GAN Project 4 Report: Image Generation with MNIST Dataset

1. Introduction

This report outlines the implementation of a Generative Adversarial Network (GAN) using the MNIST dataset, a well-known benchmark in the field of machine learning. The objective was to generate realistic images of handwritten digits, thereby demonstrating the capabilities of GANs in image synthesis. The project was carried out using Python 3 and the PyTorch library, focusing on building and training both the generator and discriminator models. This approach allows the model to learn the underlying distribution of the data, facilitating the creation of new, synthetic data that resembles the training set.

2. Dataset

The MNIST dataset comprises 60,000 training images and 10,000 test images of handwritten digits, each represented as a 28x28 grayscale image. The images are labeled with integer values ranging from 0 to 9, indicating the actual digit depicted in the image. This dataset serves as an excellent resource for evaluating image generation techniques due to its simplicity and the variety of digits.
The images are normalized to a range of [-1, 1], which is critical for the training stability of GANs.

3. Model Development

The GAN architecture consists of two main components: the generator and the discriminator.

Generator:

- The generator is responsible for creating images from random noise. In this project, the architecture included three hidden layers with sizes 256, 512, and 1024. Each layer utilizes activation functions like 'tanh' and 'LeakyReLU' to introduce non-linearity into the model. The generator's input is a random vector sampled from a normal distribution, which allows it to generate diverse images.

Discriminator:

- The discriminator's role is to distinguish between real and generated images. It shared similar architectural features with the generator but was fine-tuned to improve its classification accuracy. The output of the discriminator is a probability score indicating whether the input image is real or fake. A successful discriminator is essential for guiding the generator's learning process.

4. Hyperparameters and Optimization

Several hyperparameters were adjusted during the training process:

- **Latent Size:** 100, allowing the generator to sample a diverse range of images.

- **Batch Size:** Varied across training iterations to find the optimal size for convergence. A typical batch size used in this project was 64, which strikes a balance between training speed and model performance.
- **Epochs:** The model was trained over multiple epochs to evaluate performance over time. The training lasted for 550 epochs to ensure that both the generator and discriminator could adequately learn from each other.

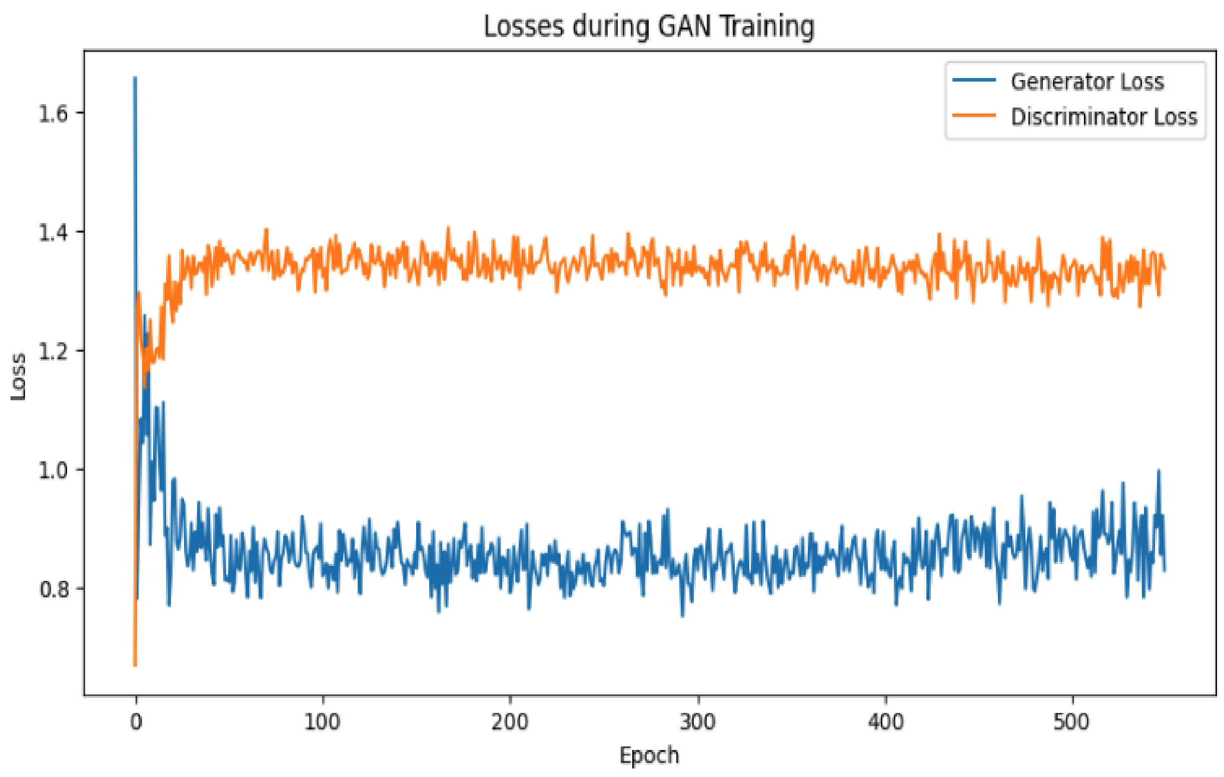
For optimization, stochastic gradient descent and various loss functions were explored, aiming to strike a balance between the generator and discriminator. The loss function used was the binary cross-entropy loss, which measures the performance of the classification model whose output is a probability value between 0 and 1.

