```python
# CELL 1: confirm GPU
import torch
print("torch:", torch.__version__)
print("cuda available:", torch.cuda.is_available())
if torch.cuda.is_available():
    print("device:", torch.cuda.get_device_name(0))
```

```
torch: 2.9.0+cu128
cuda available: True
device: Tesla T4
```

```python
# CELL 2: install dependencies
# You can skip if already satisfied, but safe to run
!pip install --quiet torch torchvision matplotlib
```

```python
# Colab cell (Code)
!pip install --upgrade torch torchvision matplotlib gdown
```

```
Requirement already satisfied: torch in
/usr/local/lib/python3.12/dist-packages (2.9.0)
Requirement already satisfied: torchvision in
/usr/local/lib/python3.12/dist-packages (0.24.0)
Requirement already satisfied: matplotlib in
/usr/local/lib/python3.12/dist-packages (3.10.7)
Requirement already satisfied: gdown in
/usr/local/lib/python3.12/dist-packages (5.2.0)
Requirement already satisfied: filelock in
/usr/local/lib/python3.12/dist-packages (from torch) (3.20.0)
Requirement already satisfied: typing-extensions>=4.10.0 in
/usr/local/lib/python3.12/dist-packages (from torch) (4.15.0)
Requirement already satisfied: setuptools in
/usr/local/lib/python3.12/dist-packages (from torch) (75.2.0)
Requirement already satisfied: sympy>=1.13.3 in
/usr/local/lib/python3.12/dist-packages (from torch) (1.13.3)
Requirement already satisfied: networkx>=2.5.1 in
/usr/local/lib/python3.12/dist-packages (from torch) (3.5)
Requirement already satisfied: jinja2 in
/usr/local/lib/python3.12/dist-packages (from torch) (3.1.6)
Requirement already satisfied: fsspec>=0.8.5 in
/usr/local/lib/python3.12/dist-packages (from torch) (2025.3.0)
Requirement already satisfied: nvidia-cuda-nvrtc-cu12==12.8.93 in
/usr/local/lib/python3.12/dist-packages (from torch) (12.8.93)
Requirement already satisfied: nvidia-cuda-runtime-cu12==12.8.90 in
/usr/local/lib/python3.12/dist-packages (from torch) (12.8.90)
Requirement already satisfied: nvidia-cuda-cupti-cu12==12.8.90 in
/usr/local/lib/python3.12/dist-packages (from torch) (12.8.90)
Requirement already satisfied: nvidia-cudnn-cu12==9.10.2.21 in
/usr/local/lib/python3.12/dist-packages (from torch) (9.10.2.21)
Requirement already satisfied: nvidia-cublas-cu12==12.8.4.1 in
/usr/local/lib/python3.12/dist-packages (from torch) (12.8.4.1)
```

```
Requirement already satisfied: nvidia-cufft-cu12==11.3.3.83 in
/usr/local/lib/python3.12/dist-packages (from torch) (11.3.3.83)
Requirement already satisfied: nvidia-curand-cu12==10.3.9.90 in
/usr/local/lib/python3.12/dist-packages (from torch) (10.3.9.90)
Requirement already satisfied: nvidia-cusolver-cu12==11.7.3.90 in
/usr/local/lib/python3.12/dist-packages (from torch) (11.7.3.90)
Requirement already satisfied: nvidia-cusparse-cu12==12.5.8.93 in
/usr/local/lib/python3.12/dist-packages (from torch) (12.5.8.93)
Requirement already satisfied: nvidia-cusparselt-cu12==0.7.1 in
/usr/local/lib/python3.12/dist-packages (from torch) (0.7.1)
Requirement already satisfied: nvidia-nccl-cu12==2.27.5 in
/usr/local/lib/python3.12/dist-packages (from torch) (2.27.5)
Requirement already satisfied: nvidia-nvshmem-cu12==3.3.20 in
/usr/local/lib/python3.12/dist-packages (from torch) (3.3.20)
Requirement already satisfied: nvidia-nvtx-cu12==12.8.90 in
/usr/local/lib/python3.12/dist-packages (from torch) (12.8.90)
Requirement already satisfied: nvidia-nvjitlink-cu12==12.8.93 in
/usr/local/lib/python3.12/dist-packages (from torch) (12.8.93)
Requirement already satisfied: nvidia-cufile-cu12==1.13.1.3 in
/usr/local/lib/python3.12/dist-packages (from torch) (1.13.1.3)
Requirement already satisfied: triton==3.5.0 in
/usr/local/lib/python3.12/dist-packages (from torch) (3.5.0)
Requirement already satisfied: numpy in
/usr/local/lib/python3.12/dist-packages (from torchvision) (2.0.2)
Requirement already satisfied: pillow!=8.3.*,>=5.3.0 in
/usr/local/lib/python3.12/dist-packages (from torchvision) (11.3.0)
Requirement already satisfied: contourpy>=1.0.1 in
/usr/local/lib/python3.12/dist-packages (from matplotlib) (1.3.3)
Requirement already satisfied: cycler>=0.10 in
/usr/local/lib/python3.12/dist-packages (from matplotlib) (0.12.1)
Requirement already satisfied: fonttools>=4.22.0 in
/usr/local/lib/python3.12/dist-packages (from matplotlib) (4.60.1)
Requirement already satisfied: kiwisolver>=1.3.1 in
/usr/local/lib/python3.12/dist-packages (from matplotlib) (1.4.9)
Requirement already satisfied: packaging>=20.0 in
/usr/local/lib/python3.12/dist-packages (from matplotlib) (25.0)
Requirement already satisfied: pyparsing>=3 in
/usr/local/lib/python3.12/dist-packages (from matplotlib) (3.2.5)
Requirement already satisfied: python-dateutil>=2.7 in
/usr/local/lib/python3.12/dist-packages (from matplotlib)
(2.9.0.post0)
Requirement already satisfied: beautifulsoup4 in
/usr/local/lib/python3.12/dist-packages (from gdown) (4.13.5)
Requirement already satisfied: requests[socks] in
/usr/local/lib/python3.12/dist-packages (from gdown) (2.32.4)
Requirement already satisfied: tqdm in /usr/local/lib/python3.12/dist-
packages (from gdown) (4.67.1)
Requirement already satisfied: six>=1.5 in
/usr/local/lib/python3.12/dist-packages (from python-dateutil>=2.7-
```

```
>matplotlib) (1.17.0)
Requirement already satisfied: mpmath<1.4,>=1.1.0 in
/usr/local/lib/python3.12/dist-packages (from sympy>=1.13.3->torch)
(1.3.0)
Requirement already satisfied: soupsieve>1.2 in
/usr/local/lib/python3.12/dist-packages (from beautifulsoup4->gdown)
(2.8)
Requirement already satisfied: MarkupSafe>=2.0 in
/usr/local/lib/python3.12/dist-packages (from jinja2->torch) (3.0.3)
Requirement already satisfied: charset_normalizer<4,>=2 in
/usr/local/lib/python3.12/dist-packages (from requests[socks]->gdown)
(3.4.4)
Requirement already satisfied: idna<4,>=2.5 in
/usr/local/lib/python3.12/dist-packages (from requests[socks]->gdown)
(3.11)
Requirement already satisfied: urllib3<3,>=1.21.1 in
/usr/local/lib/python3.12/dist-packages (from requests[socks]->gdown)
(2.5.0)
Requirement already satisfied: certifi>=2017.4.17 in
/usr/local/lib/python3.12/dist-packages (from requests[socks]->gdown)
(2025.10.5)
Requirement already satisfied: PySocks!=1.5.7,>=1.5.6 in
/usr/local/lib/python3.12/dist-packages (from requests[socks]->gdown)
(1.7.1)
```

```python
import torch
import torch.nn as nn
import torch.optim as optim
import torchvision
import torchvision.datasets as dset
import torchvision.transforms as transforms
import torchvision.utils as vutils
import matplotlib.pyplot as plt
import numpy as np

# Root directory for dataset
dataroot = "./data"

# Number of workers for dataloader
workers = 2

# Batch size
batch_size = 128

# Image size (64x64 color images)
image_size = 64

# Number of channels (RGB = 3)
nc = 3
```

```python
# Size of latent vector (z)
nz = 100

# Size of feature maps in generator
ngf = 64

# Size of feature maps in discriminator
ndf = 64

# Number of training epochs
num_epochs = 25

# Learning rate
lr = 0.0001

# Beta1 hyperparam for Adam optimizer
beta1 = 0.5

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

!mkdir -p ./data
!wget -c
https://www.dropbox.com/s/8iq1zqg4x9jqk2v/img_align_celeba.zip?dl=1 -O
./data/img_align_celeba.zip
!unzip -qq ./data/img_align_celeba.zip -d ./data/celeba
```

```
--2025-10-26 15:53:59--
https://www.dropbox.com/s/8iq1zqg4x9jqk2v/img_align_celeba.zip?dl=1
Resolving www.dropbox.com (www.dropbox.com)... 162.125.5.18,
2620:100:601d:18::a27d:512
Connecting to www.dropbox.com (www.dropbox.com)|162.125.5.18|:443...
connected.
HTTP request sent, awaiting response... 200 OK
Length: unspecified [text/html]
Saving to: ‘./data/img_align_celeba.zip’

           ./data/im     [<=>                 ]       0  --.-KB/s
./data/img_align_ce     [ <=>                ]  86.44K  --.-KB/s    in
0.05s

2025-10-26 15:53:59 (1.74 MB/s) - ‘./data/img_align_celeba.zip’ saved
[88510]

[./data/img_align_celeba.zip]
  End-of-central-directory signature not found.  Either this file is
not
  a zipfile, or it constitutes one disk of a multi-part archive.  In
the
  latter case the central directory and zipfile comment will be found
on
```

