task-2

March 14, 2023

[1]:

```python
import torch
import numpy as np
import pandas as pd
from tqdm import tqdm
import os
import h5py
import math
import pyarrow.parquet as pq
import torch.nn as nn
import torch.nn.functional as F
from torch.nn import init
from torch.utils.data import Dataset, random_split, DataLoader
from torchvision import transforms
import torch.optim as optim
from torchmetrics.classification import MulticlassAUROC, MulticlassAccuracy
```

[2]:

```python
# clearing cuda cache memory
import gc
torch.cuda.empty_cache()
gc.collect()
```

[2]:

```
0
```

[3]:

```python
os.listdir("../dataset")
```

[3]:

```
['QCDToGGQQ_IMGjet_RH1all_jet0_run0_n36272',
 'QCDToGGQQ_IMGjet_RH1all_jet0_run0_n36272.test.snappy.parquet',
 'QCDToGGQQ_IMGjet_RH1all_jet0_run1_n47540',
 'QCDToGGQQ_IMGjet_RH1all_jet0_run1_n47540.test.snappy.parquet',
 'QCDToGGQQ_IMGjet_RH1all_jet0_run2_n55494',
 'QCDToGGQQ_IMGjet_RH1all_jet0_run2_n55494.test.snappy.parquet',
 'SingleElectronPt50_IMGCROPS_n249k_RHv1.hdf5',
 'SinglePhotonPt50_IMGCROPS_n249k_RHv1.hdf5']
```

[4]:

```python
def save_ckpt(imgs,processed_dir,count):
        print("saving...")
        torch.save(imgs,f"{processed_dir}/images-jets{count}-processed.pt")
```

[5]:
```
def read_image_data(dataset_name,count="",start_split=0):
    raw_path = f"../dataset/{dataset_name}/raw/{dataset_name}.test.snappy.
 ↪parquet"
    processed_dir = f"../dataset/{dataset_name}/processed"
    imgs = None
    labels = None
    if f"images-jets{count}-processed.pt" in os.listdir(processed_dir):
        print("loading...")
        imgs = torch.load(f"{processed_dir}/images-jets{count}-processed.pt")
        # load all the label
        # this function returns all the labels
        # hence need truncate if needed seperately.
        labels = torch.load(f"{processed_dir}/labels-jets-processed.pt")
    else:
        dataset = pq.read_table(raw_path,columns=["X_jets","y"]).to_pandas()
        images_raw = dataset["X_jets"].to_numpy()[start_split:]
        labels = dataset["y"][start_split:].to_numpy().astype(np.int64)
        labels = torch.Tensor(labels).to(torch.int32)
        imgs = np.empty([0,125,125,3],dtype=np.float32)
        for inx,img in enumerate(tqdm(images_raw)):
            inx_ = inx+start_split
            img_np = np.stack([np.stack(channel) for channel in img])
            # change the shape to (125,125,3)
            img_np = img_np.transpose()
            imgs = np.vstack((imgs,np.expand_dims(img_np,axis=0)))
            if inx>0 and inx%9068==0:
                imgs = torch.Tensor(imgs)
                save_ckpt(imgs,processed_dir,f"-{str(inx_)}")
        imgs = torch.Tensor(imgs)
        save_ckpt(imgs,labels,processed_dir,"")

    return imgs,labels
```

[6]:
```
# truncated dataset => uses 25%

img_arrs, labels =␣
 ↪read_image_data("QCDToGGQQ_IMGjet_RH1all_jet0_run0_n36272","")
labels = labels[:img_arrs.shape[0]].to(torch.int64)  #truncating from full list␣
 ↪of labels
```

```
loading…
```

[7]:
```
class QuarkGluonDataset(Dataset):
    def __init__(self,split_inx, transform=None,target_transform= None):
        self.img_arrs_split = img_arrs[split_inx]
        self.labels_split = labels[split_inx]
        self.transform = transform
        self.target_transform = target_transform
    def __len__(self):
        return self.labels_split.shape[0]
    def __getitem__(self,idx):
        image=self.img_arrs_split[idx,:,:,:]
        # changing the dim of image to channels, height, width by transposing␣
 ↪the
        # original image tensor.
        image = image.permute(2,1,0)
        label = self.labels_split[idx]
        if self.transform:
            image = self.transform(image)
        if self.target_transform:
            label = self.target_transform(label)
        return image,label
```

