MNIST Digit Generation Using GAN

09.17.2023

—

By

Parmar Tarun

Introduction & Proposal

Generative models are some of the popular models in the Machine Learning field. A well trained generative model, can generate very good quality images, grammatically correct sentences and even well tuned sounds. In this project, I have used the GAN Model. GAN includes two networks which is one of the reasons why it is so powerful in generating high quality images. The first network is a generator and the second one is a discriminator. The generator's job is to generate various images from a low dimensional vector and the discriminator tries to label them as fake or real images. Both of the networks are trained while keeping the other fixed. Discriminator is trained with fake images label close to zero and real images close to 1.

Methodology

1. Network Structure

The network built for the project is a multilayer perceptron network. To build a generator network I have used keras's sequential model. The first layer, which is a dense layer, consists of 256 neurons with input size of 100 considering the size of vectors from normalized distribution, followed by an activation layer of type leaky relu with alpha as 0.3. Next the network is appended with a dense layer of 512 neurons and another leaky relu activation layer with the same alpha. For the final hidden layer total neurons are set to 1024 with a leaky relu activation function of 0.2 alpha. The output layer of this network is a dense layer of 784 units considering the 28 x 28 pixels of image as output of this model.

For the discriminator, the first hidden layer consists of 1024 neurons with input dimension of 784 considering the output of the generator followed by Leaky relu and dropout layers. The same is repeated for another 3 times with decreasing order in neurons. For the final output layer, it consists of 1 neuron considering the scalar output with sigmoid activation considering the labels being between 0 and 1.

Keeping the above network as a baseline, I have modified this baseline network and made it deeper with more hidden layers. The generator was appended with two more layers and discriminator was prepended with two layers

In both the approaches the models were compiled using the Binary cross entropy as loss function, adam as optimizer and accuracy as metric.

2. Training & Validation process

The MNIST Dataset is loaded directly using the keras library. The dataset includes a total of 70000 data points with 60000 splitted for training and 10000 for testing purposes. The testing data and labels in the dataset are irrelevant for generative models. Hence, only the training images were used without the labels.

Next, the pixel values were converted to range of 0 to 1. The image matrix was also converted to vectors as the discriminator takes input as 784. For the base model the epochs were set to 100 and for the deeper model the epochs were increased to 200. Batch size was constant with the value of 128.

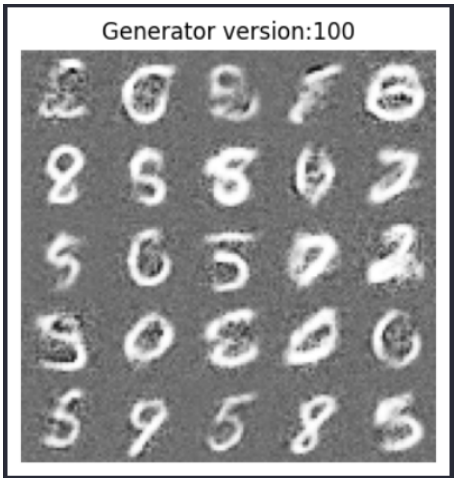
Evaluation & Results

I. Training & Validation Results

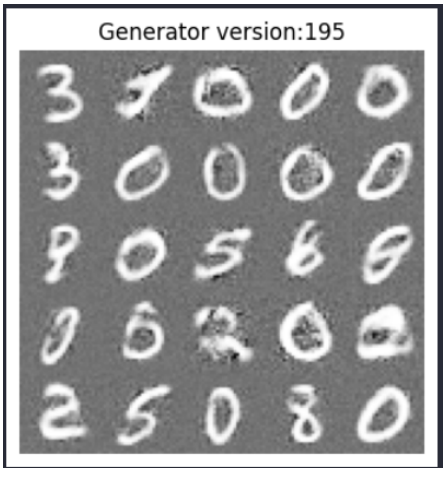
Given that the GAN models are complex models. Base model took about an hour on kaggle's gpu and more than a couple of hours for the deeper model. With the base model the generated digits were recognizable however a bit blurry. For the deeper the model the blurriness was reduced.

