main

November 16, 2023

[ ]:
```
import argparse
import os
import random
import torch
import torch.nn as nn
import torch.nn.parallel
import torch.optim as optim
import torch.utils.data
import torchvision.datasets as dset
import torchvision.transforms as transforms
import torchvision.utils as vutils
import numpy as np
import matplotlib.pyplot as plt
import matplotlib.animation as animation
from IPython.core.display import HTML

# manualSeed = 999
# print(manualSeed)

# random.seed(manualSeed)
# torch.manual_seed(manualSeed)
# torch.use_deterministic_algorithms(True)
```

[ ]:
```
dataroot = "data/celeba"

wokers = 2

batch_size = 128

image_size = 64

# number of channels in the training images
nc = 3

# size of latent vector
nz = 100
```

```
# size of feature maps in generator
ngf = 64

# size of feature maps in discriminator
ndf = 64

num_epochs = 10

# learning rate for optimizers
lr = 0.0002

# beta1 hyperparameter for adam optimizers
beta1 = 0.5

ngpu = 1

dataset = dset.ImageFolder(
    root=dataroot,
    transform=transforms.Compose(
        [
            transforms.Resize(image_size),
            transforms.CenterCrop(image_size),
            transforms.ToTensor(),
            transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5)),
        ],
    ),
)

dataloader = torch.utils.data.DataLoader(
    dataset, batch_size=batch_size, shuffle=True, num_workers=wokers
)

device = torch.device("cuda:0" if (torch.cuda.is_available() and ngpu > 0) else␣
 ↪"cpu")

real_batch = next(iter(dataloader))
plt.figure(figsize=(8, 8))
plt.axis("off")
plt.title("Training Images")
plt.imshow(
    np.transpose(
        vutils.make_grid(
            real_batch[0].to(device)[:64], padding=2, normalize=True
        ).cpu(),
        (1, 2, 0),
    )
)
```

[ ]: <matplotlib.image.AxesImage at 0x7f9069bd57d0>

[ ]:
```python
def weight_init(m):
    classname = m.__class__.__name__

    if classname.find("Conv") != -1:
        nn.init.normal_(m.weight.data, 0.0, 0.02)
    elif classname.find("BatchNorm") != -1:
        nn.init.normal_(m.weight.data, 1.0, 0.02)
        nn.init.constant_(m.bias.data, 0)
```

[ ]:
```
class Generator(nn.Module):
    def __init__(self, ngpu):
        super(Generator, self).__init__()
        self.ngpu = ngpu
        self.main = nn.Sequential(
            nn.ConvTranspose2d(nz, ngf * 8, 4, 1, 0, bias=False),
            nn.BatchNorm2d(ngf * 8),
            nn.ReLU(True),
            nn.ConvTranspose2d(ngf * 8, ngf * 4, 4, 2, 1, bias=False),
            nn.BatchNorm2d(ngf * 4),
            nn.ReLU(True),
            nn.ConvTranspose2d(ngf * 4, ngf * 2, 4, 2, 1, bias=False),
            nn.BatchNorm2d(ngf * 2),
            nn.ReLU(True),
            nn.ConvTranspose2d(ngf * 2, ngf, 4, 2, 1, bias=False),
            nn.BatchNorm2d(ngf),
            nn.ReLU(True),
            nn.ConvTranspose2d(ngf, nc, 4, 2, 1, bias=False),
            nn.Tanh(),
        )

    def forward(self, input):
        return self.main(input)
```

[ ]:
```
netG = Generator(ngpu).to(device)

if (device.type == "cuda") and (ngpu > 1):
    netG = nn.DataParallel(netG, list(range(ngpu)))

netG.apply(weight_init)
print(netG)
```

```
Generator(
  (main): Sequential(
    (0): ConvTranspose2d(100, 512, kernel_size=(4, 4), stride=(1, 1),
bias=False)
    (1): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    (2): ReLU(inplace=True)
    (3): ConvTranspose2d(512, 256, kernel_size=(4, 4), stride=(2, 2),
padding=(1, 1), bias=False)
    (4): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    (5): ReLU(inplace=True)
    (6): ConvTranspose2d(256, 128, kernel_size=(4, 4), stride=(2, 2),
padding=(1, 1), bias=False)
    (7): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
```

```
track_running_stats=True)
    (8): ReLU(inplace=True)
    (9): ConvTranspose2d(128, 64, kernel_size=(4, 4), stride=(2, 2), padding=(1,
1), bias=False)
    (10): BatchNorm2d(64, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    (11): ReLU(inplace=True)
    (12): ConvTranspose2d(64, 3, kernel_size=(4, 4), stride=(2, 2), padding=(1,
1), bias=False)
    (13): Tanh()
  )
)
```

[ ]:
```python
class Discriminator(nn.Module):
    def __init__(self, ngpu):
        super(Discriminator, self).__init__()
        self.ngpu = ngpu
        self.main = nn.Sequential(
            nn.Conv2d(nc, ndf, 4, 2, 1, bias=False),
            nn.LeakyReLU(0.2, inplace=True),
            nn.Conv2d(ndf, ndf * 2, 4, 2, 1, bias=False),
            nn.BatchNorm2d(ndf * 2),
            nn.LeakyReLU(0.2, inplace=True),
            nn.Conv2d(ndf * 2, ndf * 4, 4, 2, 1, bias=False),
            nn.BatchNorm2d(ndf * 4),
            nn.LeakyReLU(0.2, inplace=True),
            nn.Conv2d(ndf * 4, ndf * 8, 4, 2, 1, bias=False),
            nn.BatchNorm2d(ndf * 8),
            nn.LeakyReLU(0.2, inplace=True),
            nn.Conv2d(ndf * 8, 1, 4, 1, 0, bias=False),
            nn.Sigmoid(),
        )

    def forward(self, input):
        return self.main(input)
```

[ ]:
```python
netD = Discriminator(ngpu).to(device)

if (device.type == "cuda") and (ngpu > 1):
    netD = nn.DataParallel(netG, list(range(ngpu)))

netD.apply(weight_init)
print(netD)
```

```
Discriminator(
  (main): Sequential(
    (0): Conv2d(3, 64, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1),
bias=False)
```

```
    (1): LeakyReLU(negative_slope=0.2, inplace=True)
    (2): Conv2d(64, 128, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1),
bias=False)
    (3): BatchNorm2d(128, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    (4): LeakyReLU(negative_slope=0.2, inplace=True)
    (5): Conv2d(128, 256, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1),
bias=False)
    (6): BatchNorm2d(256, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    (7): LeakyReLU(negative_slope=0.2, inplace=True)
    (8): Conv2d(256, 512, kernel_size=(4, 4), stride=(2, 2), padding=(1, 1),
bias=False)
    (9): BatchNorm2d(512, eps=1e-05, momentum=0.1, affine=True,
track_running_stats=True)
    (10): LeakyReLU(negative_slope=0.2, inplace=True)
    (11): Conv2d(512, 1, kernel_size=(4, 4), stride=(1, 1), bias=False)
    (12): Sigmoid()
  )
)
```

[ ]:
```
criterion = nn.BCELoss()

fixed_noise = torch.randn(64, nz, 1, 1, device=device)

real_label = 1
fake_label = 0

optimizerD = optim.Adam(netD.parameters(), lr=lr, betas=(beta1, 0.999))
optimizerG = optim.Adam(netG.parameters(), lr=lr, betas=(beta1, 0.999))
```

