Implementing OnkoGan: Bangla Handwritten Digit Generation with Deep Convolutional Generative Adversarial Networks

Abrar Rafid Noor
170204059
Dept. of CSE, AUST

Md. Sakib Irtiza
170204081
Dept. of CSE, AUST

Labib Abdullah
170204114
Dept. of CSE, AUST

March 8, 2022

1 Introduction

Unsupervised machine learning incorporates Generative Adversarial Networks (GANs) [1], which are implemented using two neural networks. GANs have a generator that creates fake images and a discriminator that distinguishes between actual and fake images. We attempted to use OnkoGAN's DCGAN (Deep convolutional generative adversarial networks) [2] model to create Bangla handwritten digits from random noise in this project.

Furthermore, we tried to modify their model architecture to generate better results than the previous model.

2 Motivation

Bangla is the native language of the Bengal region of South Asia, which is comprised of two countries: Bangladesh and West Bengal, India. Apart from this, Bangla is also spoken in Assam, Tripura, and the Andaman Islands. This language ranks sixth among the world's most widely spoken languages. Bangla is spoken by around 30 million people. Bangla is also Bangladesh's official language. It is rooted to our ancestors. That is what inspired us to work on a particular aspect of this lovely language: generating its numbers.

The ability to generate better Bangla handwritten digits can be utilized to expand its dataset, which can then be used for future research and development of Bangla AI applications. The ability to have alternative digit styles will further improve the dataset in terms of variation.

3 Methodology

3.1 Abstract of the Model

Our model is built on two convolutional neural networks. One is the generator, the other is the discriminator. Generator takes a noise vector and tries to generate fake sample and sends it to the discriminator. Discriminator is trained on both real and fake samples, it learns to separate if a given sample is a real or fake. Based on the loss calculated, appropriate gradients is given as feedback, so that they can update their parameters. After many epochs of training, the generator learns to generate samples similar to the real ones.

We applied a technique called Spectral Normalization [3] to avoid mode collapse. This technique is used for stabilizing the discriminator, resulting in having more variations and better outputs.

3.2 Model Description

We tried to implement the exact model architecture of OnkoGAN [4] following its paper. The model's generator is a multilayer convolutional neural network. The generator starts with a fully connected dense layer with 1024 hidden node with an input size of 100. Then the output is batch normalization

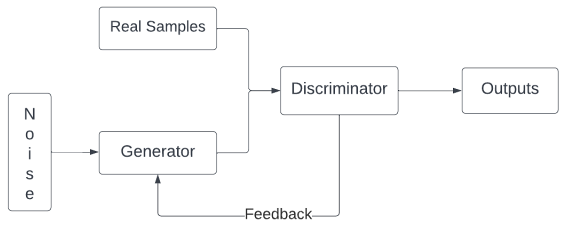

Figure 1: Abstract Architecture of the Model

layer. Again, pass with a ReLU activation. Later the output is connected to another dense layer of 6272 hidden neurons with a batch normalization and ReLU activation. Then we upsampled the output with a tuple of 2 and pass it to a convolutional layer with the size of 64 filters, 2×2 kernel, and same padding and added batch normalization and ReLU activation. And then, again upsampled the output with the size of 2 and passed it in a new convolution with a single filter, 2 × 2 kernel with tanh activation.

