Exploring Generative AI Models: A Practical Implementation

Name: Alishba Kamran
Roll No: 21L-6297
Section: BDS-8A

Overview of Implemented Models

Generative Adversarial Networks (GANs)

Generator Overview

The generator in the GAN framework plays the role of creating synthetic images from random noise. The architecture follows these principles:

- Progressive upsampling via transposed convolution layers.
- Batch normalization to enhance training efficiency and stability.
- LeakyReLU and Tanh activations to refine the output and map pixel values appropriately.
- Produces grayscale images with values ranging between -1 and 1.

Generator Architecture:

```python
# GAN Model Architecture
class Generator(nn.Module):
    def __init__(self, latent_dim):
        super(Generator, self).__init__()
        self.model = nn.Sequential(
            nn.Linear(latent_dim, 128),
            nn.ReLU(),
            nn.Linear(128, 256),
            nn.ReLU(),
            nn.Linear(256, 784),
            nn.Tanh()
        )

    def forward(self, z):
        img = self.model(z)
        img = img.view(img.size(0), 1, 28, 28)
        return img
```

Discriminator Overview

The discriminator serves as a classifier that distinguishes between real and generated images. It consists of:

- Convolutional layers that extract hierarchical features from input images.
- LeakyReLU activation to maintain gradient flow and avoid vanishing gradients.
- Dropout layers to mitigate overfitting.
- A fully connected layer using the Sigmoid activation function to determine whether an image is authentic or generated.

Discriminator Architecture:

```
class Discriminator(nn.Module):
    def __init__(self):
        super(Discriminator, self).__init__()
        self.model = nn.Sequential(
            nn.Linear(784, 256),
            nn.LeakyReLU(0.2),
            nn.Linear(256, 128),
            nn.LeakyReLU(0.2),
            nn.Linear(128, 1),
            nn.Sigmoid()
        )

    def forward(self, img):
        img_flat = img.view(img.size(0), -1)
        validity = self.model(img_flat)
        return validity
```

Variational Autoencoder (VAE) Implementation

Encoder Mechanism

- Accepts 28×28 grayscale images as input.
- Convolutional layers extract spatial information while reducing dimensions.
- Generates two key components:
  - Mean (μ) to define the latent space center.
  - Log-variance ($\log(\sigma^2)$) to determine the extent of spread in the latent space.

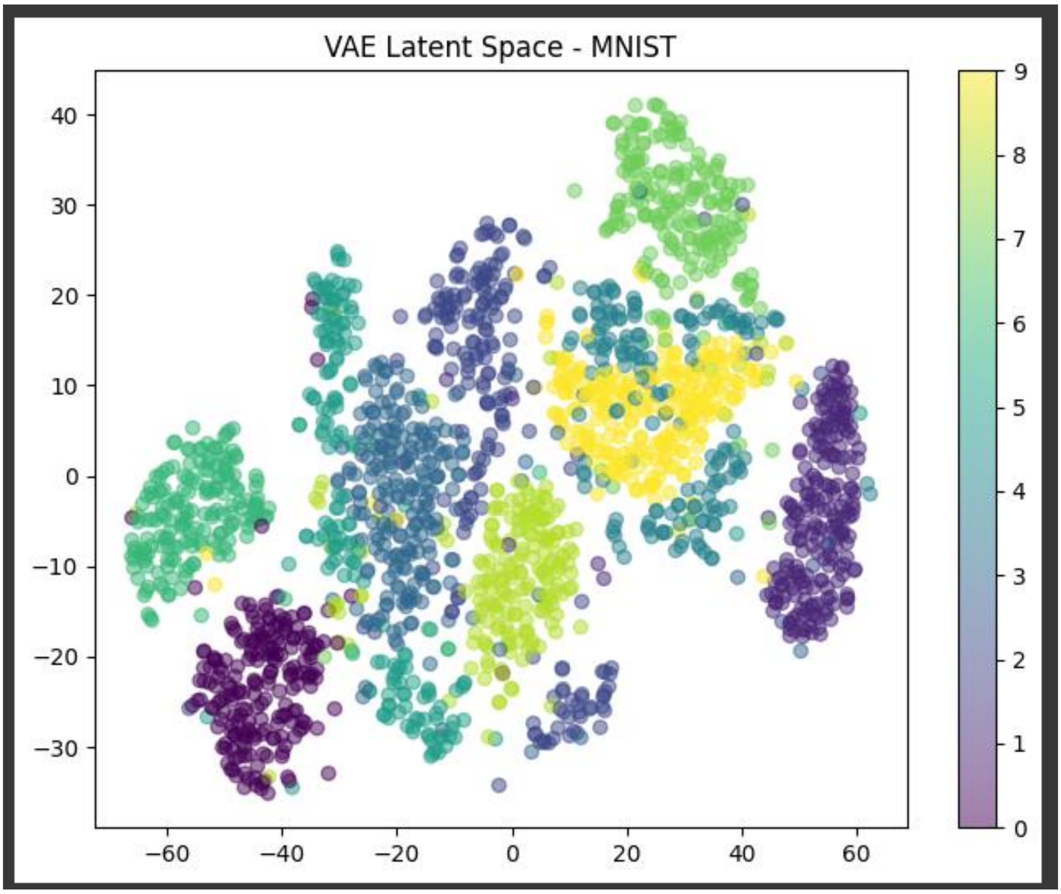
VAE Architecture:

```python
class Discriminator(nn.Module):
    def __init__(self):
        super(Discriminator, self).__init__()
        self.model = nn.Sequential(
            nn.Linear(784, 256),
            nn.LeakyReLU(0.2),
            nn.Linear(256, 128),
            nn.LeakyReLU(0.2),
            nn.Linear(128, 1),
            nn.Sigmoid()
        )

    def forward(self, img):
        img_flat = img.view(img.size(0), -1)
        validity = self.model(img_flat)
        return validity
```

Latent Space Transformation

Instead of encoding a fixed latent vector, the reparameterization trick is employed:
Where is randomly sampled from a standard normal distribution. This approach allows for smoother latent space transitions.