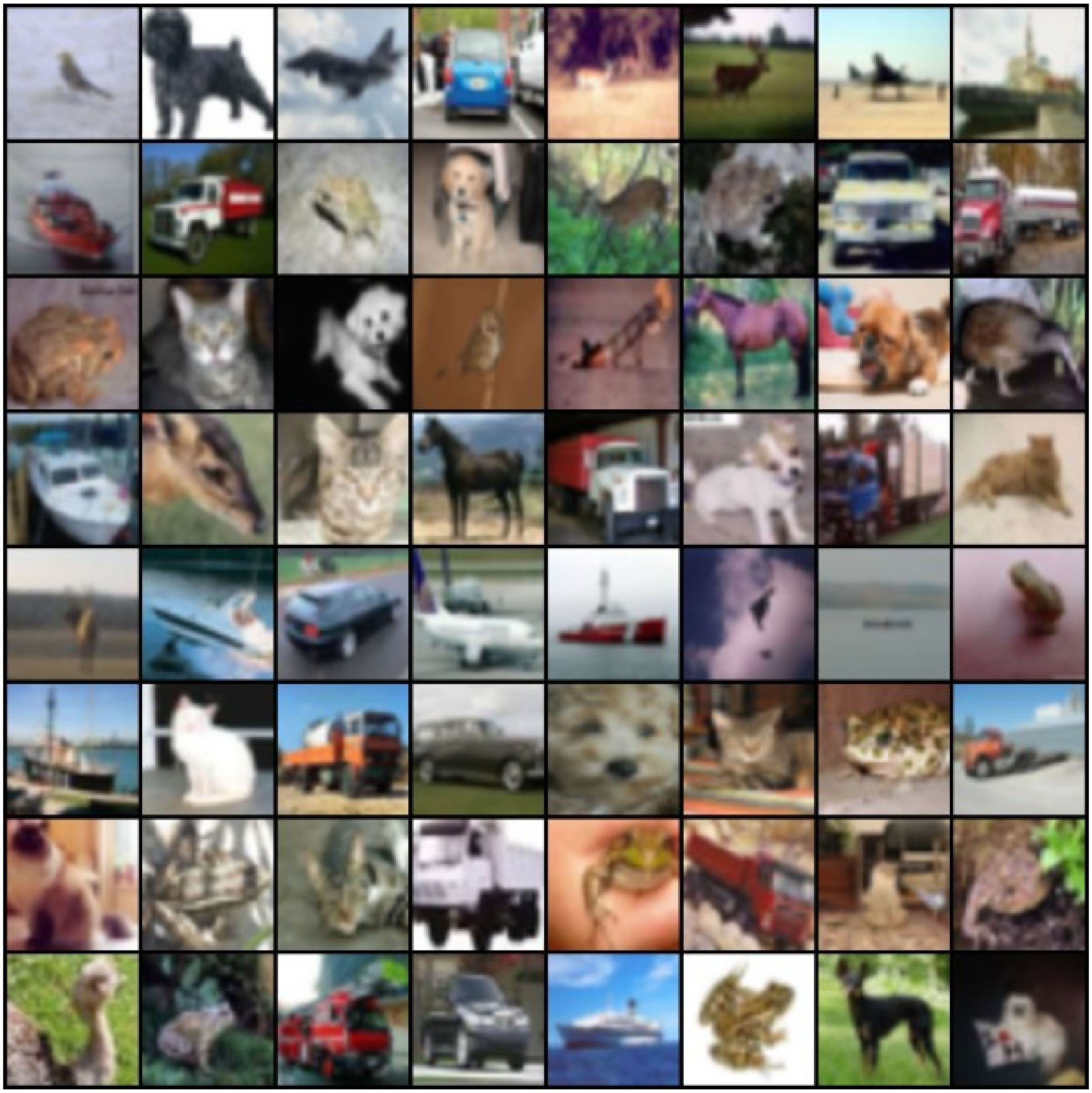
```
  the last disk(s) of this archive.
unzip:  cannot find zipfile directory in one of
./data/img_align_celeba.zip or
        ./data/img_align_celeba.zip.zip, and cannot find
./data/img_align_celeba.zip.ZIP, period.

# □ Use CIFAR-10 instead of CelebA (works immediately)
dataset = dset.CIFAR10(root='./data', download=True,
                       transform=transforms.Compose([
                           transforms.Resize(image_size),
                           transforms.CenterCrop(image_size),
                           transforms.ToTensor(),
                           transforms.Normalize((0.5, 0.5, 0.5),
                                                (0.5, 0.5, 0.5))
                       ]))

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

# Show sample training images
real_batch = next(iter(dataloader))
plt.figure(figsize=(8,8))
plt.axis("off")
plt.title("Training Images (CIFAR-10)")
plt.imshow(np.transpose(vutils.make_grid(real_batch[0][:64],
                                         padding=2,
normalize=True).cpu(), (1,2,0)))
plt.show()
```