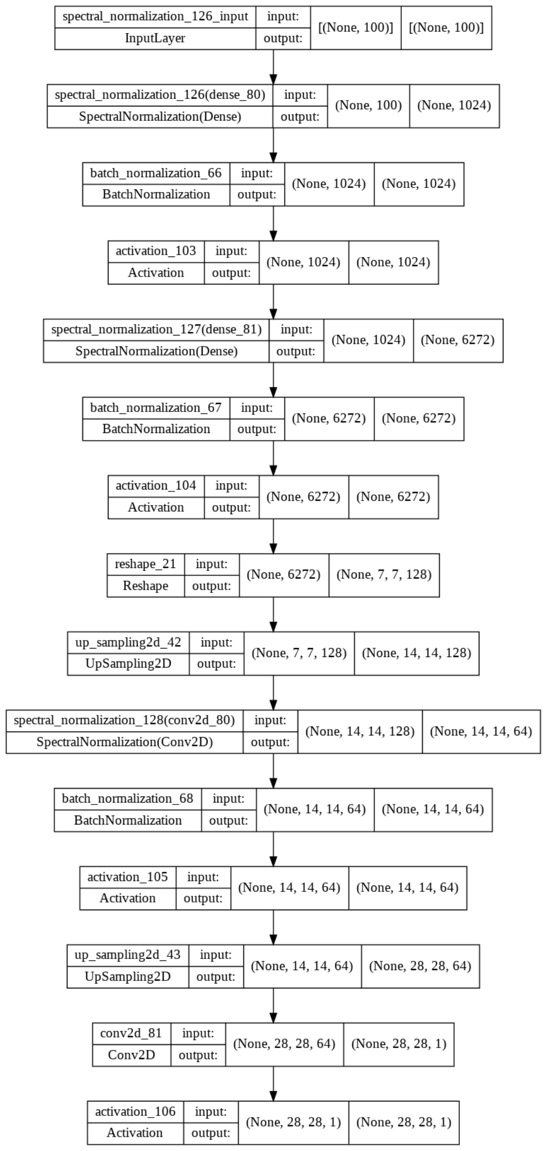
spectral_normalization_126_input | InputLayer | input: [(None, 100)] | output: [(None, 100)]
spectral_normalization_126(dense_80) | SpectralNormalization(Dense) | input: (None, 100) | output: (None, 1024)
batch_normalization_66 | BatchNormalization | input: (None, 1024) | output: (None, 1024)
activation_103 | Activation | input: (None, 1024) | output: (None, 1024)
spectral_normalization_127(dense_81) | SpectralNormalization(Dense) | input: (None, 1024) | output: (None, 6272)
batch_normalization_67 | BatchNormalization | input: (None, 6272) | output: (None, 6272)
activation_104 | Activation | input: (None, 6272) | output: (None, 6272)
reshape_21 | Reshape | input: (None, 6272) | output: (None, 7, 7, 128)
up_sampling2d_42 | UpSampling2D | input: (None, 7, 7, 128) | output: (None, 14, 14, 128)
spectral_normalization_128(conv2d_80) | SpectralNormalization(Conv2D) | input: (None, 14, 14, 128) | output: (None, 14, 14, 64)
batch_normalization_68 | BatchNormalization | input: (None, 14, 14, 64) | output: (None, 14, 14, 64)
activation_105 | Activation | input: (None, 14, 14, 64) | output: (None, 14, 14, 64)
up_sampling2d_43 | UpSampling2D | input: (None, 14, 14, 64) | output: (None, 28, 28, 64)
conv2d_81 | Conv2D | input: (None, 28, 28, 64) | output: (None, 28, 28, 1)
activation_106 | Activation | input: (None, 28, 28, 1) | output: (None, 28, 28, 1)

Figure 2: Generator Architecture of our Modified Model

Their proposed DCGAN's discriminator starts with a convolutional neural network with a size of 64 filters, 5 $\times$ 5 kernel, and same padding. The layer used LeakyReLU activation. The second layer used 128 filters with a 5$\times$5 kernel with LeakyReLU activation. The output later connects with a dense layer with 256 hidden units with 50% dropout. The final output layer has one node with sigmoid activation which gives the probability of the images being real or fake.

