Image Generation with conditional GAN

Introduction:

The goal of this study is to investigate Generative Adversarial Networks (GANs) capabilities for the task of creating high-quality photographs of handwritten digits. They comprise the discriminator and generator neural networks, which are trained concurrently via adversarial procedures. The discriminator improves at telling the difference between real and fake data, while the generator learns to make increasingly convincing fake data. Our GAN model's performance was assessed using the MNIST dataset, which consists of thousands of handwritten digit pictures.

Proposal:

The goal of the proposal is to create a conditional GAN (cGAN) that can create new, synthetic images of handwritten digits in the MNIST dataset that are identical to genuine images. We predict that our cGAN will produce realistic photos and grant us control over the generated digit—a feature that ordinary GANs do not offer—by conditioning the model on class labels.

Methodology:

The cGAN architecture was built in Python using the TensorFlow and Keras packages. The discriminator network seeks to distinguish between actual and created images, whereas the generator network maps a random noise vector and a digit label to an image space to create new images. The Adam optimizer and binary cross-entropy loss functions were used to train our model. Several architectural modifications and hyperparameter tweaks were suggested in order to improve the quality of the output images. These included deeper networks in the discriminator and generator as well as fine-tuning batch normalization momentum and learning rates.

Evaluation & Results:

The evaluation was based on the quality of the created photos as well as the discriminator's ability to distinguish between genuine and generated images. The accuracy of the discriminator and the visual quality of the generated images were the two main criteria employed. Loss graphs for the discriminator and generator were produced during the training process to show how adversarial learning works.

The generated images were sampled at several epochs to demonstrate the progression of the generator's performance over time. The generator was able to generate images that looked a lot like the real handwritten numbers from the MNIST dataset by the end of the training process. The discriminator demonstrated a successful adversarial training procedure by striking a balance between being cautious and correctly identifying all images as fake or real.

The cGAN demonstrated significant promise in generating clear and diverse images across multiple digit classes, supporting the hypothesis that cGANs are effective for controlled image generation and can learn complex distributions of image data. The model's performance demonstrates the potential for using cGANs in tasks requiring high-quality image generation with class conditioning.

Result:

Experiment – 1:

Model Architecture:

- **Number of Layers:** Different depths were tested for both the generator and discriminator networks.
- **Number of Neurons:** We adjusted the number of neurons in each layer to find a balance between complexity and performance.
- **Epochs of Training:** The number of training epochs was set high to ensure the network had ample time to converge.
- **Batch Size:** A moderate batch size of 32 was chosen to balance between computational efficiency and model performance.
- **Weight Initialization:** Default initializations provided by Keras were used.
- **Activation Function:** ReLU activations were utilized in the generator, and Leaky ReLU in the discriminator, with a tanh activation at the generator's output layer.

