ece176_finalproject_DCGAN

March 15, 2025

Datasets:

- **SVT**: Street View Text Dataset
- **USPS**: USPS Dataset
- **Synthetic Digits**: Synthetic Digits Dataset

[1]:
```
import os
import torch
import torch.nn as nn
import torch.optim as optim
import numpy as np
import matplotlib.pyplot as plt
from torch.utils.data import DataLoader, Dataset
from torchvision import datasets, transforms, utils
from PIL import Image
import requests
import zipfile
```

[2]:
```
# Function to display images
def display_image(img, title):
    img = img / 2 + 0.5  # unnormalize
    npimg = img.numpy()
    plt.imshow(np.transpose(npimg, (1, 2, 0)))
    plt.title(title)
    plt.axis("off")
    plt.show()
```

[3]:
```
# Image Generator Model
class ImageGenerator(nn.Module):
    def __init__(self):
        super(ImageGenerator, 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)

# Image Discriminator Model
class ImageDiscriminator(nn.Module):
    def __init__(self):
        super(ImageDiscriminator, 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)
```

[4]:

```
# Custom Dataset for Synthetic Digits
class SyntheticDigitsDataset(Dataset):
    def __init__(self, root, transform=None):
        self.root = root
        self.transform = transform
        self.data = []
        self.labels = []
        self.load_dataset()

    def load_dataset(self):
        data_path = os.path.join(self.root, "imgs_train")
```

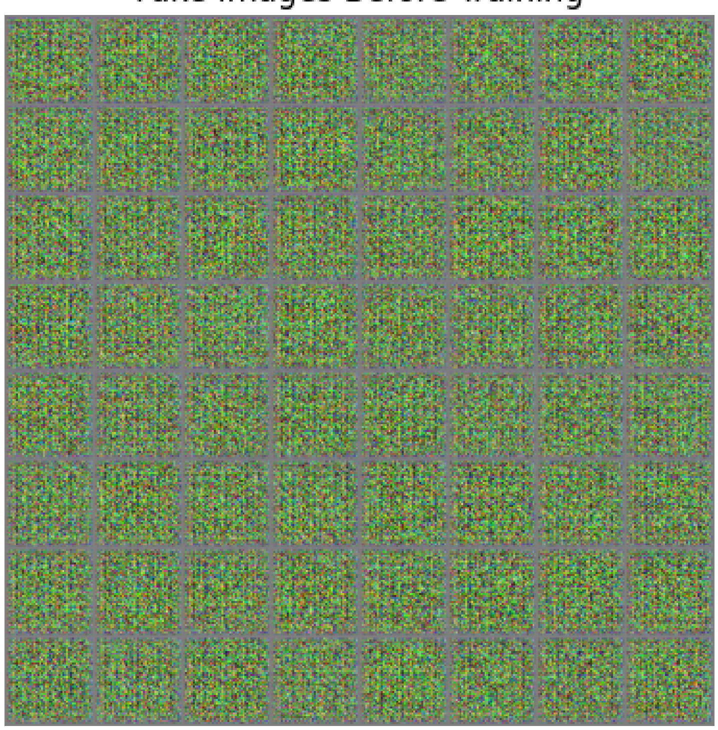
```python
        for label in os.listdir(data_path):
            label_path = os.path.join(data_path, label)
            if os.path.isdir(label_path):
                for img_name in os.listdir(label_path):
                    img_path = os.path.join(label_path, img_name)
                    self.data.append(img_path)
                    self.labels.append(int(label))

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

    def __getitem__(self, idx):
        img_path = self.data[idx]
        label = self.labels[idx]

        try:
            image = Image.open(img_path).convert('RGB').resize((64, 64))
        except Exception as e:
            print(f"Error loading image {img_path}: {e}")
            return None

        if self.transform:
            image = self.transform(image)
        return image, label

# Transformations
image_transform = transforms.Compose([
    transforms.Resize(64),
    transforms.ToTensor(),
    transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5)),
])

# Load Dataset
dataset = SyntheticDigitsDataset(root='/home/ngreenberg/.cache/kagglehub/
 ↪datasets/prasunroy/synthetic-digits/versions/1/synthetic_digits',␣
 ↪transform=image_transform)
data_loader = DataLoader(dataset, batch_size=64, shuffle=True, num_workers=1)

# Initialize Models and Optimizers
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
generator, discriminator = ImageGenerator().to(device), ImageDiscriminator().
 ↪to(device)

loss_function = nn.BCELoss()
discriminator_optimizer = optim.Adam(discriminator.parameters(), lr=0.0001,␣
 ↪betas=(0.5, 0.999))
```

```python
generator_optimizer = optim.Adam(generator.parameters(), lr=0.001, betas=(0.5, 0.999))

# Generate and show images before training
real_images, _ = next(iter(data_loader))
display_image(utils.make_grid(real_images[:64]), title="Real Images Before Training")

fixed_noise = torch.randn(64, 100, 1, 1, device=device)
with torch.no_grad():
    fake_images = generator(fixed_noise).detach().cpu()
    display_image(utils.make_grid(fake_images), title="Fake Images Before Training")

# Training loop
num_epochs = 35
real_label = 1
fake_label = 0

for epoch in range(num_epochs):
    for batch_idx, (real_data, _) in enumerate(data_loader):
        ############################
        # (1) Update Discriminator network: maximize log(D(x)) + log(1 - D(G(z)))
        ###########################
        real_data = real_data.to(device)
        batch_size = real_data.size(0)

        # Train Discriminator
        discriminator.zero_grad()
        label = torch.full((batch_size,), real_label, dtype=torch.float, device=device)
        output = discriminator(real_data)
        errD_real = loss_function(output, label)
        errD_real.backward()

        noise_input = torch.randn(batch_size, 100, 1, 1, device=device)
        fake_data = generator(noise_input)
        label.fill_(fake_label)
        output = discriminator(fake_data.detach())
        errD_fake = loss_function(output, label)
        errD_fake.backward()
        discriminator_optimizer.step()

        # Train Generator
        generator.zero_grad()
        label.fill_(real_label)
```

```
        output = discriminator(fake_data)
        errG = loss_function(output, label)
        errG.backward()
        generator_optimizer.step()

        # Print training progress
        if batch_idx % 50 == 0:
            print(f'Epoch {epoch+1} - Discriminator Loss: {errD_real.item() +␣
↪errD_fake.item():.4f} '
                  f'Generator Loss: {errG.item():.4f}')

# Generate images after training
with torch.no_grad():
    fake_images = generator(fixed_noise).detach().cpu()
    display_image(utils.make_grid(fake_images), title="Fake Images After␣
↪Training")

generator.eval()
discriminator.eval()

# Generate fake images
fixed_noise = torch.randn(1000, 100, 1, 1, device=device)
fake_images = generator(fixed_noise)  # Use 'generator' instead of 'netG'

# Get real images
real_images, _ = next(iter(DataLoader(dataset, batch_size=1000, shuffle=True)))
real_images = real_images.to(device)

# Get predictions
with torch.no_grad():
    real_preds = discriminator(real_images).squeeze()  # Use 'discriminator'␣
↪instead of 'netD'
    fake_preds = discriminator(fake_images).squeeze()

# Convert to binary classification (0 or 1)
real_correct = (real_preds > 0.5).sum().item()
fake_correct = (fake_preds < 0.5).sum().item()

# Define total correct and total samples
total_correct = real_correct + fake_correct
total_samples = len(real_images) + len(fake_images)

accuracy = (total_correct / total_samples) * 100  # Compute accuracy

# Correct print statement
print(f" Model Accuracy: {accuracy:.2f}% ({total_correct}/{total_samples}␣
↪correct predictions)")
```