[ ]:
```
img_list = []
G_losses = []
D_losses = []
iters = 0

print("Starting the Training Loop...")

for epoch in range(num_epochs):
    for i, data in enumerate(dataloader, 0):
        netD.zero_grad()

        real_cpu = data[0].to(device)
        b_size = real_cpu.size(0)
        label = torch.full((b_size,), real_label, dtype=torch.float,␣
 ↪device=device)
        output = netD(real_cpu).view(-1)
```

```
        errD_real = criterion(output, label)
        errD_real.backward()
        D_x = output.mean().item()

        noise = torch.randn(b_size, nz, 1, 1, device=device)
        fake = netG(noise)
        label.fill_(fake_label)
        output = netD(fake.detach()).view(-1)
        errD_fake = criterion(output, label)
        errD_fake.backward()
        D_G_z1 = output.mean().item()
        errD = errD_real + errD_fake
        optimizerD.step()

        netG.zero_grad()
        label.fill_(real_label)
        output = netD(fake).view(-1)
        errG = criterion(output, label)
        errG.backward()
        D_G_z2 = output.mean().item()
        optimizerG.step()

        if i % 50 == 0:
            print(
                "[%d/%d][%d/%d]\tLoss_D: %.4f\tLoss_G: %.4f\tD(x): %.
↪4f\tD(G(z)): %.4f / %.4f"
                % (
                    epoch,
                    num_epochs,
                    i,
                    len(dataloader),
                    errD.item(),
                    errG.item(),
                    D_x,
                    D_G_z1,
                    D_G_z2,
                )
            )

        G_losses.append(errG.item())
        D_losses.append(errD.item())

        if (iters % 500 == 0) or (
            (epoch == num_epochs - 1) and (i == len(dataloader) - 1)
        ):
            with torch.no_grad():
                fake = netG(fixed_noise).detach().cpu()
```

```
            img_list.append(vutils.make_grid(fake, padding=2, normalize=True))

        iters += 1
```

```
Starting the Training Loop…
[0/10][0/1583]  Loss_D: 1.3002  Loss_G: 5.3664  D(x): 0.5809    D(G(z)): 0.4150
/ 0.0070
[0/10][50/1583] Loss_D: 1.5591  Loss_G: 26.5058 D(x): 0.9450    D(G(z)): 0.6183
/ 0.0000
[0/10][100/1583]        Loss_D: 0.3188  Loss_G: 8.5724  D(x): 0.9103    D(G(z)):
0.1018 / 0.0007
[0/10][150/1583]        Loss_D: 0.3664  Loss_G: 5.6670  D(x): 0.9310    D(G(z)):
0.2113 / 0.0079
[0/10][200/1583]        Loss_D: 0.4801  Loss_G: 3.6094  D(x): 0.9280    D(G(z)):
0.2579 / 0.0622
[0/10][250/1583]        Loss_D: 0.4612  Loss_G: 9.7941  D(x): 0.7047    D(G(z)):
0.0012 / 0.0001
[0/10][300/1583]        Loss_D: 0.3894  Loss_G: 3.2959  D(x): 0.8711    D(G(z)):
0.1731 / 0.0783
[0/10][350/1583]        Loss_D: 0.9247  Loss_G: 6.2657  D(x): 0.9532    D(G(z)):
0.5191 / 0.0070
[0/10][400/1583]        Loss_D: 1.1560  Loss_G: 7.1604  D(x): 0.8436    D(G(z)):
0.5491 / 0.0041
[0/10][450/1583]        Loss_D: 0.7842  Loss_G: 5.7899  D(x): 0.8818    D(G(z)):
0.3979 / 0.0058
[0/10][500/1583]        Loss_D: 2.6364  Loss_G: 2.3789  D(x): 0.1900    D(G(z)):
0.0011 / 0.1652
[0/10][550/1583]        Loss_D: 0.6264  Loss_G: 6.6204  D(x): 0.8711    D(G(z)):
0.3058 / 0.0027
[0/10][600/1583]        Loss_D: 0.2387  Loss_G: 3.2933  D(x): 0.8846    D(G(z)):
0.0768 / 0.0612
[0/10][650/1583]        Loss_D: 0.6896  Loss_G: 3.7308  D(x): 0.6202    D(G(z)):
0.0377 / 0.0514
[0/10][700/1583]        Loss_D: 0.4637  Loss_G: 5.6521  D(x): 0.8823    D(G(z)):
0.2482 / 0.0063
[0/10][750/1583]        Loss_D: 0.6291  Loss_G: 5.0663  D(x): 0.8087    D(G(z)):
0.2580 / 0.0161
[0/10][800/1583]        Loss_D: 0.5443  Loss_G: 5.0054  D(x): 0.9456    D(G(z)):
0.3308 / 0.0109
[0/10][850/1583]        Loss_D: 1.2096  Loss_G: 4.0280  D(x): 0.4497    D(G(z)):
0.0105 / 0.0326
[0/10][900/1583]        Loss_D: 0.5365  Loss_G: 6.3879  D(x): 0.9512    D(G(z)):
0.3393 / 0.0057
[0/10][950/1583]        Loss_D: 2.0261  Loss_G: 9.1242  D(x): 0.9606    D(G(z)):
0.8018 / 0.0005
[0/10][1000/1583]       Loss_D: 0.3123  Loss_G: 4.6684  D(x): 0.8033    D(G(z)):
0.0309 / 0.0145
```

```
[0/10][1050/1583]       Loss_D: 0.5025  Loss_G: 3.2395  D(x): 0.8055    D(G(z)):
0.1972 / 0.0686
[0/10][1100/1583]       Loss_D: 0.5817  Loss_G: 5.5460  D(x): 0.9308    D(G(z)):
0.3438 / 0.0074
[0/10][1150/1583]       Loss_D: 0.3250  Loss_G: 3.4231  D(x): 0.8829    D(G(z)):
0.1440 / 0.0611
[0/10][1200/1583]       Loss_D: 0.4722  Loss_G: 3.4273  D(x): 0.7211    D(G(z)):
0.0451 / 0.0600
[0/10][1250/1583]       Loss_D: 0.4150  Loss_G: 5.0483  D(x): 0.7711    D(G(z)):
0.0332 / 0.0152
[0/10][1300/1583]       Loss_D: 1.1746  Loss_G: 10.3753 D(x): 0.9727    D(G(z)):
0.6012 / 0.0001
[0/10][1350/1583]       Loss_D: 0.4794  Loss_G: 4.4869  D(x): 0.7887    D(G(z)):
0.1465 / 0.0201
[0/10][1400/1583]       Loss_D: 0.4942  Loss_G: 3.6522  D(x): 0.7582    D(G(z)):
0.1158 / 0.0484
[0/10][1450/1583]       Loss_D: 0.4880  Loss_G: 2.5167  D(x): 0.7657    D(G(z)):
0.1167 / 0.1236
[0/10][1500/1583]       Loss_D: 0.7880  Loss_G: 5.1366  D(x): 0.9265    D(G(z)):
0.4294 / 0.0111
