Implementation of Vanilla GAN for MNIST Handwritten digits dataset

GitHub link: *https://github.com/prasadsriram01/GAN.git*

Introduction:

GAN, short for Generative Adversarial Network, is a machine learning technique that has gained tremendous attention and popularity in recent years due to its ability to generate realistic and novel content, such as images, videos, and even text. GANs consist of two neural networks: a generator network that produces new data instances, and a discriminator network that evaluates the authenticity of these instances. These two networks are trained simultaneously, with the generator trying to produce data that can fool the discriminator into thinking that it is real, while the discriminator tries to correctly distinguish between the real and generated data.

GANs have a wide range of applications, including image and video synthesis, data augmentation. They have also been used in the fashion and design industry for generating new clothing designs and in the gaming industry for creating realistic game characters and environments. In addition, GANs have been applied in medical imaging for generating high-resolution images and in the field of natural language processing for generating realistic text. In my work I have used the vanilla GAN architecture to generate MNIST dataset. MNSIT dataset is a database of handwritten digits with 60000 images. The model performed well in generating the text and the performance of the model was good as it could able to generate realistic images.

Method:

A total of 4132 images of size 28 X 28 is used as a dataset for training. The model has two neural networks, generator, and discriminator. The input to the discriminator is the images from the training set, and the generated images from the generator and output is the probability. The input to the generator is a random noise and it uses convolutional, up sampling layers to change the single dimensional array of size 100 to 28 x28 2D array representing an image. Tensor flow framework was used to build GAN and the model was trained in the Jupiter notebook.

General Problems associated with GAN:

1. Mode collapse: In GAN training, the generator might learn to produce only a limited set of outputs, ignore Mode collapse: In GAN training, the generator might learn to produce only a limited set of outputs, ignoring the diversity of possible outputs, which is called mode collapse.
   Fig.1. shows the mode collapse problem when trained to generate aeroplane image with various hyperparameters.

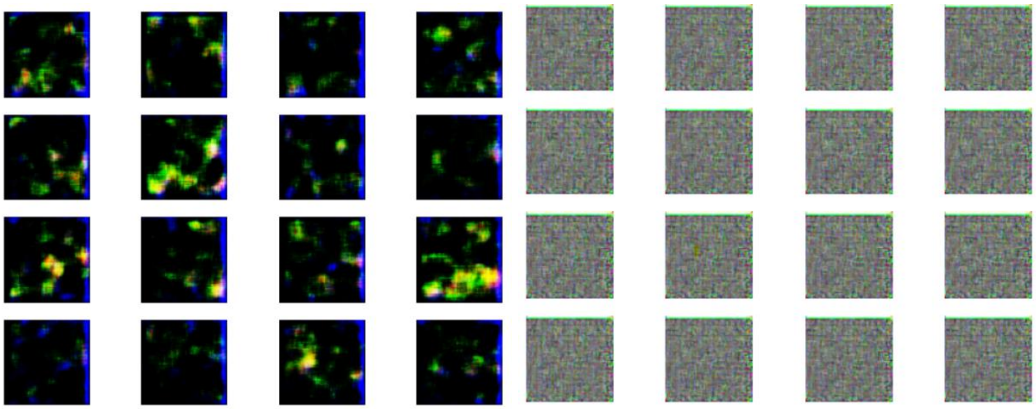

Fig.1.Mode collapse problem when trained to generate aeroplane image with various hyperparameters.

2. Training instability: GAN training is notoriously unstable, and it can be difficult to find the right hyperparameters to achieve good results.
3. Evaluation: It can be challenging to evaluate the quality of GAN-generated data. There is no universal metric that can determine how realistic or diverse the generated data is.
4. Computationally expensive: Training GANs is computationally expensive and can require significant computing resources.
5. Gradient vanishing: Gradient vanishing can occur when the gradients used to update the generator become too small, making it difficult for the generator to learn.
6. Difficulty with high-resolution images: GANs can struggle to generate high-resolution images, as the generator must produce a large number of pixels that are coherent with one another.
7. Difficulty with small datasets: GANs require a large amount of data to train effectively. If the dataset is too small, the GAN may not be able to learn the underlying distribution effectively.