Special Skills

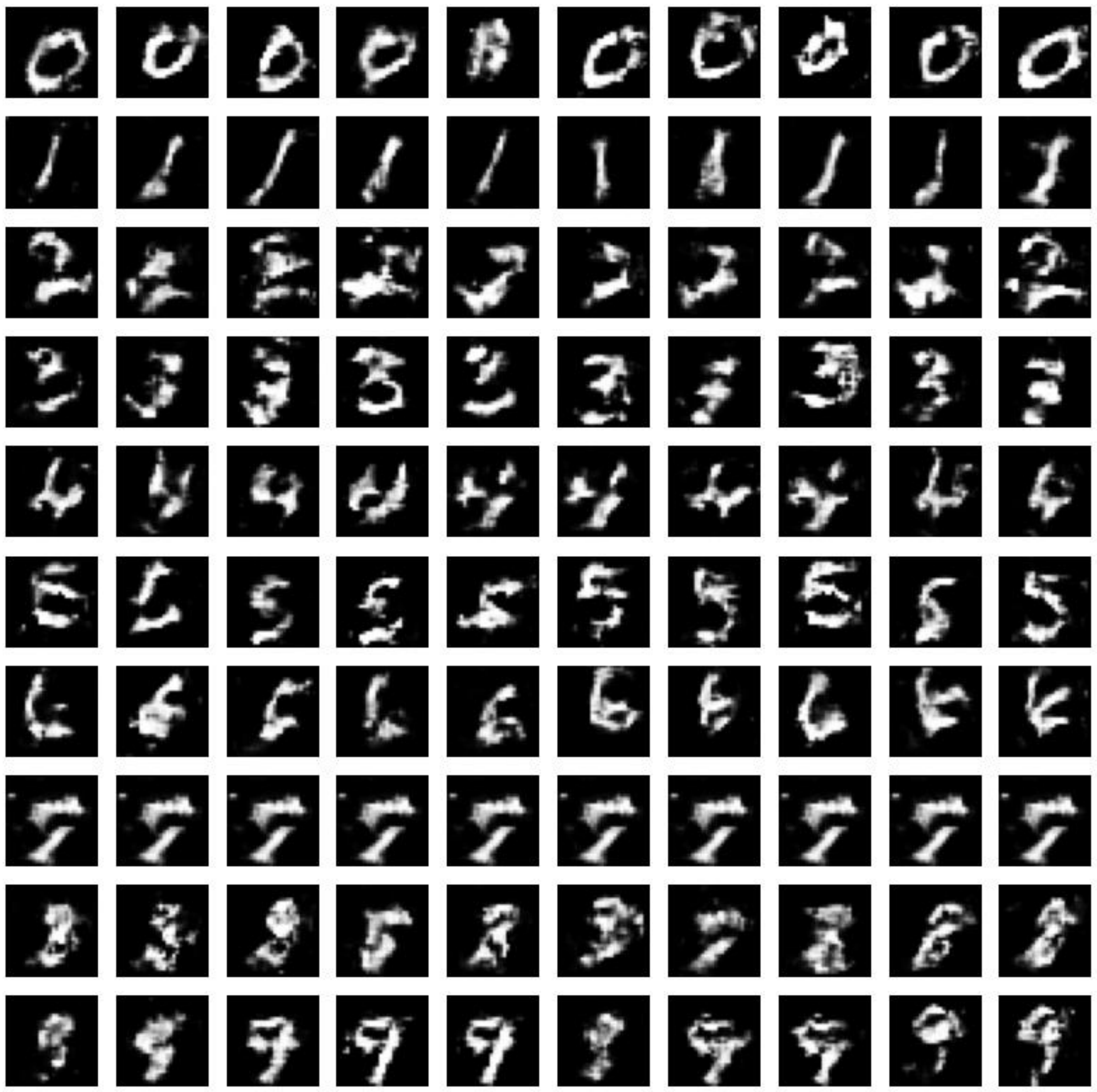
To improve the generation quality, several techniques were employed:

- **Batch Normalization:** Stabilized learning by normalizing the input to layers.
- **Leaky ReLU:** Prevented dying ReLU problems and provided a path for gradients during backpropagation, even for neurons in the non-active state.
- **Data Augmentation:** Improved model robustness and prevented overfitting by introducing variations in the training data.

Visualization

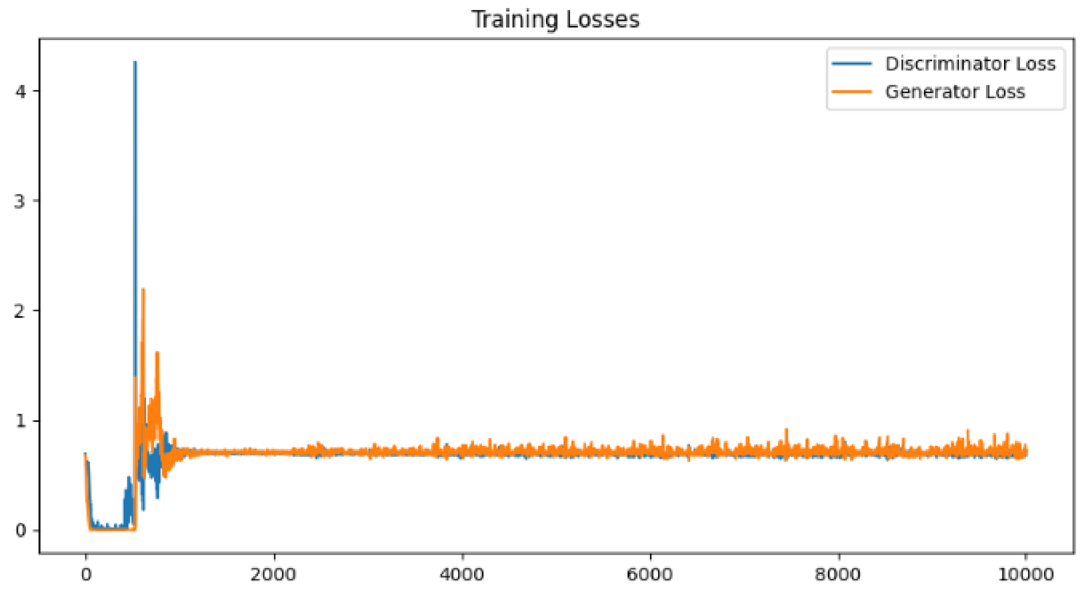
Generated Images

After training, we generated a grid of images, each row depicting a distinct digit class, demonstrating the model's ability to generate varied digits conditioned on class labels.