[8]:

```
class SeparableConv2d(nn.Module):
    def␣
 ↪__init__(self,in_channels,out_channels,kernel_size=1,stride=1,padding=0,bias=False):
 ↪
        super(SeparableConv2d,self).__init__()

        self.conv1 = nn.
 ↪Conv2d(in_channels,in_channels,kernel_size,stride,padding,groups=in_channels,bias=bias)
        self.pointwise = nn.Conv2d(in_channels,out_channels,1,1,0,1,1,bias=bias)

    def forward(self,x):
        x = self.conv1(x)
        x = self.pointwise(x)
        return x


class Block(nn.Module):
    def␣
 ↪__init__(self,in_channels,out_channels,reps,strides=1,start_with_relu=True,expand_first=Tru
 ↪
        '''
        start_with_relu: if true start with relu
        expand_first: if True latent embedding dim of the block will be␣
 ↪expanded to out_channels
                      at the beginning  else latent dim will be expanded at the␣
 ↪end
        '''
        super(Block, self).__init__()

        if out_channels != in_channels or strides!=1:
```

```python
            self.skip = nn.Conv2d(in_channels,out_channels,1,stride=strides, bias=False)
            self.skipbn = nn.BatchNorm2d(out_channels)
        else:
            self.skip=None

        self.relu = nn.ReLU(inplace=True)
        rep=[]

        filters=in_channels
        if expand_first:
            rep.append(self.relu)
            rep.append(SeparableConv2d(in_channels,out_channels,3,stride=1,padding=1,bias=False))
            rep.append(nn.BatchNorm2d(out_channels))
            filters = out_channels

        for i in range(reps-1):
            rep.append(self.relu)
            rep.append(SeparableConv2d(filters,filters,3,stride=1,padding=1,bias=False))
            rep.append(nn.BatchNorm2d(filters))

        if not expand_first:
            rep.append(self.relu)
            rep.append(SeparableConv2d(in_channels,out_channels,3,stride=1,padding=1,bias=False))
            rep.append(nn.BatchNorm2d(out_channels))

        if not start_with_relu:
            rep = rep[1:]
        else:
            rep[0] = nn.ReLU(inplace=False)

        if strides != 1:
            rep.append(nn.MaxPool2d(3,strides,1))
        self.rep = nn.Sequential(*rep)

    def forward(self,inp):
        x = self.rep(inp)

        if self.skip is not None:
            skip = self.skip(inp)
            skip = self.skipbn(skip)
        else:
            skip = inp
        x+=skip
        return x



class Xception(nn.Module):
    """
    Xception model, as specified in
    https://arxiv.org/pdf/1610.02357.pdf
    """
    def __init__(self, num_classes=2):
        """ Constructor
        Args:
            num_classes: number of classes
        """
        super(Xception, self).__init__()


        self.num_classes = num_classes

        self.conv1 = nn.Conv2d(3, 32, 3,2, 0, bias=False)
        self.bn1 = nn.BatchNorm2d(32)
        self.relu = nn.ReLU(inplace=True)

        self.conv2 = nn.Conv2d(32,64,3,bias=False)
        self.bn2 = nn.BatchNorm2d(64)
        #do relu here

        self.block1=Block(64,128,2,2,start_with_relu=False,expand_first=True)
        self.block2=Block(128,256,2,2,start_with_relu=True,expand_first=True)
        self.block3=Block(256,728,2,2,start_with_relu=True,expand_first=True)

        self.block4=Block(728,728,3,1,start_with_relu=True,expand_first=True)
        self.block5=Block(728,728,3,1,start_with_relu=True,expand_first=True)
        self.block6=Block(728,728,3,1,start_with_relu=True,expand_first=True)
        self.block7=Block(728,728,3,1,start_with_relu=True,expand_first=True)
        self.block8=Block(728,728,3,1,start_with_relu=True,expand_first=True)
        self.block9=Block(728,728,3,1,start_with_relu=True,expand_first=True)
        self.block10=Block(728,728,3,1,start_with_relu=True,expand_first=True)
        self.block11=Block(728,728,3,1,start_with_relu=True,expand_first=True)

        self.block12=Block(728,1024,2,2,start_with_relu=True,expand_first=False)

        self.conv3 = SeparableConv2d(1024,1536,3,1,1)
        self.bn3 = nn.BatchNorm2d(1536)

        #do relu here
        self.conv4 = SeparableConv2d(1536,2048,3,1,1)
        self.bn4 = nn.BatchNorm2d(2048)

        self.fc = nn.Linear(2048, num_classes)

    def forward(self, x):
        x = self.conv1(x)
        x = self.bn1(x)
        x = self.relu(x)

        x = self.conv2(x)
        x = self.bn2(x)
        x = self.relu(x)

        x = self.block1(x)
        x = self.block2(x)
        x = self.block3(x)
        x = self.block4(x)
        x = self.block5(x)
        x = self.block6(x)
        x = self.block7(x)
        x = self.block8(x)
        x = self.block9(x)
        x = self.block10(x)
        x = self.block11(x)
        x = self.block12(x)

        x = self.conv3(x)
        x = self.bn3(x)
        x = self.relu(x)


        x = self.conv4(x)
        x = self.bn4(x)
        x = self.relu(x)


        x = F.adaptive_avg_pool2d(x, (1, 1))
        x = x.view(x.size(0), -1)
        x = self.fc(x)

        return F.softmax(x,dim=1)

    def __str__(self):
        return "Xception-task2"
```