[0/10][1550/1583]       Loss_D: 0.7049  Loss_G: 4.9073  D(x): 0.9105    D(G(z)):
0.3925 / 0.0140
[1/10][0/1583]  Loss_D: 0.6150  Loss_G: 2.3010  D(x): 0.6693    D(G(z)): 0.0917
/ 0.1374
[1/10][50/1583] Loss_D: 0.4355  Loss_G: 4.0836  D(x): 0.8827    D(G(z)): 0.2250
/ 0.0269
[1/10][100/1583]        Loss_D: 0.5312  Loss_G: 4.9610  D(x): 0.9248    D(G(z)):
0.3210 / 0.0110
[1/10][150/1583]        Loss_D: 0.4154  Loss_G: 3.6153  D(x): 0.8315    D(G(z)):
0.1654 / 0.0443
[1/10][200/1583]        Loss_D: 0.5825  Loss_G: 4.0018  D(x): 0.8771    D(G(z)):
0.3106 / 0.0309
[1/10][250/1583]        Loss_D: 0.3050  Loss_G: 3.7873  D(x): 0.8588    D(G(z)):
0.1050 / 0.0350
[1/10][300/1583]        Loss_D: 0.4661  Loss_G: 2.7473  D(x): 0.7320    D(G(z)):
0.0804 / 0.0993
[1/10][350/1583]        Loss_D: 0.7285  Loss_G: 6.4599  D(x): 0.9631    D(G(z)):
0.4237 / 0.0030
[1/10][400/1583]        Loss_D: 0.4295  Loss_G: 4.0504  D(x): 0.8595    D(G(z)):
0.1914 / 0.0333
[1/10][450/1583]        Loss_D: 0.8279  Loss_G: 6.0782  D(x): 0.9467    D(G(z)):
0.4671 / 0.0052
[1/10][500/1583]        Loss_D: 0.5284  Loss_G: 4.7434  D(x): 0.8723    D(G(z)):
0.2783 / 0.0135
[1/10][550/1583]        Loss_D: 0.3508  Loss_G: 3.1864  D(x): 0.8529    D(G(z)):
0.1443 / 0.0671
[1/10][600/1583]        Loss_D: 0.4434  Loss_G: 4.2948  D(x): 0.8480    D(G(z)):
0.2016 / 0.0211
```

```
[1/10][650/1583]        Loss_D: 0.3180  Loss_G: 3.8850  D(x): 0.8891    D(G(z)):
0.1582 / 0.0339
[1/10][700/1583]        Loss_D: 0.9312  Loss_G: 5.7274  D(x): 0.9298    D(G(z)):
0.5086 / 0.0065
[1/10][750/1583]        Loss_D: 0.4579  Loss_G: 3.6078  D(x): 0.7896    D(G(z)):
0.1480 / 0.0465
[1/10][800/1583]        Loss_D: 0.4742  Loss_G: 3.7074  D(x): 0.8609    D(G(z)):
0.2401 / 0.0369
[1/10][850/1583]        Loss_D: 0.5342  Loss_G: 4.3814  D(x): 0.8831    D(G(z)):
0.2992 / 0.0196
[1/10][900/1583]        Loss_D: 0.3909  Loss_G: 3.2166  D(x): 0.8140    D(G(z)):
0.1343 / 0.0557
[1/10][950/1583]        Loss_D: 0.5557  Loss_G: 3.3441  D(x): 0.8276    D(G(z)):
0.2373 / 0.0540
[1/10][1000/1583]       Loss_D: 0.5054  Loss_G: 1.6998  D(x): 0.7155    D(G(z)):
0.1156 / 0.2192
[1/10][1050/1583]       Loss_D: 0.4200  Loss_G: 3.9009  D(x): 0.9232    D(G(z)):
0.2585 / 0.0297
[1/10][1100/1583]       Loss_D: 0.3976  Loss_G: 2.5664  D(x): 0.7859    D(G(z)):
0.1097 / 0.1103
[1/10][1150/1583]       Loss_D: 0.4174  Loss_G: 2.6396  D(x): 0.7821    D(G(z)):
0.1264 / 0.0958
[1/10][1200/1583]       Loss_D: 0.5547  Loss_G: 2.4150  D(x): 0.6753    D(G(z)):
0.0626 / 0.1282
[1/10][1250/1583]       Loss_D: 0.5070  Loss_G: 3.3619  D(x): 0.8301    D(G(z)):
0.2231 / 0.0502
[1/10][1300/1583]       Loss_D: 0.5081  Loss_G: 3.0501  D(x): 0.8605    D(G(z)):
0.2575 / 0.0674
[1/10][1350/1583]       Loss_D: 1.3050  Loss_G: 6.9609  D(x): 0.9428    D(G(z)):
0.6504 / 0.0020
[1/10][1400/1583]       Loss_D: 1.7235  Loss_G: 0.3561  D(x): 0.2707    D(G(z)):
0.0118 / 0.7401
[1/10][1450/1583]       Loss_D: 0.5198  Loss_G: 2.2282  D(x): 0.6945    D(G(z)):
0.0896 / 0.1499
[1/10][1500/1583]       Loss_D: 0.7978  Loss_G: 4.9282  D(x): 0.9071    D(G(z)):
0.4534 / 0.0110
[1/10][1550/1583]       Loss_D: 0.4954  Loss_G: 2.2161  D(x): 0.7599    D(G(z)):
0.1606 / 0.1384
[2/10][0/1583]  Loss_D: 0.3950  Loss_G: 2.8176  D(x): 0.8280    D(G(z)): 0.1611
/ 0.0742
[2/10][50/1583] Loss_D: 0.5569  Loss_G: 2.0115  D(x): 0.7049    D(G(z)): 0.1433
/ 0.1745
[2/10][100/1583]        Loss_D: 0.6158  Loss_G: 3.3171  D(x): 0.8851    D(G(z)):
0.3514 / 0.0478
[2/10][150/1583]        Loss_D: 0.5991  Loss_G: 3.3433  D(x): 0.8397    D(G(z)):
0.3103 / 0.0495
[2/10][200/1583]        Loss_D: 1.3925  Loss_G: 1.3051  D(x): 0.3209    D(G(z)):
0.0228 / 0.3388
```

```
[2/10][250/1583]        Loss_D: 0.4779  Loss_G: 3.4676  D(x): 0.8841    D(G(z)):
0.2742 / 0.0421
[2/10][300/1583]        Loss_D: 0.7613  Loss_G: 4.1292  D(x): 0.8833    D(G(z)):
0.4321 / 0.0219
[2/10][350/1583]        Loss_D: 0.6068  Loss_G: 3.7798  D(x): 0.8974    D(G(z)):
0.3480 / 0.0392
[2/10][400/1583]        Loss_D: 0.9982  Loss_G: 2.9475  D(x): 0.6620    D(G(z)):
0.3511 / 0.0862
[2/10][450/1583]        Loss_D: 0.5477  Loss_G: 2.1455  D(x): 0.6949    D(G(z)):
0.1281 / 0.1521
[2/10][500/1583]        Loss_D: 0.9797  Loss_G: 4.1544  D(x): 0.8981    D(G(z)):
0.5229 / 0.0295
[2/10][550/1583]        Loss_D: 0.5378  Loss_G: 2.7772  D(x): 0.8027    D(G(z)):
0.2330 / 0.0881
[2/10][600/1583]        Loss_D: 0.8025  Loss_G: 3.9734  D(x): 0.9115    D(G(z)):
0.4614 / 0.0279
[2/10][650/1583]        Loss_D: 0.8805  Loss_G: 1.1295  D(x): 0.5568    D(G(z)):
0.1411 / 0.3650
[2/10][700/1583]        Loss_D: 0.6854  Loss_G: 1.2721  D(x): 0.6068    D(G(z)):
0.0971 / 0.3272
[2/10][750/1583]        Loss_D: 0.4927  Loss_G: 1.9344  D(x): 0.7092    D(G(z)):
0.0940 / 0.1766
[2/10][800/1583]        Loss_D: 1.1755  Loss_G: 3.7172  D(x): 0.8947    D(G(z)):
0.5816 / 0.0379
[2/10][850/1583]        Loss_D: 1.1325  Loss_G: 0.8305  D(x): 0.4177    D(G(z)):
0.0639 / 0.4976
[2/10][900/1583]        Loss_D: 0.6465  Loss_G: 3.0357  D(x): 0.8447    D(G(z)):
