DCGAN

March 19, 2023

0.1 Stuff For Report

0.1.1 Research Paper we are 'replicating'

Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks

https://arxiv.org/pdf/1511.06434.pdf

This research paper introduces a class of powerful GANs called Deep Convolutional Generative Adversarial Networks (DCGANs), which use deep convolutional neural networks (CNNs) as building blocks for both the Generator and the Discriminator. The authors propose several architectural changes to improve the stability of GAN training, making it possible to train deeper models that generate higher-quality images.

The key contributions and findings of the paper are:

0.1.2 Architectural guidelines for stable DCGAN training:

- Use strided convolutions instead of pooling layers in the Discriminator.
- Use fractional-strided convolutions in the Generator.
- Remove fully connected hidden layers for deeper architectures.
- Use batch normalization in both the Generator and the Discriminator.
- Use ReLU activation in the Generator except for the output layer, which uses Tanh.
- Use LeakyReLU activation in the Discriminator.

0.1.3 Generator:

- Transposed convolution layer with 100 input channels, 512 output channels, 4x4 kernel size, 1 stride, and 0 padding.
- Transposed convolution layer with 512 input channels, 256 output channels, 4x4 kernel size, 2 stride, and 1 padding.
- Transposed convolution layer with 256 input channels, 128 output channels, 4x4 kernel size, 2 stride, and 1 padding.
- Transposed convolution layer with 128 input channels, 3 output channels (for RGB images), 4x4 kernel size, 2 stride, and 1 padding.

0.1.4 Discriminator:

- Convolution layer with 3 input channels (for RGB images), 128 output channels, 4x4 kernel size, 2 stride, and 1 padding.
- Convolution layer with 128 input channels, 256 output channels, 4x4 kernel size, 2 stride, and 1 padding.

- Convolution layer with 256 input channels, 512 output channels, 4x4 kernel size, 2 stride, and 1 padding.
- Convolution layer with 512 input channels, 1 output channel, 4x4 kernel size, 1 stride, and 0 padding.

0.1.5 Metrics:

- Loss D: This is the Discriminator Loss. It represents how well the Discriminator can distinguish between real and generated images during each iteration. Lower values indicate better performance.
- Loss G: This is the Generator Loss. It represents how well the Generator can create realistic images that can "fool" the Discriminator. Lower values indicate better performance.
- D(x): This is the average output probability of the Discriminator for real images (x). It represents how well the Discriminator can identify real images. Values closer to 1 indicate better performance.
- D(G(z)): This value has two parts: the numerator is the average output probability of the Discriminator for generated images before the Generator update. Lower values indicate that the Discriminator is better at identifying fake images. The denominator is the average output probability of the Discriminator for generated images after the Generator update. Higher values indicate that the Generator is better at creating realistic images that can fool the Discriminator.

0.1.6 Datasets

EMNIST - https://www.nist.gov/itl/products-and-services/emnist-dataset

CIFAR-10 - https://www.cs.toronto.edu/~kriz/cifar.html

Celeb-A - https://mmlab.ie.cuhk.edu.hk/projects/CelebA.html

0.2 DCGAN - EMNIST

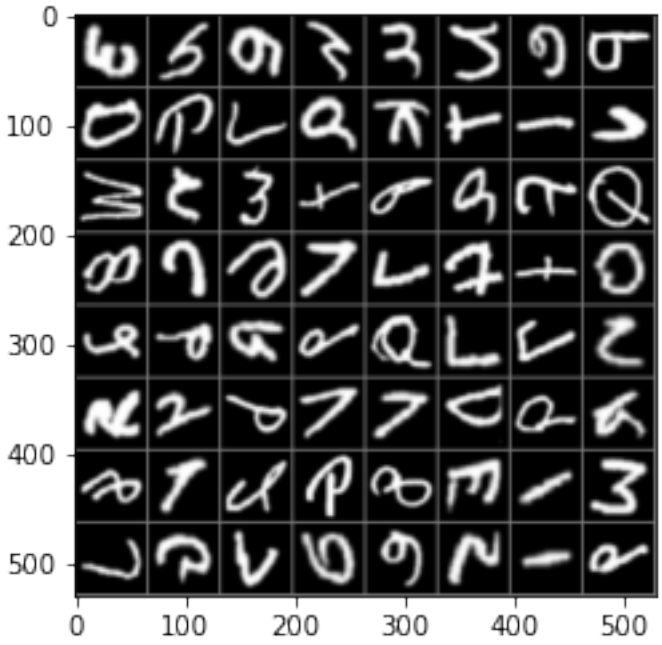
[5]:
```
import torch
import torch.nn as nn
import torch.optim as optim
import torchvision
from torch.utils.data import DataLoader
from torchvision import datasets, transforms, utils
from torchvision.datasets import EMNIST
import matplotlib.pyplot as plt
import numpy as np

# Visualize the generated images
def imshow(img):
    img = img / 2 + 0.5  # unnormalize
    np_img = img.numpy()
    plt.imshow(np.transpose(np_img, (1, 2, 0)))
    plt.show()
```

```python
# Generator
class Generator(nn.Module):
    def __init__(self, nz, ngf, nc):
        super(Generator, self).__init__()
        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, x):
        return self.main(x)

# Discriminator
class Discriminator(nn.Module):
    def __init__(self, nc, ndf):
        super(Discriminator, self).__init__()
        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, x):
        return self.main(x).view(-1)
```

```
# Hyperparameters
nz = 100
ngf = 64
ndf = 64
nc = 1  # EMNIST is grayscale, so it has only 1 channel
lr = 0.0002
beta1 = 0.5
batch_size = 128
epochs = 10
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")

# Create Generator and Discriminator
netG = Generator(nz, ngf, nc).to(device)
netD = Discriminator(nc, ndf).to(device)

# Loss function and optimizers
criterion = nn.BCELoss()
optimizerG = optim.Adam(netG.parameters(), lr=lr, betas=(beta1, 0.999))
optimizerD = optim.Adam(netD.parameters(), lr=lr, betas=(beta1, 0.999))

# Data loading and preprocessing
transform = transforms.Compose([
    transforms.Resize(64),
    transforms.ToTensor(),
    transforms.Normalize((0.5,), (0.5,)),
])

dataset = EMNIST(root='./data', split='balanced', download=True,␣
 ↪transform=transform)
dataloader = DataLoader(dataset, batch_size=batch_size, shuffle=True,␣
 ↪num_workers=2)

# Visualize a batch of real images from the EMNIST dataset
dataiter = iter(dataloader)
real_images, real_labels = dataiter.next()
grid = torchvision.utils.make_grid(real_images[:64], nrow=8, padding=2)
imshow(grid)

# ...
# Training loop
for epoch in range(epochs):
    for i, (real_images, _) in enumerate(dataloader):
        real_images = real_images.to(device)
        batch_size = real_images.size(0)

        # Train the Discriminator
```

```
        netD.zero_grad()
        real_labels = torch.full((batch_size,), 1, dtype=torch.float,␣
 ↪device=device)
        real_output = netD(real_images)
        real_loss = criterion(real_output, real_labels)
        real_loss.backward()

        noise = torch.randn(batch_size, nz, 1, 1, device=device)
        fake_images = netG(noise)
        fake_labels = torch.full((batch_size,), 0, dtype=torch.float,␣
 ↪device=device)
        fake_output = netD(fake_images.detach())
        fake_loss = criterion(fake_output, fake_labels)
        fake_loss.backward()

        D_loss = real_loss + fake_loss
        optimizerD.step()

        # Train the Generator
        netG.zero_grad()
        G_labels = torch.full((batch_size,), 1, dtype=torch.float,␣
 ↪device=device)
        G_output = netD(fake_images)
        G_loss = criterion(G_output, G_labels)
        G_loss.backward()
        optimizerG.step()

        # Calculate D(x) and D(G(z))
        D_x = real_output.mean().item()
        D_G_z1 = fake_output.mean().item()
        D_G_z2 = G_output.mean().item()

        # Print progress
        if i % 50 == 0:
            print(f"[{epoch+1}/{epochs}] [{i}/{len(dataloader)}] Loss_D:␣
 ↪{D_loss.item():.4f}\tLoss_G: {G_loss.item():.4f}\tD(x): {D_x:.4f}\tD(G(z)):␣
 ↪{D_G_z1:.4f} / {D_G_z2:.4f}")

# ...

# Generate a batch of images after training
noise = torch.randn(batch_size, nz, 1, 1, device=device)
fake_images = netG(noise).detach().cpu()

# Visualize the generated images
generated_grid = torchvision.utils.make_grid(fake_images[:64], nrow=8,␣
 ↪padding=2)
```

```
imshow(grid)

# Save the trained models
torch.save(netG.state_dict(), "dcgan_generator.pth")
torch.save(netD.state_dict(), "dcgan_discriminator.pth")
```




```
[1/10] [0/882] Loss_D: 1.4412   Loss_G: 2.7037  D(x): 0.5550    D(G(z)): 0.5590
/ 0.0709
[1/10] [50/882] Loss_D: 0.0025  Loss_G: 7.4242  D(x): 0.9992    D(G(z)): 0.0018
/ 0.0006
[1/10] [100/882] Loss_D: 0.0019 Loss_G: 7.9719  D(x): 0.9995    D(G(z)): 0.0015
/ 0.0003
[1/10] [150/882] Loss_D: 0.0006 Loss_G: 8.1752  D(x): 0.9998    D(G(z)): 0.0004
/ 0.0003
[1/10] [200/882] Loss_D: 0.0004 Loss_G: 8.6334  D(x): 0.9999    D(G(z)): 0.0002
/ 0.0002
[1/10] [250/882] Loss_D: 0.0002 Loss_G: 8.8904  D(x): 1.0000    D(G(z)): 0.0002
/ 0.0001
[1/10] [300/882] Loss_D: 0.0002 Loss_G: 9.1749  D(x): 1.0000    D(G(z)): 0.0001
/ 0.0001
[1/10] [350/882] Loss_D: 0.0001 Loss_G: 9.2901  D(x): 1.0000    D(G(z)): 0.0001
/ 0.0001
[1/10] [400/882] Loss_D: 0.0002 Loss_G: 8.9510  D(x): 1.0000    D(G(z)): 0.0002
/ 0.0001
[1/10] [450/882] Loss_D: 0.0009 Loss_G: 9.4710  D(x): 0.9997    D(G(z)): 0.0006
/ 0.0001
```

```
[1/10] [500/882] Loss_D: 0.0002 Loss_G: 9.7485  D(x): 0.9999    D(G(z)): 0.0001
/ 0.0001
[1/10] [550/882] Loss_D: 0.0318 Loss_G: 3.5893  D(x): 0.9940    D(G(z)): 0.0250
/ 0.0289
[1/10] [600/882] Loss_D: 0.5602 Loss_G: 2.5336  D(x): 0.8423    D(G(z)): 0.2874
/ 0.0909
[1/10] [650/882] Loss_D: 0.7835 Loss_G: 1.5477  D(x): 0.9278    D(G(z)): 0.4806
/ 0.2370
[1/10] [700/882] Loss_D: 0.5968 Loss_G: 1.7048  D(x): 0.7816    D(G(z)): 0.2468
/ 0.2247
[1/10] [750/882] Loss_D: 1.1111 Loss_G: 0.7210  D(x): 0.5220    D(G(z)): 0.2488
/ 0.5387
[1/10] [800/882] Loss_D: 0.4557 Loss_G: 2.5371  D(x): 0.8524    D(G(z)): 0.2243
/ 0.1084
[1/10] [850/882] Loss_D: 0.8726 Loss_G: 1.1619  D(x): 0.5810    D(G(z)): 0.2282
/ 0.3501
[2/10] [0/882] Loss_D: 0.6051   Loss_G: 1.5630  D(x): 0.6608    D(G(z)): 0.1301
/ 0.2521
[2/10] [50/882] Loss_D: 0.7845  Loss_G: 0.8275  D(x): 0.5564    D(G(z)): 0.1452
/ 0.4670
[2/10] [100/882] Loss_D: 0.5385 Loss_G: 1.9162  D(x): 0.7953    D(G(z)): 0.2505
/ 0.1648
[2/10] [150/882] Loss_D: 0.4364 Loss_G: 1.4937  D(x): 0.7701    D(G(z)): 0.1439
/ 0.2475
[2/10] [200/882] Loss_D: 0.3240 Loss_G: 1.8719  D(x): 0.8414    D(G(z)): 0.1304
/ 0.1769
[2/10] [250/882] Loss_D: 0.2693 Loss_G: 1.8602  D(x): 0.8341    D(G(z)): 0.0705
/ 0.1836
[2/10] [300/882] Loss_D: 0.2369 Loss_G: 2.2536  D(x): 0.8614    D(G(z)): 0.0764
/ 0.1283
[2/10] [350/882] Loss_D: 0.6012 Loss_G: 1.7129  D(x): 0.7582    D(G(z)): 0.2468
/ 0.2047
[2/10] [400/882] Loss_D: 0.5392 Loss_G: 2.6097  D(x): 0.8914    D(G(z)): 0.3255
/ 0.0907
[2/10] [450/882] Loss_D: 0.2489 Loss_G: 2.1143  D(x): 0.8587    D(G(z)): 0.0832
/ 0.1421
[2/10] [500/882] Loss_D: 0.6695 Loss_G: 1.8560  D(x): 0.6363    D(G(z)): 0.1136
/ 0.1956
[2/10] [550/882] Loss_D: 0.4239 Loss_G: 2.2381  D(x): 0.8478    D(G(z)): 0.2114
/ 0.1275
[2/10] [600/882] Loss_D: 1.9657 Loss_G: 0.9892  D(x): 0.2016    D(G(z)): 0.0119
/ 0.4226
[2/10] [650/882] Loss_D: 0.1898 Loss_G: 2.5399  D(x): 0.8644    D(G(z)): 0.0367
/ 0.0945
[2/10] [700/882] Loss_D: 0.4364 Loss_G: 2.0805  D(x): 0.8431    D(G(z)): 0.2151
/ 0.1505
[2/10] [750/882] Loss_D: 0.6654 Loss_G: 3.0409  D(x): 0.8730    D(G(z)): 0.3761
/ 0.0622
```

```
[2/10] [800/882] Loss_D: 0.6171 Loss_G: 2.8945  D(x): 0.9209    D(G(z)): 0.3825
/ 0.0687
[2/10] [850/882] Loss_D: 0.9565 Loss_G: 5.3156  D(x): 0.9673    D(G(z)): 0.5514
/ 0.0072
[3/10] [0/882] Loss_D: 0.1471   Loss_G: 3.7687  D(x): 0.9561    D(G(z)): 0.0937
/ 0.0301
[3/10] [50/882] Loss_D: 1.9267  Loss_G: 4.7708  D(x): 0.9929    D(G(z)): 0.8227
/ 0.0121
[3/10] [100/882] Loss_D: 0.4177 Loss_G: 3.0955  D(x): 0.8684    D(G(z)): 0.2204
/ 0.0602
[3/10] [150/882] Loss_D: 0.4231 Loss_G: 5.8882  D(x): 0.9781    D(G(z)): 0.3070
/ 0.0050
[3/10] [200/882] Loss_D: 0.2315 Loss_G: 2.6696  D(x): 0.8707    D(G(z)): 0.0806
/ 0.0926
[3/10] [250/882] Loss_D: 0.1751 Loss_G: 2.9621  D(x): 0.9043    D(G(z)): 0.0671
/ 0.0685
[3/10] [300/882] Loss_D: 0.1577 Loss_G: 3.5811  D(x): 0.9364    D(G(z)): 0.0836
/ 0.0369
[3/10] [350/882] Loss_D: 0.4375 Loss_G: 2.3459  D(x): 0.9097    D(G(z)): 0.2658
/ 0.1237
[3/10] [400/882] Loss_D: 0.6102 Loss_G: 2.2487  D(x): 0.6739    D(G(z)): 0.1509
/ 0.1434
[3/10] [450/882] Loss_D: 0.1115 Loss_G: 3.5957  D(x): 0.9612    D(G(z)): 0.0667
/ 0.0382
[3/10] [500/882] Loss_D: 4.2194 Loss_G: 0.5368  D(x): 0.0333    D(G(z)): 0.0028
/ 0.6624
[3/10] [550/882] Loss_D: 0.3758 Loss_G: 2.3010  D(x): 0.8231    D(G(z)): 0.1461
/ 0.1272
[3/10] [600/882] Loss_D: 0.4257 Loss_G: 3.0028  D(x): 0.8922    D(G(z)): 0.2485
/ 0.0652
[3/10] [650/882] Loss_D: 0.1077 Loss_G: 4.1545  D(x): 0.9596    D(G(z)): 0.0617
/ 0.0213
[3/10] [700/882] Loss_D: 0.2615 Loss_G: 2.5973  D(x): 0.8563    D(G(z)): 0.0891
/ 0.1029
[3/10] [750/882] Loss_D: 0.1202 Loss_G: 3.3427  D(x): 0.9443    D(G(z)): 0.0575
/ 0.0471
[3/10] [800/882] Loss_D: 0.6115 Loss_G: 2.7350  D(x): 0.7944    D(G(z)): 0.2675
/ 0.0976
[3/10] [850/882] Loss_D: 0.3329 Loss_G: 4.9626  D(x): 0.9789    D(G(z)): 0.2466
/ 0.0110
[4/10] [0/882] Loss_D: 0.2867   Loss_G: 4.7600  D(x): 0.9526    D(G(z)): 0.1876
/ 0.0131
[4/10] [50/882] Loss_D: 0.8615  Loss_G: 2.9440  D(x): 0.8968    D(G(z)): 0.4701
/ 0.0766
[4/10] [100/882] Loss_D: 0.4423 Loss_G: 2.0043  D(x): 0.7067    D(G(z)): 0.0420
/ 0.1835
[4/10] [150/882] Loss_D: 0.4205 Loss_G: 2.7121  D(x): 0.7893    D(G(z)): 0.1456
/ 0.0827
```

```
[4/10] [200/882] Loss_D: 0.4502 Loss_G: 3.2581  D(x): 0.9659    D(G(z)): 0.3014
/ 0.0556
[4/10] [250/882] Loss_D: 0.0888 Loss_G: 4.3567  D(x): 0.9502    D(G(z)): 0.0351
/ 0.0189
[4/10] [300/882] Loss_D: 2.9768 Loss_G: 0.9133  D(x): 0.1134    D(G(z)): 0.0151
/ 0.5041
[4/10] [350/882] Loss_D: 0.6424 Loss_G: 2.2922  D(x): 0.8790    D(G(z)): 0.3633
/ 0.1272
[4/10] [400/882] Loss_D: 0.8410 Loss_G: 2.5850  D(x): 0.9100    D(G(z)): 0.4591
/ 0.1085
[4/10] [450/882] Loss_D: 0.1557 Loss_G: 2.6091  D(x): 0.8875    D(G(z)): 0.0300
/ 0.0977
[4/10] [500/882] Loss_D: 0.0729 Loss_G: 4.6602  D(x): 0.9824    D(G(z)): 0.0526
/ 0.0132
[4/10] [550/882] Loss_D: 0.2747 Loss_G: 2.9626  D(x): 0.8763    D(G(z)): 0.1165
/ 0.0724
[4/10] [600/882] Loss_D: 0.1037 Loss_G: 4.1506  D(x): 0.9528    D(G(z)): 0.0502
/ 0.0234
[4/10] [650/882] Loss_D: 0.2569 Loss_G: 2.1809  D(x): 0.9666    D(G(z)): 0.1839
/ 0.1497
[4/10] [700/882] Loss_D: 0.1089 Loss_G: 4.0682  D(x): 0.9603    D(G(z)): 0.0614
/ 0.0250
[4/10] [750/882] Loss_D: 0.0523 Loss_G: 3.4526  D(x): 0.9821    D(G(z)): 0.0329
/ 0.0441
[4/10] [800/882] Loss_D: 0.0361 Loss_G: 5.5795  D(x): 0.9775    D(G(z)): 0.0130
/ 0.0061
[4/10] [850/882] Loss_D: 0.2690 Loss_G: 3.0911  D(x): 0.8749    D(G(z)): 0.1136
/ 0.0652
[5/10] [0/882] Loss_D: 0.1825   Loss_G: 2.7134  D(x): 0.9073    D(G(z)): 0.0732
/ 0.0958
[5/10] [50/882] Loss_D: 0.0594  Loss_G: 4.3079  D(x): 0.9724    D(G(z)): 0.0295
/ 0.0227
[5/10] [100/882] Loss_D: 0.2967 Loss_G: 2.6529  D(x): 0.7981    D(G(z)): 0.0429
/ 0.0967
[5/10] [150/882] Loss_D: 0.0609 Loss_G: 4.4050  D(x): 0.9679    D(G(z)): 0.0259
/ 0.0172
[5/10] [200/882] Loss_D: 0.0569 Loss_G: 3.8947  D(x): 0.9679    D(G(z)): 0.0227
/ 0.0293
[5/10] [250/882] Loss_D: 0.1711 Loss_G: 3.5911  D(x): 0.8934    D(G(z)): 0.0494
/ 0.0420
[5/10] [300/882] Loss_D: 0.6944 Loss_G: 3.8764  D(x): 0.9412    D(G(z)): 0.4132
/ 0.0312
[5/10] [350/882] Loss_D: 0.2405 Loss_G: 3.7573  D(x): 0.9182    D(G(z)): 0.1346
/ 0.0324
[5/10] [400/882] Loss_D: 3.0181 Loss_G: 6.9661  D(x): 0.9981    D(G(z)): 0.8709
/ 0.0037
[5/10] [450/882] Loss_D: 0.1040 Loss_G: 4.2818  D(x): 0.9758    D(G(z)): 0.0727
/ 0.0203
```