4. Training Process

The training involved alternating between updating the generator and discriminator. Initially, the generator creates synthetic images from random noise. These images, along with real images from the MNIST dataset, are fed into the discriminator, which learns to classify them correctly. The generator aims to minimize its loss while the discriminator aims to maximize its accuracy. This adversarial process is crucial for the model to converge to a point where the generator produces realistic images.

5. Loss Plot

The following plot illustrates the loss values for both the generator and discriminator over 550 epochs:



[illegible]

Explanation:
- The plot shows the loss values for the generator and discriminator. A decreasing generator loss indicates that the generator is improving its ability to create realistic images, while fluctuations in the discriminator loss reflect its adjustment to the increasing quality of generated samples. Ideally, the losses should converge to a point where the discriminator is unable to distinguish between real and generated images.

6. Epochs Summary

During training, the model underwent several epochs. The generator's loss consistently decreased, indicating an improvement in the quality of generated images. The discriminator's loss fluctuated, reflecting its adaptation to the increasing complexity of generated samples. It is important to monitor both losses to ensure a balanced training process, as one model overpowering the other can lead to failure in the GAN's learning process.

7. Problem Statement

The primary challenge was to generate realistic images of handwritten digits that could successfully fool the discriminator. Achieving a balance between the generator and discriminator was crucial to preventing one from overpowering the other. Overfitting in either model could lead to poor generalization, making it essential to monitor performance metrics closely.

8. Conclusion

This project successfully showcased the implementation of GANs for image generation using the MNIST dataset. The models effectively generated recognizable digits, achieving the goals set out in the project objectives. The ability of the GAN to learn and replicate the characteristics of handwritten digits demonstrates the potential of generative models in various applications.

9. Future Scope

Future work could involve:
- Implementing more complex architectures like Convolutional Neural Networks to enhance image quality.
- Exploring different datasets to expand the application of GANs, such as CIFAR-10 or CelebA for generating realistic images of faces.
- Fine-tuning hyperparameters further to improve the generated images and exploring adaptive learning rates.
- Investigating advanced GAN techniques, such as Progressive Growing GANs or Cycle GANs, which allow for improved image resolution and quality.

10. Final Output Image Explanation

The final output images generated by the model illustrate various handwritten digits, demonstrating the effectiveness of the GAN in replicating the characteristics of the MNIST dataset. The generated images can be analyzed to assess the model's ability to generalize and produce high-quality outputs. Analyzing the images can reveal insights into the model's performance, including any biases it may have developed during training.