 0.7878116369247437}
=====Epoch 62
Training…

Iteration: 100%|      | 794/794 [00:42<00:00, 18.64it/s]

Average training loss: 0.5731349739079511
Evaluating…
ROAUC scores:  {'Train': 0.7922171950340271, 'Validation': 0.7889771461486816}
=====Epoch 63
Training…

Iteration: 100%|      | 794/794 [00:41<00:00, 19.12it/s]

Average training loss: 0.5755593575518437
Evaluating…
ROAUC scores:  {'Train': 0.7922916412353516, 'Validation': 0.7879410982131958}
=====Epoch 64
Training…

Iteration: 100%|      | 794/794 [00:41<00:00, 19.12it/s]

Average training loss: 0.5731632108201908
Evaluating…
```

```
ROAUC scores:  {'Train': 0.7776795625686646, 'Validation': 0.7783198356628418}
=====Epoch 65
Training…

Iteration: 100%|      | 794/794 [00:41<00:00, 19.05it/s]

Average training loss: 0.5732159691133187
Evaluating…
ROAUC scores:  {'Train': 0.7833820581436157, 'Validation': 0.7821496725082397}
=====Epoch 66
Training…

Iteration: 100%|      | 794/794 [00:41<00:00, 18.95it/s]

Average training loss: 0.5734541138038227
Evaluating…
ROAUC scores:  {'Train': 0.792068600654602, 'Validation': 0.7872563600540161}
=====Epoch 67
Training…

Iteration: 100%|      | 794/794 [00:38<00:00, 20.46it/s]

Average training loss: 0.5739356318784301
Evaluating…
ROAUC scores:  {'Train': 0.7928174734115601, 'Validation': 0.7876052856445312}
=====Epoch 68
Training…

Iteration: 100%|      | 794/794 [00:41<00:00, 19.19it/s]

Average training loss: 0.5741601205367586
Evaluating…
ROAUC scores:  {'Train': 0.7923808097839355, 'Validation': 0.7879139184951782}
=====Epoch 69
Training…

Iteration: 100%|      | 794/794 [00:42<00:00, 18.55it/s]

Average training loss: 0.5737447999511618
Evaluating…
ROAUC scores:  {'Train': 0.7813278436660767, 'Validation': 0.7796964645385742}
=====Epoch 70
Training…

Iteration: 100%|      | 794/794 [00:40<00:00, 19.78it/s]

Average training loss: 0.5729288821781913
Evaluating…
ROAUC scores:  {'Train': 0.7878342270851135, 'Validation': 0.7850787043571472}
=====Epoch 71
Training…

Iteration: 100%|      | 794/794 [00:42<00:00, 18.63it/s]
```

```
Average training loss: 0.5735772976872302
Evaluating…
ROAUC scores:  {'Train': 0.7923682928085327, 'Validation': 0.7877758741378784}
=====Epoch 72
Training…

Iteration: 100%|      | 794/794 [00:42<00:00, 18.61it/s]

Average training loss: 0.5731909055478627
Evaluating…
ROAUC scores:  {'Train': 0.7925848960876465, 'Validation': 0.7882707715034485}
=====Epoch 73
Training…

Iteration: 100%|      | 794/794 [00:42<00:00, 18.61it/s]

Average training loss: 0.5736414238773004
Evaluating…
ROAUC scores:  {'Train': 0.792793869972229, 'Validation': 0.7875902652740479}
=====Epoch 74
Training…

Iteration: 100%|      | 794/794 [00:42<00:00, 18.79it/s]

Average training loss: 0.5728275962830791
Evaluating…
ROAUC scores:  {'Train': 0.783450722694397, 'Validation': 0.7826626300811768}
=====Epoch 75
Training…

Iteration: 100%|      | 794/794 [00:42<00:00, 18.77it/s]

Average training loss: 0.5726831875159698
Evaluating…
ROAUC scores:  {'Train': 0.7916743755340576, 'Validation': 0.7868238091468811}

Finished training!

ROAUC Test score: 0.7773752808570862
```

[20]:
```
([], [])
```

[28]:
```
gcn_model = GNN(num_classes = 2, num_layer =␣
 ↪2,num_post_fnn_layers=2,hasPos=True, input_node_dim=6,input_edge_dim = 1,
            gnn_type = 'gcn', emb_dim = 300, drop_ratio = 0.3).to(device)
optimizer = optim.Adam(gcn_model.parameters(), lr=1e-3)

train_model(gcn_model,optimizer)
```

```
=====Epoch 51
Training…
```

```
Iteration: 100%|      | 794/794 [00:26<00:00, 29.49it/s]

Average training loss: 0.5772895867788521
Evaluating…
ROAUC scores:  {'Train': 0.7902165651321411, 'Validation': 0.7910354137420654}
=====Epoch 52
Training…

Iteration: 100%|      | 794/794 [00:29<00:00, 27.33it/s]

Average training loss: 0.5770404302803636
Evaluating…
ROAUC scores:  {'Train': 0.7889108657836914, 'Validation': 0.7880456447601318}
=====Epoch 53
Training…

Iteration: 100%|      | 794/794 [00:29<00:00, 26.77it/s]

Average training loss: 0.5769777745008469
Evaluating…
ROAUC scores:  {'Train': 0.7869840860366821, 'Validation': 0.7861782312393188}
=====Epoch 54
Training…

Iteration: 100%|      | 794/794 [00:29<00:00, 26.64it/s]

Average training loss: 0.5756530744077577
Evaluating…
ROAUC scores:  {'Train': 0.7900853753089905, 'Validation': 0.7892134189605713}
=====Epoch 55
Training…

Iteration: 100%|      | 794/794 [00:29<00:00, 26.65it/s]