Real Images Before Training

Fake Images Before Training



```
Epoch 1 - Discriminator Loss: 1.4204 Generator Loss: 1.9398
Epoch 1 - Discriminator Loss: 2.3156 Generator Loss: 1.1651
Epoch 1 - Discriminator Loss: 1.6183 Generator Loss: 0.8033
Epoch 1 - Discriminator Loss: 1.7434 Generator Loss: 0.7995
Epoch 2 - Discriminator Loss: 1.6739 Generator Loss: 0.7493
Epoch 2 - Discriminator Loss: 1.4944 Generator Loss: 0.6487
Epoch 2 - Discriminator Loss: 1.6412 Generator Loss: 1.0122
Epoch 2 - Discriminator Loss: 1.4303 Generator Loss: 0.8400
Epoch 3 - Discriminator Loss: 1.3551 Generator Loss: 0.7717
Epoch 3 - Discriminator Loss: 1.2491 Generator Loss: 0.8824
Epoch 3 - Discriminator Loss: 1.3972 Generator Loss: 0.7446
Epoch 3 - Discriminator Loss: 1.3083 Generator Loss: 0.9217
Epoch 4 - Discriminator Loss: 1.4275 Generator Loss: 0.7337
Epoch 4 - Discriminator Loss: 1.4507 Generator Loss: 0.8174
Epoch 4 - Discriminator Loss: 1.3473 Generator Loss: 0.7798
Epoch 4 - Discriminator Loss: 1.3263 Generator Loss: 0.8175
Epoch 5 - Discriminator Loss: 1.3909 Generator Loss: 0.8179
Epoch 5 - Discriminator Loss: 1.3366 Generator Loss: 0.8161
Epoch 5 - Discriminator Loss: 1.3203 Generator Loss: 0.8456
Epoch 5 - Discriminator Loss: 1.4699 Generator Loss: 0.7956
Epoch 6 - Discriminator Loss: 1.3898 Generator Loss: 1.0368
Epoch 6 - Discriminator Loss: 1.3631 Generator Loss: 0.8314
Epoch 6 - Discriminator Loss: 1.3550 Generator Loss: 0.8463
Epoch 6 - Discriminator Loss: 1.4643 Generator Loss: 0.8370
Epoch 7 - Discriminator Loss: 1.3559 Generator Loss: 0.8874
Epoch 7 - Discriminator Loss: 1.2838 Generator Loss: 0.8798
Epoch 7 - Discriminator Loss: 1.3405 Generator Loss: 0.9456
Epoch 7 - Discriminator Loss: 1.2749 Generator Loss: 0.9796
Epoch 8 - Discriminator Loss: 1.2302 Generator Loss: 1.0186
Epoch 8 - Discriminator Loss: 1.2990 Generator Loss: 0.9719
Epoch 8 - Discriminator Loss: 1.2616 Generator Loss: 1.0948
Epoch 8 - Discriminator Loss: 1.2668 Generator Loss: 0.9659
Epoch 9 - Discriminator Loss: 1.4121 Generator Loss: 0.8489
Epoch 9 - Discriminator Loss: 1.4869 Generator Loss: 0.9922
Epoch 9 - Discriminator Loss: 1.1919 Generator Loss: 1.2868
Epoch 9 - Discriminator Loss: 1.1736 Generator Loss: 1.0938
Epoch 10 - Discriminator Loss: 1.2496 Generator Loss: 1.2768
Epoch 10 - Discriminator Loss: 1.1329 Generator Loss: 1.1724
Epoch 10 - Discriminator Loss: 1.3285 Generator Loss: 1.4085
Epoch 10 - Discriminator Loss: 1.0067 Generator Loss: 1.3839
Epoch 11 - Discriminator Loss: 1.0304 Generator Loss: 1.7931
Epoch 11 - Discriminator Loss: 0.8107 Generator Loss: 2.0651
Epoch 11 - Discriminator Loss: 1.1315 Generator Loss: 1.5869
Epoch 11 - Discriminator Loss: 1.0695 Generator Loss: 1.9086
Epoch 12 - Discriminator Loss: 1.0190 Generator Loss: 1.6880
```

```
Epoch 12 - Discriminator Loss: 1.2002 Generator Loss: 1.7340
Epoch 12 - Discriminator Loss: 1.0015 Generator Loss: 1.4826
Epoch 12 - Discriminator Loss: 0.9450 Generator Loss: 2.0384
Epoch 13 - Discriminator Loss: 1.3909 Generator Loss: 1.7538
Epoch 13 - Discriminator Loss: 1.2545 Generator Loss: 1.6264
Epoch 13 - Discriminator Loss: 1.0842 Generator Loss: 1.8673
Epoch 13 - Discriminator Loss: 1.0529 Generator Loss: 1.3349
Epoch 14 - Discriminator Loss: 1.0597 Generator Loss: 1.5925
Epoch 14 - Discriminator Loss: 1.1183 Generator Loss: 1.5679
Epoch 14 - Discriminator Loss: 1.2553 Generator Loss: 1.3655
Epoch 14 - Discriminator Loss: 1.0451 Generator Loss: 1.6476
Epoch 15 - Discriminator Loss: 1.3576 Generator Loss: 2.0070
Epoch 15 - Discriminator Loss: 0.9705 Generator Loss: 1.7616
Epoch 15 - Discriminator Loss: 1.0149 Generator Loss: 1.7590
Epoch 15 - Discriminator Loss: 1.1438 Generator Loss: 1.7118
Epoch 16 - Discriminator Loss: 0.9504 Generator Loss: 2.1665
Epoch 16 - Discriminator Loss: 1.0747 Generator Loss: 2.1056
Epoch 16 - Discriminator Loss: 0.9263 Generator Loss: 1.8709
Epoch 16 - Discriminator Loss: 0.7896 Generator Loss: 2.0406
Epoch 17 - Discriminator Loss: 0.9931 Generator Loss: 1.3915
Epoch 17 - Discriminator Loss: 1.0625 Generator Loss: 2.5836
Epoch 17 - Discriminator Loss: 0.6079 Generator Loss: 2.4788
Epoch 17 - Discriminator Loss: 0.9734 Generator Loss: 2.0299
Epoch 18 - Discriminator Loss: 0.9589 Generator Loss: 2.4427
Epoch 18 - Discriminator Loss: 1.0558 Generator Loss: 1.7799