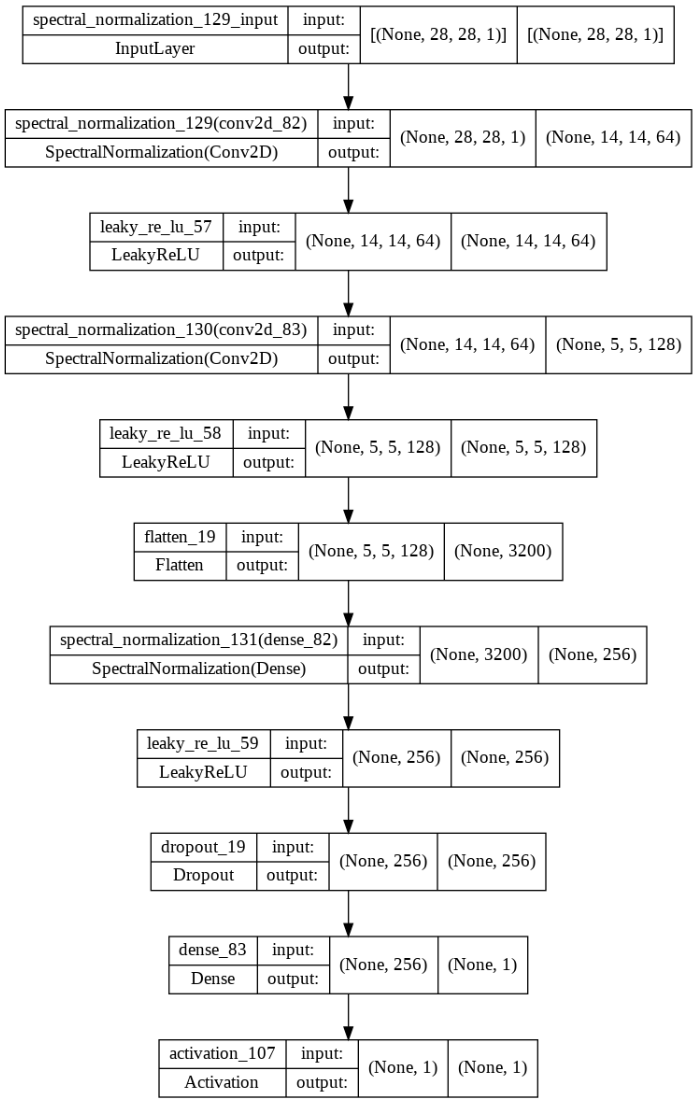

Figure 3: Discriminator Architecture of our Modified Model

4 Experiments

4.1 Dataset

The proposed method of DCGAN used 2 different datasets.

- BanglaLekha-Isolated Dataset
- Ekush Dataset.

4.1.1 Collection

BanglaLekha-Isolated Dataset has been collected from Mendeley Data's website and Ekush Dataset has been collected from their website.

4.1.2 Preprocessing

Both of the datasets were comprised of Bangla letters and digits. We extracted only the digits part for our project. Ekush Dataset was colleceted as a csv formatted file, which was able to be used directly in the training. On the otherhand, Banglalekha-Isolated Dataset was collected as folders containing images. They were all in gray-scale, but had different sizes. So they were resized into 28x28. Data from both of the datasets were normalized before training.

4.1.3 Statistics

Ekush Dataset had a total of 30830 digits.

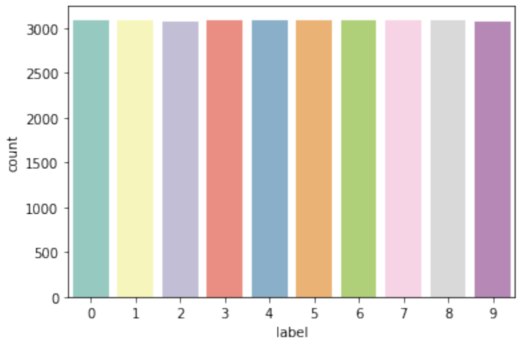

Figure 4: Frequency of Digits of Ekush Dataset

For BanglaLekha-Isolated, on average all digits had balanced number of examples, on average around 1960.

4.1.4 Description

BanglaLekha-Isolated datasets contain 19748 images where edge are looking smooth. The images are in grayscale and inverted where background as black and digit wrote as white.
All images are 28×28 pixel while preserving the aspect ratio of the images as well as edges looks smooth.

4.2 Performance Metrics

Results produced by GANs do not have a certified performance metric in the literature that can accurately measure its performance. However, the loss function vs iteration graph of the generator and discriminator models can say a lot about the generated outputs. If the loss of discriminator is greater than the loss of generator, or both the losses becomes similar-like during training, it generally produces a good output. And as the outputs are images, it can also be evaluated by humans ourselves.