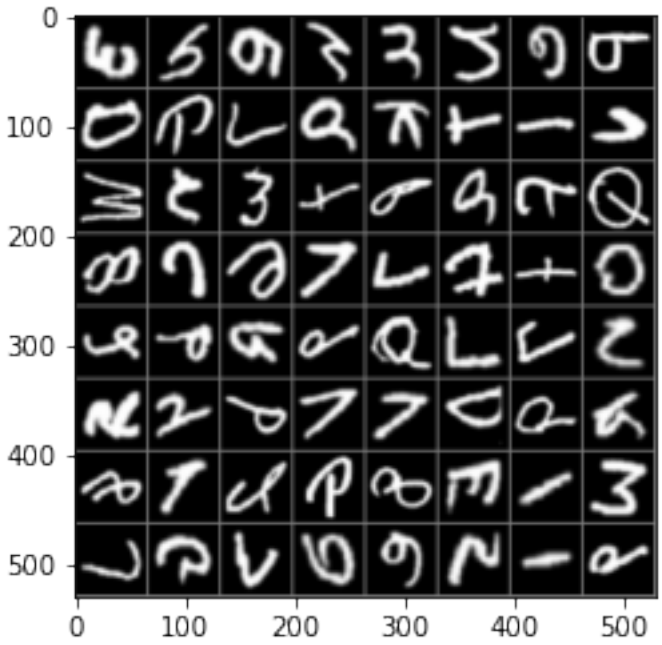
```
[5/10] [500/882] Loss_D: 0.0241 Loss_G: 5.0198  D(x): 0.9907    D(G(z)): 0.0144
/ 0.0105
[5/10] [550/882] Loss_D: 1.0344 Loss_G: 2.0503  D(x): 0.4860    D(G(z)): 0.0770
/ 0.1978
[5/10] [600/882] Loss_D: 0.6824 Loss_G: 2.8876  D(x): 0.7198    D(G(z)): 0.2328
/ 0.0793
[5/10] [650/882] Loss_D: 0.0734 Loss_G: 4.0906  D(x): 0.9888    D(G(z)): 0.0579
/ 0.0246
[5/10] [700/882] Loss_D: 0.0579 Loss_G: 4.7638  D(x): 0.9835    D(G(z)): 0.0387
/ 0.0136
[5/10] [750/882] Loss_D: 0.0721 Loss_G: 5.0217  D(x): 0.9832    D(G(z)): 0.0504
/ 0.0102
[5/10] [800/882] Loss_D: 0.9709 Loss_G: 1.3669  D(x): 0.6300    D(G(z)): 0.3357
/ 0.3101
[5/10] [850/882] Loss_D: 0.8451 Loss_G: 2.8969  D(x): 0.8983    D(G(z)): 0.4553
/ 0.0777
[6/10] [0/882] Loss_D: 0.5591   Loss_G: 4.0303  D(x): 0.8830    D(G(z)): 0.3046
/ 0.0261
[6/10] [50/882] Loss_D: 0.2194  Loss_G: 2.9256  D(x): 0.8972    D(G(z)): 0.0965
/ 0.0717
[6/10] [100/882] Loss_D: 0.2264 Loss_G: 3.8239  D(x): 0.9565    D(G(z)): 0.1550
/ 0.0286
[6/10] [150/882] Loss_D: 0.3175 Loss_G: 3.1479  D(x): 0.9449    D(G(z)): 0.2055
/ 0.0632
[6/10] [200/882] Loss_D: 0.0431 Loss_G: 4.4987  D(x): 0.9710    D(G(z)): 0.0131
/ 0.0189
[6/10] [250/882] Loss_D: 0.0380 Loss_G: 4.8379  D(x): 0.9760    D(G(z)): 0.0130
/ 0.0120
[6/10] [300/882] Loss_D: 0.0221 Loss_G: 5.0273  D(x): 0.9948    D(G(z)): 0.0165
/ 0.0101
[6/10] [350/882] Loss_D: 0.0324 Loss_G: 5.7979  D(x): 0.9777    D(G(z)): 0.0092
/ 0.0055
[6/10] [400/882] Loss_D: 0.5577 Loss_G: 1.3955  D(x): 0.6249    D(G(z)): 0.0222
/ 0.3134
[6/10] [450/882] Loss_D: 0.1446 Loss_G: 2.9403  D(x): 0.9338    D(G(z)): 0.0676
/ 0.0745
[6/10] [500/882] Loss_D: 0.0832 Loss_G: 4.3300  D(x): 0.9793    D(G(z)): 0.0586
/ 0.0191
[6/10] [550/882] Loss_D: 0.5780 Loss_G: 4.4697  D(x): 0.9769    D(G(z)): 0.3739
/ 0.0181
[6/10] [600/882] Loss_D: 0.1016 Loss_G: 4.6223  D(x): 0.9548    D(G(z)): 0.0509
/ 0.0151
[6/10] [650/882] Loss_D: 0.1610 Loss_G: 4.0265  D(x): 0.9295    D(G(z)): 0.0748
/ 0.0268
[6/10] [700/882] Loss_D: 1.0436 Loss_G: 1.8040  D(x): 0.4605    D(G(z)): 0.0157
/ 0.2622
[6/10] [750/882] Loss_D: 0.4442 Loss_G: 5.3621  D(x): 0.9863    D(G(z)): 0.3086
/ 0.0067
```

```
[6/10] [800/882] Loss_D: 0.0397 Loss_G: 4.3664  D(x): 0.9812    D(G(z)): 0.0199
/ 0.0215
[6/10] [850/882] Loss_D: 0.5905 Loss_G: 1.9971  D(x): 0.6337    D(G(z)): 0.0601
/ 0.1915
[7/10] [0/882] Loss_D: 0.2130   Loss_G: 2.4492  D(x): 0.9474    D(G(z)): 0.1323
/ 0.1369
[7/10] [50/882] Loss_D: 0.0509  Loss_G: 4.3955  D(x): 0.9622    D(G(z)): 0.0096
/ 0.0201
[7/10] [100/882] Loss_D: 1.0321 Loss_G: 1.8905  D(x): 0.7136    D(G(z)): 0.4315
/ 0.2021
[7/10] [150/882] Loss_D: 0.1291 Loss_G: 3.9691  D(x): 0.9739    D(G(z)): 0.0924
/ 0.0274
[7/10] [200/882] Loss_D: 0.1094 Loss_G: 3.9686  D(x): 0.9489    D(G(z)): 0.0515
/ 0.0282
[7/10] [250/882] Loss_D: 0.0450 Loss_G: 4.4821  D(x): 0.9746    D(G(z)): 0.0185
/ 0.0178
[7/10] [300/882] Loss_D: 0.0363 Loss_G: 5.2656  D(x): 0.9917    D(G(z)): 0.0268
/ 0.0091
[7/10] [350/882] Loss_D: 0.0267 Loss_G: 5.0880  D(x): 0.9853    D(G(z)): 0.0115
/ 0.0116
[7/10] [400/882] Loss_D: 0.2543 Loss_G: 3.4620  D(x): 0.9240    D(G(z)): 0.1460
/ 0.0425
[7/10] [450/882] Loss_D: 0.1636 Loss_G: 4.0937  D(x): 0.9567    D(G(z)): 0.1046
/ 0.0258
[7/10] [500/882] Loss_D: 0.0503 Loss_G: 4.6850  D(x): 0.9661    D(G(z)): 0.0144
/ 0.0186
[7/10] [550/882] Loss_D: 0.1436 Loss_G: 7.4163  D(x): 0.9922    D(G(z)): 0.1167
/ 0.0010
[7/10] [600/882] Loss_D: 0.1637 Loss_G: 3.6958  D(x): 0.9801    D(G(z)): 0.1228
/ 0.0369
[7/10] [650/882] Loss_D: 0.1321 Loss_G: 4.2038  D(x): 0.9826    D(G(z)): 0.1036
/ 0.0208
[7/10] [700/882] Loss_D: 0.0636 Loss_G: 5.0881  D(x): 0.9910    D(G(z)): 0.0512
/ 0.0104
[7/10] [750/882] Loss_D: 0.0273 Loss_G: 5.2532  D(x): 0.9799    D(G(z)): 0.0066
/ 0.0087
[7/10] [800/882] Loss_D: 0.0299 Loss_G: 5.3820  D(x): 0.9786    D(G(z)): 0.0079
/ 0.0094
[7/10] [850/882] Loss_D: 0.0129 Loss_G: 6.0873  D(x): 0.9918    D(G(z)): 0.0046
/ 0.0045
[8/10] [0/882] Loss_D: 0.5926   Loss_G: 2.5281  D(x): 0.8632    D(G(z)): 0.3143
/ 0.1096
[8/10] [50/882] Loss_D: 1.1064  Loss_G: 7.7863  D(x): 0.9866    D(G(z)): 0.6067
/ 0.0006
[8/10] [100/882] Loss_D: 0.2049 Loss_G: 3.9952  D(x): 0.9318    D(G(z)): 0.1074
/ 0.0305
[8/10] [150/882] Loss_D: 0.2704 Loss_G: 4.1032  D(x): 0.9588    D(G(z)): 0.1843
/ 0.0236
```

```
[8/10] [200/882] Loss_D: 0.1363 Loss_G: 3.9389  D(x): 0.8840    D(G(z)): 0.0050
/ 0.0301
[8/10] [250/882] Loss_D: 0.0333 Loss_G: 4.7803  D(x): 0.9791    D(G(z)): 0.0118
/ 0.0135
[8/10] [300/882] Loss_D: 0.3214 Loss_G: 2.5192  D(x): 0.7993    D(G(z)): 0.0695
/ 0.1289
[8/10] [350/882] Loss_D: 0.0558 Loss_G: 4.5010  D(x): 0.9806    D(G(z)): 0.0343
/ 0.0167
[8/10] [400/882] Loss_D: 0.5470 Loss_G: 2.7880  D(x): 0.6908    D(G(z)): 0.0902
/ 0.1009
[8/10] [450/882] Loss_D: 0.1155 Loss_G: 3.2487  D(x): 0.9170    D(G(z)): 0.0248
/ 0.0602
[8/10] [500/882] Loss_D: 0.0323 Loss_G: 4.5678  D(x): 0.9908    D(G(z)): 0.0221
/ 0.0166
[8/10] [550/882] Loss_D: 0.2844 Loss_G: 3.4034  D(x): 0.8428    D(G(z)): 0.0830
/ 0.0502
[8/10] [600/882] Loss_D: 0.1233 Loss_G: 3.4530  D(x): 0.9196    D(G(z)): 0.0338
/ 0.0462
[8/10] [650/882] Loss_D: 0.0797 Loss_G: 3.9409  D(x): 0.9649    D(G(z)): 0.0405
/ 0.0279
[8/10] [700/882] Loss_D: 0.0863 Loss_G: 4.4605  D(x): 0.9764    D(G(z)): 0.0571
/ 0.0193
[8/10] [750/882] Loss_D: 0.0274 Loss_G: 5.0967  D(x): 0.9865    D(G(z)): 0.0134
/ 0.0109
[8/10] [800/882] Loss_D: 0.0187 Loss_G: 5.3631  D(x): 0.9926    D(G(z)): 0.0111
/ 0.0078
[8/10] [850/882] Loss_D: 0.0122 Loss_G: 6.2027  D(x): 0.9901    D(G(z)): 0.0021
/ 0.0035
[9/10] [0/882] Loss_D: 0.9916   Loss_G: 1.5166  D(x): 0.6309    D(G(z)): 0.3039
/ 0.2755
[9/10] [50/882] Loss_D: 0.2917  Loss_G: 3.1335  D(x): 0.8794    D(G(z)): 0.1162
/ 0.0677
[9/10] [100/882] Loss_D: 0.1069 Loss_G: 5.2794  D(x): 0.9866    D(G(z)): 0.0864
/ 0.0075
[9/10] [150/882] Loss_D: 0.0545 Loss_G: 4.1988  D(x): 0.9583    D(G(z)): 0.0106
/ 0.0236
[9/10] [200/882] Loss_D: 0.0262 Loss_G: 5.1123  D(x): 0.9937    D(G(z)): 0.0194
/ 0.0093
[9/10] [250/882] Loss_D: 0.7751 Loss_G: 2.4119  D(x): 0.7961    D(G(z)): 0.3821
/ 0.1116
[9/10] [300/882] Loss_D: 0.3862 Loss_G: 5.2030  D(x): 0.9619    D(G(z)): 0.2636
/ 0.0084
[9/10] [350/882] Loss_D: 0.4932 Loss_G: 2.0473  D(x): 0.7811    D(G(z)): 0.1742
/ 0.1746
[9/10] [400/882] Loss_D: 0.0993 Loss_G: 3.8030  D(x): 0.9178    D(G(z)): 0.0087
/ 0.0329
[9/10] [450/882] Loss_D: 0.0263 Loss_G: 4.9524  D(x): 0.9815    D(G(z)): 0.0073
/ 0.0125
```

```
[9/10] [500/882] Loss_D: 0.0621 Loss_G: 6.4380  D(x): 0.9968    D(G(z)): 0.0538
/ 0.0026
[9/10] [550/882] Loss_D: 0.0213 Loss_G: 5.8027  D(x): 0.9948    D(G(z)): 0.0158
/ 0.0050
[9/10] [600/882] Loss_D: 0.0319 Loss_G: 5.0048  D(x): 0.9843    D(G(z)): 0.0152
/ 0.0128
[9/10] [650/882] Loss_D: 0.0497 Loss_G: 4.6081  D(x): 0.9583    D(G(z)): 0.0061
/ 0.0172
[9/10] [700/882] Loss_D: 0.0503 Loss_G: 4.3932  D(x): 0.9536    D(G(z)): 0.0011
/ 0.0212
[9/10] [750/882] Loss_D: 0.6070 Loss_G: 1.6911  D(x): 0.7164    D(G(z)): 0.1910
/ 0.2248
[9/10] [800/882] Loss_D: 0.2238 Loss_G: 2.6626  D(x): 0.8429    D(G(z)): 0.0225
/ 0.1045
[9/10] [850/882] Loss_D: 0.1041 Loss_G: 4.0071  D(x): 0.9577    D(G(z)): 0.0561
/ 0.0293
[10/10] [0/882] Loss_D: 1.0293  Loss_G: 4.0783  D(x): 0.9274    D(G(z)): 0.5315
/ 0.0326
[10/10] [50/882] Loss_D: 0.2543 Loss_G: 2.9967  D(x): 0.8621    D(G(z)): 0.0822
/ 0.0769
[10/10] [100/882] Loss_D: 0.3742        Loss_G: 2.8805  D(x): 0.7707    D(G(z)):
0.0716 / 0.0869
[10/10] [150/882] Loss_D: 0.1095        Loss_G: 4.8364  D(x): 0.9854    D(G(z)):
0.0843 / 0.0115
[10/10] [200/882] Loss_D: 0.0879        Loss_G: 4.3294  D(x): 0.9344    D(G(z)):
0.0166 / 0.0240
[10/10] [250/882] Loss_D: 0.0575        Loss_G: 4.1530  D(x): 0.9626    D(G(z)):
0.0176 / 0.0242
[10/10] [300/882] Loss_D: 0.0393        Loss_G: 5.1227  D(x): 0.9668    D(G(z)):
0.0049 / 0.0101
[10/10] [350/882] Loss_D: 0.9741        Loss_G: 6.8939  D(x): 0.9974    D(G(z)):
0.5497 / 0.0016
[10/10] [400/882] Loss_D: 0.0540        Loss_G: 4.4902  D(x): 0.9727    D(G(z)):
0.0252 / 0.0173
[10/10] [450/882] Loss_D: 0.0925        Loss_G: 4.1639  D(x): 0.9806    D(G(z)):
0.0674 / 0.0241
[10/10] [500/882] Loss_D: 0.0633        Loss_G: 4.7410  D(x): 0.9866    D(G(z)):
0.0472 / 0.0136
[10/10] [550/882] Loss_D: 0.1995        Loss_G: 3.5567  D(x): 0.9360    D(G(z)):
0.1159 / 0.0416
[10/10] [600/882] Loss_D: 0.0930        Loss_G: 4.1264  D(x): 0.9761    D(G(z)):
0.0626 / 0.0241
[10/10] [650/882] Loss_D: 0.1067        Loss_G: 3.6372  D(x): 0.9419    D(G(z)):
0.0396 / 0.0431
[10/10] [700/882] Loss_D: 0.0316        Loss_G: 5.0580  D(x): 0.9805    D(G(z)):
0.0112 / 0.0117
[10/10] [750/882] Loss_D: 0.0209        Loss_G: 4.6178  D(x): 0.9926    D(G(z)):
0.0132 / 0.0174
```

```
[10/10] [800/882] Loss_D: 0.0148        Loss_G: 6.2889  D(x): 0.9885    D(G(z)):
0.0030 / 0.0037
[10/10] [850/882] Loss_D: 0.0142        Loss_G: 7.1237  D(x): 0.9870    D(G(z)):
0.0009 / 0.0013
```



0.3 DCGAN - CIFAR-10 (Airplane)

[4]:
```
import torch
import torch.nn as nn
import torch.optim as optim
from torch.utils.data import DataLoader
from torchvision import transforms, datasets, utils
from torch.utils.data import Dataset
import matplotlib.pyplot as plt
import numpy as np

# Visualize the generated images
def imshow(img):
    img = img / 2 + 0.5     # unnormalize
    npimg = img.numpy()
    plt.imshow(np.transpose(npimg, (1, 2, 0)))
    plt.show()

class AirplaneDataset(Dataset):
    def __init__(self, dataset):
```

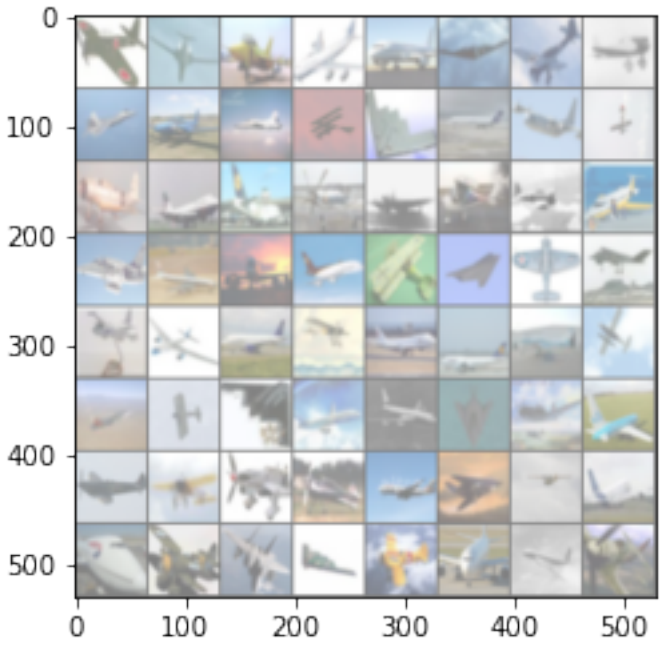
```
        self.dataset = dataset
        self.airplane_indices = [i for i, (_, label) in enumerate(self.dataset)␣
 ↪if label == 0]

    def __getitem__(self, index):
        image, label = self.dataset[self.airplane_indices[index]]
        return image, label

    def __len__(self):
        return len(self.airplane_indices)

# Generator
class Generator(nn.Module):
    def __init__(self):
        super(Generator, self).__init__()
        self.main = nn.Sequential(
            nn.ConvTranspose2d(100, 512, 4, 1, 0, bias=False),
            nn.BatchNorm2d(512),
            nn.ReLU(True),
            nn.ConvTranspose2d(512, 256, 4, 2, 1, bias=False),
            nn.BatchNorm2d(256),
            nn.ReLU(True),
            nn.ConvTranspose2d(256, 128, 4, 2, 1, bias=False),
            nn.BatchNorm2d(128),
            nn.ReLU(True),
            nn.ConvTranspose2d(128, 3, 4, 2, 1, bias=False),
            nn.Tanh()
        )

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

# Discriminator
class Discriminator(nn.Module):
    def __init__(self):
        super(Discriminator, self).__init__()
        self.main = nn.Sequential(
            nn.Conv2d(3, 128, 4, 2, 1, bias=False),
            nn.LeakyReLU(0.2, inplace=True),
            nn.Conv2d(128, 256, 4, 2, 1, bias=False),
            nn.BatchNorm2d(256),
            nn.LeakyReLU(0.2, inplace=True),
            nn.Conv2d(256, 512, 4, 2, 1, bias=False),
            nn.BatchNorm2d(512),
            nn.LeakyReLU(0.2, inplace=True),
            nn.AdaptiveAvgPool2d((1, 1)),
            nn.Conv2d(512, 1, 1, 1, 0, bias=False),
```

```
            nn.Sigmoid()
        )

    def forward(self, input):
        return self.main(input).view(-1, 1).squeeze(1)

# Load the CIFAR-10 dataset
transform = transforms.Compose([
    transforms.Resize(64),
    transforms.ToTensor(),
    transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))
])

cifar_dataset = datasets.CIFAR10(root='./data', train=True, download=True,␣
 ↪transform=transform)
airplane_dataset = AirplaneDataset(cifar_dataset)
dataloader = DataLoader(airplane_dataset, batch_size=64, shuffle=True,␣
 ↪num_workers=2)

# Get a batch of images from the dataset
dataiter = iter(dataloader)
images, labels = dataiter.next()

# Create a grid of images and display it
grid = utils.make_grid(images, nrow=8, padding=2, normalize=True)
print("Images from the CIFAR-10 dataset:")
imshow(grid)

device = torch.device("cuda" if torch.cuda.is_available() else "cpu")

# Create Generator and Discriminator
netG = Generator().to(device)
netD = Discriminator().to(device)

# Number of epochs
num_epochs = 10

# Initialize BCELoss function
criterion = nn.BCELoss()

# Create batch of latent vectors that we will use to visualize the progression␣
 ↪of the generator
fixed_noise = torch.randn(64, 100, 1, 1, device=device)

# Establish convention for real and fake labels during training
real_label = 1
fake_label = 0
```

```
# Setup Adam optimizers for both G and D
optimizerD = optim.Adam(netD.parameters(), lr=0.0002, betas=(0.5, 0.999))
optimizerG = optim.Adam(netG.parameters(), lr=0.0002, betas=(0.5, 0.999))

# Training loop
for epoch in range(num_epochs):
    for i, data in enumerate(dataloader, 0):
        # Update D network: maximize log(D(x)) + log(1 - D(G(z)))
        # Train with all-real batch
        netD.zero_grad()
        real_data = data[0].to(device)
        batch_size = real_data.size(0)
        label = torch.full((batch_size,), real_label, dtype=torch.float,␣
 ↪device=device)
        output = netD(real_data)
        errD_real = criterion(output, label)
        errD_real.backward()
        D_x = output.mean().item()

        # Train with all-fake batch
        noise = torch.randn(batch_size, 100, 1, 1, device=device)
        fake_data = netG(noise)
        label.fill_(fake_label)
        output = netD(fake_data.detach())
        errD_fake = criterion(output, label)
        errD_fake.backward()
        D_G_z1 = output.mean().item()
        errD = errD_real + errD_fake
        optimizerD.step()

        # Update G network: maximize log(D(G(z)))
        netG.zero_grad()
        label.fill_(real_label)
        output = netD(fake_data)
        errG = criterion(output, label)
        errG.backward()
        D_G_z2 = output.mean().item()
        optimizerG.step()

        # Output training stats
        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))
```