HYPER PARAMETER FOR GENERATOR AND DISCRIMINATOR	VALUE
Loss	Binary cross entropy
optimizer	Adam
Learning rate	0.0002
Epochs	300
Batch size	600

Fig.2. Image Dataset Used for Training GAN and Hyperparameters used

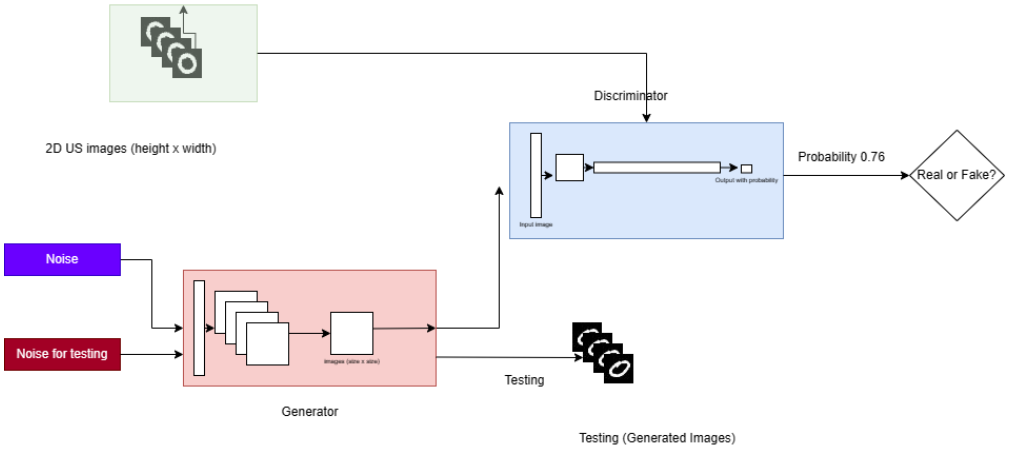

Fig.3. GAN Architecture

Results and Discussion:

The generated image looked exactly realistic, and the generated image is shown in Fig.4. Few of the pixels were not properly generated. Gaussian filter is applied as a post processing method to the generated image. Fig.5. shows the histogram of real and synthetic images and the distribution of both real and synthetic image exactly matches. The entropy value of real image is 3.62 and that of generated image is 5.05. The difference in entropy is very less between real and synthetic image.

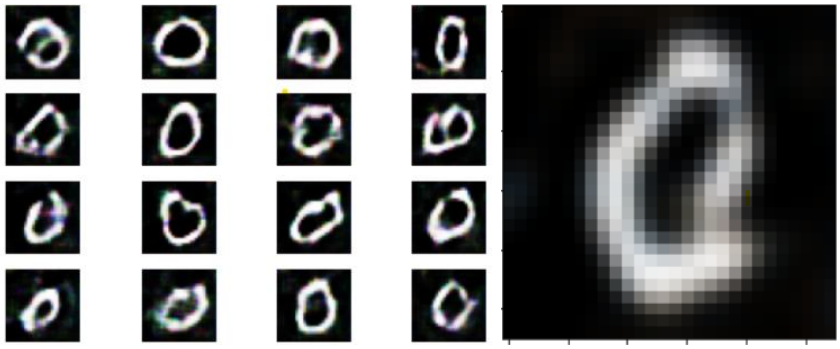

Fig.4. Generated image before and after applying filter.

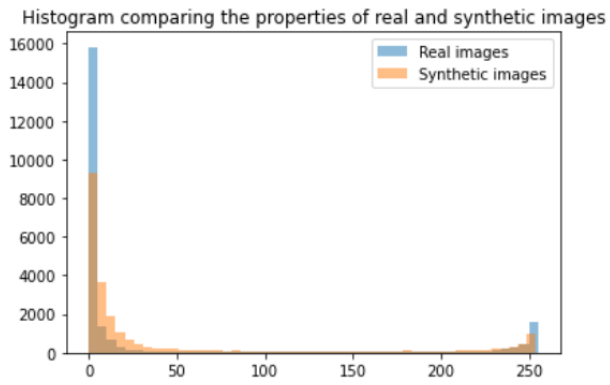

Fig.5. Histogram plot of real and synthetic image