Training Images (CIFAR-10)



```python
class Generator(nn.Module):
    def __init__(self):
        super(Generator, self).__init__()
        self.main = nn.Sequential(
            # Input Z latent vector
            nn.ConvTranspose2d(nz, ngf * 8, 4, 1, 0, bias=False),
            nn.BatchNorm2d(ngf * 8),
            nn.ReLU(True),
            # (ngf*8) x 4 x 4
            nn.ConvTranspose2d(ngf * 8, ngf * 4, 4, 2, 1, bias=False),
            nn.BatchNorm2d(ngf * 4),
```

```
            nn.ReLU(True),
            # (ngf*4) x 8 x 8
            nn.ConvTranspose2d(ngf * 4, ngf * 2, 4, 2, 1, bias=False),
            nn.BatchNorm2d(ngf * 2),
            nn.ReLU(True),
            # (ngf*2) x 16 x 16
            nn.ConvTranspose2d(ngf * 2, ngf, 4, 2, 1, bias=False),
            nn.BatchNorm2d(ngf),
            nn.ReLU(True),
            # (ngf) x 32 x 32
            nn.ConvTranspose2d(ngf, nc, 4, 2, 1, bias=False),
            nn.Tanh()
            # (nc) x 64 x 64
        )

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

class Discriminator(nn.Module):
    def __init__(self):
        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, input):
        return self.main(input)

# Create the generator
netG = Generator().to(device)
# Create the discriminator
netD = Discriminator().to(device)

# Initialize weights
def weights_init(m):
    classname = m.__class__.__name__
    if classname.find('Conv') != -1:
        nn.init.normal_(m.weight.data, 0.0, 0.02)
```

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

netG.apply(weights_init)
netD.apply(weights_init)

# Loss function
criterion = nn.BCELoss()

# Noise for generating images
fixed_noise = torch.randn(64, nz, 1, 1, device=device)

# Labels
real_label = 1.
fake_label = 0.

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

img_list = []
G_losses = []
D_losses = []

print("Starting Training Loop...")
for epoch in range(num_epochs):
    for i, data in enumerate(dataloader, 0):
        ############################
        # (1) Update D network
        ###########################
        netD.zero_grad()
        real_cpu = data[0].to(device)
        b_size = real_cpu.size(0)
        label = torch.full((b_size,), real_label, dtype=torch.float,
device=device)
        output = netD(real_cpu).view(-1)
        errD_real = criterion(output, label)
        errD_real.backward()
        D_x = output.mean().item()

