GAN

March 12, 2022

1 CE-40719: Deep Learning

1.1 HW5 - GAN (100 points)

Name:

Student No.:

1.1.1 1) Import Libraries

[ ]:
```
import numpy as np
import torch
import torch.nn as nn
import torch.nn.functional as F
import torch.optim as optim
from torchvision import datasets, transforms

import matplotlib.pyplot as plt
%matplotlib inline
plt.rcParams['figure.figsize'] = (10, 3) # set default size of plots
```

1.1.2 2) Loading Dataset (10 points)

In this notebook, you will use `MNIST` dataset to train your GAN. You can see more information about this dataset here. This dataset is a 10 class dataset. It contains 60000 grayscale images (50000 for train and 10000 for test or validation) each with shape (3, 28, 28). Every image has a corresponding label which is a number in range 0 to 9.

[ ]:
```
# MNIST Dataset
train_dataset = datasets.MNIST(root='./mnist/', train=True,␣
 ↪transform=transforms.ToTensor(), download=True)
test_dataset = datasets.MNIST(root='./mnist/', train=False,␣
 ↪transform=transforms.ToTensor(), download=True)
```

```
[ ]: # Device configuration
device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')

################ Problem 01 (5 pts) ################
# define hyper parameters
batch_size = None
d_lr = None
g_lr = None
n_epochs = None
######################## End ########################
z_dim = 100
```

```
[ ]: ################ Problem 02 (5 pts) ################
# Define Dataloaders
train_loader = None
test_loader = None
######################## End ########################
train_loader = torch.utils.data.DataLoader(dataset=train_dataset,␣
 ↪batch_size=batch_size, shuffle=True)
test_loader = torch.utils.data.DataLoader(dataset=test_dataset,␣
 ↪batch_size=batch_size, shuffle=False)
```

1.1.3 3) Defining Network (30 points)

At this stage, you should define a network that improves your GAN training and prevents problems such as mode collapse and vanishing gradients.

```
[ ]: class Discriminator(nn.Module):
    def __init__(self):
        super().__init__()

        self.discriminator = nn.Sequential(
            ################ Problem 03 (15 pts) ################
            # use linear or convolutional layer
            # use arbitrary techniques to stabilize training


            ######################## End ########################
        )

    def forward(self, x):
        return self.discriminator(x)


class Generator(nn.Module):
    def __init__(self):
```

```
        super().__init__()

        self.generator = nn.Sequential(
            ################ Problem 04 (15 pts) ################
            # use linear or convolutional layer
            # use arbitrary techniques to stabilize training


            ######################## End ########################
        )

    def forward(self, z):
        return self.generator(z)
```

1.1.4 4) Train the Network

At this step, you are going to train your network.

[ ]:
```
################ Problem 05 (5 pts) ################
# Create instances of modules (discriminator and generator)
# don't forget to put your models on device
discriminator = None
generator = None
######################## End ########################
```

[ ]:
```
################ Problem 06 (5 pts) ################
# Define two optimizer for discriminator and generator
d_optimizer = None
g_optimizer = None
######################## End ########################
```

[ ]:
```
plot_frequency = None

for epoch in range(n_epochs):
    for i, (images, labels) in enumerate(train_loader):

        ################ Problem 07 (15 pts) ################
        # put your inputs on device
        # Prepare what you need for training, like inputs for modules and␣
 ↪variables for computing loss

        z = None

        ######################## End ########################
```

```
        ################ Problem 08 (10 pts) ################
        # calculate discriminator loss and update it


        d_loss = None

        ######################## End ########################



        ################ Problem 09 (10 pts) ################
        # calculate generator loss and update it


        g_loss = None

        ######################## End ########################


    ################ Problem 10 (10 pts) ################
    # plot some of the generated pictures based on plot frequency variable

    if (epoch % plot_frequency == 0):
        pass

    ######################## End ########################

    print("epoch: {} \t discriminator last batch loss: {} \t generator last␣
↪batch loss: {}".format(epoch + 1,
                                                                            ␣
↪            d_loss.item(),
                                                                            ␣
↪            g_loss.item())
    )
```

1.1.5 5) Save Generator

Save your final generator parameters. Upload it with your other files.

[ ]:
```
################ Problem 11 (5 pts) ################
# save state dict of your generator

######################## End ########################
```