[9]:

```
device = torch.device("cuda:0" if torch.cuda.is_available() else torch.
 ↪device("cpu"))
multicls_criterion = torch.nn.CrossEntropyLoss()
```

[10]:

```
model = Xception(num_classes=2).to(device)
optimizer = optim.Adam(model.parameters(), lr=1e-3)

epochs = 13
```

[11]:

```
preprocess = transforms.Compose([
    transforms.Normalize(mean=[0.5, 0.5,0.5], std=[0.5, 0.5,0.5]),
])

train_inx, valid_inx, test_inx = random_split(range(labels.shape[0]),[0.7,0.2,0.
 ↪1],generator=torch.Generator()
                                            .manual_seed(42))

# train_inx, valid_inx, test_inx = random_split(range(labels.shape[0]),[0.005,0.
 ↪005,0.99],generator=torch.Generator()
#                                             .manual_seed(42))

train_data = QuarkGluonDataset(split_inx=train_inx,transform = preprocess)
valid_data = QuarkGluonDataset(split_inx=valid_inx,transform = preprocess)
test_data = QuarkGluonDataset(split_inx=test_inx,transform = preprocess)
# dataset = SingleElectronPhotonDataset()

train_dataloader = DataLoader(train_data,batch_size = 64, shuffle = True)
valid_dataloader = DataLoader(valid_data,batch_size = 64, shuffle = True)
test_dataloader = DataLoader(test_data,batch_size = 64, shuffle = True)
```

[12]:

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

    loss_accum = 0
    for step, batch in enumerate(tqdm(loader, desc="Iteration")):
        inputs, labels = batch
        inputs = inputs.to(device)
        labels = labels.to(device)
        output = model(inputs)
        loss= 0
        optimizer.zero_grad()
        loss += multicls_criterion(output, labels)
        loss.backward()
        optimizer.step()

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

[13]:

```
def evaluate(model, device, loader,evaluator= "roauc",isTqdm=False):
    model.eval()

    preds_list = []
    target_list = []
    iterator = enumerate(loader)
    if isTqdm:
        iterator = enumerate(tqdm(loader))
    for step, batch in iterator:
        inputs, labels = batch
        inputs = inputs.to(device)
        labels = labels.to(device)
        with torch.no_grad():
            output = model(inputs)
            preds_list.extend(output.tolist())
        target_list += batch[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()
```

[14]:

```
checkpoints_path = "../models"
checkpoints = os.listdir(checkpoints_path)
checkpoint_path = list(filter(lambda i : str(model) in i, checkpoints))
```

[15]:

```
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)
```

```python
    print("Saving model...")
    # 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)
    print('ROAUC scores: ',{'Train': train_perf_roauc, 'Validation': valid_perf_roauc})

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

```
=====Epoch 1
Training…

Iteration: 100%|     | 397/397 [02:55<00:00,  2.26it/s]