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

```
        errD = errD_real + errD_fake
        optimizerD.step()

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

        if i % 50 == 0:
            print(f"[{epoch}/{num_epochs}][{i}/{len(dataloader)}] "
                  f"Loss_D: {errD.item():.4f} Loss_G:
{errG.item():.4f} "
                  f"D(x): {D_x:.4f} D(G(z)):
{D_G_z1:.4f}/{D_G_z2:.4f}")

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

    with torch.no_grad():
        fake = netG(fixed_noise).detach().cpu()
    img_list.append(vutils.make_grid(fake, padding=2, normalize=True))
```

```
Starting Training Loop...
[0/25][0/391] Loss_D: 1.6995 Loss_G: 3.1490 D(x): 0.5540 D(G(z)):
0.5702/0.0652
[0/25][50/391] Loss_D: 0.1312 Loss_G: 8.3585 D(x): 0.9611 D(G(z)):
0.0781/0.0003
[0/25][100/391] Loss_D: 0.0468 Loss_G: 8.3772 D(x): 0.9770 D(G(z)):
0.0202/0.0003
[0/25][150/391] Loss_D: 0.0240 Loss_G: 8.6738 D(x): 0.9939 D(G(z)):
0.0176/0.0002
[0/25][200/391] Loss_D: 0.0119 Loss_G: 8.4029 D(x): 0.9963 D(G(z)):
0.0081/0.0002
[0/25][250/391] Loss_D: 0.0038 Loss_G: 8.5369 D(x): 0.9976 D(G(z)):
0.0013/0.0002
[0/25][300/391] Loss_D: 0.0073 Loss_G: 8.0444 D(x): 0.9979 D(G(z)):
0.0052/0.0003
[0/25][350/391] Loss_D: 0.0056 Loss_G: 7.9478 D(x): 0.9988 D(G(z)):
0.0044/0.0004
[1/25][0/391] Loss_D: 0.0069 Loss_G: 7.8820 D(x): 0.9976 D(G(z)):
0.0045/0.0004
[1/25][50/391] Loss_D: 0.0041 Loss_G: 8.1314 D(x): 0.9977 D(G(z)):
0.0017/0.0003
[1/25][100/391] Loss_D: 0.0149 Loss_G: 5.6273 D(x): 0.9942 D(G(z)):
```

```
0.0089/0.0039
[1/25][150/391] Loss_D: 0.1171 Loss_G: 6.3746 D(x): 0.9473 D(G(z)):
0.0481/0.0025
[1/25][200/391] Loss_D: 0.1732 Loss_G: 3.4256 D(x): 0.8952 D(G(z)):
0.0127/0.0513
[1/25][250/391] Loss_D: 0.0778 Loss_G: 6.3012 D(x): 0.9605 D(G(z)):
0.0300/0.0030
[1/25][300/391] Loss_D: 0.0863 Loss_G: 9.6991 D(x): 0.9885 D(G(z)):
0.0611/0.0001
[1/25][350/391] Loss_D: 0.0107 Loss_G: 8.3244 D(x): 0.9959 D(G(z)):
0.0063/0.0004
[2/25][0/391] Loss_D: 0.3191 Loss_G: 20.9490 D(x): 0.9891 D(G(z)):
0.2438/0.0000
[2/25][50/391] Loss_D: 0.0843 Loss_G: 6.4155 D(x): 0.9564 D(G(z)):
0.0191/0.0024
[2/25][100/391] Loss_D: 0.0688 Loss_G: 8.5886 D(x): 0.9824 D(G(z)):
0.0448/0.0002
[2/25][150/391] Loss_D: 0.1334 Loss_G: 6.0346 D(x): 0.9220 D(G(z)):
0.0073/0.0047
[2/25][200/391] Loss_D: 0.2533 Loss_G: 11.4549 D(x): 0.9794 D(G(z)):
0.1883/0.0000
[2/25][250/391] Loss_D: 0.1745 Loss_G: 5.0040 D(x): 0.8983 D(G(z)):
0.0428/0.0137
[2/25][300/391] Loss_D: 0.1124 Loss_G: 5.4058 D(x): 0.9369 D(G(z)):
0.0367/0.0062
[2/25][350/391] Loss_D: 0.2958 Loss_G: 6.3833 D(x): 0.9762 D(G(z)):
0.1685/0.0053
[3/25][0/391] Loss_D: 0.0782 Loss_G: 4.9544 D(x): 0.9502 D(G(z)):
0.0230/0.0107
[3/25][50/391] Loss_D: 0.1703 Loss_G: 5.8089 D(x): 0.8986 D(G(z)):
0.0050/0.0063
[3/25][100/391] Loss_D: 0.1080 Loss_G: 5.7404 D(x): 0.9636 D(G(z)):
0.0597/0.0053
[3/25][150/391] Loss_D: 0.1080 Loss_G: 8.1906 D(x): 0.9955 D(G(z)):
0.0907/0.0006
[3/25][200/391] Loss_D: 0.1377 Loss_G: 4.6160 D(x): 0.9258 D(G(z)):
0.0488/0.0184
[3/25][250/391] Loss_D: 0.1135 Loss_G: 4.2691 D(x): 0.9509 D(G(z)):
0.0575/0.0213
[3/25][300/391] Loss_D: 0.0667 Loss_G: 6.0128 D(x): 0.9890 D(G(z)):
0.0512/0.0038
[3/25][350/391] Loss_D: 0.0684 Loss_G: 5.9044 D(x): 0.9757 D(G(z)):
0.0397/0.0046
[4/25][0/391] Loss_D: 1.1929 Loss_G: 10.3808 D(x): 0.9911 D(G(z)):
0.5680/0.0001
[4/25][50/391] Loss_D: 0.4286 Loss_G: 3.5904 D(x): 0.7614 D(G(z)):
0.0398/0.0436
[4/25][100/391] Loss_D: 0.1730 Loss_G: 5.2042 D(x): 0.9631 D(G(z)):
0.1118/0.0092
```

```
[4/25][150/391] Loss_D: 0.4193 Loss_G: 4.1030 D(x): 0.8180 D(G(z)):
0.1328/0.0329
[4/25][200/391] Loss_D: 0.1140 Loss_G: 4.1986 D(x): 0.9492 D(G(z)):
0.0521/0.0224
[4/25][250/391] Loss_D: 0.2797 Loss_G: 4.4887 D(x): 0.8758 D(G(z)):
0.1107/0.0184
[4/25][300/391] Loss_D: 0.8116 Loss_G: 2.0735 D(x): 0.6824 D(G(z)):
0.1908/0.2067