0.3353 / 0.0720
[2/10][950/1583]        Loss_D: 0.7654  Loss_G: 2.5193  D(x): 0.7439    D(G(z)):
0.3244 / 0.1054
[2/10][1000/1583]       Loss_D: 0.5729  Loss_G: 2.0667  D(x): 0.7147    D(G(z)):
0.1584 / 0.1520
[2/10][1050/1583]       Loss_D: 0.4777  Loss_G: 2.3913  D(x): 0.8202    D(G(z)):
0.2163 / 0.1122
[2/10][1100/1583]       Loss_D: 1.0032  Loss_G: 2.5322  D(x): 0.6312    D(G(z)):
0.3401 / 0.1091
[2/10][1150/1583]       Loss_D: 0.6347  Loss_G: 2.6032  D(x): 0.7946    D(G(z)):
0.2891 / 0.0933
[2/10][1200/1583]       Loss_D: 0.7809  Loss_G: 2.2457  D(x): 0.7083    D(G(z)):
0.2925 / 0.1339
[2/10][1250/1583]       Loss_D: 0.5381  Loss_G: 2.5166  D(x): 0.8209    D(G(z)):
0.2527 / 0.1058
[2/10][1300/1583]       Loss_D: 0.5764  Loss_G: 3.1178  D(x): 0.8702    D(G(z)):
0.3247 / 0.0560
[2/10][1350/1583]       Loss_D: 0.5860  Loss_G: 1.6291  D(x): 0.6586    D(G(z)):
0.0964 / 0.2287
[2/10][1400/1583]       Loss_D: 1.0400  Loss_G: 0.4664  D(x): 0.4520    D(G(z)):
0.0987 / 0.6611
```

```
[2/10][1450/1583]       Loss_D: 0.6722  Loss_G: 2.1797  D(x): 0.7019    D(G(z)):
0.2215 / 0.1404
[2/10][1500/1583]       Loss_D: 0.8295  Loss_G: 1.4809  D(x): 0.5636    D(G(z)):
0.1256 / 0.2804
[2/10][1550/1583]       Loss_D: 0.9547  Loss_G: 4.5397  D(x): 0.8947    D(G(z)):
0.5091 / 0.0182
[3/10][0/1583]  Loss_D: 0.6242  Loss_G: 2.0804  D(x): 0.6988    D(G(z)): 0.1843
/ 0.1601
[3/10][50/1583] Loss_D: 0.5446  Loss_G: 2.7122  D(x): 0.8211    D(G(z)): 0.2629
/ 0.0842
[3/10][100/1583]        Loss_D: 0.9773  Loss_G: 3.0063  D(x): 0.8772    D(G(z)):
0.5143 / 0.0746
[3/10][150/1583]        Loss_D: 0.5856  Loss_G: 2.4304  D(x): 0.7440    D(G(z)):
0.2000 / 0.1211
[3/10][200/1583]        Loss_D: 0.5490  Loss_G: 1.9680  D(x): 0.7859    D(G(z)):
0.2198 / 0.1667
[3/10][250/1583]        Loss_D: 0.5153  Loss_G: 2.4201  D(x): 0.7830    D(G(z)):
0.2029 / 0.1099
[3/10][300/1583]        Loss_D: 0.7397  Loss_G: 2.7852  D(x): 0.8036    D(G(z)):
0.3713 / 0.0769
[3/10][350/1583]        Loss_D: 0.7672  Loss_G: 1.6470  D(x): 0.6067    D(G(z)):
0.1648 / 0.2348
[3/10][400/1583]        Loss_D: 0.7975  Loss_G: 1.4269  D(x): 0.5825    D(G(z)):
0.1247 / 0.2998
[3/10][450/1583]        Loss_D: 0.7366  Loss_G: 1.3135  D(x): 0.5691    D(G(z)):
0.0874 / 0.3178
[3/10][500/1583]        Loss_D: 0.7675  Loss_G: 3.7907  D(x): 0.8076    D(G(z)):
0.3777 / 0.0302
[3/10][550/1583]        Loss_D: 0.5428  Loss_G: 2.0399  D(x): 0.7428    D(G(z)):
0.1860 / 0.1613
[3/10][600/1583]        Loss_D: 0.8545  Loss_G: 1.7381  D(x): 0.5746    D(G(z)):
0.1548 / 0.2260
[3/10][650/1583]        Loss_D: 0.6328  Loss_G: 2.3910  D(x): 0.7620    D(G(z)):
0.2631 / 0.1273
[3/10][700/1583]        Loss_D: 0.5686  Loss_G: 1.8091  D(x): 0.6522    D(G(z)):
0.0864 / 0.2062
[3/10][750/1583]        Loss_D: 0.6976  Loss_G: 1.6283  D(x): 0.6364    D(G(z)):
0.1528 / 0.2424
[3/10][800/1583]        Loss_D: 0.6273  Loss_G: 2.8384  D(x): 0.8515    D(G(z)):
0.3394 / 0.0707
[3/10][850/1583]        Loss_D: 0.8710  Loss_G: 0.9133  D(x): 0.5054    D(G(z)):
0.0858 / 0.4488
[3/10][900/1583]        Loss_D: 0.6103  Loss_G: 2.9706  D(x): 0.8418    D(G(z)):
0.3110 / 0.0693
[3/10][950/1583]        Loss_D: 0.9465  Loss_G: 2.2653  D(x): 0.6702    D(G(z)):
0.3449 / 0.1406
[3/10][1000/1583]       Loss_D: 0.6516  Loss_G: 2.1707  D(x): 0.7000    D(G(z)):
0.2037 / 0.1426
```

```
[3/10][1050/1583]       Loss_D: 0.4721  Loss_G: 2.2969  D(x): 0.7602    D(G(z)):
0.1484 / 0.1264
[3/10][1100/1583]       Loss_D: 0.5844  Loss_G: 3.1354  D(x): 0.8347    D(G(z)):
0.3009 / 0.0522
[3/10][1150/1583]       Loss_D: 0.6346  Loss_G: 3.0313  D(x): 0.8876    D(G(z)):
0.3575 / 0.0704
[3/10][1200/1583]       Loss_D: 0.8999  Loss_G: 2.9788  D(x): 0.8412    D(G(z)):
0.4590 / 0.0691
[3/10][1250/1583]       Loss_D: 0.7309  Loss_G: 1.2432  D(x): 0.5558    D(G(z)):
0.0626 / 0.3299
[3/10][1300/1583]       Loss_D: 0.5133  Loss_G: 2.6316  D(x): 0.7820    D(G(z)):
0.2087 / 0.0847
[3/10][1350/1583]       Loss_D: 2.5759  Loss_G: 0.4903  D(x): 0.1258    D(G(z)):
0.0344 / 0.6687
[3/10][1400/1583]       Loss_D: 0.5783  Loss_G: 3.6259  D(x): 0.8858    D(G(z)):
0.3323 / 0.0354
[3/10][1450/1583]       Loss_D: 1.3331  Loss_G: 0.5660  D(x): 0.3357    D(G(z)):
0.0330 / 0.6111
[3/10][1500/1583]       Loss_D: 0.9247  Loss_G: 3.7200  D(x): 0.8756    D(G(z)):
0.4954 / 0.0362
[3/10][1550/1583]       Loss_D: 1.0936  Loss_G: 4.6652  D(x): 0.9217    D(G(z)):
0.5959 / 0.0142
[4/10][0/1583]  Loss_D: 0.6754  Loss_G: 3.4344  D(x): 0.8348    D(G(z)): 0.3514
/ 0.0411
[4/10][50/1583] Loss_D: 1.3154  Loss_G: 4.5350  D(x): 0.9163    D(G(z)): 0.6321
/ 0.0167
[4/10][100/1583]        Loss_D: 0.8015  Loss_G: 1.5834  D(x): 0.5204    D(G(z)):
0.0645 / 0.2531
[4/10][150/1583]        Loss_D: 0.4808  Loss_G: 1.9934  D(x): 0.7636    D(G(z)):
0.1582 / 0.1643
[4/10][200/1583]        Loss_D: 0.4914  Loss_G: 1.8351  D(x): 0.7441    D(G(z)):
0.1504 / 0.1892
[4/10][250/1583]        Loss_D: 0.6689  Loss_G: 2.3720  D(x): 0.7850    D(G(z)):
0.3087 / 0.1166
