Naan Mudhalvan – Generative AI course

Sree Vaasini S
2021503562
Batch 2

Abstract

Generative Adversarial Networks (GANs) have emerged as powerful tools for generating realistic synthetic data, particularly in the domain of computer vision. GANs consist of two neural networks, a generator and a discriminator, engaged in a minimax game. The generator learns to produce data that is indistinguishable from real samples, while the discriminator learns to differentiate between real and fake data. This adversarial training process results in the generator producing increasingly realistic samples over time.
This code implements a Generative Adversarial Network (GAN) using TensorFlow and Keras to generate images resembling handwritten digits from the MNIST dataset. The GAN consists of a generator and a discriminator network trained adversarially. The generator creates images from random noise, while the discriminator learns to distinguish between real and generated images. The training loop iterates over a specified number of epochs, during which the generator and discriminator are trained alternatively. Periodically, images generated by the generator are saved to disk for visualization. The code demonstrates how to build, train, and evaluate a basic GAN architecture for image generation tasks using TensorFlow and Keras.

Dataset

The dataset consists of grayscale images of handwritten digits, along with their corresponding labels.

Description of Work Implementation

1. Data Loading: loading the MNIST dataset using TensorFlow's built-in functionality

2. Data Preprocessing:

   a. The pixel values of the images are normalized to the range [-1, 1] to facilitate training. The images are also flattened into 1D arrays to be compatible with the input requirements of the neural networks.

3. Generator Architecture:

   a. The generator network is defined using Keras, with multiple densely connected layers. ReLU activation functions are used for intermediate layers, with batch normalization to stabilize training.

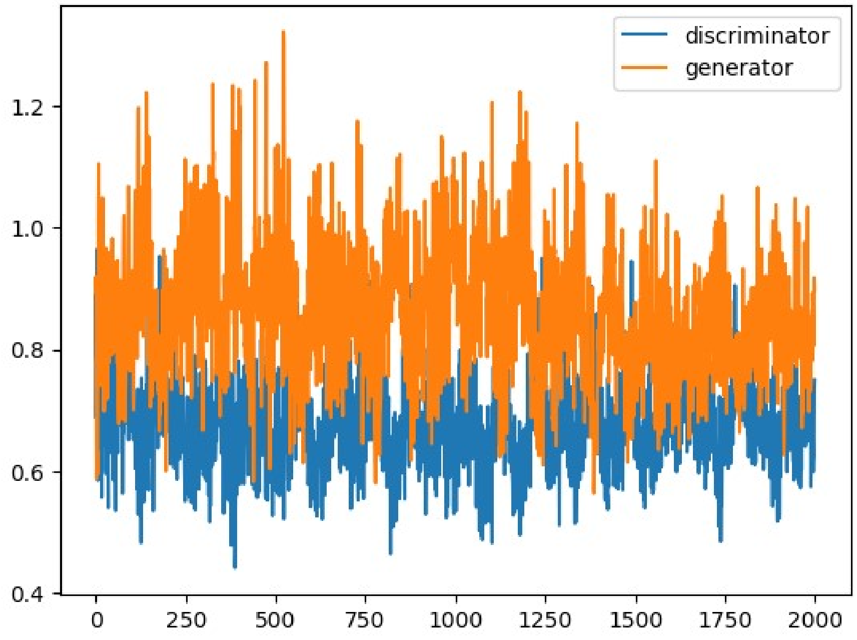
   b. The output layer uses the hyperbolic tangent (tanh) activation function to generate images in the range [-1, 1].

4. Discriminator Architecture
    a. the discriminator network is constructed using Keras, consisting of densely connected layers with Leaky ReLU activation functions.
    b. The output layer uses a sigmoid activation function to classify images as real or fake.
5. Training Setup:
    a. The discriminator is compiled with the binary cross-entropy loss function and trained to distinguish between real and fake images.
    b. The generator is trained by combining it with the frozen discriminator, effectively training the generator to fool the discriminator. The combined model is compiled with the binary cross-entropy loss function.
6. Training Loop:
    a. The GAN is trained using a loop that iterates over a specified number of epochs. In each epoch, the discriminator and generator are trained alternately, with batches of real and fake images.
    b. Periodically, the generator generates sample images, which are saved to disk for visualization.
7. Visualization:
    a. The training progress is monitored by plotting the discriminator and generator losses over epochs. sample images generated by the generator at specific epochs are displayed to observe the improvement in image quality over the course of training.

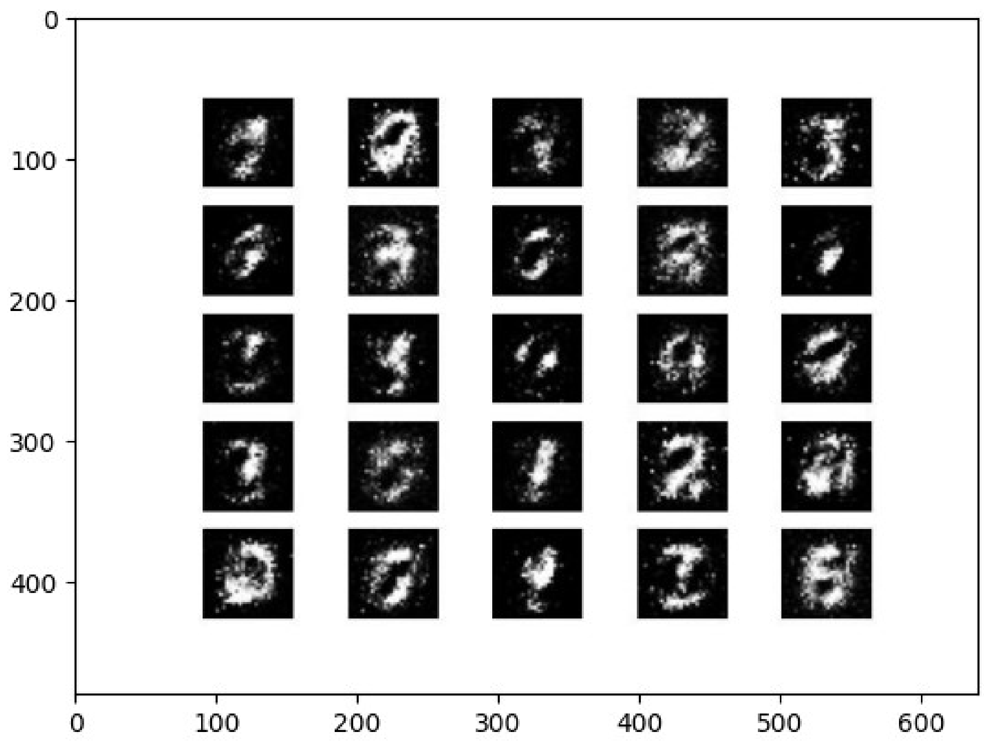
Conclusion

In conclusion, the code demonstrates the implementation of a GAN for generating synthetic handwritten digits. By training on the MNIST dataset, the GAN learns to produce realistic images that closely resemble the dataset's original samples, showcasing the effectiveness of GANs in generating high-quality synthetic data.

Output screenshots:



```
[54] <matplotlib.image.AxesImage at 0x78971f02ab60>
```


0
100
200
300
400
0
100
200
300
400
500
600