Epoch 18 - Discriminator Loss: 0.9123 Generator Loss: 2.1389
Epoch 18 - Discriminator Loss: 0.8327 Generator Loss: 2.2061
Epoch 19 - Discriminator Loss: 1.0274 Generator Loss: 1.5582
Epoch 19 - Discriminator Loss: 1.0117 Generator Loss: 1.5678
Epoch 19 - Discriminator Loss: 0.6861 Generator Loss: 1.8900
Epoch 19 - Discriminator Loss: 1.3328 Generator Loss: 1.1042
Epoch 20 - Discriminator Loss: 1.4514 Generator Loss: 3.4659
Epoch 20 - Discriminator Loss: 0.7766 Generator Loss: 1.9438
Epoch 20 - Discriminator Loss: 1.0835 Generator Loss: 1.8461
Epoch 20 - Discriminator Loss: 0.9343 Generator Loss: 1.6279
Epoch 21 - Discriminator Loss: 0.7515 Generator Loss: 2.1059
Epoch 21 - Discriminator Loss: 1.0145 Generator Loss: 1.4770
Epoch 21 - Discriminator Loss: 0.7711 Generator Loss: 1.7855
Epoch 21 - Discriminator Loss: 1.0111 Generator Loss: 1.4524
Epoch 22 - Discriminator Loss: 0.9011 Generator Loss: 2.0226
Epoch 22 - Discriminator Loss: 1.0762 Generator Loss: 2.1257
Epoch 22 - Discriminator Loss: 0.7912 Generator Loss: 1.9662
Epoch 22 - Discriminator Loss: 1.0037 Generator Loss: 1.9228
Epoch 23 - Discriminator Loss: 1.2637 Generator Loss: 3.1396
Epoch 23 - Discriminator Loss: 0.9721 Generator Loss: 2.5063
Epoch 23 - Discriminator Loss: 0.7365 Generator Loss: 1.9920
Epoch 23 - Discriminator Loss: 0.8775 Generator Loss: 1.7780
Epoch 24 - Discriminator Loss: 0.9106 Generator Loss: 1.5312
```

```
Epoch 24 - Discriminator Loss: 0.7734 Generator Loss: 1.5196
Epoch 24 - Discriminator Loss: 0.7815 Generator Loss: 2.7057
Epoch 24 - Discriminator Loss: 1.0490 Generator Loss: 2.0537
Epoch 25 - Discriminator Loss: 0.9401 Generator Loss: 2.6789
Epoch 25 - Discriminator Loss: 0.7916 Generator Loss: 1.7053
Epoch 25 - Discriminator Loss: 1.0839 Generator Loss: 1.2820
Epoch 25 - Discriminator Loss: 0.7829 Generator Loss: 1.9403
Epoch 26 - Discriminator Loss: 1.0861 Generator Loss: 0.8616
Epoch 26 - Discriminator Loss: 0.7733 Generator Loss: 1.7199
Epoch 26 - Discriminator Loss: 1.0042 Generator Loss: 2.4891
Epoch 26 - Discriminator Loss: 1.0323 Generator Loss: 1.1248
Epoch 27 - Discriminator Loss: 0.7593 Generator Loss: 1.6791
Epoch 27 - Discriminator Loss: 0.6697 Generator Loss: 1.8868
Epoch 27 - Discriminator Loss: 0.7925 Generator Loss: 1.9182
Epoch 27 - Discriminator Loss: 0.8295 Generator Loss: 2.3815
Epoch 28 - Discriminator Loss: 0.8960 Generator Loss: 1.3019
Epoch 28 - Discriminator Loss: 0.7175 Generator Loss: 1.6972
Epoch 28 - Discriminator Loss: 0.9023 Generator Loss: 2.1251
Epoch 28 - Discriminator Loss: 0.6711 Generator Loss: 2.0413
Epoch 29 - Discriminator Loss: 0.7853 Generator Loss: 2.1819
Epoch 29 - Discriminator Loss: 0.7289 Generator Loss: 1.7273
Epoch 29 - Discriminator Loss: 0.8416 Generator Loss: 1.9560
Epoch 29 - Discriminator Loss: 0.8292 Generator Loss: 1.8183
Epoch 30 - Discriminator Loss: 0.9085 Generator Loss: 1.7427
Epoch 30 - Discriminator Loss: 0.6259 Generator Loss: 1.9784
Epoch 30 - Discriminator Loss: 0.6911 Generator Loss: 1.9298
Epoch 30 - Discriminator Loss: 1.0533 Generator Loss: 1.6610
Epoch 31 - Discriminator Loss: 0.5760 Generator Loss: 1.8330
Epoch 31 - Discriminator Loss: 0.7357 Generator Loss: 2.6840
Epoch 31 - Discriminator Loss: 0.8193 Generator Loss: 2.0834
Epoch 31 - Discriminator Loss: 0.7915 Generator Loss: 2.2461
Epoch 32 - Discriminator Loss: 0.6382 Generator Loss: 3.5694
Epoch 32 - Discriminator Loss: 0.8090 Generator Loss: 2.9789
Epoch 32 - Discriminator Loss: 0.7102 Generator Loss: 2.5137
Epoch 32 - Discriminator Loss: 0.8327 Generator Loss: 2.0189
Epoch 33 - Discriminator Loss: 0.6346 Generator Loss: 2.8240
Epoch 33 - Discriminator Loss: 0.5144 Generator Loss: 2.4080
Epoch 33 - Discriminator Loss: 0.7709 Generator Loss: 1.4974
Epoch 33 - Discriminator Loss: 0.7979 Generator Loss: 1.6120
Epoch 34 - Discriminator Loss: 0.6898 Generator Loss: 2.5403
Epoch 34 - Discriminator Loss: 0.6858 Generator Loss: 2.3607
Epoch 34 - Discriminator Loss: 0.7535 Generator Loss: 2.2720
Epoch 34 - Discriminator Loss: 0.6906 Generator Loss: 3.2161
Epoch 35 - Discriminator Loss: 0.9291 Generator Loss: 3.6915
Epoch 35 - Discriminator Loss: 0.6428 Generator Loss: 2.7375
Epoch 35 - Discriminator Loss: 0.7065 Generator Loss: 2.0396
Epoch 35 - Discriminator Loss: 0.8348 Generator Loss: 2.6138
```