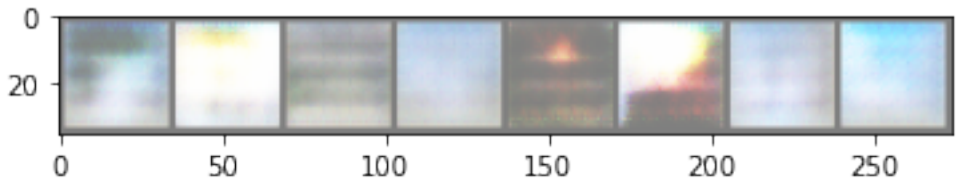
```
# Save the trained Generator and Discriminator
torch.save(netG.state_dict(), 'generator_cifar10.pth')
torch.save(netD.state_dict(), 'discriminator_cifar10.pth')

# Generate a batch of images after training
noise = torch.randn(batch_size, 100, 1, 1, device=device)
fake_images = netG(noise).detach()

# Move the images back to the CPU and convert them to a grid
fake_images = fake_images.cpu()
grid = utils.make_grid(fake_images, nrow=8, padding=2, normalize=True)

# Show the grid of images
imshow(grid)
```

```
Files already downloaded and verified
Images from the CIFAR-10 dataset:
```




```
[0/10][0/79]    Loss_D: 1.3948  Loss_G: 0.7287  D(x): 0.4950    D(G(z)): 0.4956
/ 0.4831
[0/10][50/79]   Loss_D: 1.0142  Loss_G: 1.0569  D(x): 0.5852    D(G(z)): 0.3631
/ 0.3480
[1/10][0/79]    Loss_D: 1.1486  Loss_G: 0.9254  D(x): 0.5625    D(G(z)): 0.4248
/ 0.4013
[1/10][50/79]   Loss_D: 0.9612  Loss_G: 1.1287  D(x): 0.6265    D(G(z)): 0.3869
/ 0.3286
```

```
[2/10][0/79]    Loss_D: 1.0177  Loss_G: 1.1913  D(x): 0.5882    D(G(z)): 0.3477
/ 0.3335
[2/10][50/79]   Loss_D: 1.1018  Loss_G: 1.3165  D(x): 0.5747    D(G(z)): 0.3236
/ 0.3588
[3/10][0/79]    Loss_D: 0.8497  Loss_G: 1.1664  D(x): 0.6588    D(G(z)): 0.3409
/ 0.3209
[3/10][50/79]   Loss_D: 0.6887  Loss_G: 1.3367  D(x): 0.7046    D(G(z)): 0.2799
/ 0.2746
[4/10][0/79]    Loss_D: 0.8538  Loss_G: 1.3357  D(x): 0.6586    D(G(z)): 0.3209
/ 0.2811
[4/10][50/79]   Loss_D: 0.6172  Loss_G: 1.4860  D(x): 0.7152    D(G(z)): 0.2364
/ 0.2530
[5/10][0/79]    Loss_D: 0.5102  Loss_G: 1.5910  D(x): 0.7774    D(G(z)): 0.2223
/ 0.2088
[5/10][50/79]   Loss_D: 0.4552  Loss_G: 1.8231  D(x): 0.7636    D(G(z)): 0.1608
/ 0.1669
[6/10][0/79]    Loss_D: 0.4143  Loss_G: 1.9662  D(x): 0.8073    D(G(z)): 0.1712
/ 0.1778
[6/10][50/79]   Loss_D: 0.3774  Loss_G: 1.8868  D(x): 0.8286    D(G(z)): 0.1701
/ 0.1548
[7/10][0/79]    Loss_D: 0.3880  Loss_G: 2.0402  D(x): 0.8468    D(G(z)): 0.1869
/ 0.1593
[7/10][50/79]   Loss_D: 0.2012  Loss_G: 2.4957  D(x): 0.8991    D(G(z)): 0.0893
/ 0.0850
[8/10][0/79]    Loss_D: 0.1915  Loss_G: 2.6941  D(x): 0.9039    D(G(z)): 0.0824
/ 0.0715
[8/10][50/79]   Loss_D: 0.1238  Loss_G: 2.9543  D(x): 0.9448    D(G(z)): 0.0645
/ 0.0536
[9/10][0/79]    Loss_D: 0.1135  Loss_G: 3.3470  D(x): 0.9331    D(G(z)): 0.0429
/ 0.0363
[9/10][50/79]   Loss_D: 1.0403  Loss_G: 3.0628  D(x): 0.6553    D(G(z)): 0.1640
/ 0.2879
```



[6]:
```
import torch
import torch.nn as nn
import torch.optim as optim
from torch.utils.data import DataLoader
from torchvision import transforms, datasets, utils
```

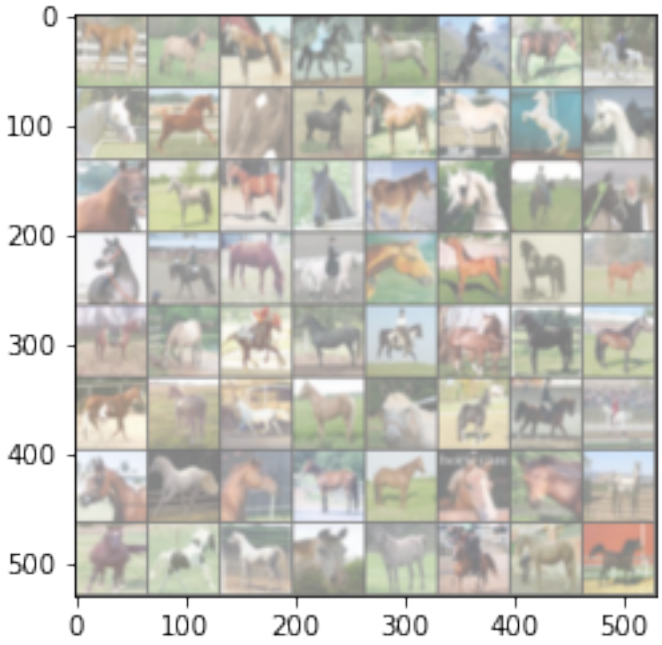
```
from torch.utils.data import Dataset
import matplotlib.pyplot as plt
import numpy as np

# Visualize the generated images
def imshow(img):
    img = img / 2 + 0.5     # unnormalize
    npimg = img.numpy()
    plt.imshow(np.transpose(npimg, (1, 2, 0)))
    plt.show()

class HorseDataset(torch.utils.data.Dataset):
    def __init__(self, dataset):
        self.dataset = dataset
        self.horse_indices = [i for i, (_, label) in enumerate(self.dataset) if␣
 ↪label == 7]

    def __getitem__(self, index):
        return self.dataset[self.horse_indices[index]]

    def __len__(self):
        return len(self.horse_indices)

# Generator
class Generator(nn.Module):
    def __init__(self):
        super(Generator, self).__init__()
        self.main = nn.Sequential(
            nn.ConvTranspose2d(100, 512, 4, 1, 0, bias=False),
            nn.BatchNorm2d(512),
            nn.ReLU(True),
            nn.ConvTranspose2d(512, 256, 4, 2, 1, bias=False),
            nn.BatchNorm2d(256),
            nn.ReLU(True),
            nn.ConvTranspose2d(256, 128, 4, 2, 1, bias=False),
            nn.BatchNorm2d(128),
            nn.ReLU(True),
            nn.ConvTranspose2d(128, 3, 4, 2, 1, bias=False),
            nn.Tanh()
        )

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

# Discriminator
class Discriminator(nn.Module):
    def __init__(self):
```

```
        super(Discriminator, self).__init__()
        self.main = nn.Sequential(
            nn.Conv2d(3, 128, 4, 2, 1, bias=False),
            nn.LeakyReLU(0.2, inplace=True),
            nn.Conv2d(128, 256, 4, 2, 1, bias=False),
            nn.BatchNorm2d(256),
            nn.LeakyReLU(0.2, inplace=True),
            nn.Conv2d(256, 512, 4, 2, 1, bias=False),
            nn.BatchNorm2d(512),
            nn.LeakyReLU(0.2, inplace=True),
            nn.AdaptiveAvgPool2d((1, 1)),
            nn.Conv2d(512, 1, 1, 1, 0, bias=False),
            nn.Sigmoid()
        )

    def forward(self, input):
        return self.main(input).view(-1, 1).squeeze(1)

# Load the CIFAR-10 dataset
transform = transforms.Compose([
    transforms.Resize(64),
    transforms.ToTensor(),
    transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))
])

cifar_dataset = datasets.CIFAR10(root='./data', train=True, download=True,␣
 ↪transform=transform)
train_horse_dataset = HorseDataset(cifar_dataset)
dataloader = DataLoader(train_horse_dataset, batch_size=64, shuffle=True,␣
 ↪num_workers=2)

# Get a batch of images from the dataset
dataiter = iter(dataloader)
images, labels = dataiter.next()

# Create a grid of images and display it
grid = utils.make_grid(images, nrow=8, padding=2, normalize=True)
print("Images from the CIFAR-10 dataset:")
imshow(grid)

device = torch.device("cuda" if torch.cuda.is_available() else "cpu")

# Create Generator and Discriminator
netG = Generator().to(device)
netD = Discriminator().to(device)

# Number of epochs
```

```
num_epochs = 10

# Initialize BCELoss function
criterion = nn.BCELoss()

# Create batch of latent vectors that we will use to visualize the progression␣
 ↪of the generator
fixed_noise = torch.randn(64, 100, 1, 1, device=device)

# Establish convention for real and fake labels during training
real_label = 1
fake_label = 0

# Setup Adam optimizers for both G and D
optimizerD = optim.Adam(netD.parameters(), lr=0.0002, betas=(0.5, 0.999))
optimizerG = optim.Adam(netG.parameters(), lr=0.0002, betas=(0.5, 0.999))

# Training loop
for epoch in range(num_epochs):
    for i, data in enumerate(dataloader, 0):
        # Update D network: maximize log(D(x)) + log(1 - D(G(z)))
        # Train with all-real batch
        netD.zero_grad()
        real_data = data[0].to(device)
        batch_size = real_data.size(0)
        label = torch.full((batch_size,), real_label, dtype=torch.float,␣
 ↪device=device)
        output = netD(real_data)
        errD_real = criterion(output, label)
        errD_real.backward()
        D_x = output.mean().item()

        # Train with all-fake batch
        noise = torch.randn(batch_size, 100, 1, 1, device=device)
        fake_data = netG(noise)
        label.fill_(fake_label)
        output = netD(fake_data.detach())
        errD_fake = criterion(output, label)
        errD_fake.backward()
        D_G_z1 = output.mean().item()
        errD = errD_real + errD_fake
        optimizerD.step()

        # Update G network: maximize log(D(G(z)))
        netG.zero_grad()
        label.fill_(real_label)
        output = netD(fake_data)
```

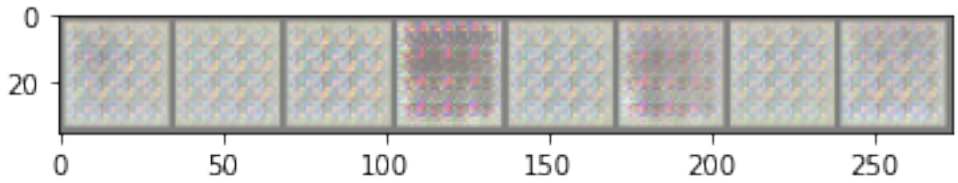
```
        errG = criterion(output, label)
        errG.backward()
        D_G_z2 = output.mean().item()
        optimizerG.step()

        # Output training stats
        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))

# Save the trained Generator and Discriminator
torch.save(netG.state_dict(), 'generator_cifar10.pth')
torch.save(netD.state_dict(), 'discriminator_cifar10.pth')

# Generate a batch of images after training
noise = torch.randn(batch_size, 100, 1, 1, device=device)
fake_images = netG(noise).detach()

# Move the images back to the CPU and convert them to a grid
fake_images = fake_images.cpu()
grid = utils.make_grid(fake_images, nrow=8, padding=2, normalize=True)

# Show the grid of images
imshow(grid)
```

```
Files already downloaded and verified
Images from the CIFAR-10 dataset:
```




```
[0/10][0/79]    Loss_D: 1.4064  Loss_G: 0.6391  D(x): 0.5509    D(G(z)): 0.5531
/ 0.5284
[0/10][50/79]   Loss_D: 0.9898  Loss_G: 1.0562  D(x): 0.6132    D(G(z)): 0.3780
/ 0.3490
[1/10][0/79]    Loss_D: 1.1343  Loss_G: 1.0638  D(x): 0.5308    D(G(z)): 0.3776
/ 0.3506
[1/10][50/79]   Loss_D: 0.9337  Loss_G: 1.1060  D(x): 0.6452    D(G(z)): 0.3757
/ 0.3383
[2/10][0/79]    Loss_D: 1.1186  Loss_G: 1.2260  D(x): 0.5617    D(G(z)): 0.3406
/ 0.3261
[2/10][50/79]   Loss_D: 0.6790  Loss_G: 1.4441  D(x): 0.7279    D(G(z)): 0.2839
/ 0.2751
[3/10][0/79]    Loss_D: 0.9966  Loss_G: 1.4984  D(x): 0.7027    D(G(z)): 0.3593
/ 0.2845
[3/10][50/79]   Loss_D: 0.7161  Loss_G: 1.7135  D(x): 0.7060    D(G(z)): 0.2517
/ 0.2310
[4/10][0/79]    Loss_D: 0.6420  Loss_G: 1.5547  D(x): 0.7334    D(G(z)): 0.2389
/ 0.2665
[4/10][50/79]   Loss_D: 0.4434  Loss_G: 1.9503  D(x): 0.8619    D(G(z)): 0.2469
/ 0.1472
[5/10][0/79]    Loss_D: 0.4773  Loss_G: 1.8853  D(x): 0.7618    D(G(z)): 0.1583
/ 0.1681
[5/10][50/79]   Loss_D: 0.3195  Loss_G: 2.3159  D(x): 0.8604    D(G(z)): 0.1054
/ 0.1394
[6/10][0/79]    Loss_D: 0.8834  Loss_G: 2.3895  D(x): 0.6674    D(G(z)): 0.2107
/ 0.1596
```

```
[6/10][50/79]   Loss_D: 0.1980  Loss_G: 2.5579  D(x): 0.9333    D(G(z)): 0.1192
/ 0.0824
[7/10][0/79]    Loss_D: 0.2425  Loss_G: 2.2819  D(x): 0.9447    D(G(z)): 0.1655
/ 0.1077
[7/10][50/79]   Loss_D: 0.1059  Loss_G: 2.9022  D(x): 0.9627    D(G(z)): 0.0654
/ 0.0563
[8/10][0/79]    Loss_D: 0.1034  Loss_G: 3.2395  D(x): 0.9460    D(G(z)): 0.0461
/ 0.0409
[8/10][50/79]   Loss_D: 0.1157  Loss_G: 3.8263  D(x): 0.9614    D(G(z)): 0.0660
/ 0.1828
[9/10][0/79]    Loss_D: 0.1903  Loss_G: 4.7651  D(x): 0.9646    D(G(z)): 0.1341
/ 0.2695
[9/10][50/79]   Loss_D: 0.0908  Loss_G: 3.6585  D(x): 0.9549    D(G(z)): 0.0432
/ 0.0265
```



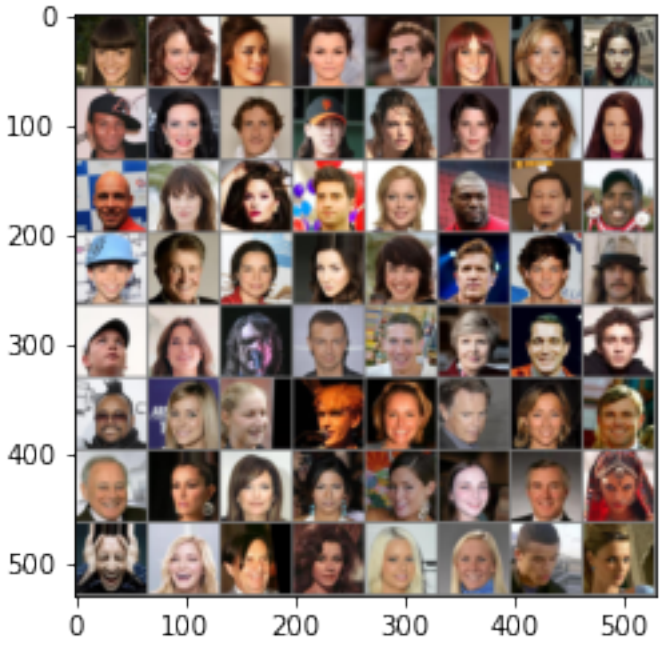
0.4 DCGAN - Celeb-A

[3]:
```python
import torch
import torch.nn as nn
import torch.optim as optim
from torch.utils.data import DataLoader
from torchvision import transforms, datasets, utils
import matplotlib.pyplot as plt
import numpy as np
import os

# Display images
def imshow(img):
    img = img / 2 + 0.5     # unnormalize
    img = img.cpu()  # Move the tensor to CPU
    npimg = img.numpy()  # Convert the tensor to a NumPy array
    plt.imshow(np.transpose(npimg, (1, 2, 0)))
    plt.show()

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

```
        self.main = nn.Sequential(
            nn.ConvTranspose2d(100, 512, 4, 1, 0, bias=False),
            nn.BatchNorm2d(512),
            nn.ReLU(True),
            nn.ConvTranspose2d(512, 256, 4, 2, 1, bias=False),
            nn.BatchNorm2d(256),
            nn.ReLU(True),
            nn.ConvTranspose2d(256, 128, 4, 2, 1, bias=False),
            nn.BatchNorm2d(128),
            nn.ReLU(True),
            nn.ConvTranspose2d(128, 64, 4, 2, 1, bias=False),
            nn.BatchNorm2d(64),
            nn.ReLU(True),
            nn.ConvTranspose2d(64, 3, 4, 2, 1, bias=False),
            nn.Tanh()
        )

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

# Discriminator
class Discriminator(nn.Module):
    def __init__(self):
        super(Discriminator, self).__init__()
        self.main = nn.Sequential(
            nn.Conv2d(3, 64, 4, 2, 1, bias=False),
            nn.LeakyReLU(0.2, inplace=True),
            nn.Conv2d(64, 128, 4, 2, 1, bias=False),
            nn.BatchNorm2d(128),
            nn.LeakyReLU(0.2, inplace=True),
            nn.Conv2d(128, 256, 4, 2, 1, bias=False),
            nn.BatchNorm2d(256),
            nn.LeakyReLU(0.2, inplace=True),
            nn.Conv2d(256, 512, 4, 2, 1, bias=False),
            nn.BatchNorm2d(512),
            nn.LeakyReLU(0.2, inplace=True),
            nn.Conv2d(512, 1, 4, 1, 0, bias=False),
            nn.Sigmoid()
        )

    def forward(self, input):
        return self.main(input).view(-1, 1).squeeze(1)