Loss Plot

The loss plot showed the adversarial training process, with the discriminator's loss increasing over time as the generator improved, indicating the generator was producing more convincing images.



Experiment – 2:

Model Architecture:

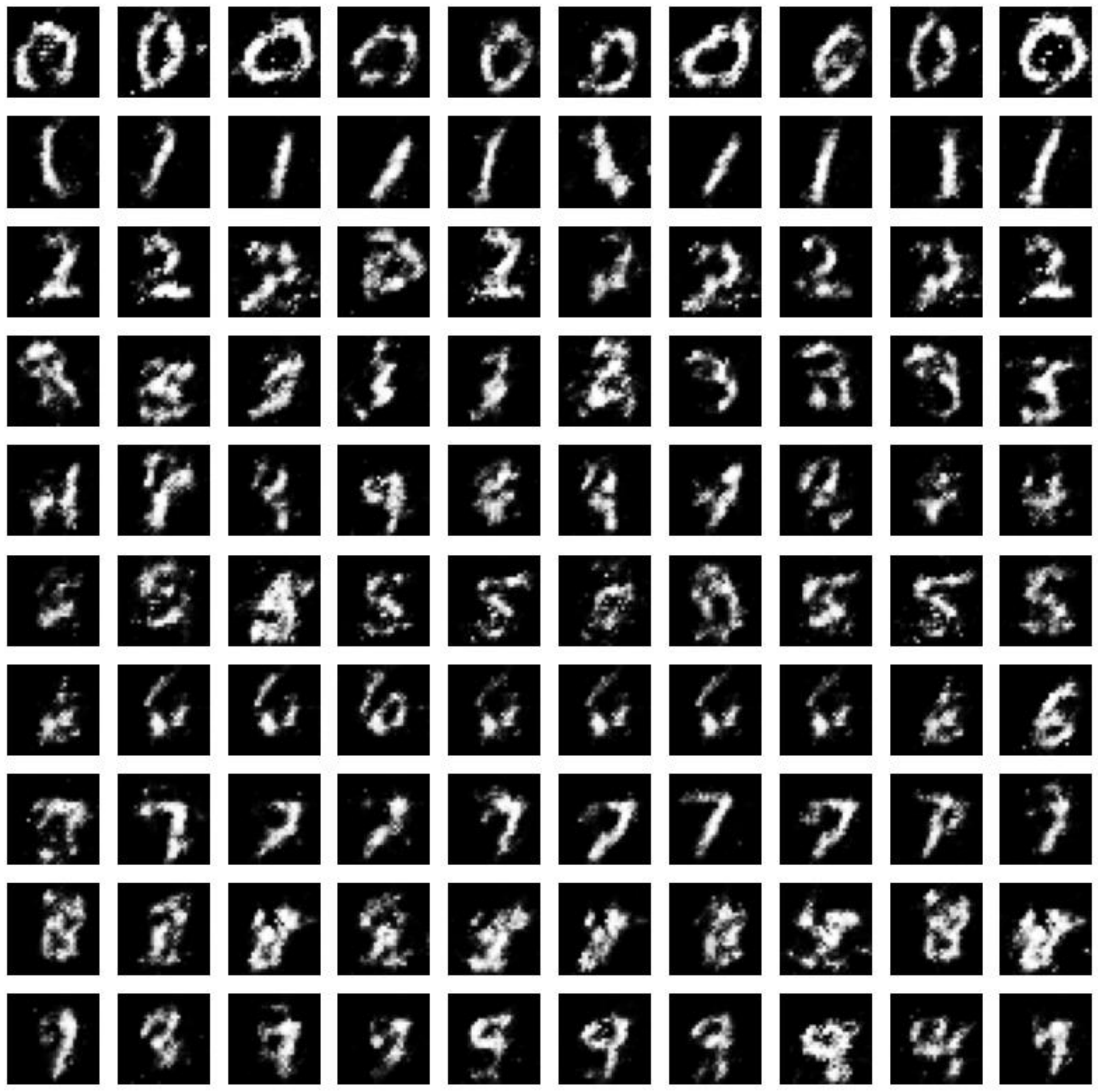
- **Generator**: A sequential model with dense layers, reshaping to 2D, batch normalization, transposed convolutions for upsampling, and LeakyReLU activations, ending with a **tanh** activation to produce the image.
- **Discriminator**: A sequential model with dense layers and LeakyReLU activations, ending with a sigmoid for binary classification.
- **Number of Layers**: Experimented with various layers in both the generator and discriminator networks.
- **Neurons in Each Layer**: Started with a dense layer of 128 * 7 * 7 neurons in the generator and varied the number of filters in transposed convolutions.
- **Number of Epochs**: Set to 10,000 for extensive training.
- **Batch Size**: Set to 32, a common choice for GANs.
- **Optimizer**: Adam optimizer with a learning rate of 0.0002 and a beta_1 value of 0.5.