Fake Images After Training

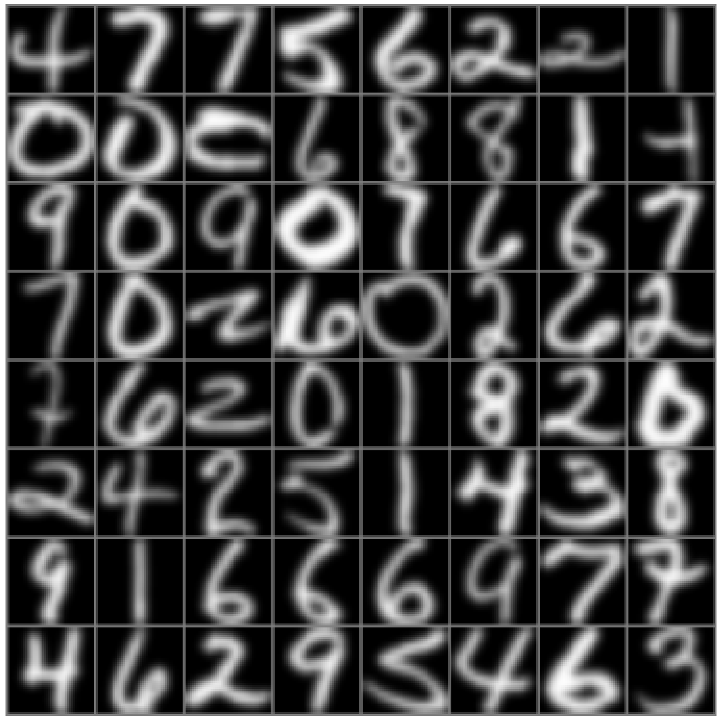
```
Model Accuracy: 81.35% (1627/2000 correct predictions)
```

[7]:
```
# USPS Dataset
# Transformations
image_transform = transforms.Compose([
    transforms.Resize(64),
    transforms.Grayscale(num_output_channels=3),
    transforms.ToTensor(),
    transforms.Normalize((0.5,), (0.5,))
])

# Load USPS dataset
dataset = datasets.USPS(root='./data', train=True, download=True,
 ↪transform=image_transform)
data_loader = DataLoader(dataset, batch_size=64, shuffle=True, num_workers=1)

device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
generator, discriminator = ImageGenerator().to(device), ImageDiscriminator().
 ↪to(device)

loss_function = nn.BCELoss()
```