# Set the path to the extracted Celeb-A dataset folder
celeba_path = './data/celeba/img_align_celeba'

# Define the data transformation
```

```
transform = transforms.Compose([
    transforms.Resize(64),
    transforms.CenterCrop(64),
    transforms.ToTensor(),
    transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5)),
])

# Load the Celeb-A dataset
celeba_dataset = datasets.ImageFolder(os.path.dirname(celeba_path),␣
 ↪transform=transform)
dataloader = torch.utils.data.DataLoader(celeba_dataset, batch_size=64,␣
 ↪shuffle=True, num_workers=2)

# Display a batch of Celeb-A images before training
dataiter = iter(dataloader)
images, _ = dataiter.next()
imshow(utils.make_grid(images))

device = torch.device("cuda" if torch.cuda.is_available() else "cpu")

# Number of epochs
num_epochs = 10

# Initialize BCELoss function
criterion = nn.BCELoss()

# Create batch of latent vectors that we will use to visualize the progression␣
 ↪of the generator
fixed_noise = torch.randn(64, 100, 1, 1, device=device)

# Establish convention for real and fake labels during training
real_label = 1
fake_label = 0

netG = Generator().to(device)
netD = Discriminator().to(device)

# Setup Adam optimizers for both G and D
optimizerD = optim.Adam(netD.parameters(), lr=0.0002, betas=(0.5, 0.999))
optimizerG = optim.Adam(netG.parameters(), lr=0.0002, betas=(0.5, 0.999))

# Training loop
for epoch in range(num_epochs):
    for i, data in enumerate(dataloader, 0):
        # Update D network: maximize log(D(x)) + log(1 - D(G(z)))
        # Train with all-real batch
        netD.zero_grad()
```

```
        real_data = data[0].to(device)
        batch_size = real_data.size(0)
        label = torch.full((batch_size,), real_label, dtype=torch.float,␣
 ↪device=device)
        output = netD(real_data)
        errD_real = criterion(output, label)
        errD_real.backward()
        D_x = output.mean().item()

        # Train with all-fake batch
        noise = torch.randn(batch_size, 100, 1, 1, device=device)
        fake_data = netG(noise)
        label.fill_(fake_label)
        output = netD(fake_data.detach())
        errD_fake = criterion(output, label)
        errD_fake.backward()
        D_G_z1 = output.mean().item()
        errD = errD_real + errD_fake
        optimizerD.step()

        # Update G network: maximize log(D(G(z)))
        netG.zero_grad()
        label.fill_(real_label)
        output = netD(fake_data)
        errG = criterion(output, label)
        errG.backward()
        D_G_z2 = output.mean().item()
        optimizerG.step()

        # Output training stats
        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))

# Generate a batch of images after training
noise = torch.randn(batch_size, 100, 1, 1, device=device)
fake_images = netG(noise).detach()
imshow(utils.make_grid(fake_images))

# Save the trained Generator and Discriminator
torch.save(netG.state_dict(), 'generator_celeba.pth')
torch.save(netD.state_dict(), 'discriminator_celeba.pth')
```




```
[0/10][0/3166]  Loss_D: 1.3840  Loss_G: 3.1961  D(x): 0.5814    D(G(z)): 0.5564
/ 0.0429
[0/10][50/3166] Loss_D: 0.1647  Loss_G: 16.8214 D(x): 0.9138    D(G(z)): 0.0000
/ 0.0000
[0/10][100/3166]        Loss_D: 0.2343  Loss_G: 11.0997 D(x): 0.8464    D(G(z)):
0.0001 / 0.0000
[0/10][150/3166]        Loss_D: 0.8444  Loss_G: 3.3980  D(x): 0.9599    D(G(z)):
0.5100 / 0.0405
[0/10][200/3166]        Loss_D: 1.0781  Loss_G: 4.8283  D(x): 0.8655    D(G(z)):
0.5816 / 0.0106
[0/10][250/3166]        Loss_D: 2.1442  Loss_G: 3.9403  D(x): 0.2192    D(G(z)):
0.0060 / 0.0405
[0/10][300/3166]        Loss_D: 1.2314  Loss_G: 2.5140  D(x): 0.4203    D(G(z)):
0.0181 / 0.0987
[0/10][350/3166]        Loss_D: 0.4344  Loss_G: 2.4359  D(x): 0.7266    D(G(z)):
0.0684 / 0.1113
[0/10][400/3166]        Loss_D: 0.7402  Loss_G: 2.0682  D(x): 0.5886    D(G(z)):
0.1114 / 0.1573
[0/10][450/3166]        Loss_D: 0.3831  Loss_G: 2.8049  D(x): 0.7918    D(G(z)):
0.1189 / 0.0760
[0/10][500/3166]        Loss_D: 0.5713  Loss_G: 3.7558  D(x): 0.8814    D(G(z)):
0.3350 / 0.0313
[0/10][550/3166]        Loss_D: 0.6581  Loss_G: 1.7813  D(x): 0.6634    D(G(z)):
0.1344 / 0.2089
[0/10][600/3166]        Loss_D: 0.5415  Loss_G: 3.8739  D(x): 0.9009    D(G(z)):
0.3244 / 0.0268
```

```
[0/10][650/3166]        Loss_D: 0.3056  Loss_G: 4.4164  D(x): 0.8179    D(G(z)):
0.0523 / 0.0209
[0/10][700/3166]        Loss_D: 0.4124  Loss_G: 4.7968  D(x): 0.9079    D(G(z)):
0.2479 / 0.0114
[0/10][750/3166]        Loss_D: 0.8636  Loss_G: 5.6566  D(x): 0.9138    D(G(z)):
0.4932 / 0.0050
[0/10][800/3166]        Loss_D: 0.8026  Loss_G: 3.5748  D(x): 0.5287    D(G(z)):
0.0217 / 0.0392
[0/10][850/3166]        Loss_D: 0.7472  Loss_G: 5.2760  D(x): 0.8309    D(G(z)):
0.3881 / 0.0062
[0/10][900/3166]        Loss_D: 0.5171  Loss_G: 2.8541  D(x): 0.7843    D(G(z)):
0.1988 / 0.0683
[0/10][950/3166]        Loss_D: 0.5198  Loss_G: 2.6263  D(x): 0.7412    D(G(z)):
0.1683 / 0.0901
[0/10][1000/3166]       Loss_D: 1.0584  Loss_G: 7.9073  D(x): 0.9405    D(G(z)):
0.5671 / 0.0005
[0/10][1050/3166]       Loss_D: 0.7290  Loss_G: 3.7033  D(x): 0.9072    D(G(z)):
0.4266 / 0.0309
[0/10][1100/3166]       Loss_D: 0.4914  Loss_G: 4.1457  D(x): 0.8707    D(G(z)):
0.2686 / 0.0200
[0/10][1150/3166]       Loss_D: 0.9661  Loss_G: 4.3992  D(x): 0.7672    D(G(z)):
0.4329 / 0.0170
[0/10][1200/3166]       Loss_D: 0.7861  Loss_G: 3.0770  D(x): 0.5414    D(G(z)):
0.0301 / 0.0680
[0/10][1250/3166]       Loss_D: 0.8747  Loss_G: 6.0635  D(x): 0.8910    D(G(z)):
0.4615 / 0.0040
[0/10][1300/3166]       Loss_D: 0.3898  Loss_G: 3.4634  D(x): 0.9064    D(G(z)):
0.2274 / 0.0409
[0/10][1350/3166]       Loss_D: 0.5737  Loss_G: 1.8929  D(x): 0.6542    D(G(z)):
0.0830 / 0.1803
[0/10][1400/3166]       Loss_D: 0.4824  Loss_G: 4.7775  D(x): 0.9685    D(G(z)):
0.3266 / 0.0108
[0/10][1450/3166]       Loss_D: 0.9993  Loss_G: 5.5555  D(x): 0.9450    D(G(z)):
0.5520 / 0.0056
[0/10][1500/3166]       Loss_D: 0.9173  Loss_G: 6.1429  D(x): 0.9294    D(G(z)):
0.5229 / 0.0031
[0/10][1550/3166]       Loss_D: 0.3340  Loss_G: 2.8755  D(x): 0.8305    D(G(z)):
0.1231 / 0.0730
[0/10][1600/3166]       Loss_D: 0.3143  Loss_G: 4.5168  D(x): 0.8003    D(G(z)):
0.0561 / 0.0227
[0/10][1650/3166]       Loss_D: 0.2189  Loss_G: 4.3099  D(x): 0.8643    D(G(z)):
0.0534 / 0.0195
[0/10][1700/3166]       Loss_D: 0.5351  Loss_G: 2.5866  D(x): 0.6610    D(G(z)):
0.0490 / 0.0960
[0/10][1750/3166]       Loss_D: 0.8557  Loss_G: 5.7879  D(x): 0.9654    D(G(z)):
0.4970 / 0.0046
[0/10][1800/3166]       Loss_D: 0.7327  Loss_G: 6.7600  D(x): 0.9592    D(G(z)):
0.4599 / 0.0017
```

```
[0/10][1850/3166]       Loss_D: 1.0395  Loss_G: 0.8733  D(x): 0.4166    D(G(z)):
0.0091 / 0.4773
[0/10][1900/3166]       Loss_D: 1.0213  Loss_G: 1.1247  D(x): 0.4561    D(G(z)):
0.0806 / 0.4021
[0/10][1950/3166]       Loss_D: 1.4417  Loss_G: 2.0353  D(x): 0.3283    D(G(z)):
0.0114 / 0.1830
[0/10][2000/3166]       Loss_D: 0.4432  Loss_G: 4.6633  D(x): 0.9066    D(G(z)):
0.2491 / 0.0126
[0/10][2050/3166]       Loss_D: 1.2692  Loss_G: 1.4600  D(x): 0.3548    D(G(z)):
0.0110 / 0.2721
[0/10][2100/3166]       Loss_D: 0.4717  Loss_G: 5.3558  D(x): 0.9707    D(G(z)):
0.3232 / 0.0080
[0/10][2150/3166]       Loss_D: 0.3976  Loss_G: 3.3973  D(x): 0.8371    D(G(z)):
0.1763 / 0.0419
[0/10][2200/3166]       Loss_D: 0.8336  Loss_G: 6.7513  D(x): 0.9665    D(G(z)):
0.5067 / 0.0020
[0/10][2250/3166]       Loss_D: 0.3953  Loss_G: 3.8106  D(x): 0.8841    D(G(z)):
0.2122 / 0.0291
[0/10][2300/3166]       Loss_D: 0.5574  Loss_G: 5.3741  D(x): 0.8746    D(G(z)):
0.3103 / 0.0061
[0/10][2350/3166]       Loss_D: 0.3775  Loss_G: 4.3556  D(x): 0.9025    D(G(z)):
0.2195 / 0.0177
[0/10][2400/3166]       Loss_D: 0.4462  Loss_G: 4.5833  D(x): 0.9182    D(G(z)):
0.2826 / 0.0127
[0/10][2450/3166]       Loss_D: 0.3281  Loss_G: 4.6813  D(x): 0.9152    D(G(z)):
0.1948 / 0.0125
[0/10][2500/3166]       Loss_D: 0.4008  Loss_G: 2.9326  D(x): 0.7361    D(G(z)):
0.0533 / 0.0752
[0/10][2550/3166]       Loss_D: 0.6978  Loss_G: 7.1003  D(x): 0.9680    D(G(z)):
0.4386 / 0.0013
[0/10][2600/3166]       Loss_D: 0.3369  Loss_G: 4.6393  D(x): 0.9558    D(G(z)):
0.2249 / 0.0158
[0/10][2650/3166]       Loss_D: 0.4198  Loss_G: 2.9550  D(x): 0.8062    D(G(z)):
0.1636 / 0.0713
[0/10][2700/3166]       Loss_D: 0.4083  Loss_G: 2.6705  D(x): 0.8306    D(G(z)):
0.1707 / 0.0866
[0/10][2750/3166]       Loss_D: 0.1688  Loss_G: 3.1551  D(x): 0.9408    D(G(z)):
0.0959 / 0.0587
[0/10][2800/3166]       Loss_D: 0.5198  Loss_G: 3.1151  D(x): 0.8203    D(G(z)):
0.2356 / 0.0541
[0/10][2850/3166]       Loss_D: 0.9453  Loss_G: 5.8782  D(x): 0.9408    D(G(z)):
0.5369 / 0.0040
[0/10][2900/3166]       Loss_D: 0.3144  Loss_G: 3.8440  D(x): 0.8960    D(G(z)):
0.1683 / 0.0281
[0/10][2950/3166]       Loss_D: 0.7963  Loss_G: 3.3191  D(x): 0.5234    D(G(z)):
0.0033 / 0.0577
[0/10][3000/3166]       Loss_D: 0.3756  Loss_G: 4.1365  D(x): 0.8804    D(G(z)):
0.1999 / 0.0221
```

```
[0/10][3050/3166]       Loss_D: 0.5044  Loss_G: 1.4161  D(x): 0.7446    D(G(z)):
0.1513 / 0.2712
[0/10][3100/3166]       Loss_D: 0.4106  Loss_G: 3.7301  D(x): 0.8625    D(G(z)):
0.2015 / 0.0331
[0/10][3150/3166]       Loss_D: 0.4170  Loss_G: 3.5692  D(x): 0.8530    D(G(z)):
0.1953 / 0.0390
[1/10][0/3166]  Loss_D: 0.2809  Loss_G: 4.1908  D(x): 0.9299    D(G(z)): 0.1717
/ 0.0217
[1/10][50/3166] Loss_D: 1.1832  Loss_G: 8.5540  D(x): 0.9776    D(G(z)): 0.6353
/ 0.0003
[1/10][100/3166]        Loss_D: 0.3299  Loss_G: 3.6557  D(x): 0.8681    D(G(z)):
0.1530 / 0.0356
[1/10][150/3166]        Loss_D: 1.1106  Loss_G: 7.1840  D(x): 0.9777    D(G(z)):
0.6136 / 0.0012
[1/10][200/3166]        Loss_D: 0.8758  Loss_G: 4.9791  D(x): 0.9282    D(G(z)):
0.5005 / 0.0124
[1/10][250/3166]        Loss_D: 0.2984  Loss_G: 3.5498  D(x): 0.9391    D(G(z)):
0.1928 / 0.0382
[1/10][300/3166]        Loss_D: 2.0110  Loss_G: 1.0709  D(x): 0.1896    D(G(z)):
0.0010 / 0.4147
[1/10][350/3166]        Loss_D: 0.3966  Loss_G: 4.0739  D(x): 0.9238    D(G(z)):
0.2371 / 0.0245
[1/10][400/3166]        Loss_D: 0.4963  Loss_G: 3.3017  D(x): 0.7819    D(G(z)):
0.1779 / 0.0470
[1/10][450/3166]        Loss_D: 0.1536  Loss_G: 4.0128  D(x): 0.9255    D(G(z)):
0.0636 / 0.0255
[1/10][500/3166]        Loss_D: 0.4616  Loss_G: 2.1609  D(x): 0.7369    D(G(z)):
0.0830 / 0.1429
[1/10][550/3166]        Loss_D: 1.0333  Loss_G: 6.5139  D(x): 0.9690    D(G(z)):
0.5768 / 0.0027
[1/10][600/3166]        Loss_D: 0.2105  Loss_G: 2.4316  D(x): 0.8939    D(G(z)):
0.0834 / 0.1131
[1/10][650/3166]        Loss_D: 0.3661  Loss_G: 1.9005  D(x): 0.7606    D(G(z)):
0.0482 / 0.1838
[1/10][700/3166]        Loss_D: 0.3565  Loss_G: 2.8610  D(x): 0.9043    D(G(z)):
0.2040 / 0.0768
[1/10][750/3166]        Loss_D: 0.2826  Loss_G: 3.3919  D(x): 0.9035    D(G(z)):
0.1476 / 0.0417
[1/10][800/3166]        Loss_D: 0.2534  Loss_G: 4.1671  D(x): 0.9281    D(G(z)):
0.1428 / 0.0207
[1/10][850/3166]        Loss_D: 0.3027  Loss_G: 3.8553  D(x): 0.9397    D(G(z)):
0.1954 / 0.0291
[1/10][900/3166]        Loss_D: 0.3017  Loss_G: 3.8346  D(x): 0.9019    D(G(z)):
0.1621 / 0.0305
[1/10][950/3166]        Loss_D: 0.2496  Loss_G: 3.1908  D(x): 0.8846    D(G(z)):
0.1052 / 0.0514
[1/10][1000/3166]       Loss_D: 0.3668  Loss_G: 3.5830  D(x): 0.7392    D(G(z)):
0.0212 / 0.0447
```

```
[1/10][1050/3166]       Loss_D: 0.3680  Loss_G: 4.0481  D(x): 0.9460    D(G(z)):
0.2501 / 0.0229
[1/10][1100/3166]       Loss_D: 0.2904  Loss_G: 2.9276  D(x): 0.8399    D(G(z)):
0.0940 / 0.0706
[1/10][1150/3166]       Loss_D: 0.8883  Loss_G: 1.2934  D(x): 0.5550    D(G(z)):
0.1181 / 0.3532
[1/10][1200/3166]       Loss_D: 0.2862  Loss_G: 3.1509  D(x): 0.8368    D(G(z)):
0.0885 / 0.0610
[1/10][1250/3166]       Loss_D: 0.5832  Loss_G: 4.4908  D(x): 0.6074    D(G(z)):
0.0058 / 0.0246
[1/10][1300/3166]       Loss_D: 0.3866  Loss_G: 2.5761  D(x): 0.7393    D(G(z)):
0.0410 / 0.1009
[1/10][1350/3166]       Loss_D: 0.7050  Loss_G: 3.1283  D(x): 0.5729    D(G(z)):
0.0060 / 0.0692
[1/10][1400/3166]       Loss_D: 0.4732  Loss_G: 1.1748  D(x): 0.6929    D(G(z)):
0.0375 / 0.3629
[1/10][1450/3166]       Loss_D: 0.5373  Loss_G: 2.2592  D(x): 0.6425    D(G(z)):
0.0140 / 0.1477
[1/10][1500/3166]       Loss_D: 0.2869  Loss_G: 3.3750  D(x): 0.7918    D(G(z)):
0.0228 / 0.0475
[1/10][1550/3166]       Loss_D: 0.2456  Loss_G: 2.7269  D(x): 0.8457    D(G(z)):
0.0586 / 0.0917
[1/10][1600/3166]       Loss_D: 0.5784  Loss_G: 5.9123  D(x): 0.9109    D(G(z)):
0.3449 / 0.0044
[1/10][1650/3166]       Loss_D: 0.1036  Loss_G: 4.6830  D(x): 0.9657    D(G(z)):
0.0633 / 0.0145
[1/10][1700/3166]       Loss_D: 0.1809  Loss_G: 4.0416  D(x): 0.9100    D(G(z)):
0.0756 / 0.0266
[1/10][1750/3166]       Loss_D: 0.7689  Loss_G: 3.1260  D(x): 0.5330    D(G(z)):
0.0081 / 0.0704
[1/10][1800/3166]       Loss_D: 0.2239  Loss_G: 4.5559  D(x): 0.9488    D(G(z)):
0.1488 / 0.0137
[1/10][1850/3166]       Loss_D: 0.1957  Loss_G: 4.2919  D(x): 0.9639    D(G(z)):
0.1365 / 0.0190
[1/10][1900/3166]       Loss_D: 1.2417  Loss_G: 1.1556  D(x): 0.4031    D(G(z)):
0.0481 / 0.3889
[1/10][1950/3166]       Loss_D: 0.2127  Loss_G: 3.1391  D(x): 0.8546    D(G(z)):
0.0444 / 0.0600
[1/10][2000/3166]       Loss_D: 0.2386  Loss_G: 4.6090  D(x): 0.9477    D(G(z)):
0.1554 / 0.0142
[1/10][2050/3166]       Loss_D: 0.5651  Loss_G: 1.9125  D(x): 0.6619    D(G(z)):
0.0431 / 0.2158
[1/10][2100/3166]       Loss_D: 1.0816  Loss_G: 9.1986  D(x): 0.9822    D(G(z)):
0.5774 / 0.0002
[1/10][2150/3166]       Loss_D: 0.1558  Loss_G: 3.2400  D(x): 0.8806    D(G(z)):
0.0196 / 0.0630
[1/10][2200/3166]       Loss_D: 0.5165  Loss_G: 2.8293  D(x): 0.6792    D(G(z)):
0.0429 / 0.1075
```