Average training loss: 0.5920324786484091
Saving model…
Evaluating…
ROAUC scores:  {'Train': 0.7894700169563293, 'Validation': 0.785323977470398}
=====Epoch 2
Training…

Iteration: 100%|     | 397/397 [03:25<00:00,  1.93it/s]

Average training loss: 0.5744664172831951
Saving model…
Evaluating…
ROAUC scores:  {'Train': 0.7900659441947937, 'Validation': 0.7892624139785767}
=====Epoch 3
Training…

Iteration: 100%|     | 397/397 [03:26<00:00,  1.92it/s]

Average training loss: 0.5720023585657028
Saving model…
Evaluating…
ROAUC scores:  {'Train': 0.7965438961982727, 'Validation': 0.7908732891082764}
=====Epoch 4
```

```
Training…

Iteration: 100%|     | 397/397 [03:27<00:00,  1.91it/s]

Average training loss: 0.5693950557738767
Saving model…
Evaluating…
ROAUC scores:  {'Train': 0.799289882183075, 'Validation': 0.7927624583244324}
=====Epoch 5
Training…

Iteration: 100%|     | 397/397 [03:28<00:00,  1.90it/s]

Average training loss: 0.5667623525452554
Saving model…
Evaluating…
ROAUC scores:  {'Train': 0.7954740524291992, 'Validation': 0.7896384000778198}
=====Epoch 6
Training…

Iteration: 100%|     | 397/397 [03:30<00:00,  1.89it/s]

Average training loss: 0.5650653058395578
Saving model…
Evaluating…
ROAUC scores:  {'Train': 0.8014592528343201, 'Validation': 0.791454017162323}
=====Epoch 7
Training…

Iteration: 100%|     | 397/397 [03:28<00:00,  1.91it/s]

Average training loss: 0.5621103882339199
Saving model…
Evaluating…
ROAUC scores:  {'Train': 0.8037586212158203, 'Validation': 0.7877793312072754}
=====Epoch 8
Training…

Iteration: 100%|     | 397/397 [03:25<00:00,  1.93it/s]

Average training loss: 0.5591421389309525
Saving model…
Evaluating…
ROAUC scores:  {'Train': 0.8118031620979309, 'Validation': 0.7920047044754028}
=====Epoch 9
Training…

Iteration: 100%|     | 397/397 [03:26<00:00,  1.92it/s]

Average training loss: 0.5569321213651364
Saving model…
Evaluating…
ROAUC scores:  {'Train': 0.8165521025657654, 'Validation': 0.7909567356109619}
```

```
=====Epoch 10
Training…

Iteration: 100%|      | 397/397 [03:23<00:00,  1.95it/s]

Average training loss: 0.551441257636553
Saving model…
Evaluating…
ROAUC scores:  {'Train': 0.8156135678291321, 'Validation': 0.7847069501876831}
=====Epoch 11
Training…

Iteration: 100%|      | 397/397 [03:23<00:00,  1.95it/s]

Average training loss: 0.5477477962184012
Saving model…
Evaluating…
ROAUC scores:  {'Train': 0.8186319470405579, 'Validation': 0.7774256467819214}
=====Epoch 12
Training…

Iteration: 100%|      | 397/397 [03:25<00:00,  1.93it/s]

Average training loss: 0.5427978107851158
Saving model…
Evaluating…
ROAUC scores:  {'Train': 0.8174840211868286, 'Validation': 0.7749207019805908}
=====Epoch 13
Training…

Iteration: 100%|      | 397/397 [03:35<00:00,  1.84it/s]

Average training loss: 0.5357407721073861
Saving model…
Evaluating…
ROAUC scores:  {'Train': 0.8252854347229004, 'Validation': 0.7775824069976807}

Finished training!

ROAUC Test score: 0.7694913148880005
```

[16]:

```
tot_dataloader = DataLoader(QuarkGluonDataset(split_inx=list(range(labels.
 ↪shape[0])),
                                                              transform =␣
 ↪preprocess))
print('\nROAUC Total score: {}'.
 ↪format(evaluate(model,device,tot_dataloader,isTqdm=True)))
```

```
100%|      | 36272/36272 [10:51<00:00, 55.69it/s]


ROAUC Total score: 0.8105034828186035
```