[4/25][350/391] Loss_D: 0.1329 Loss_G: 5.8495 D(x): 0.9437 D(G(z)):
0.0612/0.0067
[5/25][0/391] Loss_D: 0.3923 Loss_G: 3.5737 D(x): 0.7269 D(G(z)):
0.0076/0.0393
[5/25][50/391] Loss_D: 0.0678 Loss_G: 3.2096 D(x): 0.9824 D(G(z)):
0.0469/0.0692
[5/25][100/391] Loss_D: 0.5854 Loss_G: 1.0366 D(x): 0.6467 D(G(z)):
0.0370/0.4327
[5/25][150/391] Loss_D: 0.1105 Loss_G: 4.2320 D(x): 0.9329 D(G(z)):
0.0330/0.0252
[5/25][200/391] Loss_D: 0.5049 Loss_G: 4.9501 D(x): 0.8834 D(G(z)):
0.2590/0.0134
[5/25][250/391] Loss_D: 0.0838 Loss_G: 4.3595 D(x): 0.9536 D(G(z)):
0.0336/0.0192
[5/25][300/391] Loss_D: 0.3643 Loss_G: 7.7349 D(x): 0.9931 D(G(z)):
0.2463/0.0017
[5/25][350/391] Loss_D: 0.0918 Loss_G: 4.5764 D(x): 0.9639 D(G(z)):
0.0495/0.0166
[6/25][0/391] Loss_D: 0.1746 Loss_G: 3.2673 D(x): 0.8728 D(G(z)):
0.0284/0.0638
[6/25][50/391] Loss_D: 0.1681 Loss_G: 4.2433 D(x): 0.9022 D(G(z)):
0.0530/0.0233
[6/25][100/391] Loss_D: 0.7003 Loss_G: 2.4243 D(x): 0.5804 D(G(z)):
0.0015/0.1519
[6/25][150/391] Loss_D: 0.2448 Loss_G: 3.0620 D(x): 0.8964 D(G(z)):
0.1069/0.0692
[6/25][200/391] Loss_D: 1.1748 Loss_G: 2.1320 D(x): 0.4446 D(G(z)):
0.0025/0.1858
[6/25][250/391] Loss_D: 0.2240 Loss_G: 3.7028 D(x): 0.9063 D(G(z)):
0.1013/0.0395
[6/25][300/391] Loss_D: 0.3222 Loss_G: 3.1065 D(x): 0.8392 D(G(z)):
0.1075/0.0661
[6/25][350/391] Loss_D: 1.2455 Loss_G: 8.9119 D(x): 0.9678 D(G(z)):
0.5950/0.0004
[7/25][0/391] Loss_D: 0.1887 Loss_G: 4.9018 D(x): 0.9717 D(G(z)):
0.1374/0.0123
[7/25][50/391] Loss_D: 0.2250 Loss_G: 3.8597 D(x): 0.9013 D(G(z)):
0.1002/0.0325
[7/25][100/391] Loss_D: 0.2342 Loss_G: 3.3527 D(x): 0.8686 D(G(z)):
0.0733/0.0529
[7/25][150/391] Loss_D: 0.9015 Loss_G: 2.8168 D(x): 0.7542 D(G(z)):
```

```
0.3721/0.1021
[7/25][200/391] Loss_D: 0.2523 Loss_G: 4.3030 D(x): 0.9514 D(G(z)):
0.1686/0.0189
[7/25][250/391] Loss_D: 0.2060 Loss_G: 4.7691 D(x): 0.9529 D(G(z)):
0.1359/0.0121
[7/25][300/391] Loss_D: 0.1846 Loss_G: 4.2250 D(x): 0.8558 D(G(z)):
0.0154/0.0236
[7/25][350/391] Loss_D: 2.8398 Loss_G: 2.2214 D(x): 0.2175 D(G(z)):
0.0230/0.3110
[8/25][0/391] Loss_D: 0.2963 Loss_G: 4.2419 D(x): 0.9464 D(G(z)):
0.1905/0.0231
[8/25][50/391] Loss_D: 0.2485 Loss_G: 3.5950 D(x): 0.9612 D(G(z)):
0.1716/0.0391
[8/25][100/391] Loss_D: 0.2147 Loss_G: 3.9419 D(x): 0.9097 D(G(z)):
0.0986/0.0314
[8/25][150/391] Loss_D: 0.3683 Loss_G: 3.7753 D(x): 0.9209 D(G(z)):
0.2128/0.0349
[8/25][200/391] Loss_D: 0.2535 Loss_G: 3.4017 D(x): 0.9274 D(G(z)):
0.1516/0.0481
[8/25][250/391] Loss_D: 0.2017 Loss_G: 3.8699 D(x): 0.8771 D(G(z)):
0.0510/0.0296
[8/25][300/391] Loss_D: 0.2775 Loss_G: 3.1548 D(x): 0.8226 D(G(z)):
0.0535/0.0679
[8/25][350/391] Loss_D: 0.3068 Loss_G: 4.2829 D(x): 0.9218 D(G(z)):
0.1834/0.0212
[9/25][0/391] Loss_D: 0.1945 Loss_G: 3.0187 D(x): 0.9156 D(G(z)):
0.0913/0.0697
[9/25][50/391] Loss_D: 0.7263 Loss_G: 1.9701 D(x): 0.5890 D(G(z)):
0.0772/0.1848
[9/25][100/391] Loss_D: 0.1486 Loss_G: 3.9156 D(x): 0.9173 D(G(z)):
0.0542/0.0335
[9/25][150/391] Loss_D: 0.3813 Loss_G: 6.0692 D(x): 0.9400 D(G(z)):
0.2491/0.0043
[9/25][200/391] Loss_D: 0.5548 Loss_G: 4.8223 D(x): 0.8992 D(G(z)):
0.3225/0.0143
[9/25][250/391] Loss_D: 0.1584 Loss_G: 3.2223 D(x): 0.9260 D(G(z)):
0.0735/0.0550
[9/25][300/391] Loss_D: 1.2624 Loss_G: 0.1766 D(x): 0.3812 D(G(z)):
0.0323/0.8584
[9/25][350/391] Loss_D: 0.4237 Loss_G: 2.2743 D(x): 0.7181 D(G(z)):
0.0425/0.1365
[10/25][0/391] Loss_D: 1.7963 Loss_G: 1.3993 D(x): 0.2618 D(G(z)):
0.0030/0.3303
[10/25][50/391] Loss_D: 0.8590 Loss_G: 6.5633 D(x): 0.9291 D(G(z)):
0.4865/0.0026
[10/25][100/391] Loss_D: 0.3513 Loss_G: 4.1012 D(x): 0.8901 D(G(z)):
0.1873/0.0246
[10/25][150/391] Loss_D: 0.5087 Loss_G: 3.5556 D(x): 0.9760 D(G(z)):
0.3221/0.0450
```

```
[10/25][200/391] Loss_D: 0.3261 Loss_G: 2.7832 D(x): 0.8684 D(G(z)):
0.1436/0.0854
[10/25][250/391] Loss_D: 1.3850 Loss_G: 0.0550 D(x): 0.3335 D(G(z)):
0.0070/0.9484