```
[1/10][2250/3166]       Loss_D: 0.2103  Loss_G: 2.8844  D(x): 0.8755    D(G(z)):
0.0619 / 0.0776
[1/10][2300/3166]       Loss_D: 0.3781  Loss_G: 4.2161  D(x): 0.8632    D(G(z)):
0.1730 / 0.0212
[1/10][2350/3166]       Loss_D: 0.2032  Loss_G: 3.2461  D(x): 0.8893    D(G(z)):
0.0627 / 0.0546
[1/10][2400/3166]       Loss_D: 0.2593  Loss_G: 3.7845  D(x): 0.8504    D(G(z)):
0.0771 / 0.0331
[1/10][2450/3166]       Loss_D: 0.7991  Loss_G: 6.3027  D(x): 0.9624    D(G(z)):
0.4530 / 0.0032
[1/10][2500/3166]       Loss_D: 0.3125  Loss_G: 3.2360  D(x): 0.8978    D(G(z)):
0.1585 / 0.0522
[1/10][2550/3166]       Loss_D: 0.2556  Loss_G: 3.9172  D(x): 0.9142    D(G(z)):
0.1317 / 0.0298
[1/10][2600/3166]       Loss_D: 0.2399  Loss_G: 4.4129  D(x): 0.9641    D(G(z)):
0.1668 / 0.0181
[1/10][2650/3166]       Loss_D: 1.0113  Loss_G: 2.8599  D(x): 0.4723    D(G(z)):
0.0051 / 0.1103
[1/10][2700/3166]       Loss_D: 0.2220  Loss_G: 3.9839  D(x): 0.9056    D(G(z)):
0.0991 / 0.0259
[1/10][2750/3166]       Loss_D: 0.3106  Loss_G: 4.1940  D(x): 0.7776    D(G(z)):
0.0219 / 0.0281
[1/10][2800/3166]       Loss_D: 0.3497  Loss_G: 5.5455  D(x): 0.9864    D(G(z)):
0.2506 / 0.0056
[1/10][2850/3166]       Loss_D: 6.2065  Loss_G: 0.1884  D(x): 0.0074    D(G(z)):
0.0002 / 0.8540
[1/10][2900/3166]       Loss_D: 0.2650  Loss_G: 3.5871  D(x): 0.8625    D(G(z)):
0.0960 / 0.0355
[1/10][2950/3166]       Loss_D: 0.1789  Loss_G: 3.4897  D(x): 0.8916    D(G(z)):
0.0519 / 0.0483
[1/10][3000/3166]       Loss_D: 0.3477  Loss_G: 2.8336  D(x): 0.8184    D(G(z)):
0.1155 / 0.0882
[1/10][3050/3166]       Loss_D: 0.2971  Loss_G: 5.7076  D(x): 0.9682    D(G(z)):
0.2136 / 0.0046
[1/10][3100/3166]       Loss_D: 0.1798  Loss_G: 4.3567  D(x): 0.9257    D(G(z)):
0.0868 / 0.0193
[1/10][3150/3166]       Loss_D: 1.0908  Loss_G: 2.7461  D(x): 0.4389    D(G(z)):
0.0115 / 0.1305
[2/10][0/3166]  Loss_D: 0.2730  Loss_G: 2.7644  D(x): 0.8267    D(G(z)): 0.0586
/ 0.1114
[2/10][50/3166] Loss_D: 0.2577  Loss_G: 4.5858  D(x): 0.9397    D(G(z)): 0.1606
/ 0.0143
[2/10][100/3166]        Loss_D: 1.2366  Loss_G: 8.6955  D(x): 0.9943    D(G(z)):
0.6079 / 0.0003
[2/10][150/3166]        Loss_D: 0.2987  Loss_G: 2.1276  D(x): 0.8207    D(G(z)):
0.0749 / 0.1507
[2/10][200/3166]        Loss_D: 0.0966  Loss_G: 4.5805  D(x): 0.9684    D(G(z)):
0.0595 / 0.0169
```

```
[2/10][250/3166]        Loss_D: 0.1971  Loss_G: 4.1933  D(x): 0.9269    D(G(z)):
0.1000 / 0.0237
[2/10][300/3166]        Loss_D: 0.4961  Loss_G: 5.5479  D(x): 0.9483    D(G(z)):
0.3098 / 0.0054
[2/10][350/3166]        Loss_D: 0.2378  Loss_G: 3.8739  D(x): 0.9413    D(G(z)):
0.1452 / 0.0293
[2/10][400/3166]        Loss_D: 0.1406  Loss_G: 3.7524  D(x): 0.9753    D(G(z)):
0.1036 / 0.0336
[2/10][450/3166]        Loss_D: 1.0173  Loss_G: 1.4191  D(x): 0.4633    D(G(z)):
0.0300 / 0.3171
[2/10][500/3166]        Loss_D: 0.4339  Loss_G: 1.9865  D(x): 0.7126    D(G(z)):
0.0241 / 0.2135
[2/10][550/3166]        Loss_D: 0.1968  Loss_G: 3.7856  D(x): 0.9429    D(G(z)):
0.1202 / 0.0303
[2/10][600/3166]        Loss_D: 0.7007  Loss_G: 3.2930  D(x): 0.5796    D(G(z)):
0.0200 / 0.0867
[2/10][650/3166]        Loss_D: 0.5970  Loss_G: 4.3374  D(x): 0.9888    D(G(z)):
0.3664 / 0.0268
[2/10][700/3166]        Loss_D: 0.2183  Loss_G: 3.8060  D(x): 0.9565    D(G(z)):
0.1513 / 0.0298
[2/10][750/3166]        Loss_D: 0.1337  Loss_G: 3.8198  D(x): 0.9428    D(G(z)):
0.0650 / 0.0295
[2/10][800/3166]        Loss_D: 0.4416  Loss_G: 4.9928  D(x): 0.6914    D(G(z)):
0.0045 / 0.0141
[2/10][850/3166]        Loss_D: 0.5860  Loss_G: 7.1301  D(x): 0.9784    D(G(z)):
0.3781 / 0.0015
[2/10][900/3166]        Loss_D: 0.2595  Loss_G: 2.7925  D(x): 0.8378    D(G(z)):
0.0512 / 0.0853
[2/10][950/3166]        Loss_D: 0.1532  Loss_G: 3.8898  D(x): 0.9122    D(G(z)):
0.0486 / 0.0318
[2/10][1000/3166]       Loss_D: 0.1848  Loss_G: 3.9922  D(x): 0.8678    D(G(z)):
0.0288 / 0.0314
[2/10][1050/3166]       Loss_D: 0.3597  Loss_G: 2.8644  D(x): 0.7758    D(G(z)):
0.0522 / 0.0780
[2/10][1100/3166]       Loss_D: 0.2169  Loss_G: 2.6972  D(x): 0.8465    D(G(z)):
0.0381 / 0.0908
[2/10][1150/3166]       Loss_D: 0.1607  Loss_G: 3.6943  D(x): 0.9450    D(G(z)):
0.0932 / 0.0316
[2/10][1200/3166]       Loss_D: 0.4019  Loss_G: 4.6050  D(x): 0.9599    D(G(z)):
0.2647 / 0.0135
[2/10][1250/3166]       Loss_D: 0.3196  Loss_G: 2.4814  D(x): 0.7811    D(G(z)):
0.0325 / 0.1339
[2/10][1300/3166]       Loss_D: 0.4156  Loss_G: 2.2160  D(x): 0.7244    D(G(z)):
0.0345 / 0.1914
[2/10][1350/3166]       Loss_D: 0.5132  Loss_G: 1.8442  D(x): 0.6601    D(G(z)):
0.0226 / 0.2383
[2/10][1400/3166]       Loss_D: 0.2207  Loss_G: 3.2156  D(x): 0.9448    D(G(z)):
0.1361 / 0.0612
```

```
[2/10][1450/3166]       Loss_D: 0.1412  Loss_G: 4.1286  D(x): 0.9411    D(G(z)):
0.0721 / 0.0257
[2/10][1500/3166]       Loss_D: 0.1575  Loss_G: 2.7864  D(x): 0.8859    D(G(z)):
0.0230 / 0.0878
[2/10][1550/3166]       Loss_D: 0.3196  Loss_G: 4.6444  D(x): 0.7733    D(G(z)):
0.0080 / 0.0179
[2/10][1600/3166]       Loss_D: 0.4541  Loss_G: 3.1546  D(x): 0.7138    D(G(z)):
0.0106 / 0.0657
[2/10][1650/3166]       Loss_D: 0.6922  Loss_G: 8.3977  D(x): 0.9961    D(G(z)):
0.4344 / 0.0003
[2/10][1700/3166]       Loss_D: 0.2746  Loss_G: 3.4161  D(x): 0.8819    D(G(z)):
0.1171 / 0.0457
[2/10][1750/3166]       Loss_D: 0.3452  Loss_G: 3.9243  D(x): 0.7796    D(G(z)):
0.0262 / 0.0411
[2/10][1800/3166]       Loss_D: 0.1673  Loss_G: 3.0245  D(x): 0.8981    D(G(z)):
0.0490 / 0.0803
[2/10][1850/3166]       Loss_D: 0.7712  Loss_G: 1.1186  D(x): 0.5194    D(G(z)):
0.0074 / 0.4326
[2/10][1900/3166]       Loss_D: 0.4426  Loss_G: 2.8235  D(x): 0.7152    D(G(z)):
0.0280 / 0.1091
[2/10][1950/3166]       Loss_D: 0.2490  Loss_G: 3.6448  D(x): 0.8623    D(G(z)):
0.0731 / 0.0427
[2/10][2000/3166]       Loss_D: 0.3103  Loss_G: 4.8157  D(x): 0.9726    D(G(z)):
0.2128 / 0.0123
[2/10][2050/3166]       Loss_D: 0.1296  Loss_G: 4.0421  D(x): 0.9648    D(G(z)):
0.0771 / 0.0305
[2/10][2100/3166]       Loss_D: 0.2067  Loss_G: 4.3545  D(x): 0.9718    D(G(z)):
0.1433 / 0.0223
[2/10][2150/3166]       Loss_D: 0.2048  Loss_G: 2.8736  D(x): 0.9066    D(G(z)):
0.0864 / 0.0997
[2/10][2200/3166]       Loss_D: 0.6089  Loss_G: 6.6437  D(x): 0.9795    D(G(z)):
0.3797 / 0.0021
[2/10][2250/3166]       Loss_D: 0.2206  Loss_G: 4.2816  D(x): 0.9819    D(G(z)):
0.1648 / 0.0209
[2/10][2300/3166]       Loss_D: 0.2259  Loss_G: 3.3566  D(x): 0.8570    D(G(z)):
0.0410 / 0.0567
[2/10][2350/3166]       Loss_D: 0.3186  Loss_G: 5.0782  D(x): 0.9844    D(G(z)):
0.2214 / 0.0082
[2/10][2400/3166]       Loss_D: 0.1908  Loss_G: 3.8591  D(x): 0.8955    D(G(z)):
0.0652 / 0.0368
[2/10][2450/3166]       Loss_D: 0.4880  Loss_G: 3.1975  D(x): 0.9548    D(G(z)):
0.3065 / 0.0587
[2/10][2500/3166]       Loss_D: 0.4543  Loss_G: 3.3061  D(x): 0.6832    D(G(z)):
0.0172 / 0.0701
[2/10][2550/3166]       Loss_D: 0.4094  Loss_G: 3.4412  D(x): 0.7665    D(G(z)):
0.0778 / 0.0500
[2/10][2600/3166]       Loss_D: 0.1235  Loss_G: 4.4730  D(x): 0.9842    D(G(z)):
0.0966 / 0.0159
```

```
[2/10][2650/3166]       Loss_D: 0.7778  Loss_G: 3.3146  D(x): 0.7683    D(G(z)):
0.2857 / 0.0654
[2/10][2700/3166]       Loss_D: 0.2592  Loss_G: 3.3174  D(x): 0.8722    D(G(z)):
0.0947 / 0.0533
[2/10][2750/3166]       Loss_D: 0.3090  Loss_G: 1.9648  D(x): 0.7855    D(G(z)):
0.0335 / 0.1853
[2/10][2800/3166]       Loss_D: 0.1249  Loss_G: 3.9631  D(x): 0.9344    D(G(z)):
0.0515 / 0.0287
[2/10][2850/3166]       Loss_D: 1.9819  Loss_G: 2.3070  D(x): 0.2670    D(G(z)):
0.0034 / 0.2209
[2/10][2900/3166]       Loss_D: 0.2906  Loss_G: 3.2339  D(x): 0.7865    D(G(z)):
0.0189 / 0.0675
[2/10][2950/3166]       Loss_D: 0.2277  Loss_G: 3.8578  D(x): 0.9609    D(G(z)):
0.1566 / 0.0298
[2/10][3000/3166]       Loss_D: 0.1426  Loss_G: 3.3801  D(x): 0.9161    D(G(z)):
0.0477 / 0.0463
[2/10][3050/3166]       Loss_D: 0.1648  Loss_G: 4.4141  D(x): 0.9411    D(G(z)):
0.0886 / 0.0209
[2/10][3100/3166]       Loss_D: 0.3210  Loss_G: 5.6268  D(x): 0.9850    D(G(z)):
0.2282 / 0.0057
[2/10][3150/3166]       Loss_D: 0.3196  Loss_G: 2.8380  D(x): 0.7555    D(G(z)):
0.0080 / 0.0912
[3/10][0/3166]  Loss_D: 0.2336  Loss_G: 4.9007  D(x): 0.9748    D(G(z)): 0.1738
/ 0.0099
[3/10][50/3166] Loss_D: 1.0090  Loss_G: 3.5906  D(x): 0.7540    D(G(z)): 0.3989
/ 0.0535
[3/10][100/3166]        Loss_D: 0.2882  Loss_G: 5.3558  D(x): 0.9519    D(G(z)):
0.1966 / 0.0068
[3/10][150/3166]        Loss_D: 0.2322  Loss_G: 5.6864  D(x): 0.9687    D(G(z)):
0.1568 / 0.0052
[3/10][200/3166]        Loss_D: 0.2112  Loss_G: 5.5088  D(x): 0.9471    D(G(z)):
0.1276 / 0.0064
[3/10][250/3166]        Loss_D: 0.4592  Loss_G: 3.8878  D(x): 0.8556    D(G(z)):
0.1909 / 0.0356
[3/10][300/3166]        Loss_D: 0.4451  Loss_G: 1.7811  D(x): 0.7137    D(G(z)):
0.0418 / 0.2265
[3/10][350/3166]        Loss_D: 0.1827  Loss_G: 3.5136  D(x): 0.9611    D(G(z)):
0.1219 / 0.0440
[3/10][400/3166]        Loss_D: 0.2568  Loss_G: 4.6833  D(x): 0.9568    D(G(z)):
0.1492 / 0.0136
[3/10][450/3166]        Loss_D: 0.1303  Loss_G: 4.9319  D(x): 0.9794    D(G(z)):
0.0980 / 0.0100
[3/10][500/3166]        Loss_D: 0.8107  Loss_G: 6.3270  D(x): 0.9321    D(G(z)):
0.4181 / 0.0043
[3/10][550/3166]        Loss_D: 0.9537  Loss_G: 9.4864  D(x): 0.9987    D(G(z)):
0.5001 / 0.0002
[3/10][600/3166]        Loss_D: 0.2143  Loss_G: 5.0569  D(x): 0.9341    D(G(z)):
0.1235 / 0.0089
```

```
[3/10][650/3166]        Loss_D: 0.2081  Loss_G: 3.0726  D(x): 0.8815    D(G(z)):
0.0666 / 0.0655
[3/10][700/3166]        Loss_D: 1.5428  Loss_G: 8.1632  D(x): 0.9713    D(G(z)):
0.6804 / 0.0008
[3/10][750/3166]        Loss_D: 1.3291  Loss_G: 9.0299  D(x): 0.9964    D(G(z)):
0.6470 / 0.0002
[3/10][800/3166]        Loss_D: 0.2721  Loss_G: 4.7157  D(x): 0.9882    D(G(z)):
0.1939 / 0.0121
[3/10][850/3166]        Loss_D: 0.2788  Loss_G: 2.8701  D(x): 0.8558    D(G(z)):
0.0894 / 0.0852
[3/10][900/3166]        Loss_D: 0.1312  Loss_G: 3.1600  D(x): 0.9209    D(G(z)):
0.0396 / 0.0717
[3/10][950/3166]        Loss_D: 0.4863  Loss_G: 3.0296  D(x): 0.8342    D(G(z)):
0.2057 / 0.0885
[3/10][1000/3166]       Loss_D: 0.1960  Loss_G: 5.2495  D(x): 0.8514    D(G(z)):
0.0104 / 0.0111
[3/10][1050/3166]       Loss_D: 0.3444  Loss_G: 3.3872  D(x): 0.7574    D(G(z)):
0.0257 / 0.0733
[3/10][1100/3166]       Loss_D: 1.7542  Loss_G: 7.5556  D(x): 0.9856    D(G(z)):
0.6954 / 0.0015
[3/10][1150/3166]       Loss_D: 0.4520  Loss_G: 6.5896  D(x): 0.9859    D(G(z)):
0.3120 / 0.0021
[3/10][1200/3166]       Loss_D: 0.2969  Loss_G: 5.2409  D(x): 0.9835    D(G(z)):
0.2155 / 0.0079
[3/10][1250/3166]       Loss_D: 0.5633  Loss_G: 4.8784  D(x): 0.9637    D(G(z)):
0.3470 / 0.0121
[3/10][1300/3166]       Loss_D: 0.2176  Loss_G: 4.2932  D(x): 0.9549    D(G(z)):
0.1370 / 0.0217
[3/10][1350/3166]       Loss_D: 0.1929  Loss_G: 5.2484  D(x): 0.9784    D(G(z)):
0.1491 / 0.0071
[3/10][1400/3166]       Loss_D: 0.3686  Loss_G: 2.6347  D(x): 0.7465    D(G(z)):
0.0111 / 0.1186
[3/10][1450/3166]       Loss_D: 0.3789  Loss_G: 3.0780  D(x): 0.8104    D(G(z)):
0.1150 / 0.0701
[3/10][1500/3166]       Loss_D: 2.1080  Loss_G: 1.1745  D(x): 0.2326    D(G(z)):
0.0057 / 0.4607
[3/10][1550/3166]       Loss_D: 0.2049  Loss_G: 3.0908  D(x): 0.8644    D(G(z)):
0.0344 / 0.0741
[3/10][1600/3166]       Loss_D: 2.8067  Loss_G: 10.9756 D(x): 0.9994    D(G(z)):
0.8897 / 0.0001
[3/10][1650/3166]       Loss_D: 0.3235  Loss_G: 3.0366  D(x): 0.7946    D(G(z)):
0.0492 / 0.0880
[3/10][1700/3166]       Loss_D: 0.1209  Loss_G: 4.4046  D(x): 0.9386    D(G(z)):
0.0486 / 0.0190
[3/10][1750/3166]       Loss_D: 1.1252  Loss_G: 4.7618  D(x): 0.4500    D(G(z)):
0.0011 / 0.0280
[3/10][1800/3166]       Loss_D: 0.1319  Loss_G: 3.9336  D(x): 0.9828    D(G(z)):
0.0951 / 0.0327
```

```
[3/10][1850/3166]       Loss_D: 0.2763  Loss_G: 2.9965  D(x): 0.8770    D(G(z)):
0.1063 / 0.0709
[3/10][1900/3166]       Loss_D: 0.1772  Loss_G: 3.6222  D(x): 0.8703    D(G(z)):
0.0216 / 0.0426
[3/10][1950/3166]       Loss_D: 0.0764  Loss_G: 4.1422  D(x): 0.9728    D(G(z)):
0.0457 / 0.0228
[3/10][2000/3166]       Loss_D: 0.2196  Loss_G: 2.6252  D(x): 0.8777    D(G(z)):
0.0689 / 0.1164
[3/10][2050/3166]       Loss_D: 0.1876  Loss_G: 3.5807  D(x): 0.8817    D(G(z)):
0.0400 / 0.0408
[3/10][2100/3166]       Loss_D: 0.1393  Loss_G: 3.0916  D(x): 0.9077    D(G(z)):
0.0342 / 0.0683
[3/10][2150/3166]       Loss_D: 0.6497  Loss_G: 3.9016  D(x): 0.8848    D(G(z)):
0.3430 / 0.0305
[3/10][2200/3166]       Loss_D: 0.2707  Loss_G: 3.3456  D(x): 0.8602    D(G(z)):
0.0890 / 0.0540
[3/10][2250/3166]       Loss_D: 0.2839  Loss_G: 4.9590  D(x): 0.9543    D(G(z)):
0.1795 / 0.0138
[3/10][2300/3166]       Loss_D: 0.1560  Loss_G: 4.0654  D(x): 0.9250    D(G(z)):
0.0644 / 0.0296
[3/10][2350/3166]       Loss_D: 0.2132  Loss_G: 3.8555  D(x): 0.9815    D(G(z)):
0.1571 / 0.0288
[3/10][2400/3166]       Loss_D: 0.1448  Loss_G: 3.7526  D(x): 0.9445    D(G(z)):
0.0762 / 0.0327
[3/10][2450/3166]       Loss_D: 0.9761  Loss_G: 1.7150  D(x): 0.4825    D(G(z)):
0.0044 / 0.3020
[3/10][2500/3166]       Loss_D: 0.5981  Loss_G: 7.2703  D(x): 0.9833    D(G(z)):
0.3892 / 0.0012
[3/10][2550/3166]       Loss_D: 0.6735  Loss_G: 5.6720  D(x): 0.8806    D(G(z)):
0.3156 / 0.0061
[3/10][2600/3166]       Loss_D: 0.1006  Loss_G: 4.5128  D(x): 0.9616    D(G(z)):
0.0543 / 0.0177
[3/10][2650/3166]       Loss_D: 0.1121  Loss_G: 4.2802  D(x): 0.9776    D(G(z)):
0.0787 / 0.0199
[3/10][2700/3166]       Loss_D: 0.1448  Loss_G: 4.7159  D(x): 0.9488    D(G(z)):
0.0821 / 0.0136
[3/10][2750/3166]       Loss_D: 0.1970  Loss_G: 3.3787  D(x): 0.9269    D(G(z)):
0.0982 / 0.0551
[3/10][2800/3166]       Loss_D: 0.1772  Loss_G: 5.1968  D(x): 0.8593    D(G(z)):
0.0119 / 0.0116
[3/10][2850/3166]       Loss_D: 0.2069  Loss_G: 3.7812  D(x): 0.8824    D(G(z)):
0.0422 / 0.0382
[3/10][2900/3166]       Loss_D: 0.1549  Loss_G: 3.5803  D(x): 0.9201    D(G(z)):
0.0537 / 0.0483
[3/10][2950/3166]       Loss_D: 0.2089  Loss_G: 3.1871  D(x): 0.8743    D(G(z)):
0.0495 / 0.0695
[3/10][3000/3166]       Loss_D: 0.1333  Loss_G: 6.4304  D(x): 0.9068    D(G(z)):
0.0273 / 0.0040
```