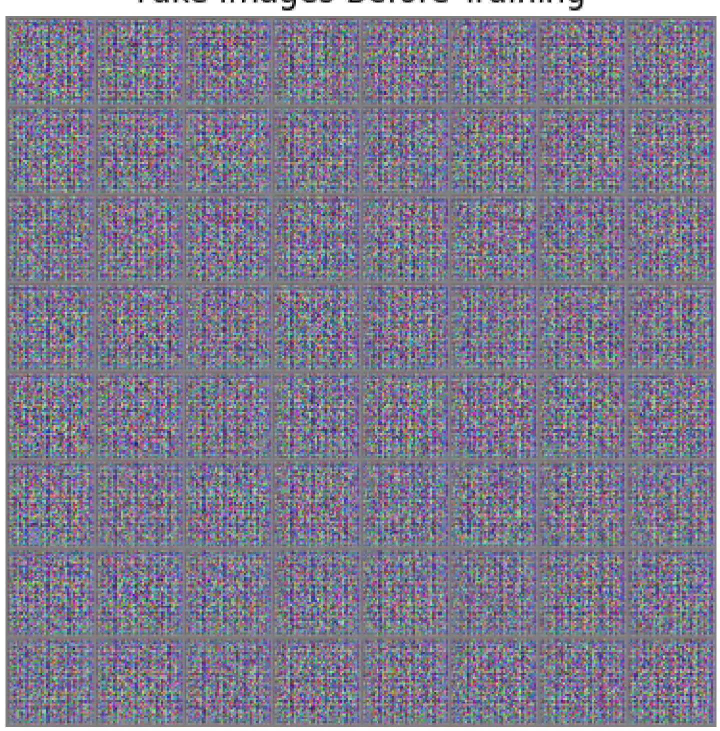
```
discriminator_optimizer = optim.Adam(discriminator.parameters(), lr=0.002,␣
↪betas=(0.5, 0.999))
generator_optimizer = optim.Adam(generator.parameters(), lr=0.002, betas=(0.5,␣
↪0.999))

# Pre-training visualization
fixed_noise = torch.randn(64, 100, 1, 1, device=device)
def visualize(title, images):
    display_image(utils.make_grid(images.cpu()), title=title)

real_images, _ = next(iter(data_loader))
visualize("Real Images Before Training", real_images[:64])
visualize("Fake Images Before Training", generator(fixed_noise).detach())

# Training Loop
num_epochs = 35
real_label, fake_label = 1, 0

for epoch in range(num_epochs):
    for batch_idx, (real_data, _) in enumerate(data_loader, start=1):
        ############################
        # (1) Update Discriminator network: maximize log(D(x)) + log(1 -␣
↪D(G(z)))
        ###########################
        batch_size = real_data.size(0)
        real_data = real_data.to(device)
        label_real = torch.full((batch_size,), real_label, dtype=torch.float,␣
↪device=device)
        label_fake = torch.full((batch_size,), fake_label, dtype=torch.float,␣
↪device=device)



        # Train Discriminator
        discriminator.zero_grad()
        errD_real = loss_function(discriminator(real_data), label_real)
        errD_real.backward()

        noise_input = torch.randn(batch_size, 100, 1, 1, device=device)
        fake_data = generator(noise_input)
        errD_fake = loss_function(discriminator(fake_data.detach()), label_fake)
        errD_fake.backward()
        discriminator_optimizer.step()

        # Train Generator
        generator.zero_grad()
        errG = loss_function(discriminator(fake_data), label_real)
        errG.backward()
```

```
        generator_optimizer.step()

        if batch_idx % 50 == 0:
            print(f'Epoch {epoch+1} - Discriminator Loss: {errD_real.item() +␣
 ↪errD_fake.item():.4f} '
                  f'Generator Loss: {errG.item():.4f}')

# Post-training visualization
with torch.no_grad():
    visualize("Fake Images After Training", generator(fixed_noise))

# Model Evaluation
# Model Evaluation
generator.eval()
discriminator.eval()

# Generate fake images
fixed_noise = torch.randn(1000, 100, 1, 1, device=device)
fake_images = generator(fixed_noise)  # Use 'generator' instead of 'netG'

# Get real images
real_images, _ = next(iter(DataLoader(dataset, batch_size=1000, shuffle=True)))
real_images = real_images.to(device)

# Get predictions
with torch.no_grad():
    real_preds = discriminator(real_images).squeeze()  # Use 'discriminator'␣
 ↪instead of 'netD'
    fake_preds = discriminator(fake_images).squeeze()

# Convert to binary classification (0 or 1)
real_correct = (real_preds > 0.5).sum().item()
fake_correct = (fake_preds < 0.5).sum().item()

# Define total correct and total samples
total_correct = real_correct + fake_correct
total_samples = len(real_images) + len(fake_images)

accuracy = (total_correct / total_samples) * 100  # Compute accuracy

# Correct print statement
print(f" Model Accuracy: {accuracy:.2f}% ({total_correct}/{total_samples}␣
 ↪correct predictions)")
```