[10/25][300/391] Loss_D: 0.3788 Loss_G: 3.9000 D(x): 0.9291 D(G(z)):
0.2410/0.0264
[10/25][350/391] Loss_D: 0.2914 Loss_G: 3.6505 D(x): 0.9539 D(G(z)):
0.2013/0.0366
[11/25][0/391] Loss_D: 0.4845 Loss_G: 4.6684 D(x): 0.9367 D(G(z)):
0.3127/0.0150
[11/25][50/391] Loss_D: 0.1509 Loss_G: 4.0109 D(x): 0.9664 D(G(z)):
0.1038/0.0290
[11/25][100/391] Loss_D: 0.2649 Loss_G: 4.7008 D(x): 0.9636 D(G(z)):
0.1870/0.0143
[11/25][150/391] Loss_D: 0.4664 Loss_G: 6.2561 D(x): 0.9735 D(G(z)):
0.3165/0.0029
[11/25][200/391] Loss_D: 0.3120 Loss_G: 3.1057 D(x): 0.8564 D(G(z)):
0.1290/0.0639
[11/25][250/391] Loss_D: 0.2105 Loss_G: 3.1205 D(x): 0.9116 D(G(z)):
0.1002/0.0646
[11/25][300/391] Loss_D: 0.2467 Loss_G: 2.8824 D(x): 0.8741 D(G(z)):
0.0923/0.0769
[11/25][350/391] Loss_D: 0.1702 Loss_G: 3.8355 D(x): 0.9326 D(G(z)):
0.0876/0.0318
[12/25][0/391] Loss_D: 0.3843 Loss_G: 3.8459 D(x): 0.8978 D(G(z)):
0.2185/0.0326
[12/25][50/391] Loss_D: 0.2032 Loss_G: 3.6179 D(x): 0.9385 D(G(z)):
0.1209/0.0385
[12/25][100/391] Loss_D: 0.2431 Loss_G: 3.4220 D(x): 0.8944 D(G(z)):
0.1133/0.0468
[12/25][150/391] Loss_D: 0.1889 Loss_G: 3.3942 D(x): 0.9288 D(G(z)):
0.1018/0.0439
[12/25][200/391] Loss_D: 0.2670 Loss_G: 5.0052 D(x): 0.9579 D(G(z)):
0.1852/0.0094
[12/25][250/391] Loss_D: 1.2594 Loss_G: 1.4995 D(x): 0.5915 D(G(z)):
0.3846/0.2857
[12/25][300/391] Loss_D: 0.8007 Loss_G: 1.2191 D(x): 0.5554 D(G(z)):
0.0522/0.3834
[12/25][350/391] Loss_D: 0.3379 Loss_G: 2.8372 D(x): 0.8672 D(G(z)):
0.1597/0.0787
[13/25][0/391] Loss_D: 0.4118 Loss_G: 3.4969 D(x): 0.7857 D(G(z)):
0.1198/0.0540
[13/25][50/391] Loss_D: 0.2123 Loss_G: 3.7122 D(x): 0.8480 D(G(z)):
0.0373/0.0352
[13/25][100/391] Loss_D: 0.1086 Loss_G: 3.4445 D(x): 0.9611 D(G(z)):
0.0643/0.0426
[13/25][150/391] Loss_D: 0.0888 Loss_G: 4.1466 D(x): 0.9771 D(G(z)):
0.0607/0.0238
[13/25][200/391] Loss_D: 0.1308 Loss_G: 4.1258 D(x): 0.9313 D(G(z)):
```

```
0.0545/0.0242
[13/25][250/391] Loss_D: 0.3727 Loss_G: 2.7534 D(x): 0.7271 D(G(z)):
0.0164/0.1018
[13/25][300/391] Loss_D: 0.2510 Loss_G: 2.7527 D(x): 0.8825 D(G(z)):
0.1041/0.0823
[13/25][350/391] Loss_D: 0.2273 Loss_G: 3.6269 D(x): 0.9390 D(G(z)):
0.1406/0.0361
[14/25][0/391] Loss_D: 0.1290 Loss_G: 3.7982 D(x): 0.9159 D(G(z)):
0.0364/0.0346
[14/25][50/391] Loss_D: 0.1366 Loss_G: 3.8706 D(x): 0.9155 D(G(z)):
0.0421/0.0278
[14/25][100/391] Loss_D: 0.1076 Loss_G: 5.3778 D(x): 0.9114 D(G(z)):
0.0085/0.0073
[14/25][150/391] Loss_D: 0.1653 Loss_G: 4.6264 D(x): 0.9525 D(G(z)):
0.1035/0.0162
[14/25][200/391] Loss_D: 0.1607 Loss_G: 3.7224 D(x): 0.9495 D(G(z)):
0.0982/0.0336
[14/25][250/391] Loss_D: 0.8654 Loss_G: 2.6314 D(x): 0.8766 D(G(z)):
0.4479/0.1058
[14/25][300/391] Loss_D: 1.6326 Loss_G: 0.6062 D(x): 0.2791 D(G(z)):
0.0727/0.5963
[14/25][350/391] Loss_D: 0.3619 Loss_G: 4.7480 D(x): 0.9661 D(G(z)):
0.2589/0.0124
[15/25][0/391] Loss_D: 0.2653 Loss_G: 3.8635 D(x): 0.9132 D(G(z)):
0.1475/0.0310
[15/25][50/391] Loss_D: 0.1219 Loss_G: 3.8145 D(x): 0.9462 D(G(z)):
0.0615/0.0311
[15/25][100/391] Loss_D: 0.2631 Loss_G: 4.5027 D(x): 0.9904 D(G(z)):
0.2056/0.0163
[15/25][150/391] Loss_D: 0.1059 Loss_G: 4.1397 D(x): 0.9742 D(G(z)):
0.0748/0.0212
[15/25][200/391] Loss_D: 0.0422 Loss_G: 5.0042 D(x): 0.9754 D(G(z)):
0.0167/0.0099
[15/25][250/391] Loss_D: 2.4384 Loss_G: 4.9549 D(x): 0.9467 D(G(z)):
0.8636/0.0179
[15/25][300/391] Loss_D: 0.1755 Loss_G: 4.1028 D(x): 0.9510 D(G(z)):
0.1074/0.0248
[15/25][350/391] Loss_D: 0.5474 Loss_G: 1.8550 D(x): 0.6600 D(G(z)):
0.0666/0.1998
[16/25][0/391] Loss_D: 0.2624 Loss_G: 4.3307 D(x): 0.9553 D(G(z)):
0.1827/0.0194
[16/25][50/391] Loss_D: 0.2039 Loss_G: 3.5183 D(x): 0.8522 D(G(z)):
0.0342/0.0448
[16/25][100/391] Loss_D: 0.1169 Loss_G: 3.3994 D(x): 0.9361 D(G(z)):
0.0462/0.0437
[16/25][150/391] Loss_D: 0.1150 Loss_G: 4.2710 D(x): 0.9882 D(G(z)):
0.0950/0.0194
[16/25][200/391] Loss_D: 0.2623 Loss_G: 4.7425 D(x): 0.9458 D(G(z)):
0.1671/0.0154
```