II. Performance Comparison

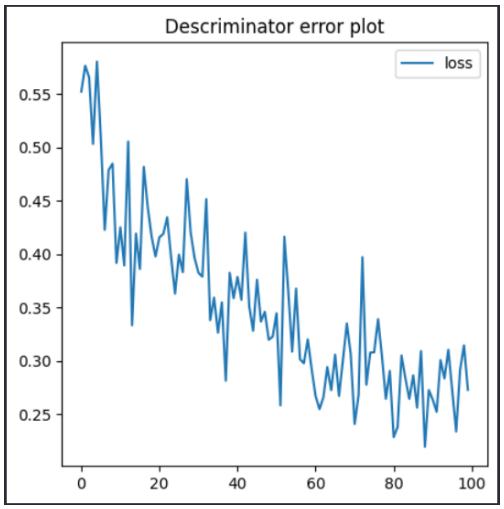
The following figures depicts the comparative results of both the models used in this project.



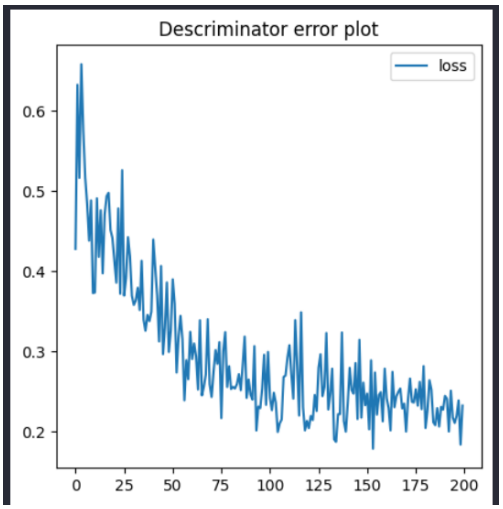
BASE MODEL



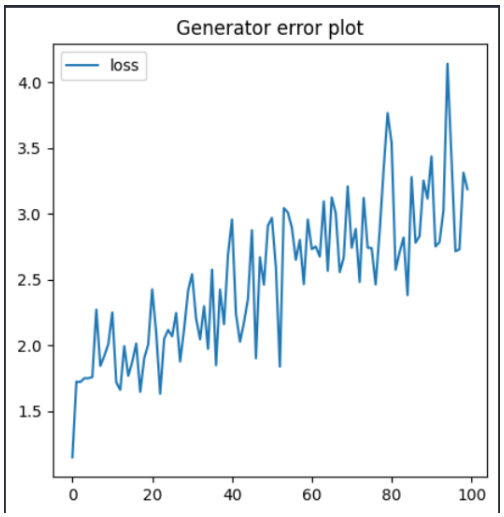
DEEPER MODEL



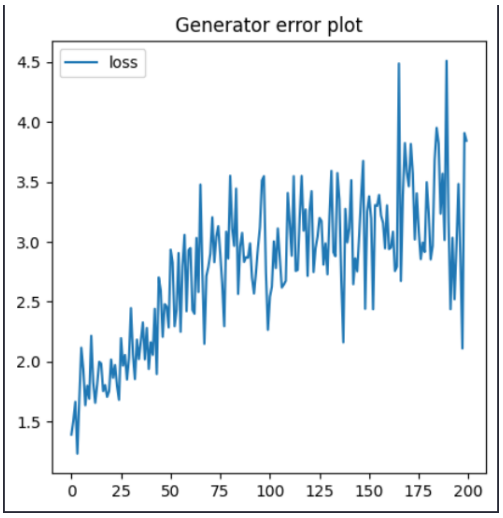
Base Model



Deeper Model



Base Model



Deeper Model

III. Hyperparameter

The following hyperparameters were used / tweaked while training process of the models:

- ***Number of layers***: Hidden layers were increased from 3 in base model to 5 in the deeper model.
- ***Dropout***: A constant dropout of 0.3 was used in the discriminator of both the models. The 0.3 dropout indicates that neurons have 0.3 probability of dropping out during the training process.
- ***Activation function***: The activation function for hidden layers for both models was leaky relu. The activation function for the output layer of the generator was tanh and for the discriminator, softmax in both the models.
- ***Number of epochs***: Different epochs' values were used in the models in an experiment to get better results than the previous one.

Conclusion

The GAN model with a deeper neural network resulted in some good quality image generation. Generative models such as GAN for generating handwritten digit images proved to be a good choice in developing a better model.

With less hidden layers in the base model outputted some blurry digits however the increase in the hidden layers in the next model with larger epochs really helped the model to generate better results.