[4/10][300/1583]        Loss_D: 0.6901  Loss_G: 2.0626  D(x): 0.6592    D(G(z)):
0.1867 / 0.1628
[4/10][350/1583]        Loss_D: 0.9388  Loss_G: 1.0210  D(x): 0.4809    D(G(z)):
0.0853 / 0.4183
[4/10][400/1583]        Loss_D: 0.6433  Loss_G: 1.7744  D(x): 0.7112    D(G(z)):
0.2124 / 0.2133
[4/10][450/1583]        Loss_D: 0.5434  Loss_G: 2.7423  D(x): 0.7989    D(G(z)):
0.2427 / 0.0811
[4/10][500/1583]        Loss_D: 0.6965  Loss_G: 2.9836  D(x): 0.8334    D(G(z)):
0.3635 / 0.0630
[4/10][550/1583]        Loss_D: 0.5570  Loss_G: 2.7025  D(x): 0.7899    D(G(z)):
0.2403 / 0.0821
[4/10][600/1583]        Loss_D: 0.6574  Loss_G: 1.8016  D(x): 0.6750    D(G(z)):
0.1777 / 0.2055
```

```
[4/10][650/1583]        Loss_D: 0.7318  Loss_G: 1.2587  D(x): 0.6013    D(G(z)):
0.1471 / 0.3249
[4/10][700/1583]        Loss_D: 0.7771  Loss_G: 1.3766  D(x): 0.5520    D(G(z)):
0.0740 / 0.2995
[4/10][750/1583]        Loss_D: 1.3912  Loss_G: 0.7068  D(x): 0.3173    D(G(z)):
0.0357 / 0.5420
[4/10][800/1583]        Loss_D: 0.7566  Loss_G: 3.4140  D(x): 0.8610    D(G(z)):
0.4072 / 0.0478
[4/10][850/1583]        Loss_D: 0.8019  Loss_G: 3.0653  D(x): 0.8814    D(G(z)):
0.4361 / 0.0687
[4/10][900/1583]        Loss_D: 0.5238  Loss_G: 2.6011  D(x): 0.8267    D(G(z)):
0.2480 / 0.0933
[4/10][950/1583]        Loss_D: 1.1802  Loss_G: 4.2918  D(x): 0.9016    D(G(z)):
0.5937 / 0.0221
[4/10][1000/1583]       Loss_D: 0.4246  Loss_G: 3.2705  D(x): 0.8695    D(G(z)):
0.2256 / 0.0482
[4/10][1050/1583]       Loss_D: 0.7819  Loss_G: 3.9572  D(x): 0.9010    D(G(z)):
0.4421 / 0.0282
[4/10][1100/1583]       Loss_D: 0.7209  Loss_G: 1.7695  D(x): 0.6491    D(G(z)):
0.1767 / 0.2137
[4/10][1150/1583]       Loss_D: 1.9600  Loss_G: 0.6096  D(x): 0.2054    D(G(z)):
0.0387 / 0.6001
[4/10][1200/1583]       Loss_D: 0.6486  Loss_G: 3.0216  D(x): 0.8676    D(G(z)):
0.3640 / 0.0616
[4/10][1250/1583]       Loss_D: 0.4560  Loss_G: 2.3554  D(x): 0.7772    D(G(z)):
0.1557 / 0.1183
[4/10][1300/1583]       Loss_D: 0.6011  Loss_G: 1.5766  D(x): 0.6350    D(G(z)):
0.0773 / 0.2608
[4/10][1350/1583]       Loss_D: 0.5508  Loss_G: 2.8567  D(x): 0.8491    D(G(z)):
0.2914 / 0.0744
[4/10][1400/1583]       Loss_D: 0.5511  Loss_G: 3.1016  D(x): 0.8512    D(G(z)):
0.2927 / 0.0606
[4/10][1450/1583]       Loss_D: 0.3941  Loss_G: 2.9615  D(x): 0.8632    D(G(z)):
0.1971 / 0.0712
[4/10][1500/1583]       Loss_D: 0.4972  Loss_G: 3.1722  D(x): 0.8131    D(G(z)):
0.2157 / 0.0549
[4/10][1550/1583]       Loss_D: 0.4908  Loss_G: 2.7449  D(x): 0.8200    D(G(z)):
0.2214 / 0.0855
[5/10][0/1583]  Loss_D: 0.5613  Loss_G: 2.0747  D(x): 0.7209    D(G(z)): 0.1682
/ 0.1520
[5/10][50/1583] Loss_D: 0.4651  Loss_G: 2.3589  D(x): 0.7876    D(G(z)): 0.1808
/ 0.1110
[5/10][100/1583]        Loss_D: 1.6769  Loss_G: 4.0183  D(x): 0.7546    D(G(z)):
0.6826 / 0.0314
[5/10][150/1583]        Loss_D: 0.6835  Loss_G: 1.9465  D(x): 0.6758    D(G(z)):
0.2007 / 0.1765
[5/10][200/1583]        Loss_D: 0.4045  Loss_G: 3.1263  D(x): 0.8231    D(G(z)):
0.1621 / 0.0608
```

```
[5/10][250/1583]        Loss_D: 0.4707  Loss_G: 2.3372  D(x): 0.7658    D(G(z)):
0.1487 / 0.1202
[5/10][300/1583]        Loss_D: 0.4466  Loss_G: 3.2894  D(x): 0.9104    D(G(z)):
0.2701 / 0.0497
[5/10][350/1583]        Loss_D: 0.6803  Loss_G: 3.2483  D(x): 0.9191    D(G(z)):
0.4099 / 0.0510
[5/10][400/1583]        Loss_D: 0.5231  Loss_G: 2.7686  D(x): 0.8799    D(G(z)):
0.2886 / 0.0891
[5/10][450/1583]        Loss_D: 0.6944  Loss_G: 2.0719  D(x): 0.6128    D(G(z)):
0.1124 / 0.1608
[5/10][500/1583]        Loss_D: 0.5364  Loss_G: 2.7175  D(x): 0.8300    D(G(z)):
0.2679 / 0.0821
[5/10][550/1583]        Loss_D: 0.4268  Loss_G: 2.3963  D(x): 0.8007    D(G(z)):
0.1604 / 0.1204
[5/10][600/1583]        Loss_D: 0.6918  Loss_G: 2.1879  D(x): 0.6831    D(G(z)):
0.2081 / 0.1469
[5/10][650/1583]        Loss_D: 0.6307  Loss_G: 1.9255  D(x): 0.7412    D(G(z)):
0.2418 / 0.1770
[5/10][700/1583]        Loss_D: 0.5471  Loss_G: 2.1154  D(x): 0.7169    D(G(z)):
0.1555 / 0.1571
[5/10][750/1583]        Loss_D: 1.1326  Loss_G: 0.7689  D(x): 0.3947    D(G(z)):
0.0277 / 0.5534
[5/10][800/1583]        Loss_D: 0.7908  Loss_G: 4.7098  D(x): 0.9448    D(G(z)):
0.4707 / 0.0138
[5/10][850/1583]        Loss_D: 0.5688  Loss_G: 4.0265  D(x): 0.9513    D(G(z)):
0.3710 / 0.0250
[5/10][900/1583]        Loss_D: 1.0469  Loss_G: 3.3382  D(x): 0.8139    D(G(z)):
0.4796 / 0.0560
[5/10][950/1583]        Loss_D: 0.6586  Loss_G: 1.8025  D(x): 0.6127    D(G(z)):
0.0759 / 0.2116
[5/10][1000/1583]       Loss_D: 0.7450  Loss_G: 1.3680  D(x): 0.5450    D(G(z)):
0.0445 / 0.3170
[5/10][1050/1583]       Loss_D: 0.4880  Loss_G: 2.7812  D(x): 0.8058    D(G(z)):
0.2037 / 0.0797
[5/10][1100/1583]       Loss_D: 0.4865  Loss_G: 2.2381  D(x): 0.7702    D(G(z)):
0.1707 / 0.1378
[5/10][1150/1583]       Loss_D: 0.4253  Loss_G: 2.8506  D(x): 0.8517    D(G(z)):
0.2093 / 0.0801
[5/10][1200/1583]       Loss_D: 0.6595  Loss_G: 2.1307  D(x): 0.6849    D(G(z)):
0.1839 / 0.1611
[5/10][1250/1583]       Loss_D: 1.9396  Loss_G: 0.6265  D(x): 0.2183    D(G(z)):