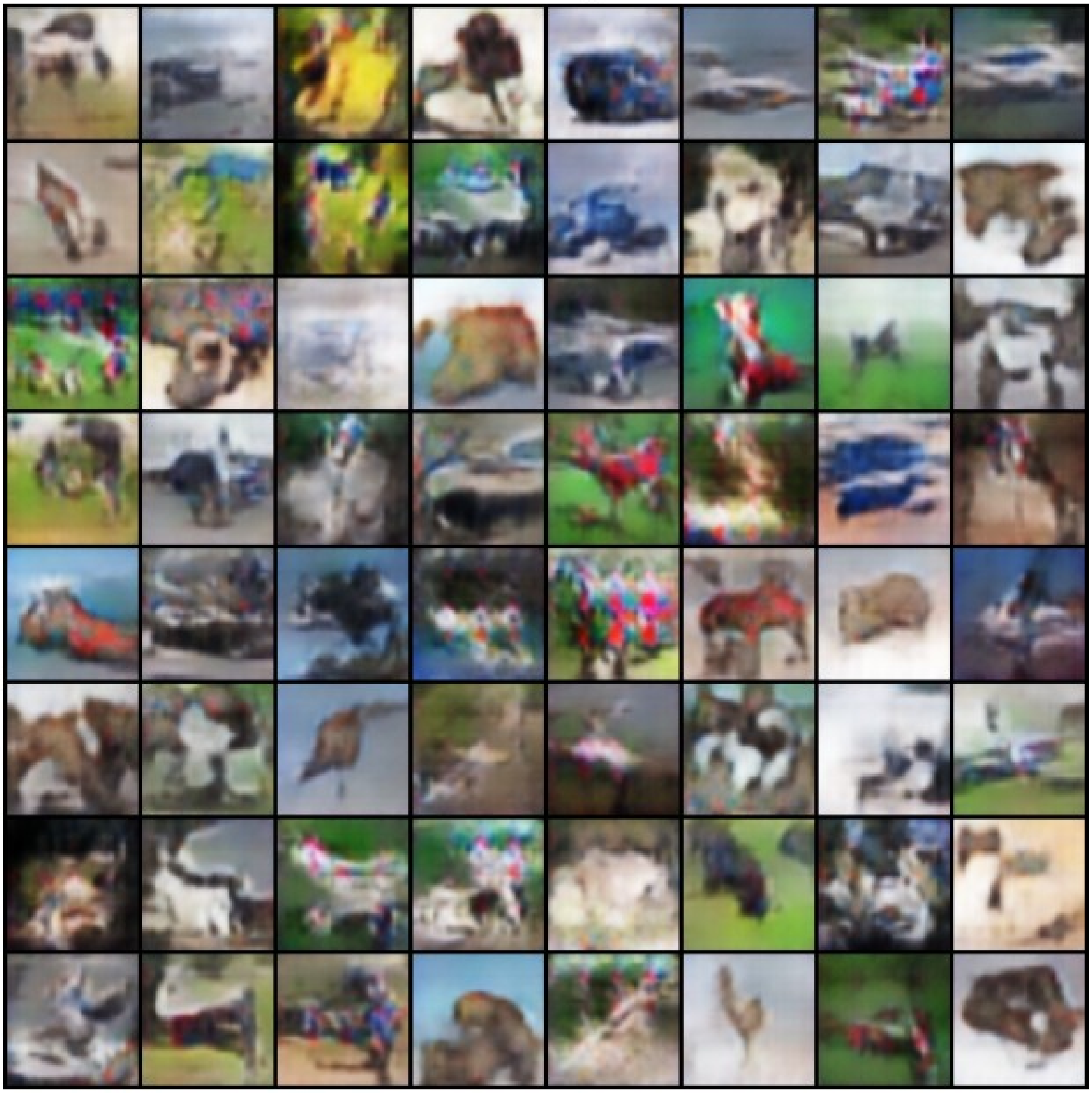
```
[16/25][250/391] Loss_D: 0.0445 Loss_G: 4.4541 D(x): 0.9887 D(G(z)): 0.0310/0.0218
[16/25][300/391] Loss_D: 0.9215 Loss_G: 1.4098 D(x): 0.5029 D(G(z)): 0.0816/0.3155
[16/25][350/391] Loss_D: 0.6643 Loss_G: 3.5190 D(x): 0.5867 D(G(z)): 0.0159/0.0501
[17/25][0/391] Loss_D: 1.1184 Loss_G: 9.3496 D(x): 0.9955 D(G(z)): 0.5975/0.0002
[17/25][50/391] Loss_D: 0.0813 Loss_G: 4.3990 D(x): 0.9532 D(G(z)): 0.0311/0.0192
[17/25][100/391] Loss_D: 0.1311 Loss_G: 4.3251 D(x): 0.9761 D(G(z)): 0.0947/0.0211
[17/25][150/391] Loss_D: 0.0434 Loss_G: 4.2660 D(x): 0.9800 D(G(z)): 0.0224/0.0200
[17/25][200/391] Loss_D: 0.0749 Loss_G: 4.3653 D(x): 0.9835 D(G(z)): 0.0553/0.0173
[17/25][250/391] Loss_D: 6.9083 Loss_G: 1.2525 D(x): 0.0027 D(G(z)): 0.0000/0.3947
[17/25][300/391] Loss_D: 0.8164 Loss_G: 1.8789 D(x): 0.5453 D(G(z)): 0.0696/0.2347
[17/25][350/391] Loss_D: 0.2270 Loss_G: 3.3939 D(x): 0.8940 D(G(z)): 0.0974/0.0481
[18/25][0/391] Loss_D: 0.9260 Loss_G: 5.5013 D(x): 0.9356 D(G(z)): 0.5354/0.0062
[18/25][50/391] Loss_D: 0.0909 Loss_G: 3.7871 D(x): 0.9632 D(G(z)): 0.0505/0.0295
[18/25][100/391] Loss_D: 0.0495 Loss_G: 4.5994 D(x): 0.9774 D(G(z)): 0.0258/0.0141
[18/25][150/391] Loss_D: 0.1145 Loss_G: 5.0294 D(x): 0.9850 D(G(z)): 0.0914/0.0094
[18/25][200/391] Loss_D: 0.3034 Loss_G: 2.8363 D(x): 0.9951 D(G(z)): 0.2332/0.0840
[18/25][250/391] Loss_D: 0.2073 Loss_G: 4.2906 D(x): 0.9565 D(G(z)): 0.1428/0.0196
[18/25][300/391] Loss_D: 0.0962 Loss_G: 3.3561 D(x): 0.9480 D(G(z)): 0.0400/0.0519
[18/25][350/391] Loss_D: 0.8917 Loss_G: 2.3016 D(x): 0.7495 D(G(z)): 0.3905/0.1348
[19/25][0/391] Loss_D: 0.1452 Loss_G: 4.0228 D(x): 0.9409 D(G(z)): 0.0759/0.0284
[19/25][50/391] Loss_D: 0.0510 Loss_G: 4.5671 D(x): 0.9865 D(G(z)): 0.0357/0.0165
[19/25][100/391] Loss_D: 0.0501 Loss_G: 4.2670 D(x): 0.9797 D(G(z)): 0.0285/0.0213
[19/25][150/391] Loss_D: 0.1045 Loss_G: 4.3078 D(x): 0.9659 D(G(z)): 0.0639/0.0204
[19/25][200/391] Loss_D: 0.0589 Loss_G: 4.5762 D(x): 0.9732 D(G(z)): 0.0307/0.0150
[19/25][250/391] Loss_D: 0.0441 Loss_G: 4.3697 D(x): 0.9811 D(G(z)):
```

```
0.0242/0.0171
[19/25][300/391] Loss_D: 0.1092 Loss_G: 4.0528 D(x): 0.9577 D(G(z)):
0.0595/0.0250
[19/25][350/391] Loss_D: 0.1081 Loss_G: 3.6139 D(x): 0.9438 D(G(z)):
0.0456/0.0395
[20/25][0/391] Loss_D: 0.0691 Loss_G: 5.1977 D(x): 0.9877 D(G(z)):
0.0532/0.0094
[20/25][50/391] Loss_D: 0.5547 Loss_G: 2.1792 D(x): 0.6452 D(G(z)):
0.0482/0.1551
[20/25][100/391] Loss_D: 0.1754 Loss_G: 2.7115 D(x): 0.8711 D(G(z)):
0.0207/0.0962
[20/25][150/391] Loss_D: 0.0511 Loss_G: 4.9766 D(x): 0.9597 D(G(z)):
0.0091/0.0101
[20/25][200/391] Loss_D: 0.6795 Loss_G: 2.2007 D(x): 0.7876 D(G(z)):
0.3219/0.1415
[20/25][250/391] Loss_D: 1.8940 Loss_G: 9.2496 D(x): 0.9985 D(G(z)):
0.7765/0.0002
[20/25][300/391] Loss_D: 0.0660 Loss_G: 4.0386 D(x): 0.9847 D(G(z)):
0.0477/0.0293
[20/25][350/391] Loss_D: 0.5060 Loss_G: 4.9167 D(x): 0.9817 D(G(z)):
0.3343/0.0127
[21/25][0/391] Loss_D: 0.1278 Loss_G: 4.3051 D(x): 0.9804 D(G(z)):
0.0968/0.0202
[21/25][50/391] Loss_D: 0.0901 Loss_G: 4.4530 D(x): 0.9697 D(G(z)):
0.0559/0.0166
[21/25][100/391] Loss_D: 0.0466 Loss_G: 4.8611 D(x): 0.9887 D(G(z)):
0.0338/0.0123
[21/25][150/391] Loss_D: 0.0843 Loss_G: 5.7204 D(x): 0.9918 D(G(z)):
0.0717/0.0049
[21/25][200/391] Loss_D: 0.6620 Loss_G: 2.2757 D(x): 0.7706 D(G(z)):
0.2745/0.1574
[21/25][250/391] Loss_D: 0.0866 Loss_G: 4.6316 D(x): 0.9788 D(G(z)):
0.0588/0.0161
[21/25][300/391] Loss_D: 0.7829 Loss_G: 1.8976 D(x): 0.6360 D(G(z)):
0.2080/0.1864
[21/25][350/391] Loss_D: 0.9674 Loss_G: 0.5611 D(x): 0.4553 D(G(z)):
0.0125/0.6187
[22/25][0/391] Loss_D: 0.1874 Loss_G: 2.6702 D(x): 0.8725 D(G(z)):
0.0389/0.0955
[22/25][50/391] Loss_D: 0.0913 Loss_G: 4.8661 D(x): 0.9251 D(G(z)):
0.0100/0.0114
[22/25][100/391] Loss_D: 1.3047 Loss_G: 3.1944 D(x): 0.9584 D(G(z)):
0.6225/0.0588
[22/25][150/391] Loss_D: 0.1017 Loss_G: 3.4309 D(x): 0.9441 D(G(z)):
0.0400/0.0491
[22/25][200/391] Loss_D: 0.0588 Loss_G: 5.0003 D(x): 0.9536 D(G(z)):
0.0101/0.0105
[22/25][250/391] Loss_D: 0.6797 Loss_G: 2.8983 D(x): 0.8462 D(G(z)):
0.3572/0.0744
```