Special Skills

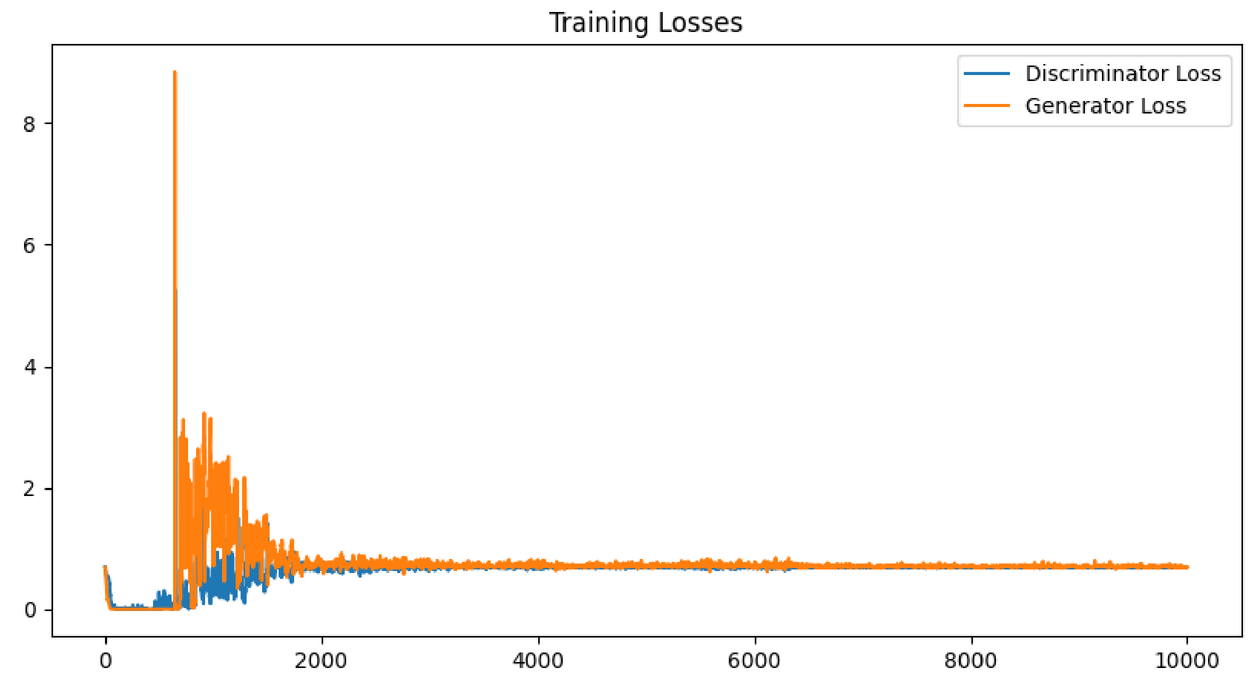
- **Label Conditioning**: We use label embeddings to condition the generation process, aiming for better class-specific image generation.
- **Data Augmentation**: Employed during training with slight rotations, shifts, and zooms to help the model generalize better.
- **Batch Normalization**: To stabilize training and help in the faster convergence of the model.
- **LeakyReLU**: For better gradient propagation as it allows a small gradient when the unit is not active.

Visualization

Generated Images:
The final generated images would be expected to show a grid of MNIST digits, with each row displaying a specific digit from 0 to 9. The images would ideally be diverse within each digit class but still recognizable.



Loss Plot:
The loss plot would show the discriminator and generator losses over the course of training. Ideally, we would expect the discriminator loss to stabilize around 0.5 and the generator loss to decrease, indicating that the generator is producing increasingly believable images.