Real Images Before Training



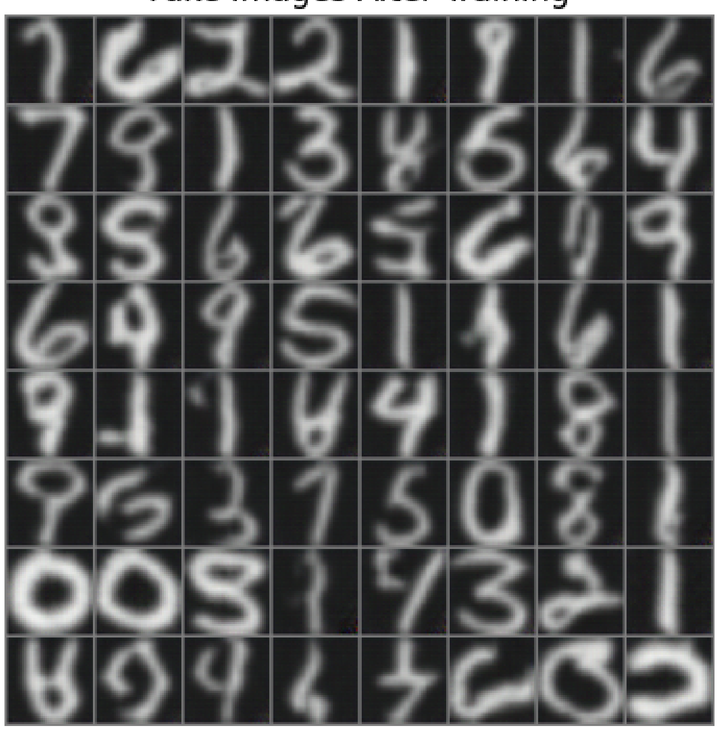
Fake Images Before Training



```
Epoch 1 - Discriminator Loss: 1.2134 Generator Loss: 2.1676
Epoch 1 - Discriminator Loss: 1.3147 Generator Loss: 1.0865
Epoch 2 - Discriminator Loss: 1.3489 Generator Loss: 5.0309
Epoch 2 - Discriminator Loss: 1.3484 Generator Loss: 0.5912
Epoch 3 - Discriminator Loss: 1.4141 Generator Loss: 1.4120
Epoch 3 - Discriminator Loss: 1.6248 Generator Loss: 0.7487
Epoch 4 - Discriminator Loss: 1.1913 Generator Loss: 1.0329
Epoch 4 - Discriminator Loss: 1.3861 Generator Loss: 1.0061
Epoch 5 - Discriminator Loss: 1.3719 Generator Loss: 0.9272
Epoch 5 - Discriminator Loss: 1.4196 Generator Loss: 2.1481
Epoch 6 - Discriminator Loss: 1.2437 Generator Loss: 0.8487
Epoch 6 - Discriminator Loss: 1.2601 Generator Loss: 1.1220
Epoch 7 - Discriminator Loss: 1.2207 Generator Loss: 1.7656
Epoch 7 - Discriminator Loss: 1.3804 Generator Loss: 1.2798
Epoch 8 - Discriminator Loss: 1.2838 Generator Loss: 0.9507
Epoch 8 - Discriminator Loss: 1.2753 Generator Loss: 1.0324
Epoch 9 - Discriminator Loss: 1.6415 Generator Loss: 0.4072
Epoch 9 - Discriminator Loss: 1.1103 Generator Loss: 0.9708
Epoch 10 - Discriminator Loss: 1.3809 Generator Loss: 1.9597
Epoch 10 - Discriminator Loss: 1.2366 Generator Loss: 2.6908
Epoch 11 - Discriminator Loss: 1.0994 Generator Loss: 2.4691
Epoch 11 - Discriminator Loss: 1.4035 Generator Loss: 1.1129
Epoch 12 - Discriminator Loss: 1.5471 Generator Loss: 2.3819
Epoch 12 - Discriminator Loss: 1.2360 Generator Loss: 1.3622
Epoch 13 - Discriminator Loss: 1.5467 Generator Loss: 0.7391
Epoch 13 - Discriminator Loss: 1.2275 Generator Loss: 0.8886
Epoch 14 - Discriminator Loss: 0.9235 Generator Loss: 1.4488
Epoch 14 - Discriminator Loss: 1.2663 Generator Loss: 1.2951
Epoch 15 - Discriminator Loss: 0.9300 Generator Loss: 1.7403
Epoch 15 - Discriminator Loss: 1.1849 Generator Loss: 1.2428
Epoch 16 - Discriminator Loss: 0.0406 Generator Loss: 4.5459
Epoch 16 - Discriminator Loss: 0.0690 Generator Loss: 4.6696
Epoch 17 - Discriminator Loss: 0.0068 Generator Loss: 6.3078
Epoch 17 - Discriminator Loss: 0.0024 Generator Loss: 6.8298
Epoch 18 - Discriminator Loss: 0.0020 Generator Loss: 7.3134
Epoch 18 - Discriminator Loss: 0.0006 Generator Loss: 8.0254
Epoch 19 - Discriminator Loss: 0.0012 Generator Loss: 7.8373
Epoch 19 - Discriminator Loss: 0.0003 Generator Loss: 13.3177
Epoch 20 - Discriminator Loss: 0.0018 Generator Loss: 7.3422
Epoch 20 - Discriminator Loss: 0.0003 Generator Loss: 8.5866
Epoch 21 - Discriminator Loss: 0.0004 Generator Loss: 8.3975
Epoch 21 - Discriminator Loss: 0.0001 Generator Loss: 10.4373
Epoch 22 - Discriminator Loss: 1.1000 Generator Loss: 5.8259
Epoch 22 - Discriminator Loss: 0.5054 Generator Loss: 2.7216
Epoch 23 - Discriminator Loss: 0.4369 Generator Loss: 3.5242
```

```
Epoch 23 - Discriminator Loss: 0.9825 Generator Loss: 2.1754
Epoch 24 - Discriminator Loss: 0.4815 Generator Loss: 2.7820
Epoch 24 - Discriminator Loss: 1.2990 Generator Loss: 1.1502
Epoch 25 - Discriminator Loss: 0.8272 Generator Loss: 2.3475
Epoch 25 - Discriminator Loss: 0.9584 Generator Loss: 1.0205
Epoch 26 - Discriminator Loss: 1.3425 Generator Loss: 0.8776
Epoch 26 - Discriminator Loss: 1.0201 Generator Loss: 1.8083
Epoch 27 - Discriminator Loss: 1.3951 Generator Loss: 1.7012
Epoch 27 - Discriminator Loss: 1.2140 Generator Loss: 2.8120
Epoch 28 - Discriminator Loss: 0.5138 Generator Loss: 2.2658
Epoch 28 - Discriminator Loss: 0.1551 Generator Loss: 3.2965
Epoch 29 - Discriminator Loss: 0.9166 Generator Loss: 1.1516
Epoch 29 - Discriminator Loss: 0.8062 Generator Loss: 2.8240
Epoch 30 - Discriminator Loss: 2.0519 Generator Loss: 0.9770
Epoch 30 - Discriminator Loss: 0.7901 Generator Loss: 2.0921
Epoch 31 - Discriminator Loss: 0.0398 Generator Loss: 4.7936
Epoch 31 - Discriminator Loss: 1.1375 Generator Loss: 1.0784
Epoch 32 - Discriminator Loss: 1.0865 Generator Loss: 3.2530
Epoch 32 - Discriminator Loss: 1.3082 Generator Loss: 2.2531
Epoch 33 - Discriminator Loss: 0.8098 Generator Loss: 1.4495
Epoch 33 - Discriminator Loss: 1.3380 Generator Loss: 2.0772
Epoch 34 - Discriminator Loss: 0.7925 Generator Loss: 1.2365
Epoch 34 - Discriminator Loss: 1.3890 Generator Loss: 4.5039
Epoch 35 - Discriminator Loss: 0.6376 Generator Loss: 2.7063
Epoch 35 - Discriminator Loss: 0.7380 Generator Loss: 3.6645
```

Fake Images After Training



```
Model Accuracy: 73.90% (1478/2000 correct predictions)
```