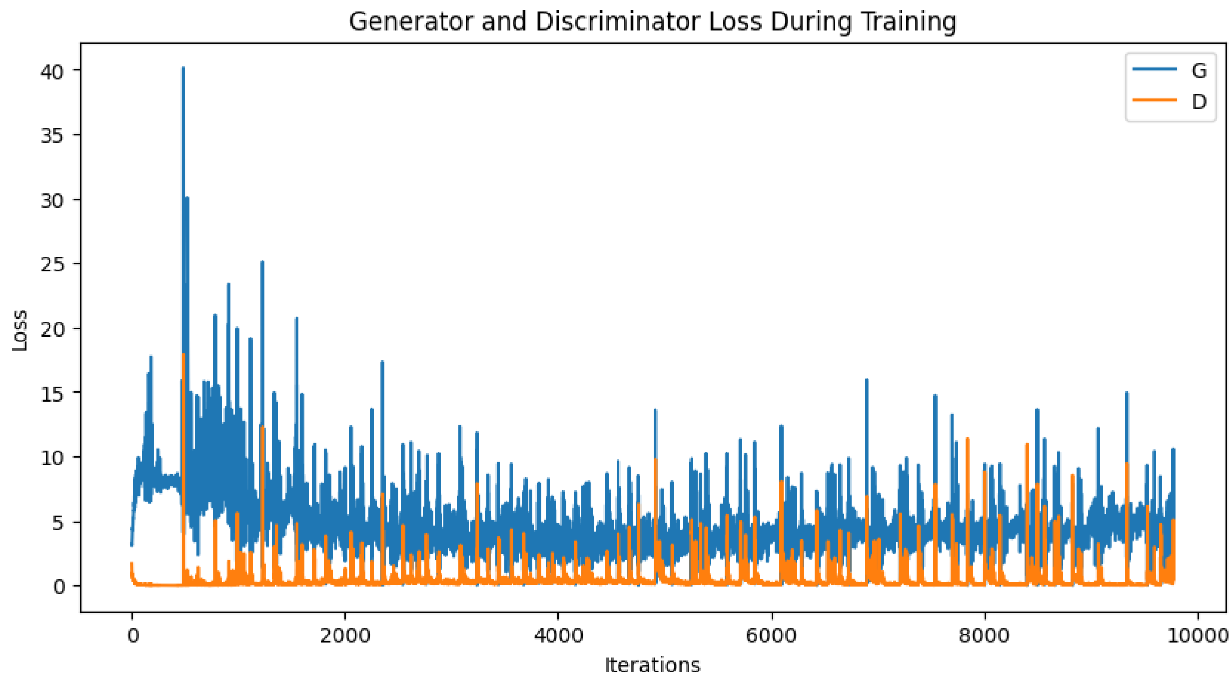
```
[22/25][300/391] Loss_D: 0.5644 Loss_G: 8.2055 D(x): 0.9928 D(G(z)):
0.3852/0.0004
[22/25][350/391] Loss_D: 0.0431 Loss_G: 4.5117 D(x): 0.9818 D(G(z)):
0.0238/0.0161
[23/25][0/391] Loss_D: 0.0463 Loss_G: 4.4239 D(x): 0.9684 D(G(z)):
0.0134/0.0172
[23/25][50/391] Loss_D: 0.0979 Loss_G: 6.0663 D(x): 0.9912 D(G(z)):
0.0824/0.0032
[23/25][100/391] Loss_D: 0.7022 Loss_G: 3.9991 D(x): 0.8125 D(G(z)):
0.3565/0.0254
[23/25][150/391] Loss_D: 0.1026 Loss_G: 3.4476 D(x): 0.9355 D(G(z)):
0.0325/0.0487
[23/25][200/391] Loss_D: 0.0599 Loss_G: 5.3748 D(x): 0.9887 D(G(z)):
0.0454/0.0085
[23/25][250/391] Loss_D: 0.0510 Loss_G: 4.0855 D(x): 0.9668 D(G(z)):
0.0163/0.0254
[23/25][300/391] Loss_D: 0.0330 Loss_G: 4.9451 D(x): 0.9738 D(G(z)):
0.0061/0.0112
[23/25][350/391] Loss_D: 1.2160 Loss_G: 1.5570 D(x): 0.4451 D(G(z)):
0.0448/0.2868
[24/25][0/391] Loss_D: 0.0601 Loss_G: 3.8026 D(x): 0.9798 D(G(z)):
0.0379/0.0341
[24/25][50/391] Loss_D: 0.0593 Loss_G: 5.2611 D(x): 0.9529 D(G(z)):
0.0098/0.0085
[24/25][100/391] Loss_D: 0.0217 Loss_G: 5.1558 D(x): 0.9905 D(G(z)):
0.0119/0.0096
[24/25][150/391] Loss_D: 0.4477 Loss_G: 2.4772 D(x): 0.8775 D(G(z)):
0.2166/0.1405
[24/25][200/391] Loss_D: 0.6628 Loss_G: 6.5809 D(x): 0.9890 D(G(z)):
0.4235/0.0022
[24/25][250/391] Loss_D: 0.0538 Loss_G: 5.1306 D(x): 0.9730 D(G(z)):
0.0250/0.0103
[24/25][300/391] Loss_D: 0.6559 Loss_G: 2.0413 D(x): 0.6094 D(G(z)):
0.0690/0.1772
[24/25][350/391] Loss_D: 0.4575 Loss_G: 2.4877 D(x): 0.8344 D(G(z)):
0.2040/0.1238

plt.figure(figsize=(8,8))
plt.axis("off")
plt.title("Fake Images")
plt.imshow(np.transpose(img_list[-1], (1,2,0)))
plt.show()
```

Fake Images



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



Generator and Discriminator Loss During Training
G
D
Loss
Iterations
0
5
10
15
20
25
30
35
40
0
2000
4000
6000
8000
10000