0.0479 / 0.6046
[5/10][1300/1583]       Loss_D: 0.6080  Loss_G: 1.5130  D(x): 0.7437    D(G(z)):
0.2264 / 0.2585
[5/10][1350/1583]       Loss_D: 1.0917  Loss_G: 0.6374  D(x): 0.4188    D(G(z)):
0.0365 / 0.5837
[5/10][1400/1583]       Loss_D: 0.5842  Loss_G: 2.8466  D(x): 0.8406    D(G(z)):
0.2978 / 0.0746
```

```
[5/10][1450/1583]       Loss_D: 0.3509  Loss_G: 2.8440  D(x): 0.8116    D(G(z)):
0.1147 / 0.0824
[5/10][1500/1583]       Loss_D: 1.5262  Loss_G: 0.3862  D(x): 0.3777    D(G(z)):
0.2060 / 0.7312
[5/10][1550/1583]       Loss_D: 0.5578  Loss_G: 1.5548  D(x): 0.6840    D(G(z)):
0.1090 / 0.2468
[6/10][0/1583]  Loss_D: 0.7388  Loss_G: 1.2495  D(x): 0.5931    D(G(z)): 0.1203
/ 0.3462
[6/10][50/1583] Loss_D: 0.5803  Loss_G: 3.3133  D(x): 0.9037    D(G(z)): 0.3524
/ 0.0465
[6/10][100/1583]        Loss_D: 0.6301  Loss_G: 1.5457  D(x): 0.6866    D(G(z)):
0.1780 / 0.2724
[6/10][150/1583]        Loss_D: 1.2526  Loss_G: 5.0420  D(x): 0.9616    D(G(z)):
0.6387 / 0.0109
[6/10][200/1583]        Loss_D: 0.8472  Loss_G: 1.1213  D(x): 0.5046    D(G(z)):
0.0489 / 0.3915
[6/10][250/1583]        Loss_D: 3.1972  Loss_G: 5.7986  D(x): 0.9706    D(G(z)):
0.9284 / 0.0090
[6/10][300/1583]        Loss_D: 0.6408  Loss_G: 1.6895  D(x): 0.6248    D(G(z)):
0.1003 / 0.2383
[6/10][350/1583]        Loss_D: 0.5169  Loss_G: 3.2863  D(x): 0.8640    D(G(z)):
0.2747 / 0.0487
[6/10][400/1583]        Loss_D: 0.9310  Loss_G: 2.7829  D(x): 0.8219    D(G(z)):
0.4510 / 0.0868
[6/10][450/1583]        Loss_D: 0.6039  Loss_G: 2.6829  D(x): 0.7895    D(G(z)):
0.2683 / 0.0882
[6/10][500/1583]        Loss_D: 0.6171  Loss_G: 3.5275  D(x): 0.8788    D(G(z)):
0.3479 / 0.0411
[6/10][550/1583]        Loss_D: 0.7941  Loss_G: 4.7749  D(x): 0.9446    D(G(z)):
0.4726 / 0.0127
[6/10][600/1583]        Loss_D: 0.6087  Loss_G: 1.6567  D(x): 0.6467    D(G(z)):
0.0972 / 0.2363
[6/10][650/1583]        Loss_D: 0.4873  Loss_G: 3.5100  D(x): 0.9504    D(G(z)):
0.3171 / 0.0422
[6/10][700/1583]        Loss_D: 0.6010  Loss_G: 4.1132  D(x): 0.8864    D(G(z)):
0.3456 / 0.0209
[6/10][750/1583]        Loss_D: 1.6876  Loss_G: 5.3635  D(x): 0.9348    D(G(z)):
0.7361 / 0.0083
[6/10][800/1583]        Loss_D: 0.4935  Loss_G: 3.2439  D(x): 0.9112    D(G(z)):
0.2948 / 0.0539
[6/10][850/1583]        Loss_D: 0.4624  Loss_G: 2.0757  D(x): 0.7016    D(G(z)):
0.0626 / 0.1568
[6/10][900/1583]        Loss_D: 0.5708  Loss_G: 2.1730  D(x): 0.7555    D(G(z)):
0.2180 / 0.1415
[6/10][950/1583]        Loss_D: 0.4442  Loss_G: 2.6885  D(x): 0.8303    D(G(z)):
0.1991 / 0.0885
[6/10][1000/1583]       Loss_D: 0.6410  Loss_G: 1.5757  D(x): 0.5879    D(G(z)):
0.0418 / 0.2523
```

```
[6/10][1050/1583]       Loss_D: 0.3602  Loss_G: 3.2989  D(x): 0.8500    D(G(z)):
0.1586 / 0.0524
[6/10][1100/1583]       Loss_D: 0.5261  Loss_G: 1.9257  D(x): 0.6976    D(G(z)):
0.0988 / 0.1810
[6/10][1150/1583]       Loss_D: 0.5006  Loss_G: 2.0165  D(x): 0.7231    D(G(z)):
0.1230 / 0.1692
[6/10][1200/1583]       Loss_D: 0.6034  Loss_G: 3.5019  D(x): 0.9137    D(G(z)):
0.3671 / 0.0427
[6/10][1250/1583]       Loss_D: 0.4917  Loss_G: 2.2765  D(x): 0.7244    D(G(z)):
0.1000 / 0.1371
[6/10][1300/1583]       Loss_D: 0.7576  Loss_G: 1.5767  D(x): 0.5617    D(G(z)):
0.0874 / 0.2676
[6/10][1350/1583]       Loss_D: 0.7392  Loss_G: 1.2197  D(x): 0.5754    D(G(z)):
0.0805 / 0.3765
[6/10][1400/1583]       Loss_D: 0.6274  Loss_G: 2.0145  D(x): 0.7019    D(G(z)):
0.1847 / 0.1675
[6/10][1450/1583]       Loss_D: 0.4314  Loss_G: 2.6456  D(x): 0.7228    D(G(z)):
0.0542 / 0.0943
[6/10][1500/1583]       Loss_D: 0.3603  Loss_G: 3.0909  D(x): 0.8567    D(G(z)):
0.1600 / 0.0605
[6/10][1550/1583]       Loss_D: 1.0331  Loss_G: 3.6338  D(x): 0.8452    D(G(z)):
0.5044 / 0.0442
[7/10][0/1583]  Loss_D: 0.4583  Loss_G: 2.8229  D(x): 0.8208    D(G(z)): 0.1948
/ 0.0762
[7/10][50/1583] Loss_D: 0.6427  Loss_G: 1.3175  D(x): 0.6686    D(G(z)): 0.1621
/ 0.3099
[7/10][100/1583]        Loss_D: 0.4840  Loss_G: 1.8941  D(x): 0.6930    D(G(z)):
0.0745 / 0.2032
[7/10][150/1583]        Loss_D: 0.5518  Loss_G: 2.0217  D(x): 0.7305    D(G(z)):
0.1680 / 0.1740
[7/10][200/1583]        Loss_D: 0.6680  Loss_G: 3.7873  D(x): 0.8989    D(G(z)):
0.3828 / 0.0337
[7/10][250/1583]        Loss_D: 0.3931  Loss_G: 3.0152  D(x): 0.8671    D(G(z)):
0.2005 / 0.0638
[7/10][300/1583]        Loss_D: 0.4980  Loss_G: 4.2029  D(x): 0.9522    D(G(z)):
0.3387 / 0.0194
[7/10][350/1583]        Loss_D: 0.6460  Loss_G: 1.5703  D(x): 0.6253    D(G(z)):
0.0962 / 0.2505
[7/10][400/1583]        Loss_D: 0.3378  Loss_G: 2.7986  D(x): 0.8386    D(G(z)):
0.1244 / 0.0805
[7/10][450/1583]        Loss_D: 1.7369  Loss_G: 0.6589  D(x): 0.2503    D(G(z)):
0.0207 / 0.5783
[7/10][500/1583]        Loss_D: 0.4799  Loss_G: 1.7699  D(x): 0.7200    D(G(z)):
0.1089 / 0.2127
[7/10][550/1583]        Loss_D: 0.4291  Loss_G: 2.3210  D(x): 0.8741    D(G(z)):
0.2299 / 0.1303
[7/10][600/1583]        Loss_D: 0.3562  Loss_G: 2.4944  D(x): 0.8540    D(G(z)):
0.1612 / 0.0989
```