[6]:
```
# Define the path to your dataset
image_directory = '/home/ngreenberg/.cache/kagglehub/datasets/nageshsingh/
↪the-street-view-text-dataset/versions/1/img'

# Custom Dataset for SVT Images
class StreetViewDataset(Dataset):
    def __init__(self, directory, transform=None):
        self.directory = directory
        self.transform = transform
        self.image_paths = [os.path.join(directory, image) for image in os.
↪listdir(directory) if image.endswith('.jpg')]

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

    def __getitem__(self, index):
        image_path = self.image_paths[index]
        image = Image.open(image_path).convert('RGB')

        if self.transform:
            image = self.transform(image)
        else:
            # Default transformation: resize to 64x64
            default_transform = transforms.Compose([
                transforms.Resize((64, 64)),
                transforms.ToTensor(),
                transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5))
            ])
            image = default_transform(image)

        return image

# Transformations
image_transform = transforms.Compose([
    transforms.Resize((64, 64)),  # Resize all images to 64x64
    transforms.ToTensor(),
    transforms.Normalize((0.5, 0.5, 0.5), (0.5, 0.5, 0.5)),  # Normalize to␣
↪[-1, 1]
])

# Initialize Dataset and DataLoader
dataset = StreetViewDataset(directory=image_directory,␣
↪transform=image_transform)
```

```
data_loader = DataLoader(dataset, batch_size=64, shuffle=True, num_workers=2)

# Initialize models, optimizers, and loss function
device = torch.device("cuda" if torch.cuda.is_available() else "cpu")
generator = ImageGenerator().to(device)
discriminator = ImageDiscriminator().to(device)

loss_function = nn.BCELoss()
discriminator_optimizer = optim.Adam(discriminator.parameters(), lr=0.0002,␣
↪betas=(0.5, 0.999))
generator_optimizer = optim.Adam(generator.parameters(), lr=0.0002, betas=(0.5,␣
↪0.999))

# Generate images BEFORE training
real_images_batch = next(iter(data_loader))
display_image(utils.make_grid(real_images_batch[:64]), title="Real Images␣
↪Before Training")

noise_input = torch.randn(64, 100, 1, 1, device=device)
with torch.no_grad():
    fake_images_batch = generator(noise_input).detach().cpu()
    display_image(utils.make_grid(fake_images_batch), title="Fake Images Before␣
↪Training")

# Training loop
epochs = 100
real_label = 1
fake_label = 0

for epoch in range(epochs):
    for batch_idx, data_batch in enumerate(data_loader, 0):
        ############################
        # (1) Update Discriminator network: maximize log(D(x)) + log(1 -␣
↪D(G(z)))
        ###########################
        # Train with all-real batch
        discriminator.zero_grad()
        real_data = data_batch.to(device)
        batch_size = real_data.size(0)
        real_labels = torch.full((batch_size,), real_label, dtype=torch.float,␣
↪device=device)
        real_output = discriminator(real_data)
        real_loss = loss_function(real_output, real_labels)
        real_loss.backward()
        D_real = real_output.mean().item()
```

```
        # Train with all-fake batch
        noise_input = torch.randn(batch_size, 100, 1, 1, device=device)
        fake_data = generator(noise_input)
        fake_labels = torch.full((batch_size,), fake_label, dtype=torch.float,␣
 ↪device=device)
        fake_output = discriminator(fake_data.detach())
        fake_loss = loss_function(fake_output, fake_labels)
        fake_loss.backward()
        D_fake = fake_output.mean().item()
        discriminator_loss = real_loss + fake_loss
        discriminator_optimizer.step()

        # Train Generator
        generator.zero_grad()
        real_labels.fill_(real_label)
        generator_output = discriminator(fake_data)
        generator_loss = loss_function(generator_output, real_labels)
        generator_loss.backward()
        generator_optimizer.step()

        # Print training progress every 100 batches
        if batch_idx % 100 == 0:
            print(f'Epoch {epoch+1} - Discriminator Loss: {discriminator_loss.
 ↪item():.4f} '
                  f'Generator Loss: {generator_loss.item():.4f}')

# Generate images AFTER training
with torch.no_grad():
    fake_images_batch = generator(noise_input).detach().cpu()
    display_image(utils.make_grid(fake_images_batch), title="Fake Images After␣
 ↪Training")

# Model Evaluation
generator.eval()
discriminator.eval()

# Generate fake images for evaluation
evaluation_noise = torch.randn(1000, 100, 1, 1, device=device)
generated_images = generator(evaluation_noise)

# Get real images
real_images_batch = next(iter(DataLoader(dataset, batch_size=1000,␣
 ↪shuffle=True)))
real_images_batch = real_images_batch.to(device)

# Get predictions from the discriminator
with torch.no_grad():
```

```
    real_predictions = torch.sigmoid(discriminator(real_images_batch)).squeeze()
    fake_predictions = torch.sigmoid(discriminator(generated_images)).squeeze()

# Convert to binary classification (0 or 1)
correct_real_predictions = (real_predictions > 0.5).sum().item()
correct_fake_predictions = (fake_predictions <= 0.5).sum().item()

total_samples = len(real_images_batch) + len(generated_images)
accuracy = (correct_real_predictions + correct_fake_predictions) / total_samples * 100

# Print final accuracy
print(f"  Model Accuracy: {accuracy:.2f}% ({correct_real_predictions + correct_fake_predictions}/{total_samples} correct predictions)")
```

```
Epoch 1 - Discriminator Loss: 1.3976 Generator Loss: 3.2044
Epoch 2 - Discriminator Loss: 0.3430 Generator Loss: 6.1691
Epoch 3 - Discriminator Loss: 0.2667 Generator Loss: 9.4412
Epoch 4 - Discriminator Loss: 0.0856 Generator Loss: 14.5835
Epoch 5 - Discriminator Loss: 0.0571 Generator Loss: 12.5232
Epoch 6 - Discriminator Loss: 0.0489 Generator Loss: 12.3733