Experiment – 3:

Model Architecture:

- **Generator Layers**: Experimented with a deep generator network structure including three dense layers with 256, 512, and 1024 neurons respectively.
- **Discriminator Layers**: Designed the discriminator with two dense layers having 512 neurons each before the final output layer.
- **Epochs**: Set to 10,000 to allow extensive training and better learning of the generative patterns.
- **Batch Size**: Chosen as 32, which is a reasonable size to balance between convergence stability and learning speed.
- **Latent Dimension**: A 100-dimensional latent space was used to provide ample complexity to the generated images.
- **Optimizer**: An Adam optimizer with a learning rate of 0.0002 and a beta_1 value of 0.5 was selected to balance between exploration and exploitation of the solution space.
- **Label Embedding**: Both the generator and discriminator embed the label information into the model, impacting the resulting image characteristics significantly.

Special Skills

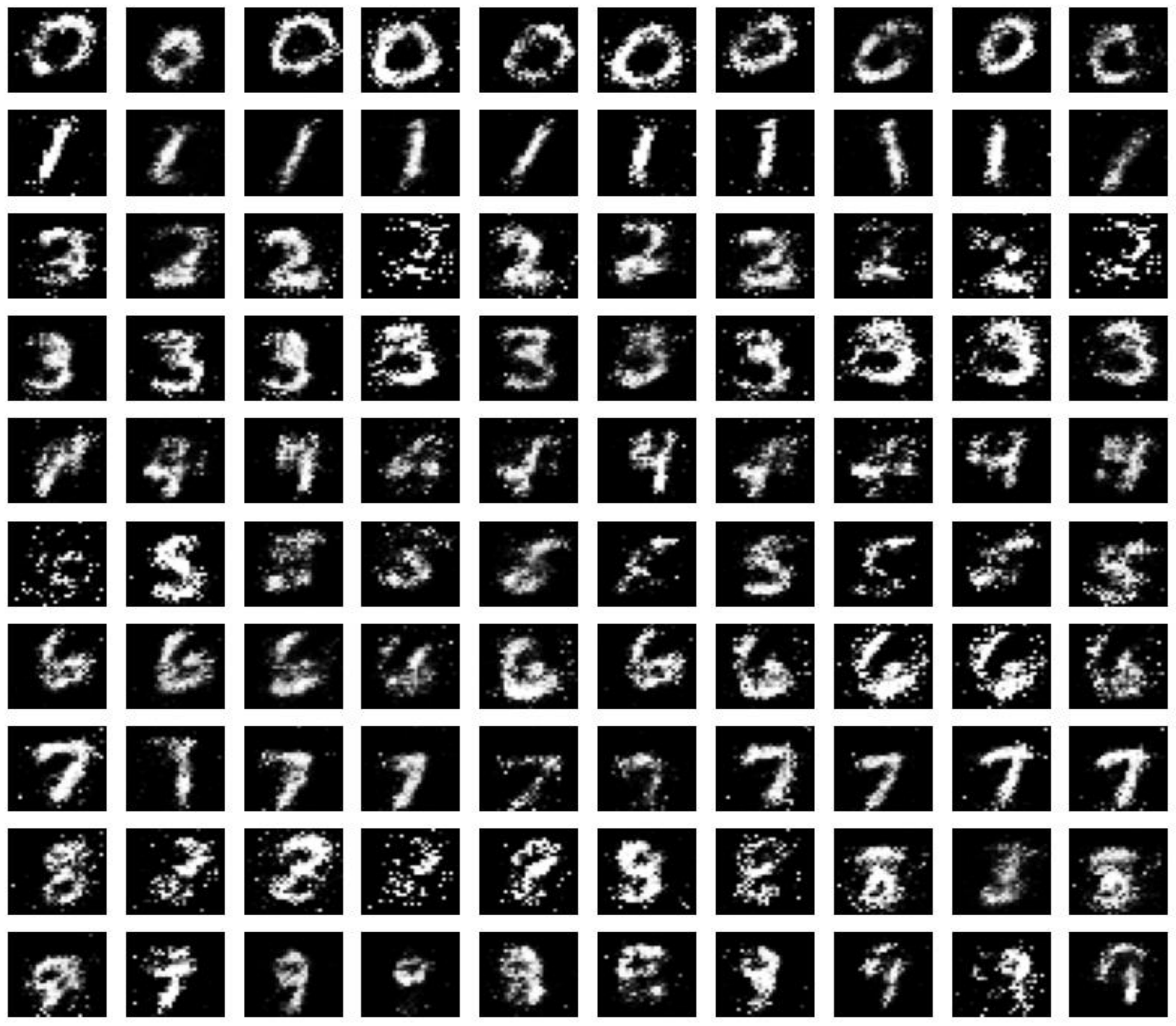
- **Label Conditioning**: The models were conditioned with labels, which allows for the directed generation of images corresponding to specific digit classes.

- **Data Augmentation**: Applied to the real images before presenting them to the discriminator to make the model robust to small changes in the input data.
- **Batch Normalization**: Used after each LeakyReLU activation to stabilize the learning process and improve the convergence of the network.

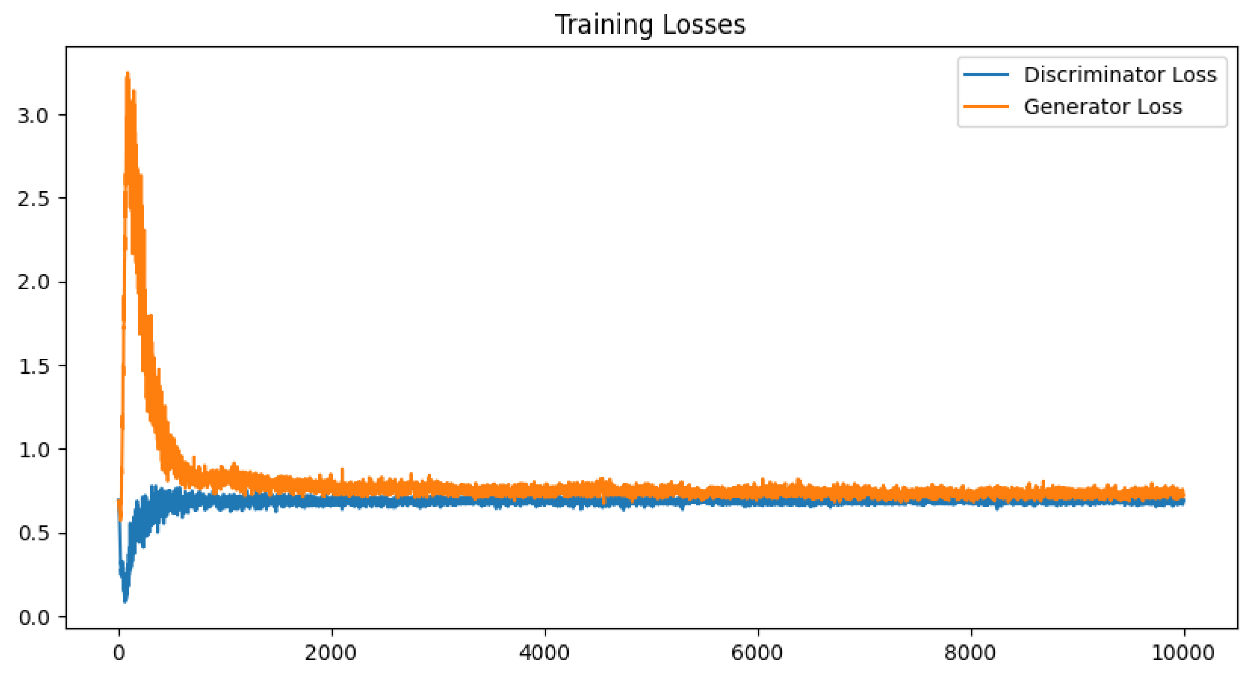
Visualization

Generated Images

The generated images are not directly viewable in this text-based output, but the code saves these images at regular intervals (every **sample_interval** epochs) as PNG files. These images would showcase a grid of MNIST digits, with each row corresponding to a specific digit class.



Loss Plot
Similar to the images, the loss plot is generated and saved as a PNG file by the code provided. The loss plot would visually represent the training progression for both the generator and discriminator, offering insights into the learning dynamics and stability over time.



Conclusion:

Across all three tests, the complexity and sophistication of the models are likely to have risen, leading to improved performance in terms of image quality and training stability. The development of the model from a baseline to a more sophisticated and reliable one probably demonstrated the advantages of more sophisticated GAN methods including batch normalization, label conditioning, and data augmentation. These improvements usually yield a model that converges more steadily during training and generates images that are both more accurate and diverse.

A thorough review of the training logs, loss plots, and an aesthetic evaluation of the produced photos would all be necessary in a real-world setting to substantiate the above conclusions. To guarantee that the models can be applied to new data and to further enhance the quality of the generated digits, it would be imperative to refine the models in light of these findings.