ASSIGNMENT REPORT:

Introduction:

This code implements a Generative Adversarial Network (GAN) for generating synthetic images from the MNIST dataset. GANs are a type of generative model that consists of two neural networks, a generator and a discriminator, trained in an adversarial manner. The generator's goal is to produce realistic fake images, while the discriminator's goal is to distinguish between real and fake images. Through this adversarial training process, the generator learns to generate increasingly realistic images.

Literature review:

https://arxiv.org/abs/1710.07035

https://www.youtube.com/watch?v=79IvwU3G5_Q&t=928s

https://github.com/zephyr2403/dcgan_MNIST

Design Decisions

1. Model Architecture: The code uses a multilayer perceptron (MLP) architecture for both the generator and discriminator networks. The discriminator consists of four linear layers with LeakyReLU activations and dropout regularization. The generator has a similar structure but with a Tanh activation in the output layer to produce images in the range [-1, 1].

2. Loss Function: The code uses the Binary Cross-Entropy with Logits Loss (BCEWithLogitsLoss) for both the discriminator and generator losses. This loss function is commonly used in binary classification tasks, such as the discriminator's task of distinguishing between real and fake images.

3. Data Preprocessing: The MNIST dataset is loaded and transformed to tensors using the torchvision library. The pixel values are scaled to the range [-1, 1] before feeding them to the discriminator to match the range of the generator's output.

Training Details

1. Dataset: The code uses the MNIST dataset, which consists of 60,000 training and 10,000 test grayscale images of handwritten digits (0-9).

2. Hyperparameters: The training is performed for 100 epochs with a batch size of 64. The Adam optimizer is used for both the generator and discriminator with a learning rate of 0.002. A dropout rate of 0.3 is applied to the hidden layers of both networks for regularization.

3. Training Loop: The training loop alternates between training the discriminator and the generator. For each batch of real images, the discriminator is trained on both real and fake (generated) images. The generator is then trained to generate fake images that can fool the discriminator into classifying them as real.

4. Evaluation: During training, the losses for the discriminator and generator are computed and stored for each epoch. Additionally, a fixed batch of latent vectors is used to generate and save fake images at the end of each epoch. These generated images can be used to visually evaluate the progress of the generator's learning.

Evaluation Metrics

GANs are typically evaluated using qualitative measures, such as visual inspection of the generated images. Since there is no explicit ground truth for the generated images, quantitative metrics like classification accuracy or mean squared error are not directly applicable.

In this code, the primary evaluation metric is the visual quality of the generated images. By displaying the generated images at different epochs, one can assess the generator's ability to capture the distribution of the real data and generate realistic-looking images.

Additionally, the discriminator and generator losses can provide insight into the training dynamics and convergence of the adversarial process. Ideally, the discriminator loss should decrease, indicating its ability to better distinguish between real and fake images, while the generator loss should increase, indicating its ability to generate more realistic fake images that can fool the discriminator.

Results: