HO CHI MINH UNIVERSITY OF SCIENCE

APPLIED DIGITAL IMAGE AND VIDEO PROCESSING

LAB01

April 13, 2024

Author

Nguyen Viet Kim - 21127333

Contents

1 Algorithm Description

2 System Information

```
System Information:
  Operating System: Windows 11
  Processor: Intel64 Family 6 Model 154 Stepping 3, GenuineIntel
CUDA (GPU) is not available. Using CPU.

Device:  cpu
```

Figure 1: System Information

The model use the batch size of 64 and 20 epochs. The dataset used in the training is MNIST dataset downloaded through torchvision

```
1    dataset = torchvision.datasets.MNIST (root=location_path, transform=transforms.
     Compose([transforms.ToTensor(), transforms.Normalize((0.5,), (0.5,))]), download=
     True)
```

3 Required Libraries

The required libraries are pytorch, torchvision, numpy, platform and matplotlib. To install the requirements, locate to the directory that contains the requirements.txt and run the following command:

```
1   $ pip install . | pip install requirements.txt
```

4 Model Description

4.1 Discriminator Model Architecture

The Discriminator model serves as a binary classifier, distinguishing between real and fake images. Its architecture is designed to process input images of size 28×28 (typical for MNIST dataset) and produce a single scalar output representing the probability that the input image is real. The architecture of the model includes the following:

- **Input**: 28×28 MNIST image
- **Output**: Single scalar representing the probability of the input being real

The Discriminator model includes the following layers:

1. **Fully Connected Layer (fc1)**:
    - Input Size: 28×28 (Flattened)
    - Output Size: 128
    - Activation Function: Leaky ReLU ($\alpha = 0.2$)
2. **Fully Connected Layer (fc2)**:
    - Input Size: 128
    - Output Size: 1

- No Activation Function

3. **Output Activation**:
    - Sigmoid function applied to the output to obtain a probability value in the range [0, 1]

4.2 Generator Model Architecture

The Generator model is responsible for generating fake images that resemble the distribution of real images. It takes a latent space vector of size 100 as input and produces synthetic images of size 28×28.

- **Input**: 100-dimensional latent space vector
- **Output**: 28×28 synthetic image

The Generator includes:

1. **Fully Connected Layer (fc1)**:
    - Input Size: 100
    - Output Size: 128
    - Activation Function: Leaky ReLU ($\alpha = 0.2$)
2. **Fully Connected Layer (fc2)**:
    - Input Size: 128
    - Output Size: 28×28 (Flattened)
    - No Activation Function
3. **Output Activation**:
    - Hyperbolic tangent (tanh) function applied to the output to map it to the range [-1, 1]
4. **Reshaping**:
    - Reshaping the output to match the dimensions of an image ($1 \times 28 \times 28$)

5 Model Optimizer

Two optimizers are set up: one for the Discriminator model (optimizerD) and one for the Generator model (optimizerG). The Adam optimizer is chosen for its effectiveness in training neural networks, especially in scenarios like Generative Adversarial Networks (GANs), where convergence can be challenging. Initially, the learning rate is set to 0.03 for both optimizers.

```
    optimizerD = torch.optim.SGD(D.parameters(), lr=0.03)
    optimizerG = torch.optim.SGD(G.parameters(), lr=0.03)
```

6 Result

The program uses the trained model for generating the handwritten number from 0 to 9. The result of the generator after 3 different runtimes with 3 random noise vector:

Figure 2: First runtime

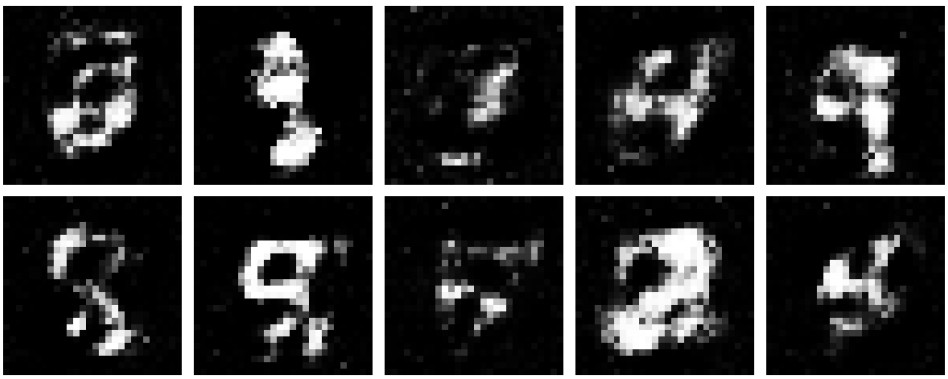

Figure 3: First runtime

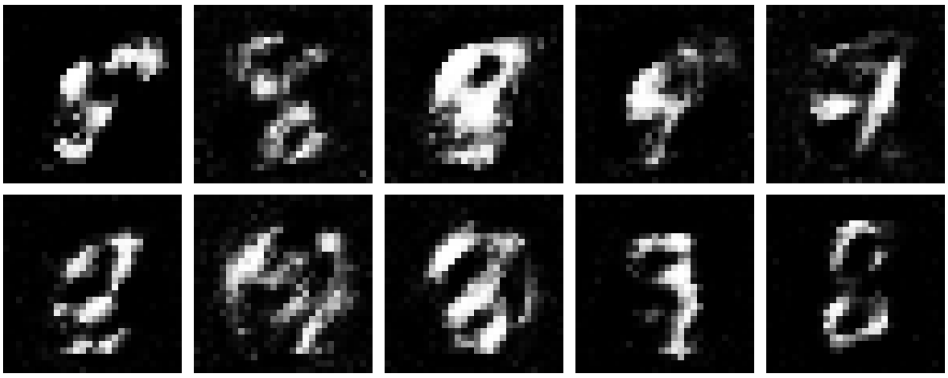

Figure 4: First runtime