```
[3/10][3050/3166]       Loss_D: 0.2149  Loss_G: 3.2877  D(x): 0.8489    D(G(z)):
0.0353 / 0.0546
[3/10][3100/3166]       Loss_D: 0.3317  Loss_G: 6.2781  D(x): 0.9948    D(G(z)):
0.2413 / 0.0025
[3/10][3150/3166]       Loss_D: 0.2581  Loss_G: 3.9598  D(x): 0.9444    D(G(z)):
0.1554 / 0.0298
[4/10][0/3166]  Loss_D: 0.2189  Loss_G: 4.5155  D(x): 0.9508    D(G(z)): 0.1277
/ 0.0208
[4/10][50/3166] Loss_D: 0.1082  Loss_G: 3.4802  D(x): 0.9532    D(G(z)): 0.0536
/ 0.0447
[4/10][100/3166]        Loss_D: 0.1130  Loss_G: 4.7106  D(x): 0.9809    D(G(z)):
0.0837 / 0.0133
[4/10][150/3166]        Loss_D: 0.1903  Loss_G: 3.1225  D(x): 0.9320    D(G(z)):
0.0988 / 0.0764
[4/10][200/3166]        Loss_D: 0.2100  Loss_G: 5.1357  D(x): 0.9212    D(G(z)):
0.0940 / 0.0107
[4/10][250/3166]        Loss_D: 0.1706  Loss_G: 4.4546  D(x): 0.9234    D(G(z)):
0.0683 / 0.0259
[4/10][300/3166]        Loss_D: 0.3315  Loss_G: 1.8590  D(x): 0.8018    D(G(z)):
0.0656 / 0.2176
[4/10][350/3166]        Loss_D: 0.1979  Loss_G: 3.6915  D(x): 0.9318    D(G(z)):
0.1067 / 0.0353
[4/10][400/3166]        Loss_D: 0.1692  Loss_G: 4.2103  D(x): 0.9228    D(G(z)):
0.0730 / 0.0296
[4/10][450/3166]        Loss_D: 0.0782  Loss_G: 3.7248  D(x): 0.9841    D(G(z)):
0.0566 / 0.0394
[4/10][500/3166]        Loss_D: 0.1110  Loss_G: 3.7020  D(x): 0.9363    D(G(z)):
0.0403 / 0.0391
[4/10][550/3166]        Loss_D: 0.0743  Loss_G: 4.5605  D(x): 0.9812    D(G(z)):
0.0513 / 0.0164
[4/10][600/3166]        Loss_D: 0.4282  Loss_G: 9.3291  D(x): 0.9942    D(G(z)):
0.3057 / 0.0001
[4/10][650/3166]        Loss_D: 0.2406  Loss_G: 5.2941  D(x): 0.9480    D(G(z)):
0.1507 / 0.0085
[4/10][700/3166]        Loss_D: 0.0925  Loss_G: 3.5982  D(x): 0.9540    D(G(z)):
0.0414 / 0.0531
[4/10][750/3166]        Loss_D: 0.2603  Loss_G: 5.1156  D(x): 0.9806    D(G(z)):
0.1901 / 0.0109
[4/10][800/3166]        Loss_D: 0.1541  Loss_G: 3.9738  D(x): 0.8891    D(G(z)):
0.0250 / 0.0383
[4/10][850/3166]        Loss_D: 0.2445  Loss_G: 3.6540  D(x): 0.8606    D(G(z)):
0.0633 / 0.0441
[4/10][900/3166]        Loss_D: 0.1537  Loss_G: 4.1029  D(x): 0.8822    D(G(z)):
0.0088 / 0.0310
[4/10][950/3166]        Loss_D: 0.1991  Loss_G: 3.7793  D(x): 0.8548    D(G(z)):
0.0124 / 0.0417
[4/10][1000/3166]       Loss_D: 0.1222  Loss_G: 4.5830  D(x): 0.9463    D(G(z)):
0.0587 / 0.0200
```

```
[4/10][1050/3166]       Loss_D: 0.1679  Loss_G: 5.0239  D(x): 0.8656    D(G(z)):
0.0056 / 0.0161
[4/10][1100/3166]       Loss_D: 0.1078  Loss_G: 4.4157  D(x): 0.9471    D(G(z)):
0.0466 / 0.0210
[4/10][1150/3166]       Loss_D: 0.1272  Loss_G: 3.7310  D(x): 0.9002    D(G(z)):
0.0134 / 0.0451
[4/10][1200/3166]       Loss_D: 0.0863  Loss_G: 4.4514  D(x): 0.9563    D(G(z)):
0.0371 / 0.0181
[4/10][1250/3166]       Loss_D: 0.1656  Loss_G: 4.2598  D(x): 0.9406    D(G(z)):
0.0885 / 0.0235
[4/10][1300/3166]       Loss_D: 0.1488  Loss_G: 5.2031  D(x): 0.9789    D(G(z)):
0.1037 / 0.0098
[4/10][1350/3166]       Loss_D: 1.0732  Loss_G: 6.4218  D(x): 0.9553    D(G(z)):
0.5312 / 0.0043
[4/10][1400/3166]       Loss_D: 0.2321  Loss_G: 3.2292  D(x): 0.8362    D(G(z)):
0.0222 / 0.0668
[4/10][1450/3166]       Loss_D: 0.9781  Loss_G: 2.9072  D(x): 0.5587    D(G(z)):
0.0194 / 0.1607
[4/10][1500/3166]       Loss_D: 0.1296  Loss_G: 5.5408  D(x): 0.9791    D(G(z)):
0.0913 / 0.0066
[4/10][1550/3166]       Loss_D: 0.5806  Loss_G: 8.0275  D(x): 0.9748    D(G(z)):
0.3544 / 0.0007
[4/10][1600/3166]       Loss_D: 0.2443  Loss_G: 3.9402  D(x): 0.8389    D(G(z)):
0.0125 / 0.0365
[4/10][1650/3166]       Loss_D: 0.2997  Loss_G: 4.3719  D(x): 0.7701    D(G(z)):
0.0049 / 0.0267
[4/10][1700/3166]       Loss_D: 1.3515  Loss_G: 6.7456  D(x): 0.9896    D(G(z)):
0.6223 / 0.0036
[4/10][1750/3166]       Loss_D: 0.1688  Loss_G: 4.2759  D(x): 0.9358    D(G(z)):
0.0831 / 0.0234
[4/10][1800/3166]       Loss_D: 0.1210  Loss_G: 4.4168  D(x): 0.9236    D(G(z)):
0.0355 / 0.0216
[4/10][1850/3166]       Loss_D: 0.2401  Loss_G: 6.0210  D(x): 0.9618    D(G(z)):
0.1640 / 0.0041
[4/10][1900/3166]       Loss_D: 0.0990  Loss_G: 3.5711  D(x): 0.9245    D(G(z)):
0.0141 / 0.0488
[4/10][1950/3166]       Loss_D: 0.1354  Loss_G: 4.3963  D(x): 0.9196    D(G(z)):
0.0390 / 0.0215
[4/10][2000/3166]       Loss_D: 1.8193  Loss_G: 0.6771  D(x): 0.2737    D(G(z)):
0.0007 / 0.6034
[4/10][2050/3166]       Loss_D: 0.1238  Loss_G: 5.0608  D(x): 0.9065    D(G(z)):
0.0172 / 0.0143
[4/10][2100/3166]       Loss_D: 0.2204  Loss_G: 3.0449  D(x): 0.8464    D(G(z)):
0.0140 / 0.0808
[4/10][2150/3166]       Loss_D: 1.2562  Loss_G: 10.6426 D(x): 0.9751    D(G(z)):
0.5836 / 0.0001
[4/10][2200/3166]       Loss_D: 1.0343  Loss_G: 2.5853  D(x): 0.4855    D(G(z)):
0.0049 / 0.1358
```

```
[4/10][2250/3166]       Loss_D: 0.1947  Loss_G: 3.5109  D(x): 0.8938    D(G(z)):
0.0572 / 0.0485
[4/10][2300/3166]       Loss_D: 0.1970  Loss_G: 3.3535  D(x): 0.8573    D(G(z)):
0.0247 / 0.0656
[4/10][2350/3166]       Loss_D: 0.2756  Loss_G: 2.9793  D(x): 0.8215    D(G(z)):
0.0416 / 0.0783
[4/10][2400/3166]       Loss_D: 0.0812  Loss_G: 4.0240  D(x): 0.9514    D(G(z)):
0.0265 / 0.0318
[4/10][2450/3166]       Loss_D: 0.0689  Loss_G: 4.7712  D(x): 0.9782    D(G(z)):
0.0429 / 0.0152
[4/10][2500/3166]       Loss_D: 0.1345  Loss_G: 5.1345  D(x): 0.9521    D(G(z)):
0.0682 / 0.0107
[4/10][2550/3166]       Loss_D: 0.2484  Loss_G: 4.2956  D(x): 0.9021    D(G(z)):
0.1125 / 0.0226
[4/10][2600/3166]       Loss_D: 0.2410  Loss_G: 3.6604  D(x): 0.8501    D(G(z)):
0.0411 / 0.0522
[4/10][2650/3166]       Loss_D: 0.4648  Loss_G: 7.5549  D(x): 0.9841    D(G(z)):
0.3169 / 0.0007
[4/10][2700/3166]       Loss_D: 0.1244  Loss_G: 5.1756  D(x): 0.9268    D(G(z)):
0.0326 / 0.0110
[4/10][2750/3166]       Loss_D: 0.1608  Loss_G: 5.0012  D(x): 0.9482    D(G(z)):
0.0908 / 0.0120
[4/10][2800/3166]       Loss_D: 0.0730  Loss_G: 6.0507  D(x): 0.9641    D(G(z)):
0.0337 / 0.0048
[4/10][2850/3166]       Loss_D: 0.0765  Loss_G: 5.3006  D(x): 0.9858    D(G(z)):
0.0557 / 0.0092
[4/10][2900/3166]       Loss_D: 0.1298  Loss_G: 3.9201  D(x): 0.9423    D(G(z)):
0.0597 / 0.0353
[4/10][2950/3166]       Loss_D: 0.1997  Loss_G: 4.2319  D(x): 0.8556    D(G(z)):
0.0106 / 0.0247
[4/10][3000/3166]       Loss_D: 0.1536  Loss_G: 4.8477  D(x): 0.9549    D(G(z)):
0.0888 / 0.0130
[4/10][3050/3166]       Loss_D: 0.5074  Loss_G: 7.1217  D(x): 0.9768    D(G(z)):
0.3207 / 0.0020
[4/10][3100/3166]       Loss_D: 0.0699  Loss_G: 5.1929  D(x): 0.9687    D(G(z)):
0.0308 / 0.0139
[4/10][3150/3166]       Loss_D: 0.2387  Loss_G: 7.1646  D(x): 0.9964    D(G(z)):
0.1863 / 0.0012
[5/10][0/3166]  Loss_D: 0.0993  Loss_G: 4.2458  D(x): 0.9397    D(G(z)): 0.0203
/ 0.0249
[5/10][50/3166] Loss_D: 0.1652  Loss_G: 4.0447  D(x): 0.8822    D(G(z)): 0.0116
/ 0.0419
[5/10][100/3166]        Loss_D: 0.1390  Loss_G: 3.9907  D(x): 0.8883    D(G(z)):
0.0071 / 0.0360
[5/10][150/3166]        Loss_D: 0.1339  Loss_G: 5.1667  D(x): 0.9288    D(G(z)):
0.0389 / 0.0098
[5/10][200/3166]        Loss_D: 0.2626  Loss_G: 5.1875  D(x): 0.9552    D(G(z)):
0.1539 / 0.0102
```

```
[5/10][250/3166]        Loss_D: 0.1719  Loss_G: 3.6237  D(x): 0.8896    D(G(z)):
0.0275 / 0.0469
[5/10][300/3166]        Loss_D: 0.1800  Loss_G: 2.9484  D(x): 0.8625    D(G(z)):
0.0182 / 0.0973
[5/10][350/3166]        Loss_D: 0.1525  Loss_G: 5.1183  D(x): 0.9575    D(G(z)):
0.0919 / 0.0100
[5/10][400/3166]        Loss_D: 0.0782  Loss_G: 4.6311  D(x): 0.9412    D(G(z)):
0.0132 / 0.0202
[5/10][450/3166]        Loss_D: 0.1750  Loss_G: 5.2667  D(x): 0.9874    D(G(z)):
0.1154 / 0.0094
[5/10][500/3166]        Loss_D: 0.1680  Loss_G: 4.0007  D(x): 0.9069    D(G(z)):
0.0582 / 0.0301
[5/10][550/3166]        Loss_D: 0.0712  Loss_G: 5.1212  D(x): 0.9942    D(G(z)):
0.0605 / 0.0116
[5/10][600/3166]        Loss_D: 0.2399  Loss_G: 3.9889  D(x): 0.8716    D(G(z)):
0.0538 / 0.0380
[5/10][650/3166]        Loss_D: 0.1859  Loss_G: 5.1761  D(x): 0.8698    D(G(z)):
0.0173 / 0.0140
[5/10][700/3166]        Loss_D: 0.0985  Loss_G: 5.2534  D(x): 0.9743    D(G(z)):
0.0655 / 0.0098
[5/10][750/3166]        Loss_D: 0.1445  Loss_G: 5.4719  D(x): 0.9778    D(G(z)):
0.1068 / 0.0056
[5/10][800/3166]        Loss_D: 0.2152  Loss_G: 3.8093  D(x): 0.8285    D(G(z)):
0.0057 / 0.0455
[5/10][850/3166]        Loss_D: 2.3607  Loss_G: 2.3772  D(x): 0.1993    D(G(z)):
0.0001 / 0.1897
[5/10][900/3166]        Loss_D: 0.1376  Loss_G: 4.4854  D(x): 0.9387    D(G(z)):
0.0637 / 0.0202
[5/10][950/3166]        Loss_D: 1.8901  Loss_G: 2.2450  D(x): 0.2713    D(G(z)):
0.0011 / 0.2248
[5/10][1000/3166]       Loss_D: 0.1020  Loss_G: 4.9071  D(x): 0.9386    D(G(z)):
0.0313 / 0.0154
[5/10][1050/3166]       Loss_D: 0.1292  Loss_G: 4.3041  D(x): 0.9480    D(G(z)):
0.0531 / 0.0250
[5/10][1100/3166]       Loss_D: 0.2414  Loss_G: 2.7133  D(x): 0.8222    D(G(z)):
0.0128 / 0.1206
[5/10][1150/3166]       Loss_D: 0.0512  Loss_G: 5.4865  D(x): 0.9986    D(G(z)):
0.0459 / 0.0093
[5/10][1200/3166]       Loss_D: 0.0419  Loss_G: 6.1554  D(x): 0.9731    D(G(z)):
0.0133 / 0.0038
[5/10][1250/3166]       Loss_D: 0.2184  Loss_G: 6.6420  D(x): 0.9863    D(G(z)):
0.1531 / 0.0030
[5/10][1300/3166]       Loss_D: 1.9472  Loss_G: 10.8927 D(x): 0.9979    D(G(z)):
0.7455 / 0.0001
[5/10][1350/3166]       Loss_D: 0.1402  Loss_G: 4.2547  D(x): 0.9274    D(G(z)):
0.0540 / 0.0311
[5/10][1400/3166]       Loss_D: 0.0773  Loss_G: 4.4958  D(x): 0.9632    D(G(z)):
0.0362 / 0.0205
```

```
[5/10][1450/3166]       Loss_D: 0.3105  Loss_G: 3.5901  D(x): 0.8199    D(G(z)):
0.0281 / 0.0597
[5/10][1500/3166]       Loss_D: 1.0615  Loss_G: 10.4718 D(x): 0.9979    D(G(z)):
0.5436 / 0.0001
[5/10][1550/3166]       Loss_D: 0.2202  Loss_G: 2.6162  D(x): 0.8724    D(G(z)):
0.0432 / 0.1281
[5/10][1600/3166]       Loss_D: 0.1214  Loss_G: 4.1072  D(x): 0.9623    D(G(z)):
0.0685 / 0.0229
[5/10][1650/3166]       Loss_D: 0.0502  Loss_G: 5.1189  D(x): 0.9838    D(G(z)):
0.0324 / 0.0111
[5/10][1700/3166]       Loss_D: 0.2534  Loss_G: 5.9831  D(x): 0.9832    D(G(z)):
0.1740 / 0.0040
[5/10][1750/3166]       Loss_D: 0.2115  Loss_G: 5.2591  D(x): 0.9879    D(G(z)):
0.1559 / 0.0084
[5/10][1800/3166]       Loss_D: 0.9508  Loss_G: 9.2732  D(x): 0.9946    D(G(z)):
0.4773 / 0.0002
[5/10][1850/3166]       Loss_D: 0.1643  Loss_G: 5.5512  D(x): 0.9950    D(G(z)):
0.1313 / 0.0061
[5/10][1900/3166]       Loss_D: 1.1232  Loss_G: 2.6677  D(x): 0.4590    D(G(z)):
0.0050 / 0.1492
[5/10][1950/3166]       Loss_D: 0.7516  Loss_G: 5.8409  D(x): 0.9903    D(G(z)):
0.3970 / 0.0085
[5/10][2000/3166]       Loss_D: 0.1659  Loss_G: 3.7345  D(x): 0.9075    D(G(z)):
0.0506 / 0.0448
[5/10][2050/3166]       Loss_D: 0.2526  Loss_G: 3.2480  D(x): 0.8169    D(G(z)):
0.0101 / 0.0845
[5/10][2100/3166]       Loss_D: 0.4930  Loss_G: 4.5260  D(x): 0.9408    D(G(z)):
0.2637 / 0.0199
[5/10][2150/3166]       Loss_D: 0.1742  Loss_G: 4.3105  D(x): 0.8933    D(G(z)):
0.0423 / 0.0234
[5/10][2200/3166]       Loss_D: 0.1995  Loss_G: 5.8669  D(x): 0.9550    D(G(z)):
0.1227 / 0.0072
[5/10][2250/3166]       Loss_D: 0.1351  Loss_G: 4.3958  D(x): 0.9029    D(G(z)):
0.0214 / 0.0319
[5/10][2300/3166]       Loss_D: 0.1835  Loss_G: 5.4104  D(x): 0.9530    D(G(z)):
0.1109 / 0.0084
[5/10][2350/3166]       Loss_D: 0.2373  Loss_G: 4.2546  D(x): 0.8369    D(G(z)):
0.0084 / 0.0344
[5/10][2400/3166]       Loss_D: 0.0639  Loss_G: 5.4114  D(x): 0.9832    D(G(z)):
0.0424 / 0.0087
[5/10][2450/3166]       Loss_D: 0.1598  Loss_G: 5.9218  D(x): 0.9858    D(G(z)):
0.1223 / 0.0039
[5/10][2500/3166]       Loss_D: 0.0806  Loss_G: 4.2461  D(x): 0.9561    D(G(z)):
0.0318 / 0.0279
[5/10][2550/3166]       Loss_D: 0.2743  Loss_G: 5.0144  D(x): 0.9865    D(G(z)):
0.1890 / 0.0133
[5/10][2600/3166]       Loss_D: 0.0647  Loss_G: 6.4305  D(x): 0.9481    D(G(z)):
0.0071 / 0.0042
```