Average training loss: 0.5754305484613184
Evaluating…
ROAUC scores:  {'Train': 0.7903851270675659, 'Validation': 0.7889273166656494}
=====Epoch 56
Training…

Iteration: 100%|      | 794/794 [00:34<00:00, 23.07it/s]

Average training loss: 0.5763846651582935
Evaluating…
ROAUC scores:  {'Train': 0.7909694910049438, 'Validation': 0.7873558402061462}
=====Epoch 57
Training…

Iteration: 100%|      | 794/794 [00:35<00:00, 22.15it/s]

Average training loss: 0.5776246293290737
Evaluating…
ROAUC scores:  {'Train': 0.7902017831802368, 'Validation': 0.7877715826034546}
```

```
=====Epoch 58
Training…

Iteration: 100%|      | 794/794 [00:34<00:00, 22.94it/s]

Average training loss: 0.577743453440498
Evaluating…
ROAUC scores:  {'Train': 0.7895077466964722, 'Validation': 0.786736011505127}
=====Epoch 59
Training…

Iteration: 100%|      | 794/794 [00:34<00:00, 22.86it/s]

Average training loss: 0.576131787348154
Evaluating…
ROAUC scores:  {'Train': 0.790465772151947, 'Validation': 0.7894949913024902}
=====Epoch 60
Training…

Iteration: 100%|      | 794/794 [00:34<00:00, 22.73it/s]

Average training loss: 0.5767226024373653
Evaluating…
ROAUC scores:  {'Train': 0.7838032245635986, 'Validation': 0.7847281694412231}
=====Epoch 61
Training…

Iteration: 100%|      | 794/794 [00:34<00:00, 22.81it/s]

Average training loss: 0.575934233608414
Evaluating…
ROAUC scores:  {'Train': 0.7910096645355225, 'Validation': 0.789182722568512}
=====Epoch 62
Training…

Iteration: 100%|      | 794/794 [00:39<00:00, 20.13it/s]

Average training loss: 0.5767635426082899
Evaluating…
ROAUC scores:  {'Train': 0.7783926129341125, 'Validation': 0.7763255834579468}
=====Epoch 63
Training…

Iteration: 100%|      | 794/794 [00:37<00:00, 21.06it/s]

Average training loss: 0.5754161258638656
Evaluating…
ROAUC scores:  {'Train': 0.7910293340682983, 'Validation': 0.7879165410995483}
=====Epoch 64
Training…

Iteration: 100%|      | 794/794 [00:32<00:00, 24.26it/s]

Average training loss: 0.5751496818789307
Evaluating…
```

```
ROAUC scores:  {'Train': 0.7920236587524414, 'Validation': 0.7917516231536865}
=====Epoch 65
Training…

Iteration: 100%|      | 794/794 [00:31<00:00, 25.11it/s]

Average training loss: 0.5760714168887895
Evaluating…
ROAUC scores:  {'Train': 0.7902299165725708, 'Validation': 0.7889820337295532}
=====Epoch 66
Training…

Iteration: 100%|      | 794/794 [00:30<00:00, 26.07it/s]

Average training loss: 0.5749656383877136
Evaluating…
ROAUC scores:  {'Train': 0.7914806604385376, 'Validation': 0.7903842926025391}
=====Epoch 67
Training…

Iteration: 100%|      | 794/794 [00:30<00:00, 26.32it/s]

Average training loss: 0.57651697406991
Evaluating…
ROAUC scores:  {'Train': 0.7914169430732727, 'Validation': 0.7912519574165344}
=====Epoch 68
Training…

Iteration: 100%|      | 794/794 [00:30<00:00, 26.38it/s]

Average training loss: 0.574033105215738
Evaluating…
ROAUC scores:  {'Train': 0.7872203588485718, 'Validation': 0.7851631045341492}
=====Epoch 69
Training…

Iteration: 100%|      | 794/794 [00:30<00:00, 26.34it/s]

Average training loss: 0.5748955247083899
Evaluating…
ROAUC scores:  {'Train': 0.7918460965156555, 'Validation': 0.7897984981536865}
=====Epoch 70
Training…

Iteration: 100%|      | 794/794 [00:30<00:00, 26.43it/s]

Average training loss: 0.5746962269697742
Evaluating…
ROAUC scores:  {'Train': 0.7917401790618896, 'Validation': 0.7904717326164246}
=====Epoch 71
Training…

Iteration: 100%|      | 794/794 [00:29<00:00, 26.63it/s]
```

```
Average training loss: 0.5746052057226599
Evaluating…
ROAUC scores:  {'Train': 0.7912579774856567, 'Validation': 0.789975643157959}
=====Epoch 72
Training…

Iteration: 100%|      | 794/794 [00:29<00:00, 26.69it/s]

Average training loss: 0.5770513055381606
Evaluating…
ROAUC scores:  {'Train': 0.7912329435348511, 'Validation': 0.7897987365722656}
=====Epoch 73
Training…

Iteration: 100%|      | 794/794 [00:30<00:00, 26.32it/s]

Average training loss: 0.5754396517946377
Evaluating…
ROAUC scores:  {'Train': 0.790923535823822, 'Validation': 0.7898480296134949}
=====Epoch 74
Training…

Iteration: 100%|      | 794/794 [00:29<00:00, 26.65it/s]

Average training loss: 0.5748563607408658
Evaluating…
ROAUC scores:  {'Train': 0.7889094352722168, 'Validation': 0.7857871651649475}
=====Epoch 75
Training…

Iteration: 100%|      | 794/794 [00:29<00:00, 26.73it/s]

Average training loss: 0.5754139746166297
Evaluating…
ROAUC scores:  {'Train': 0.7905126810073853, 'Validation': 0.7884494066238403}

Finished training!

ROAUC Test score: 0.7769407033920288
```

[28]: ([], [])