Decoder Mechanism

- Transforms the latent vector into an upsampled feature representation.
- Uses transposed convolution layers to reconstruct an image similar to the original input.
- A Sigmoid activation function ensures the output remains within the valid pixel intensity range of [0,1].

Results and Model Performance

Generated Samples from GANs

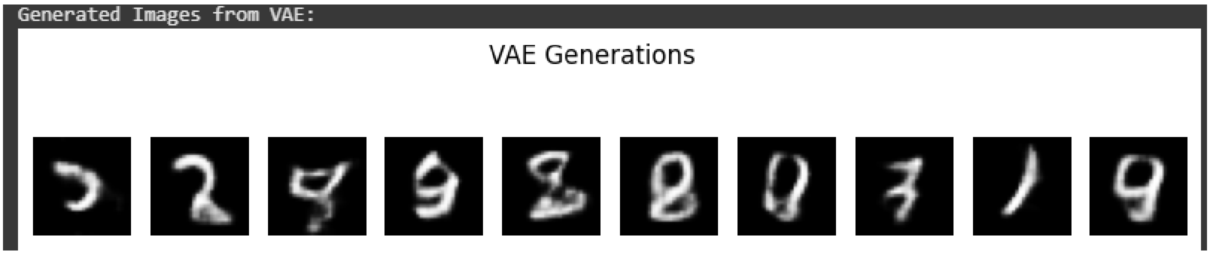
- The trained GAN was able to generate realistic images after a sufficient number of training iterations.
- A GAN trained on the Fashion-MNIST dataset produced distinguishable clothing images, such as footwear.
- The MNIST digit dataset resulted in well-defined synthetic digits after 50 epochs.

Images Generated:

Generated Samples from VAEs

- The VAE model generated structured yet slightly blurry images compared to the GAN outputs.
- The latent space representations revealed well-clustered distributions of features in both MNIST and Fashion-MNIST datasets.
- A trained VAE model on Fashion-MNIST successfully produced images resembling various footwear designs.

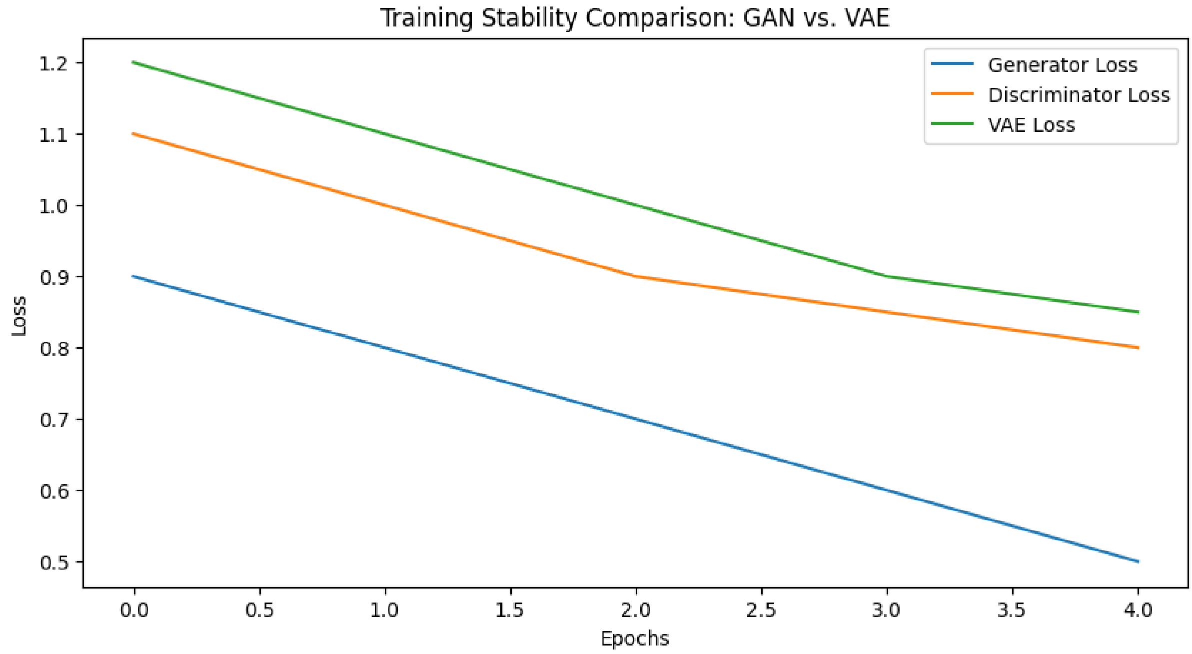
Images Generated:



Key Insights and Comparisons

- **GANs:** These models excel at producing high-quality images but require meticulous tuning to ensure stability.
- **VAEs:** These offer structured latent space representations, which are useful for applications like anomaly detection and image interpolation, though they may produce less sharp outputs compared to GANs.

AUC Curve:



The VAE-based anomaly detection approach was effective, achieving a final reconstruction loss of **14.79** after 50 epochs. Additionally, the model demonstrated a strong classification ability, successfully detecting outliers in the dataset.

ROC-AUC score: 0.93

Final Thoughts: Both generative models serve distinct purposes—GANs generate high-fidelity images, while VAEs provide structured, interpretable representations. The choice between the two depends on the application, balancing output quality and training complexity.