```
[5/10][2650/3166]       Loss_D: 0.1580  Loss_G: 5.0526  D(x): 0.9319    D(G(z)):
0.0665 / 0.0104
[5/10][2700/3166]       Loss_D: 0.1893  Loss_G: 4.7937  D(x): 0.8876    D(G(z)):
0.0237 / 0.0194
[5/10][2750/3166]       Loss_D: 0.1213  Loss_G: 4.4481  D(x): 0.9200    D(G(z)):
0.0195 / 0.0241
[5/10][2800/3166]       Loss_D: 0.2041  Loss_G: 4.3910  D(x): 0.8996    D(G(z)):
0.0703 / 0.0223
[5/10][2850/3166]       Loss_D: 0.1831  Loss_G: 5.4385  D(x): 0.9914    D(G(z)):
0.1372 / 0.0064
[5/10][2900/3166]       Loss_D: 0.1870  Loss_G: 4.3263  D(x): 0.9378    D(G(z)):
0.1012 / 0.0222
[5/10][2950/3166]       Loss_D: 0.2758  Loss_G: 6.4534  D(x): 0.9622    D(G(z)):
0.1715 / 0.0041
[5/10][3000/3166]       Loss_D: 0.0727  Loss_G: 4.6154  D(x): 0.9778    D(G(z)):
0.0460 / 0.0206
[5/10][3050/3166]       Loss_D: 0.2338  Loss_G: 5.0817  D(x): 0.9509    D(G(z)):
0.1389 / 0.0115
[5/10][3100/3166]       Loss_D: 0.1065  Loss_G: 5.1846  D(x): 0.9377    D(G(z)):
0.0164 / 0.0125
[5/10][3150/3166]       Loss_D: 0.0798  Loss_G: 4.9466  D(x): 0.9690    D(G(z)):
0.0444 / 0.0129
[6/10][0/3166]  Loss_D: 1.7054  Loss_G: 0.0739  D(x): 0.2751    D(G(z)): 0.0003
/ 0.9393
[6/10][50/3166] Loss_D: 0.5300  Loss_G: 4.3621  D(x): 0.7045    D(G(z)): 0.0030
/ 0.0394
[6/10][100/3166]        Loss_D: 0.1169  Loss_G: 5.8489  D(x): 0.9939    D(G(z)):
0.0893 / 0.0059
[6/10][150/3166]        Loss_D: 0.1143  Loss_G: 5.5151  D(x): 0.9880    D(G(z)):
0.0908 / 0.0076
[6/10][200/3166]        Loss_D: 0.4091  Loss_G: 2.3836  D(x): 0.7351    D(G(z)):
0.0073 / 0.1661
[6/10][250/3166]        Loss_D: 0.0367  Loss_G: 5.4233  D(x): 0.9907    D(G(z)):
0.0263 / 0.0080
[6/10][300/3166]        Loss_D: 0.0622  Loss_G: 4.9398  D(x): 0.9590    D(G(z)):
0.0174 / 0.0170
[6/10][350/3166]        Loss_D: 0.0935  Loss_G: 5.6503  D(x): 0.9830    D(G(z)):
0.0690 / 0.0055
[6/10][400/3166]        Loss_D: 0.0893  Loss_G: 4.3522  D(x): 0.9423    D(G(z)):
0.0218 / 0.0239
[6/10][450/3166]        Loss_D: 0.2988  Loss_G: 6.0096  D(x): 0.9548    D(G(z)):
0.1619 / 0.0041
[6/10][500/3166]        Loss_D: 0.2433  Loss_G: 3.7784  D(x): 0.8426    D(G(z)):
0.0287 / 0.0471
[6/10][550/3166]        Loss_D: 0.1357  Loss_G: 4.7312  D(x): 0.9443    D(G(z)):
0.0628 / 0.0177
[6/10][600/3166]        Loss_D: 0.2860  Loss_G: 5.4004  D(x): 0.9227    D(G(z)):
0.1320 / 0.0100
```

```
[6/10][650/3166]        Loss_D: 0.3187  Loss_G: 1.9105  D(x): 0.7756    D(G(z)):
0.0127 / 0.2458
[6/10][700/3166]        Loss_D: 0.1309  Loss_G: 5.1091  D(x): 0.9806    D(G(z)):
0.0937 / 0.0092
[6/10][750/3166]        Loss_D: 0.0605  Loss_G: 5.3967  D(x): 0.9847    D(G(z)):
0.0395 / 0.0113
[6/10][800/3166]        Loss_D: 0.6239  Loss_G: 10.6706 D(x): 0.9982    D(G(z)):
0.3835 / 0.0000
[6/10][850/3166]        Loss_D: 0.0610  Loss_G: 4.5287  D(x): 0.9615    D(G(z)):
0.0190 / 0.0189
[6/10][900/3166]        Loss_D: 0.9124  Loss_G: 9.4006  D(x): 0.9964    D(G(z)):
0.4817 / 0.0002
[6/10][950/3166]        Loss_D: 1.4963  Loss_G: 1.1168  D(x): 0.4422    D(G(z)):
0.0160 / 0.4920
[6/10][1000/3166]       Loss_D: 0.1340  Loss_G: 5.1283  D(x): 0.9706    D(G(z)):
0.0854 / 0.0144
[6/10][1050/3166]       Loss_D: 0.1954  Loss_G: 3.5504  D(x): 0.8539    D(G(z)):
0.0118 / 0.0569
[6/10][1100/3166]       Loss_D: 0.9223  Loss_G: 3.2721  D(x): 0.5600    D(G(z)):
0.0021 / 0.0942
[6/10][1150/3166]       Loss_D: 0.0924  Loss_G: 4.7703  D(x): 0.9461    D(G(z)):
0.0310 / 0.0184
[6/10][1200/3166]       Loss_D: 0.1246  Loss_G: 4.3755  D(x): 0.9160    D(G(z)):
0.0271 / 0.0283
[6/10][1250/3166]       Loss_D: 0.3298  Loss_G: 7.1686  D(x): 0.9901    D(G(z)):
0.2365 / 0.0013
[6/10][1300/3166]       Loss_D: 0.1562  Loss_G: 6.8217  D(x): 0.9936    D(G(z)):
0.1027 / 0.0032
[6/10][1350/3166]       Loss_D: 0.0758  Loss_G: 4.9188  D(x): 0.9736    D(G(z)):
0.0436 / 0.0135
[6/10][1400/3166]       Loss_D: 0.2778  Loss_G: 8.0948  D(x): 0.9985    D(G(z)):
0.2099 / 0.0004
[6/10][1450/3166]       Loss_D: 0.0382  Loss_G: 6.3820  D(x): 0.9749    D(G(z)):
0.0119 / 0.0045
[6/10][1500/3166]       Loss_D: 0.2002  Loss_G: 4.9285  D(x): 0.8469    D(G(z)):
0.0060 / 0.0200
[6/10][1550/3166]       Loss_D: 0.1952  Loss_G: 3.7836  D(x): 0.9072    D(G(z)):
0.0752 / 0.0392
[6/10][1600/3166]       Loss_D: 0.1672  Loss_G: 4.8337  D(x): 0.9693    D(G(z)):
0.1077 / 0.0144
[6/10][1650/3166]       Loss_D: 0.1887  Loss_G: 4.7573  D(x): 0.9549    D(G(z)):
0.1146 / 0.0167
[6/10][1700/3166]       Loss_D: 0.1797  Loss_G: 5.1365  D(x): 0.9380    D(G(z)):
0.0942 / 0.0136
[6/10][1750/3166]       Loss_D: 0.0617  Loss_G: 5.2297  D(x): 0.9551    D(G(z)):
0.0082 / 0.0108
[6/10][1800/3166]       Loss_D: 0.6446  Loss_G: 5.4932  D(x): 0.9157    D(G(z)):
0.3282 / 0.0086
```

```
[6/10][1850/3166]       Loss_D: 0.1872  Loss_G: 3.7993  D(x): 0.8712    D(G(z)):
0.0275 / 0.0423
[6/10][1900/3166]       Loss_D: 0.1336  Loss_G: 4.5865  D(x): 0.9488    D(G(z)):
0.0701 / 0.0179
[6/10][1950/3166]       Loss_D: 0.3044  Loss_G: 1.6092  D(x): 0.7827    D(G(z)):
0.0065 / 0.2998
[6/10][2000/3166]       Loss_D: 0.2297  Loss_G: 6.3071  D(x): 0.9931    D(G(z)):
0.1828 / 0.0027
[6/10][2050/3166]       Loss_D: 0.1322  Loss_G: 3.5260  D(x): 0.9137    D(G(z)):
0.0219 / 0.0515
[6/10][2100/3166]       Loss_D: 0.1003  Loss_G: 4.7435  D(x): 0.9839    D(G(z)):
0.0744 / 0.0152
[6/10][2150/3166]       Loss_D: 0.0513  Loss_G: 5.5423  D(x): 0.9596    D(G(z)):
0.0083 / 0.0079
[6/10][2200/3166]       Loss_D: 0.1115  Loss_G: 4.3328  D(x): 0.9093    D(G(z)):
0.0046 / 0.0270
[6/10][2250/3166]       Loss_D: 0.2695  Loss_G: 1.4047  D(x): 0.8007    D(G(z)):
0.0130 / 0.3496
[6/10][2300/3166]       Loss_D: 0.1879  Loss_G: 5.4392  D(x): 0.9808    D(G(z)):
0.1346 / 0.0082
[6/10][2350/3166]       Loss_D: 0.7805  Loss_G: 2.7083  D(x): 0.5618    D(G(z)):
0.0012 / 0.1435
[6/10][2400/3166]       Loss_D: 0.0519  Loss_G: 5.9486  D(x): 0.9719    D(G(z)):
0.0214 / 0.0059
[6/10][2450/3166]       Loss_D: 0.5398  Loss_G: 5.4223  D(x): 0.9333    D(G(z)):
0.3034 / 0.0089
[6/10][2500/3166]       Loss_D: 0.2178  Loss_G: 5.1208  D(x): 0.9828    D(G(z)):
0.1644 / 0.0092
[6/10][2550/3166]       Loss_D: 0.1182  Loss_G: 4.5809  D(x): 0.9358    D(G(z)):
0.0447 / 0.0218
[6/10][2600/3166]       Loss_D: 0.0676  Loss_G: 5.5309  D(x): 0.9917    D(G(z)):
0.0532 / 0.0062
[6/10][2650/3166]       Loss_D: 0.1585  Loss_G: 4.0368  D(x): 0.8877    D(G(z)):
0.0210 / 0.0357
[6/10][2700/3166]       Loss_D: 0.0347  Loss_G: 6.5849  D(x): 0.9706    D(G(z)):
0.0040 / 0.0030
[6/10][2750/3166]       Loss_D: 0.0491  Loss_G: 5.1092  D(x): 0.9841    D(G(z)):
0.0302 / 0.0112
[6/10][2800/3166]       Loss_D: 0.1898  Loss_G: 7.2285  D(x): 0.9938    D(G(z)):
0.1389 / 0.0014
[6/10][2850/3166]       Loss_D: 0.1577  Loss_G: 4.0527  D(x): 0.9221    D(G(z)):
0.0477 / 0.0308
[6/10][2900/3166]       Loss_D: 0.1870  Loss_G: 6.7249  D(x): 0.9905    D(G(z)):
0.1264 / 0.0020
[6/10][2950/3166]       Loss_D: 0.1881  Loss_G: 4.4725  D(x): 0.9390    D(G(z)):
0.0964 / 0.0235
[6/10][3000/3166]       Loss_D: 0.2307  Loss_G: 7.1399  D(x): 0.9788    D(G(z)):
0.1527 / 0.0015
```

```
[6/10][3050/3166]       Loss_D: 0.3287  Loss_G: 3.0298  D(x): 0.7944    D(G(z)):
0.0233 / 0.1023
[6/10][3100/3166]       Loss_D: 0.4053  Loss_G: 7.5448  D(x): 0.9907    D(G(z)):
0.2644 / 0.0009
[6/10][3150/3166]       Loss_D: 0.0615  Loss_G: 4.9970  D(x): 0.9852    D(G(z)):
0.0427 / 0.0125
[7/10][0/3166]  Loss_D: 0.0801  Loss_G: 4.3229  D(x): 0.9470    D(G(z)): 0.0218
/ 0.0264
[7/10][50/3166] Loss_D: 0.0291  Loss_G: 5.2289  D(x): 0.9871    D(G(z)): 0.0154
/ 0.0103
[7/10][100/3166]        Loss_D: 0.1509  Loss_G: 4.1806  D(x): 0.9066    D(G(z)):
0.0363 / 0.0347
[7/10][150/3166]        Loss_D: 0.1794  Loss_G: 7.4450  D(x): 0.9855    D(G(z)):
0.1320 / 0.0012
[7/10][200/3166]        Loss_D: 0.1424  Loss_G: 6.8739  D(x): 0.9917    D(G(z)):
0.0871 / 0.0030
[7/10][250/3166]        Loss_D: 0.3371  Loss_G: 6.6195  D(x): 0.9376    D(G(z)):
0.1956 / 0.0027
[7/10][300/3166]        Loss_D: 0.2262  Loss_G: 5.8721  D(x): 0.9795    D(G(z)):
0.1622 / 0.0060
[7/10][350/3166]        Loss_D: 0.0825  Loss_G: 4.6912  D(x): 0.9876    D(G(z)):
0.0643 / 0.0142
[7/10][400/3166]        Loss_D: 0.1394  Loss_G: 6.5347  D(x): 0.9864    D(G(z)):
0.1055 / 0.0026
[7/10][450/3166]        Loss_D: 0.2659  Loss_G: 1.7156  D(x): 0.8165    D(G(z)):
0.0120 / 0.2535
[7/10][500/3166]        Loss_D: 0.0891  Loss_G: 5.2627  D(x): 0.9671    D(G(z)):
0.0476 / 0.0099
[7/10][550/3166]        Loss_D: 0.3539  Loss_G: 5.6445  D(x): 0.9807    D(G(z)):
0.2156 / 0.0066
[7/10][600/3166]        Loss_D: 0.2539  Loss_G: 3.6809  D(x): 0.8283    D(G(z)):
0.0083 / 0.0599
[7/10][650/3166]        Loss_D: 0.1174  Loss_G: 4.6507  D(x): 0.9103    D(G(z)):
0.0124 / 0.0238
[7/10][700/3166]        Loss_D: 0.0931  Loss_G: 5.3654  D(x): 0.9683    D(G(z)):
0.0540 / 0.0104
[7/10][750/3166]        Loss_D: 0.0756  Loss_G: 5.2991  D(x): 0.9518    D(G(z)):
0.0203 / 0.0104
[7/10][800/3166]        Loss_D: 0.0942  Loss_G: 4.5699  D(x): 0.9271    D(G(z)):
0.0089 / 0.0214
[7/10][850/3166]        Loss_D: 0.0889  Loss_G: 5.6317  D(x): 0.9873    D(G(z)):
0.0690 / 0.0086
[7/10][900/3166]        Loss_D: 0.0733  Loss_G: 5.5918  D(x): 0.9889    D(G(z)):
0.0560 / 0.0063
[7/10][950/3166]        Loss_D: 0.0740  Loss_G: 5.4238  D(x): 0.9849    D(G(z)):
0.0525 / 0.0071
[7/10][1000/3166]       Loss_D: 1.5058  Loss_G: 0.8515  D(x): 0.3625    D(G(z)):
0.0027 / 0.5353
```

```
[7/10][1050/3166]       Loss_D: 0.1695  Loss_G: 6.1764  D(x): 0.9668    D(G(z)):
0.0944 / 0.0044
[7/10][1100/3166]       Loss_D: 0.2022  Loss_G: 6.6979  D(x): 0.9387    D(G(z)):
0.1050 / 0.0028
[7/10][1150/3166]       Loss_D: 0.0623  Loss_G: 5.0381  D(x): 0.9799    D(G(z)):
0.0389 / 0.0115
[7/10][1200/3166]       Loss_D: 0.3091  Loss_G: 3.1361  D(x): 0.8336    D(G(z)):
0.0735 / 0.0806
[7/10][1250/3166]       Loss_D: 0.2189  Loss_G: 4.5666  D(x): 0.8770    D(G(z)):
0.0484 / 0.0288
[7/10][1300/3166]       Loss_D: 1.1385  Loss_G: 12.0128 D(x): 0.9945    D(G(z)):
0.5506 / 0.0000
[7/10][1350/3166]       Loss_D: 0.2725  Loss_G: 6.5393  D(x): 0.9533    D(G(z)):
0.1716 / 0.0026
[7/10][1400/3166]       Loss_D: 0.0907  Loss_G: 4.6645  D(x): 0.9796    D(G(z)):
0.0515 / 0.0167
[7/10][1450/3166]       Loss_D: 0.1003  Loss_G: 5.4366  D(x): 0.9875    D(G(z)):
0.0741 / 0.0073
[7/10][1500/3166]       Loss_D: 0.1616  Loss_G: 6.1578  D(x): 0.9926    D(G(z)):
0.1322 / 0.0037
[7/10][1550/3166]       Loss_D: 0.1302  Loss_G: 4.7074  D(x): 0.8937    D(G(z)):
0.0069 / 0.0201
[7/10][1600/3166]       Loss_D: 0.0356  Loss_G: 5.9745  D(x): 0.9720    D(G(z)):
0.0062 / 0.0060
[7/10][1650/3166]       Loss_D: 0.4635  Loss_G: 13.5098 D(x): 0.9952    D(G(z)):
0.2871 / 0.0000
[7/10][1700/3166]       Loss_D: 0.3956  Loss_G: 5.5835  D(x): 0.9913    D(G(z)):
0.2472 / 0.0081
[7/10][1750/3166]       Loss_D: 0.2760  Loss_G: 6.2081  D(x): 0.9442    D(G(z)):
0.1346 / 0.0041
[7/10][1800/3166]       Loss_D: 1.7451  Loss_G: 2.7283  D(x): 0.3029    D(G(z)):
0.0001 / 0.1420
[7/10][1850/3166]       Loss_D: 0.0774  Loss_G: 5.2506  D(x): 0.9900    D(G(z)):
0.0562 / 0.0130
[7/10][1900/3166]       Loss_D: 0.4044  Loss_G: 7.2019  D(x): 0.9750    D(G(z)):
0.2794 / 0.0014
[7/10][1950/3166]       Loss_D: 0.1009  Loss_G: 5.6210  D(x): 0.9228    D(G(z)):
0.0131 / 0.0068
[7/10][2000/3166]       Loss_D: 0.4107  Loss_G: 8.1259  D(x): 0.9906    D(G(z)):
0.2942 / 0.0005
[7/10][2050/3166]       Loss_D: 0.1399  Loss_G: 4.4258  D(x): 0.9184    D(G(z)):
0.0390 / 0.0252
[7/10][2100/3166]       Loss_D: 0.0780  Loss_G: 5.1167  D(x): 0.9808    D(G(z)):
0.0536 / 0.0117
[7/10][2150/3166]       Loss_D: 0.0509  Loss_G: 4.8570  D(x): 0.9689    D(G(z)):
0.0174 / 0.0162
[7/10][2200/3166]       Loss_D: 0.2790  Loss_G: 6.6695  D(x): 0.9419    D(G(z)):
0.1534 / 0.0027
```

```
[7/10][2250/3166]       Loss_D: 0.1587  Loss_G: 6.2467  D(x): 0.9515    D(G(z)):
0.0883 / 0.0042
[7/10][2300/3166]       Loss_D: 0.4328  Loss_G: 3.0812  D(x): 0.7065    D(G(z)):
0.0042 / 0.1002
[7/10][2350/3166]       Loss_D: 0.0576  Loss_G: 4.4909  D(x): 0.9707    D(G(z)):
0.0257 / 0.0219
[7/10][2400/3166]       Loss_D: 0.0888  Loss_G: 5.0247  D(x): 0.9815    D(G(z)):
0.0628 / 0.0133
[7/10][2450/3166]       Loss_D: 0.0477  Loss_G: 4.6127  D(x): 0.9728    D(G(z)):
0.0182 / 0.0184
[7/10][2500/3166]       Loss_D: 0.1520  Loss_G: 4.9642  D(x): 0.9184    D(G(z)):
0.0496 / 0.0157
[7/10][2550/3166]       Loss_D: 0.1349  Loss_G: 4.0088  D(x): 0.9121    D(G(z)):
0.0224 / 0.0382
[7/10][2600/3166]       Loss_D: 0.1029  Loss_G: 5.3803  D(x): 0.9231    D(G(z)):
0.0160 / 0.0131
[7/10][2650/3166]       Loss_D: 0.1519  Loss_G: 4.0654  D(x): 0.9181    D(G(z)):
0.0445 / 0.0386
[7/10][2700/3166]       Loss_D: 0.3574  Loss_G: 3.4713  D(x): 0.7724    D(G(z)):
0.0420 / 0.0883
[7/10][2750/3166]       Loss_D: 0.1072  Loss_G: 4.2855  D(x): 0.9184    D(G(z)):
0.0136 / 0.0368
[7/10][2800/3166]       Loss_D: 0.1213  Loss_G: 4.7051  D(x): 0.9691    D(G(z)):
0.0751 / 0.0136
[7/10][2850/3166]       Loss_D: 0.0785  Loss_G: 5.6190  D(x): 0.9687    D(G(z)):
0.0429 / 0.0070
[7/10][2900/3166]       Loss_D: 0.1299  Loss_G: 5.1534  D(x): 0.9505    D(G(z)):
0.0624 / 0.0104
[7/10][2950/3166]       Loss_D: 0.0519  Loss_G: 5.8673  D(x): 0.9782    D(G(z)):
0.0261 / 0.0058
[7/10][3000/3166]       Loss_D: 0.0977  Loss_G: 5.2334  D(x): 0.9858    D(G(z)):
0.0738 / 0.0102
[7/10][3050/3166]       Loss_D: 0.2686  Loss_G: 8.6192  D(x): 0.9881    D(G(z)):
0.1757 / 0.0004
[7/10][3100/3166]       Loss_D: 0.1425  Loss_G: 4.1135  D(x): 0.9278    D(G(z)):
0.0454 / 0.0384
[7/10][3150/3166]       Loss_D: 0.1900  Loss_G: 7.8092  D(x): 0.9997    D(G(z)):
0.1536 / 0.0008
[8/10][0/3166]  Loss_D: 0.1042  Loss_G: 4.7485  D(x): 0.9197    D(G(z)): 0.0113
/ 0.0182
[8/10][50/3166] Loss_D: 0.0512  Loss_G: 4.5068  D(x): 0.9961    D(G(z)): 0.0414
/ 0.0179
[8/10][100/3166]        Loss_D: 0.5699  Loss_G: 7.2121  D(x): 0.9985    D(G(z)):
0.3527 / 0.0017
[8/10][150/3166]        Loss_D: 0.0588  Loss_G: 5.5877  D(x): 0.9741    D(G(z)):
0.0300 / 0.0073
[8/10][200/3166]        Loss_D: 0.2670  Loss_G: 9.2201  D(x): 0.9988    D(G(z)):
0.2055 / 0.0002
```