Epoch 7 - Discriminator Loss: 0.0477 Generator Loss: 13.5750
Epoch 8 - Discriminator Loss: 0.0176 Generator Loss: 9.5794
Epoch 9 - Discriminator Loss: 0.0634 Generator Loss: 12.7333
Epoch 10 - Discriminator Loss: 1.1853 Generator Loss: 18.5286
Epoch 11 - Discriminator Loss: 0.0153 Generator Loss: 12.5842
Epoch 12 - Discriminator Loss: 0.0218 Generator Loss: 14.3519
Epoch 13 - Discriminator Loss: 0.3845 Generator Loss: 18.5853
Epoch 14 - Discriminator Loss: 0.0251 Generator Loss: 11.1795
Epoch 15 - Discriminator Loss: 0.5953 Generator Loss: 13.5450
Epoch 16 - Discriminator Loss: 1.3327 Generator Loss: 13.0960
Epoch 17 - Discriminator Loss: 0.1372 Generator Loss: 6.1798
Epoch 18 - Discriminator Loss: 0.2290 Generator Loss: 5.2222
Epoch 19 - Discriminator Loss: 1.9983 Generator Loss: 7.2043
Epoch 20 - Discriminator Loss: 0.7163 Generator Loss: 4.4243
Epoch 21 - Discriminator Loss: 2.4273 Generator Loss: 3.3733
Epoch 22 - Discriminator Loss: 0.8401 Generator Loss: 4.6207
```

```
Epoch 23 - Discriminator Loss: 0.8765 Generator Loss: 4.1280
Epoch 24 - Discriminator Loss: 0.6174 Generator Loss: 3.8445
Epoch 25 - Discriminator Loss: 0.5954 Generator Loss: 2.9404
Epoch 26 - Discriminator Loss: 0.9838 Generator Loss: 3.6764
Epoch 27 - Discriminator Loss: 0.9870 Generator Loss: 3.9636
Epoch 28 - Discriminator Loss: 0.8436 Generator Loss: 5.1797
Epoch 29 - Discriminator Loss: 0.3799 Generator Loss: 3.0438
Epoch 30 - Discriminator Loss: 0.5059 Generator Loss: 4.3964
Epoch 31 - Discriminator Loss: 0.9481 Generator Loss: 5.6132
Epoch 32 - Discriminator Loss: 0.3853 Generator Loss: 3.6561
Epoch 33 - Discriminator Loss: 0.3555 Generator Loss: 3.1753
Epoch 34 - Discriminator Loss: 1.9237 Generator Loss: 1.5829
Epoch 35 - Discriminator Loss: 0.3927 Generator Loss: 3.4170
Epoch 36 - Discriminator Loss: 0.7080 Generator Loss: 1.9447
Epoch 37 - Discriminator Loss: 0.2448 Generator Loss: 4.1261
Epoch 38 - Discriminator Loss: 0.9778 Generator Loss: 4.9157
Epoch 39 - Discriminator Loss: 1.0457 Generator Loss: 7.3484
Epoch 40 - Discriminator Loss: 0.3756 Generator Loss: 4.2588
Epoch 41 - Discriminator Loss: 0.8762 Generator Loss: 6.0929
Epoch 42 - Discriminator Loss: 0.9825 Generator Loss: 2.0850
Epoch 43 - Discriminator Loss: 0.8168 Generator Loss: 3.9999
Epoch 44 - Discriminator Loss: 0.3501 Generator Loss: 3.5177
Epoch 45 - Discriminator Loss: 0.5041 Generator Loss: 3.7941
Epoch 46 - Discriminator Loss: 0.9057 Generator Loss: 2.8545
Epoch 47 - Discriminator Loss: 0.3446 Generator Loss: 2.7077
Epoch 48 - Discriminator Loss: 0.4026 Generator Loss: 3.6352
Epoch 49 - Discriminator Loss: 0.5609 Generator Loss: 4.7670
Epoch 50 - Discriminator Loss: 0.3513 Generator Loss: 3.5058
Epoch 51 - Discriminator Loss: 0.4555 Generator Loss: 1.6146
Epoch 52 - Discriminator Loss: 0.5281 Generator Loss: 2.8317
Epoch 53 - Discriminator Loss: 0.5796 Generator Loss: 4.5590
Epoch 54 - Discriminator Loss: 0.4449 Generator Loss: 4.4765
Epoch 55 - Discriminator Loss: 0.4096 Generator Loss: 2.5719
Epoch 56 - Discriminator Loss: 0.6381 Generator Loss: 4.4751
Epoch 57 - Discriminator Loss: 0.4900 Generator Loss: 2.9316
Epoch 58 - Discriminator Loss: 0.8780 Generator Loss: 5.3754
Epoch 59 - Discriminator Loss: 0.4219 Generator Loss: 3.4180
Epoch 60 - Discriminator Loss: 0.3760 Generator Loss: 4.3366
Epoch 61 - Discriminator Loss: 0.6445 Generator Loss: 2.3226
Epoch 62 - Discriminator Loss: 0.4568 Generator Loss: 3.8143
Epoch 63 - Discriminator Loss: 1.1783 Generator Loss: 3.2062
Epoch 64 - Discriminator Loss: 0.6320 Generator Loss: 4.3956
Epoch 65 - Discriminator Loss: 0.9974 Generator Loss: 2.4449
Epoch 66 - Discriminator Loss: 0.5055 Generator Loss: 4.2561
Epoch 67 - Discriminator Loss: 0.5413 Generator Loss: 3.4116
Epoch 68 - Discriminator Loss: 0.6752 Generator Loss: 6.5895
Epoch 69 - Discriminator Loss: 0.4003 Generator Loss: 5.4946
Epoch 70 - Discriminator Loss: 0.2740 Generator Loss: 5.1221
```

```
Epoch 71 - Discriminator Loss: 0.4663 Generator Loss: 5.2035
Epoch 72 - Discriminator Loss: 0.4243 Generator Loss: 5.2571
Epoch 73 - Discriminator Loss: 0.3357 Generator Loss: 3.3061
Epoch 74 - Discriminator Loss: 0.2665 Generator Loss: 4.0591
Epoch 75 - Discriminator Loss: 1.2102 Generator Loss: 5.2737
Epoch 76 - Discriminator Loss: 0.7662 Generator Loss: 6.8074
Epoch 77 - Discriminator Loss: 0.4395 Generator Loss: 4.2145
Epoch 78 - Discriminator Loss: 0.2051 Generator Loss: 4.0692
Epoch 79 - Discriminator Loss: 0.3453 Generator Loss: 3.8035
Epoch 80 - Discriminator Loss: 0.2791 Generator Loss: 4.8405
Epoch 81 - Discriminator Loss: 0.5751 Generator Loss: 4.3533
Epoch 82 - Discriminator Loss: 0.3992 Generator Loss: 3.3420
Epoch 83 - Discriminator Loss: 0.3341 Generator Loss: 4.2784
Epoch 84 - Discriminator Loss: 0.3576 Generator Loss: 4.7651
Epoch 85 - Discriminator Loss: 0.2752 Generator Loss: 5.2177
Epoch 86 - Discriminator Loss: 0.2426 Generator Loss: 5.0206
Epoch 87 - Discriminator Loss: 0.4754 Generator Loss: 6.6318
Epoch 88 - Discriminator Loss: 0.2452 Generator Loss: 4.3895
Epoch 89 - Discriminator Loss: 0.5731 Generator Loss: 8.2381
Epoch 90 - Discriminator Loss: 0.6925 Generator Loss: 7.7093
Epoch 91 - Discriminator Loss: 0.3151 Generator Loss: 4.1164
Epoch 92 - Discriminator Loss: 0.2633 Generator Loss: 3.7816
Epoch 93 - Discriminator Loss: 0.3341 Generator Loss: 4.4008
Epoch 94 - Discriminator Loss: 0.3329 Generator Loss: 6.3003
Epoch 95 - Discriminator Loss: 0.8028 Generator Loss: 7.0365
Epoch 96 - Discriminator Loss: 2.1721 Generator Loss: 5.1612
Epoch 97 - Discriminator Loss: 1.0019 Generator Loss: 8.8703
Epoch 98 - Discriminator Loss: 0.6990 Generator Loss: 5.3886
Epoch 99 - Discriminator Loss: 0.3709 Generator Loss: 5.5475
Epoch 100 - Discriminator Loss: 0.4613 Generator Loss: 5.9332
```

```
Model Accuracy: 25.93% (350/1350 correct predictions)
```