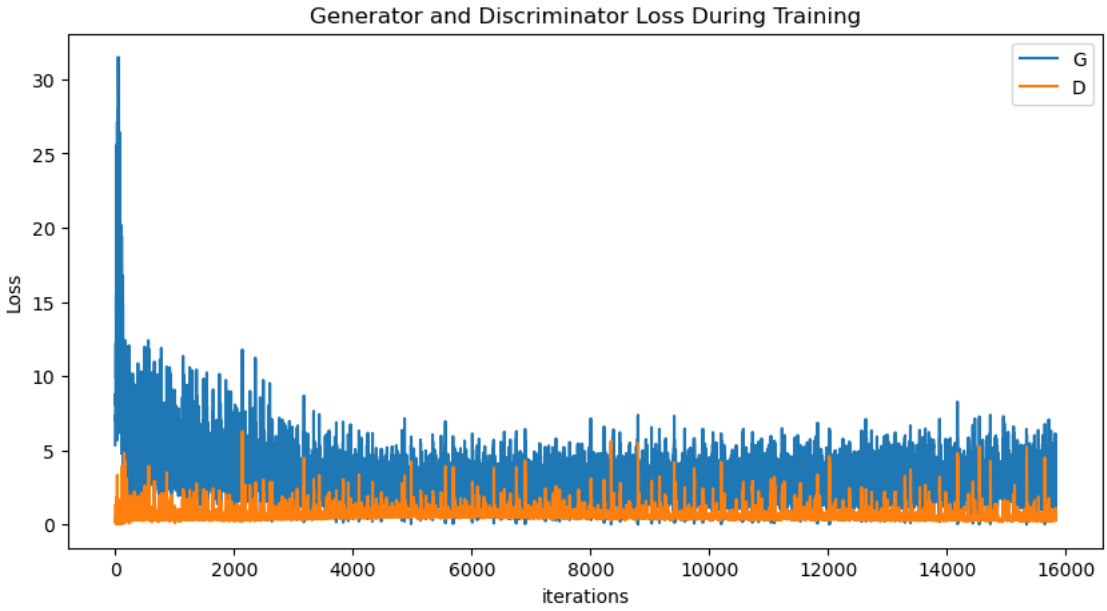
```
[7/10][650/1583]        Loss_D: 0.4594  Loss_G: 3.5051  D(x): 0.9036    D(G(z)):
0.2699 / 0.0413
[7/10][700/1583]        Loss_D: 0.4751  Loss_G: 2.6655  D(x): 0.7915    D(G(z)):
0.1806 / 0.0914
[7/10][750/1583]        Loss_D: 0.8548  Loss_G: 1.2122  D(x): 0.5770    D(G(z)):
0.1445 / 0.3448
[7/10][800/1583]        Loss_D: 0.5340  Loss_G: 2.7373  D(x): 0.7776    D(G(z)):
0.2107 / 0.0869
[7/10][850/1583]        Loss_D: 0.5359  Loss_G: 2.7120  D(x): 0.7709    D(G(z)):
0.2010 / 0.0844
[7/10][900/1583]        Loss_D: 0.3710  Loss_G: 2.8163  D(x): 0.8205    D(G(z)):
0.1354 / 0.0851
[7/10][950/1583]        Loss_D: 1.1073  Loss_G: 3.0721  D(x): 0.8424    D(G(z)):
0.5402 / 0.0643
[7/10][1000/1583]       Loss_D: 0.4141  Loss_G: 2.5424  D(x): 0.7799    D(G(z)):
0.1204 / 0.1042
[7/10][1050/1583]       Loss_D: 0.6546  Loss_G: 2.7264  D(x): 0.7031    D(G(z)):
0.2058 / 0.0978
[7/10][1100/1583]       Loss_D: 0.4719  Loss_G: 3.6829  D(x): 0.9080    D(G(z)):
0.2769 / 0.0365
[7/10][1150/1583]       Loss_D: 0.5060  Loss_G: 1.8239  D(x): 0.7739    D(G(z)):
0.1915 / 0.1942
[7/10][1200/1583]       Loss_D: 0.4031  Loss_G: 2.2230  D(x): 0.7967    D(G(z)):
0.1356 / 0.1417
[7/10][1250/1583]       Loss_D: 0.7308  Loss_G: 1.3562  D(x): 0.5485    D(G(z)):
0.0355 / 0.3072
[7/10][1300/1583]       Loss_D: 0.9708  Loss_G: 1.5459  D(x): 0.4645    D(G(z)):
0.0515 / 0.2797
[7/10][1350/1583]       Loss_D: 0.5766  Loss_G: 1.5832  D(x): 0.6441    D(G(z)):
0.0632 / 0.2541
[7/10][1400/1583]       Loss_D: 0.5048  Loss_G: 2.0800  D(x): 0.6810    D(G(z)):
0.0657 / 0.1698
[7/10][1450/1583]       Loss_D: 0.5139  Loss_G: 4.1287  D(x): 0.9314    D(G(z)):
0.3110 / 0.0251
[7/10][1500/1583]       Loss_D: 0.6216  Loss_G: 3.5085  D(x): 0.8827    D(G(z)):
0.3525 / 0.0425
[7/10][1550/1583]       Loss_D: 0.5448  Loss_G: 3.2460  D(x): 0.8272    D(G(z)):
0.2549 / 0.0544
[8/10][0/1583]  Loss_D: 0.4110  Loss_G: 3.4221  D(x): 0.8591    D(G(z)): 0.1969
/ 0.0457
[8/10][50/1583] Loss_D: 0.5400  Loss_G: 3.6789  D(x): 0.9120    D(G(z)): 0.3199
/ 0.0369
[8/10][100/1583]        Loss_D: 0.3185  Loss_G: 3.7238  D(x): 0.9400    D(G(z)):
0.2050 / 0.0345
[8/10][150/1583]        Loss_D: 0.3334  Loss_G: 3.1562  D(x): 0.9079    D(G(z)):
0.1920 / 0.0572
[8/10][200/1583]        Loss_D: 0.3789  Loss_G: 3.1811  D(x): 0.8347    D(G(z)):
0.1599 / 0.0548
```

```
[8/10][250/1583]        Loss_D: 0.5488  Loss_G: 3.3802  D(x): 0.9103    D(G(z)):
0.3246 / 0.0510
[8/10][300/1583]        Loss_D: 0.5672  Loss_G: 2.1145  D(x): 0.6826    D(G(z)):
0.1138 / 0.1752
[8/10][350/1583]        Loss_D: 0.4688  Loss_G: 2.4857  D(x): 0.7308    D(G(z)):
0.0953 / 0.1201
[8/10][400/1583]        Loss_D: 0.4282  Loss_G: 2.0439  D(x): 0.7669    D(G(z)):
0.1151 / 0.1724
[8/10][450/1583]        Loss_D: 0.4199  Loss_G: 2.4845  D(x): 0.7286    D(G(z)):
0.0607 / 0.1175
[8/10][500/1583]        Loss_D: 0.4359  Loss_G: 2.1610  D(x): 0.7745    D(G(z)):
0.1283 / 0.1493
[8/10][550/1583]        Loss_D: 0.5275  Loss_G: 1.5287  D(x): 0.6895    D(G(z)):
0.0899 / 0.2745
[8/10][600/1583]        Loss_D: 0.4889  Loss_G: 1.9957  D(x): 0.6824    D(G(z)):
0.0406 / 0.1909
[8/10][650/1583]        Loss_D: 0.8706  Loss_G: 4.7255  D(x): 0.9538    D(G(z)):
0.4996 / 0.0142
[8/10][700/1583]        Loss_D: 0.5339  Loss_G: 4.1607  D(x): 0.9697    D(G(z)):
0.3512 / 0.0245
[8/10][750/1583]        Loss_D: 0.4914  Loss_G: 2.5349  D(x): 0.7749    D(G(z)):
0.1710 / 0.1139
[8/10][800/1583]        Loss_D: 0.6120  Loss_G: 4.5815  D(x): 0.9436    D(G(z)):
0.3807 / 0.0155
[8/10][850/1583]        Loss_D: 0.4059  Loss_G: 2.3228  D(x): 0.8159    D(G(z)):
0.1592 / 0.1290
[8/10][900/1583]        Loss_D: 0.4019  Loss_G: 2.8448  D(x): 0.7779    D(G(z)):
0.1005 / 0.0878
[8/10][950/1583]        Loss_D: 0.9933  Loss_G: 0.9337  D(x): 0.4531    D(G(z)):
0.0359 / 0.4583
[8/10][1000/1583]       Loss_D: 0.4824  Loss_G: 2.0618  D(x): 0.7045    D(G(z)):
0.0637 / 0.1819
[8/10][1050/1583]       Loss_D: 0.3998  Loss_G: 2.5957  D(x): 0.8408    D(G(z)):
0.1756 / 0.1071
[8/10][1100/1583]       Loss_D: 0.3186  Loss_G: 2.6664  D(x): 0.8289    D(G(z)):
0.1065 / 0.0970
[8/10][1150/1583]       Loss_D: 0.3302  Loss_G: 2.6973  D(x): 0.8667    D(G(z)):
0.1535 / 0.0867
[8/10][1200/1583]       Loss_D: 0.2929  Loss_G: 2.2298  D(x): 0.8324    D(G(z)):
0.0868 / 0.1419
[8/10][1250/1583]       Loss_D: 0.4821  Loss_G: 2.3548  D(x): 0.7689    D(G(z)):
0.1509 / 0.1284
[8/10][1300/1583]       Loss_D: 0.4433  Loss_G: 2.4200  D(x): 0.8020    D(G(z)):
0.1651 / 0.1127
[8/10][1350/1583]       Loss_D: 0.3329  Loss_G: 3.0396  D(x): 0.8622    D(G(z)):
0.1482 / 0.0673
[8/10][1400/1583]       Loss_D: 0.6386  Loss_G: 2.4250  D(x): 0.7177    D(G(z)):
0.2096 / 0.1195
```

```
[8/10][1450/1583]       Loss_D: 0.8590  Loss_G: 4.2922  D(x): 0.9487    D(G(z)):
0.4978 / 0.0252
[8/10][1500/1583]       Loss_D: 0.2399  Loss_G: 3.4750  D(x): 0.8861    D(G(z)):
0.0965 / 0.0447
[8/10][1550/1583]       Loss_D: 0.4477  Loss_G: 2.8879  D(x): 0.7757    D(G(z)):
0.1397 / 0.0770
[9/10][0/1583]  Loss_D: 0.3304  Loss_G: 2.9775  D(x): 0.7983    D(G(z)): 0.0781
/ 0.0819
[9/10][50/1583] Loss_D: 0.4836  Loss_G: 1.9386  D(x): 0.7092    D(G(z)): 0.0780
/ 0.2038
[9/10][100/1583]        Loss_D: 0.3887  Loss_G: 2.7089  D(x): 0.8489    D(G(z)):
0.1749 / 0.0926
[9/10][150/1583]        Loss_D: 0.3158  Loss_G: 3.0308  D(x): 0.8742    D(G(z)):
0.1446 / 0.0683
[9/10][200/1583]        Loss_D: 0.4455  Loss_G: 2.7617  D(x): 0.7745    D(G(z)):
0.1307 / 0.0919
[9/10][250/1583]        Loss_D: 0.3778  Loss_G: 2.2505  D(x): 0.8524    D(G(z)):
0.1668 / 0.1348
[9/10][300/1583]        Loss_D: 2.3168  Loss_G: 4.7014  D(x): 0.9417    D(G(z)):
0.8262 / 0.0191
[9/10][350/1583]        Loss_D: 0.6384  Loss_G: 1.3468  D(x): 0.6386    D(G(z)):
0.0798 / 0.3202
[9/10][400/1583]        Loss_D: 0.3609  Loss_G: 2.3744  D(x): 0.7645    D(G(z)):
0.0596 / 0.1321
[9/10][450/1583]        Loss_D: 0.4469  Loss_G: 2.2142  D(x): 0.7142    D(G(z)):
0.0591 / 0.1482
[9/10][500/1583]        Loss_D: 0.8408  Loss_G: 1.0987  D(x): 0.5604    D(G(z)):
0.1074 / 0.3884
[9/10][550/1583]        Loss_D: 0.4663  Loss_G: 4.1167  D(x): 0.9215    D(G(z)):
0.2900 / 0.0221
[9/10][600/1583]        Loss_D: 0.9487  Loss_G: 1.0021  D(x): 0.4655    D(G(z)):
0.0267 / 0.4332
[9/10][650/1583]        Loss_D: 0.3561  Loss_G: 2.3660  D(x): 0.8383    D(G(z)):
0.1386 / 0.1320
[9/10][700/1583]        Loss_D: 0.4360  Loss_G: 1.5980  D(x): 0.7940    D(G(z)):
0.1357 / 0.2535
[9/10][750/1583]        Loss_D: 1.2382  Loss_G: 6.1342  D(x): 0.9731    D(G(z)):
0.6177 / 0.0043
[9/10][800/1583]        Loss_D: 0.8239  Loss_G: 1.8945  D(x): 0.5118    D(G(z)):
0.0304 / 0.2269
[9/10][850/1583]        Loss_D: 0.3395  Loss_G: 3.6450  D(x): 0.9042    D(G(z)):
0.1935 / 0.0369
[9/10][900/1583]        Loss_D: 0.5982  Loss_G: 2.3272  D(x): 0.6826    D(G(z)):
0.1173 / 0.1404
[9/10][950/1583]        Loss_D: 0.4496  Loss_G: 3.2270  D(x): 0.9052    D(G(z)):
0.2584 / 0.0552
[9/10][1000/1583]       Loss_D: 0.7411  Loss_G: 4.0571  D(x): 0.9640    D(G(z)):
0.4552 / 0.0248
```

```
[9/10][1050/1583]       Loss_D: 0.2377  Loss_G: 3.1040  D(x): 0.9014    D(G(z)):
0.1145 / 0.0620
[9/10][1100/1583]       Loss_D: 1.3294  Loss_G: 0.8515  D(x): 0.4049    D(G(z)):
0.1815 / 0.5046
[9/10][1150/1583]       Loss_D: 0.9030  Loss_G: 1.0263  D(x): 0.4984    D(G(z)):
0.0530 / 0.4135
[9/10][1200/1583]       Loss_D: 0.3822  Loss_G: 2.5964  D(x): 0.8120    D(G(z)):
0.1367 / 0.0993
[9/10][1250/1583]       Loss_D: 0.4998  Loss_G: 3.9357  D(x): 0.8443    D(G(z)):
0.2356 / 0.0295
[9/10][1300/1583]       Loss_D: 0.4348  Loss_G: 3.0091  D(x): 0.7777    D(G(z)):
0.1221 / 0.0704
[9/10][1350/1583]       Loss_D: 0.2083  Loss_G: 3.3659  D(x): 0.8894    D(G(z)):
0.0767 / 0.0546
[9/10][1400/1583]       Loss_D: 4.5155  Loss_G: 6.7676  D(x): 0.9974    D(G(z)):
0.9780 / 0.0032
[9/10][1450/1583]       Loss_D: 0.3444  Loss_G: 3.0516  D(x): 0.8526    D(G(z)):
0.1424 / 0.0691
[9/10][1500/1583]       Loss_D: 0.1941  Loss_G: 3.8814  D(x): 0.9292    D(G(z)):
0.1064 / 0.0285
[9/10][1550/1583]       Loss_D: 0.7652  Loss_G: 5.5152  D(x): 0.9693    D(G(z)):
0.4575 / 0.0061
```