```
[8/10][250/3166]        Loss_D: 0.2195  Loss_G: 5.2668  D(x): 0.9425    D(G(z)):
0.1250 / 0.0101
[8/10][300/3166]        Loss_D: 0.4333  Loss_G: 3.5567  D(x): 0.7730    D(G(z)):
0.0677 / 0.0554
[8/10][350/3166]        Loss_D: 0.1057  Loss_G: 4.8960  D(x): 0.9170    D(G(z)):
0.0101 / 0.0211
[8/10][400/3166]        Loss_D: 0.0984  Loss_G: 4.8055  D(x): 0.9254    D(G(z)):
0.0110 / 0.0197
[8/10][450/3166]        Loss_D: 0.3338  Loss_G: 7.6440  D(x): 0.9927    D(G(z)):
0.2140 / 0.0011
[8/10][500/3166]        Loss_D: 0.0826  Loss_G: 7.1976  D(x): 0.9317    D(G(z)):
0.0059 / 0.0032
[8/10][550/3166]        Loss_D: 0.0401  Loss_G: 5.5639  D(x): 0.9875    D(G(z)):
0.0263 / 0.0076
[8/10][600/3166]        Loss_D: 0.1012  Loss_G: 5.8901  D(x): 0.9723    D(G(z)):
0.0573 / 0.0061
[8/10][650/3166]        Loss_D: 0.0902  Loss_G: 5.3609  D(x): 0.9760    D(G(z)):
0.0593 / 0.0082
[8/10][700/3166]        Loss_D: 0.0363  Loss_G: 5.7176  D(x): 0.9789    D(G(z)):
0.0134 / 0.0081
[8/10][750/3166]        Loss_D: 0.0552  Loss_G: 5.0847  D(x): 0.9908    D(G(z)):
0.0436 / 0.0097
[8/10][800/3166]        Loss_D: 0.0425  Loss_G: 5.4987  D(x): 0.9820    D(G(z)):
0.0222 / 0.0097
[8/10][850/3166]        Loss_D: 0.4716  Loss_G: 4.6530  D(x): 0.7188    D(G(z)):
0.0055 / 0.0389
[8/10][900/3166]        Loss_D: 1.4160  Loss_G: 1.9005  D(x): 0.3731    D(G(z)):
0.0028 / 0.2819
[8/10][950/3166]        Loss_D: 0.1890  Loss_G: 5.1397  D(x): 0.8813    D(G(z)):
0.0346 / 0.0187
[8/10][1000/3166]       Loss_D: 0.6457  Loss_G: 10.6764 D(x): 0.9985    D(G(z)):
0.3895 / 0.0000
[8/10][1050/3166]       Loss_D: 0.0582  Loss_G: 6.7441  D(x): 0.9778    D(G(z)):
0.0254 / 0.0031
[8/10][1100/3166]       Loss_D: 0.2258  Loss_G: 4.7461  D(x): 0.8345    D(G(z)):
0.0044 / 0.0284
[8/10][1150/3166]       Loss_D: 0.0745  Loss_G: 4.1970  D(x): 0.9510    D(G(z)):
0.0207 / 0.0334
[8/10][1200/3166]       Loss_D: 0.0920  Loss_G: 4.4553  D(x): 0.9452    D(G(z)):
0.0260 / 0.0239
[8/10][1250/3166]       Loss_D: 0.1707  Loss_G: 4.4351  D(x): 0.9538    D(G(z)):
0.0913 / 0.0693
[8/10][1300/3166]       Loss_D: 0.2363  Loss_G: 2.6823  D(x): 0.8322    D(G(z)):
0.0190 / 0.1485
[8/10][1350/3166]       Loss_D: 0.5742  Loss_G: 4.6132  D(x): 0.7051    D(G(z)):
0.0054 / 0.0546
[8/10][1400/3166]       Loss_D: 0.1375  Loss_G: 5.6734  D(x): 0.9476    D(G(z)):
0.0703 / 0.0066
```

```
[8/10][1450/3166]       Loss_D: 0.0790  Loss_G: 4.6866  D(x): 0.9437    D(G(z)):
0.0172 / 0.0197
[8/10][1500/3166]       Loss_D: 0.0596  Loss_G: 4.3146  D(x): 0.9566    D(G(z)):
0.0122 / 0.0282
[8/10][1550/3166]       Loss_D: 0.0858  Loss_G: 6.0017  D(x): 0.9641    D(G(z)):
0.0391 / 0.0054
[8/10][1600/3166]       Loss_D: 0.0702  Loss_G: 4.7321  D(x): 0.9639    D(G(z)):
0.0297 / 0.0161
[8/10][1650/3166]       Loss_D: 0.0725  Loss_G: 4.8240  D(x): 0.9409    D(G(z)):
0.0070 / 0.0229
[8/10][1700/3166]       Loss_D: 0.3176  Loss_G: 3.1347  D(x): 0.7996    D(G(z)):
0.0313 / 0.1085
[8/10][1750/3166]       Loss_D: 0.1437  Loss_G: 4.2756  D(x): 0.9174    D(G(z)):
0.0383 / 0.0329
[8/10][1800/3166]       Loss_D: 0.2046  Loss_G: 6.7040  D(x): 0.9904    D(G(z)):
0.1506 / 0.0019
[8/10][1850/3166]       Loss_D: 0.0603  Loss_G: 5.5268  D(x): 0.9549    D(G(z)):
0.0112 / 0.0104
[8/10][1900/3166]       Loss_D: 0.1122  Loss_G: 5.7726  D(x): 0.9790    D(G(z)):
0.0584 / 0.0051
[8/10][1950/3166]       Loss_D: 0.1804  Loss_G: 3.5080  D(x): 0.8955    D(G(z)):
0.0372 / 0.0619
[8/10][2000/3166]       Loss_D: 0.3483  Loss_G: 7.7772  D(x): 0.9928    D(G(z)):
0.2329 / 0.0007
[8/10][2050/3166]       Loss_D: 0.0531  Loss_G: 5.5316  D(x): 0.9896    D(G(z)):
0.0399 / 0.0101
[8/10][2100/3166]       Loss_D: 0.1278  Loss_G: 6.2121  D(x): 0.9940    D(G(z)):
0.1036 / 0.0036
[8/10][2150/3166]       Loss_D: 0.2259  Loss_G: 6.7816  D(x): 0.9942    D(G(z)):
0.1553 / 0.0017
[8/10][2200/3166]       Loss_D: 0.4529  Loss_G: 9.0192  D(x): 0.9979    D(G(z)):
0.2822 / 0.0003
[8/10][2250/3166]       Loss_D: 0.1223  Loss_G: 3.6559  D(x): 0.9175    D(G(z)):
0.0227 / 0.0574
[8/10][2300/3166]       Loss_D: 0.3596  Loss_G: 8.8176  D(x): 0.9869    D(G(z)):
0.2246 / 0.0003
[8/10][2350/3166]       Loss_D: 0.0694  Loss_G: 5.6434  D(x): 0.9732    D(G(z)):
0.0386 / 0.0077
[8/10][2400/3166]       Loss_D: 0.1152  Loss_G: 6.9942  D(x): 0.9940    D(G(z)):
0.0952 / 0.0015
[8/10][2450/3166]       Loss_D: 0.4025  Loss_G: 3.7537  D(x): 0.8652    D(G(z)):
0.1562 / 0.0379
[8/10][2500/3166]       Loss_D: 0.1435  Loss_G: 4.2844  D(x): 0.9130    D(G(z)):
0.0348 / 0.0265
[8/10][2550/3166]       Loss_D: 0.0952  Loss_G: 6.1404  D(x): 0.9898    D(G(z)):
0.0680 / 0.0049
[8/10][2600/3166]       Loss_D: 0.0581  Loss_G: 5.4976  D(x): 0.9792    D(G(z)):
0.0338 / 0.0084
```

```
[8/10][2650/3166]       Loss_D: 0.0549  Loss_G: 5.2661  D(x): 0.9569    D(G(z)):
0.0069 / 0.0109
[8/10][2700/3166]       Loss_D: 0.1088  Loss_G: 4.8374  D(x): 0.9268    D(G(z)):
0.0259 / 0.0205
[8/10][2750/3166]       Loss_D: 0.0478  Loss_G: 5.0052  D(x): 0.9837    D(G(z)):
0.0294 / 0.0140
[8/10][2800/3166]       Loss_D: 0.0664  Loss_G: 5.9265  D(x): 0.9469    D(G(z)):
0.0054 / 0.0089
[8/10][2850/3166]       Loss_D: 0.3112  Loss_G: 1.9335  D(x): 0.7769    D(G(z)):
0.0019 / 0.2737
[8/10][2900/3166]       Loss_D: 0.0468  Loss_G: 5.6446  D(x): 0.9737    D(G(z)):
0.0184 / 0.0087
[8/10][2950/3166]       Loss_D: 0.0291  Loss_G: 7.4361  D(x): 0.9846    D(G(z)):
0.0127 / 0.0025
[8/10][3000/3166]       Loss_D: 0.0643  Loss_G: 5.8844  D(x): 0.9533    D(G(z)):
0.0116 / 0.0086
[8/10][3050/3166]       Loss_D: 0.1138  Loss_G: 4.0589  D(x): 0.9398    D(G(z)):
0.0377 / 0.0356
[8/10][3100/3166]       Loss_D: 0.0565  Loss_G: 4.3258  D(x): 0.9647    D(G(z)):
0.0171 / 0.0275
[8/10][3150/3166]       Loss_D: 0.0814  Loss_G: 5.2150  D(x): 0.9490    D(G(z)):
0.0231 / 0.0166
[9/10][0/3166]  Loss_D: 0.3421  Loss_G: 3.6770  D(x): 0.9574    D(G(z)): 0.1926
/ 0.0496
[9/10][50/3166] Loss_D: 0.2940  Loss_G: 7.4135  D(x): 0.9967    D(G(z)): 0.2003
/ 0.0012
[9/10][100/3166]        Loss_D: 0.0577  Loss_G: 6.2147  D(x): 0.9675    D(G(z)):
0.0216 / 0.0063
[9/10][150/3166]        Loss_D: 0.6543  Loss_G: 13.8023 D(x): 0.9921    D(G(z)):
0.3708 / 0.0000
[9/10][200/3166]        Loss_D: 0.5339  Loss_G: 3.1715  D(x): 0.7859    D(G(z)):
0.1171 / 0.0802
[9/10][250/3166]        Loss_D: 0.2279  Loss_G: 7.1039  D(x): 0.9887    D(G(z)):
0.1433 / 0.0015
[9/10][300/3166]        Loss_D: 0.0837  Loss_G: 5.8660  D(x): 0.9810    D(G(z)):
0.0567 / 0.0073
[9/10][350/3166]        Loss_D: 0.0712  Loss_G: 5.5250  D(x): 0.9725    D(G(z)):
0.0376 / 0.0086
[9/10][400/3166]        Loss_D: 0.1100  Loss_G: 5.2075  D(x): 0.9813    D(G(z)):
0.0654 / 0.0116
[9/10][450/3166]        Loss_D: 0.2203  Loss_G: 4.7715  D(x): 0.9124    D(G(z)):
0.0794 / 0.0167
[9/10][500/3166]        Loss_D: 0.1757  Loss_G: 4.9153  D(x): 0.8878    D(G(z)):
0.0259 / 0.0185
[9/10][550/3166]        Loss_D: 1.2343  Loss_G: 9.3441  D(x): 0.9874    D(G(z)):
0.5328 / 0.0008
[9/10][600/3166]        Loss_D: 0.1026  Loss_G: 4.8368  D(x): 0.9636    D(G(z)):
0.0565 / 0.0142
```

```
[9/10][650/3166]        Loss_D: 0.1109  Loss_G: 6.3800  D(x): 0.9166    D(G(z)):
0.0109 / 0.0057
[9/10][700/3166]        Loss_D: 0.1103  Loss_G: 4.5003  D(x): 0.9425    D(G(z)):
0.0381 / 0.0263
[9/10][750/3166]        Loss_D: 0.0947  Loss_G: 4.0601  D(x): 0.9826    D(G(z)):
0.0689 / 0.0246
[9/10][800/3166]        Loss_D: 0.0335  Loss_G: 5.9956  D(x): 0.9946    D(G(z)):
0.0266 / 0.0062
[9/10][850/3166]        Loss_D: 0.1025  Loss_G: 6.0728  D(x): 0.9887    D(G(z)):
0.0811 / 0.0038
[9/10][900/3166]        Loss_D: 0.0668  Loss_G: 4.9133  D(x): 0.9685    D(G(z)):
0.0309 / 0.0127
[9/10][950/3166]        Loss_D: 0.2268  Loss_G: 8.7479  D(x): 0.8315    D(G(z)):
0.0004 / 0.0009
[9/10][1000/3166]       Loss_D: 0.1025  Loss_G: 4.2059  D(x): 0.9365    D(G(z)):
0.0166 / 0.0300
[9/10][1050/3166]       Loss_D: 0.6133  Loss_G: 0.6383  D(x): 0.6286    D(G(z)):
0.0029 / 0.6987
[9/10][1100/3166]       Loss_D: 0.0957  Loss_G: 4.7747  D(x): 0.9327    D(G(z)):
0.0101 / 0.0220
[9/10][1150/3166]       Loss_D: 0.2676  Loss_G: 9.2607  D(x): 0.9965    D(G(z)):
0.1905 / 0.0002
[9/10][1200/3166]       Loss_D: 0.0510  Loss_G: 4.9237  D(x): 0.9967    D(G(z)):
0.0445 / 0.0120
[9/10][1250/3166]       Loss_D: 0.0491  Loss_G: 6.2637  D(x): 0.9842    D(G(z)):
0.0303 / 0.0045
[9/10][1300/3166]       Loss_D: 0.0647  Loss_G: 6.4978  D(x): 0.9958    D(G(z)):
0.0560 / 0.0029
[9/10][1350/3166]       Loss_D: 0.1487  Loss_G: 8.0022  D(x): 0.9681    D(G(z)):
0.0857 / 0.0013
[9/10][1400/3166]       Loss_D: 0.0474  Loss_G: 6.5959  D(x): 0.9708    D(G(z)):
0.0161 / 0.0041
[9/10][1450/3166]       Loss_D: 0.0534  Loss_G: 4.0419  D(x): 0.9973    D(G(z)):
0.0460 / 0.0386
[9/10][1500/3166]       Loss_D: 0.3856  Loss_G: 9.6570  D(x): 0.9957    D(G(z)):
0.2526 / 0.0001
[9/10][1550/3166]       Loss_D: 0.1005  Loss_G: 5.5240  D(x): 0.9826    D(G(z)):
0.0687 / 0.0091
[9/10][1600/3166]       Loss_D: 0.0633  Loss_G: 5.6491  D(x): 0.9928    D(G(z)):
0.0456 / 0.0070
[9/10][1650/3166]       Loss_D: 0.1556  Loss_G: 4.2859  D(x): 0.8795    D(G(z)):
0.0054 / 0.0328
[9/10][1700/3166]       Loss_D: 0.3130  Loss_G: 3.9136  D(x): 0.8131    D(G(z)):
0.0459 / 0.0629
[9/10][1750/3166]       Loss_D: 0.0622  Loss_G: 5.7381  D(x): 0.9723    D(G(z)):
0.0297 / 0.0080
[9/10][1800/3166]       Loss_D: 0.1100  Loss_G: 3.5784  D(x): 0.9720    D(G(z)):
0.0641 / 0.0595
```

```
[9/10][1850/3166]       Loss_D: 0.0886  Loss_G: 6.0951  D(x): 0.9851    D(G(z)):
0.0642 / 0.0047
[9/10][1900/3166]       Loss_D: 0.0762  Loss_G: 6.5652  D(x): 0.9883    D(G(z)):
0.0578 / 0.0060
[9/10][1950/3166]       Loss_D: 0.1217  Loss_G: 5.6341  D(x): 0.9366    D(G(z)):
0.0378 / 0.0105
[9/10][2000/3166]       Loss_D: 0.0400  Loss_G: 5.2169  D(x): 0.9903    D(G(z)):
0.0287 / 0.0127
[9/10][2050/3166]       Loss_D: 0.2387  Loss_G: 1.9555  D(x): 0.8273    D(G(z)):
0.0042 / 0.2268
[9/10][2100/3166]       Loss_D: 0.3595  Loss_G: 4.9125  D(x): 0.9010    D(G(z)):
0.1674 / 0.0183
[9/10][2150/3166]       Loss_D: 0.0844  Loss_G: 6.4712  D(x): 0.9437    D(G(z)):
0.0142 / 0.0068
[9/10][2200/3166]       Loss_D: 0.0561  Loss_G: 4.8452  D(x): 0.9831    D(G(z)):
0.0366 / 0.0167
[9/10][2250/3166]       Loss_D: 0.0601  Loss_G: 5.2457  D(x): 0.9566    D(G(z)):
0.0117 / 0.0136
[9/10][2300/3166]       Loss_D: 0.0367  Loss_G: 5.3744  D(x): 0.9900    D(G(z)):
0.0250 / 0.0109
[9/10][2350/3166]       Loss_D: 0.0744  Loss_G: 4.9670  D(x): 0.9630    D(G(z)):
0.0308 / 0.0147
[9/10][2400/3166]       Loss_D: 0.0455  Loss_G: 5.6936  D(x): 0.9916    D(G(z)):
0.0344 / 0.0081
[9/10][2450/3166]       Loss_D: 0.1871  Loss_G: 4.6678  D(x): 0.9679    D(G(z)):
0.1263 / 0.0155
[9/10][2500/3166]       Loss_D: 0.1262  Loss_G: 5.3990  D(x): 0.8986    D(G(z)):
0.0084 / 0.0121
[9/10][2550/3166]       Loss_D: 0.0576  Loss_G: 5.8728  D(x): 0.9614    D(G(z)):
0.0145 / 0.0090
[9/10][2600/3166]       Loss_D: 0.1006  Loss_G: 6.7276  D(x): 0.9960    D(G(z)):
0.0855 / 0.0020
[9/10][2650/3166]       Loss_D: 0.1782  Loss_G: 8.5822  D(x): 0.9845    D(G(z)):
0.1186 / 0.0007
[9/10][2700/3166]       Loss_D: 0.0903  Loss_G: 3.8615  D(x): 0.9315    D(G(z)):
0.0134 / 0.0464
[9/10][2750/3166]       Loss_D: 0.2485  Loss_G: 5.3204  D(x): 0.9815    D(G(z)):
0.1694 / 0.0116
[9/10][2800/3166]       Loss_D: 0.1270  Loss_G: 6.9989  D(x): 0.9127    D(G(z)):
0.0117 / 0.0036
[9/10][2850/3166]       Loss_D: 0.1473  Loss_G: 5.7211  D(x): 0.9611    D(G(z)):
0.0866 / 0.0079
[9/10][2900/3166]       Loss_D: 0.0800  Loss_G: 4.4613  D(x): 0.9724    D(G(z)):
0.0468 / 0.0201
[9/10][2950/3166]       Loss_D: 0.0919  Loss_G: 4.9013  D(x): 0.9352    D(G(z)):
0.0168 / 0.0219
[9/10][3000/3166]       Loss_D: 0.0559  Loss_G: 5.3102  D(x): 0.9578    D(G(z)):
0.0088 / 0.0115
```

```
[9/10][3050/3166]       Loss_D: 0.0426  Loss_G: 5.3165  D(x): 0.9767    D(G(z)):
0.0173 / 0.0108
[9/10][3100/3166]       Loss_D: 0.0479  Loss_G: 5.7214  D(x): 0.9764    D(G(z)):
0.0218 / 0.0081
[9/10][3150/3166]       Loss_D: 0.0713  Loss_G: 5.1483  D(x): 0.9479    D(G(z)):
0.0121 / 0.0186
```