[ ]:
```python
plt.figure(figsize=(10, 5))
plt.title("Generator and Discriminator Loss During Training")
plt.plot(G_losses, label="G")
plt.plot(D_losses, label="D")
plt.xlabel("iterations")
plt.ylabel("Loss")
plt.legend()
plt.show()
```



[ ]:
```
fig = plt.figure(figsize=(8, 8))
plt.axis("off")
ims = [[plt.imshow(np.transpose(i, (1, 2, 0)), animated=True)] for i in␣
 ↪img_list]
ani = animation.ArtistAnimation(fig, ims, interval=1000, repeat_delay=1000,␣
 ↪blit=True)

HTML(ani.to_jshtml())
```

```
Animation size has reached 21289351 bytes, exceeding the limit of 20971520.0. If
you're sure you want a larger animation embedded, set the animation.embed_limit
rc parameter to a larger value (in MB). This and further frames will be dropped.
```

[ ]:
```
<IPython.core.display.HTML object>
```

[ ]:
```
real_batch = next(iter(dataloader))

plt.figure(figsize=(15, 15))
plt.subplot(1, 2, 1)
plt.axis("off")
plt.title("Real Images")
plt.imshow(
    np.transpose(
        vutils.make_grid(
            real_batch[0].to(device)[:64], padding=5, normalize=True
        ).cpu(),
        (1, 2, 0),
    )
```

```
)

plt.subplot(1, 2, 2)
plt.axis("off")
plt.title("Fake Images")
plt.imshow(np.transpose(img_list[-1], (1, 2, 0)))
plt.show()
```