4.3 Evaluation

4.3.1 Performance of OnkoGAN Model-like Architecture

The loss of the model after training with Ekush Dataset is given below:

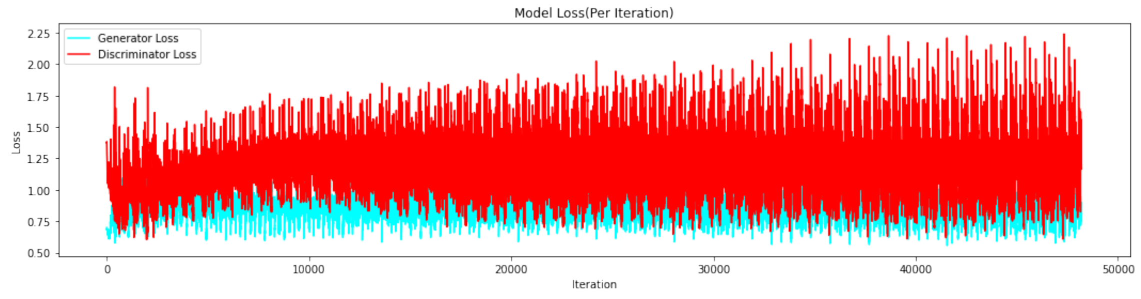

Figure 5: Loss vs Iteration Graph for Ekush Dataset

The loss of the model after training with BanglaLekha-Isolated Dataset is given below:

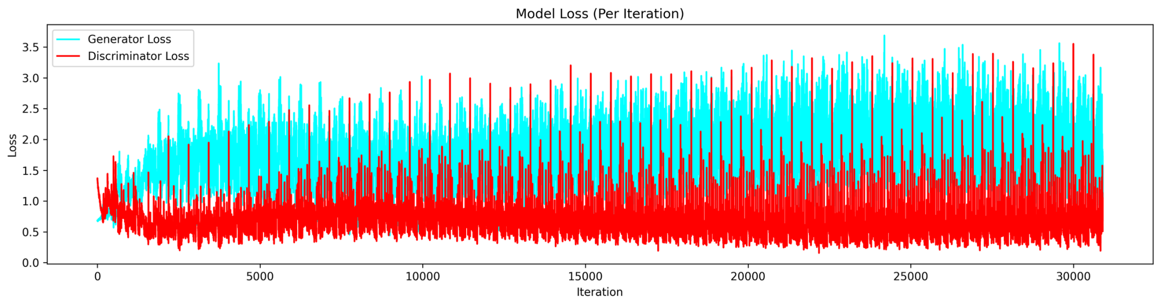

Figure 6: Loss vs Iteration Graph for BanglaLekha-Isolated Dataset

4.3.2 Performance of our Proposed Model Architecture

The generated outputs after training with Ekush Dataset is given below:

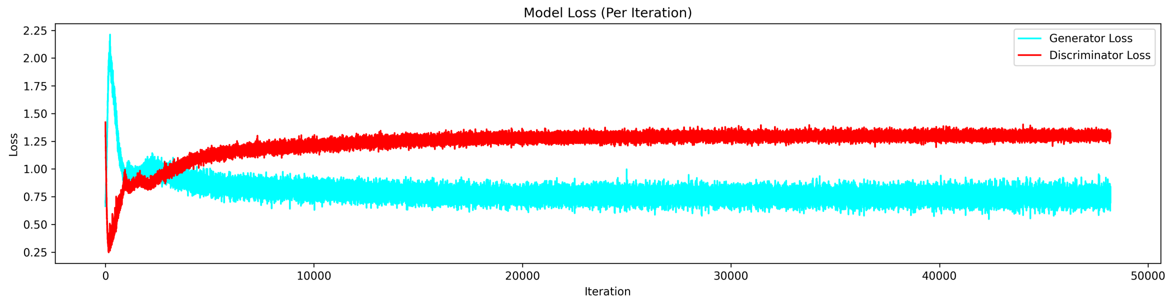

Figure 7: Loss vs Iteration Graph for Ekush Dataset

The generated outputs after training with BanglaLekha-Isolated Dataset is given below:

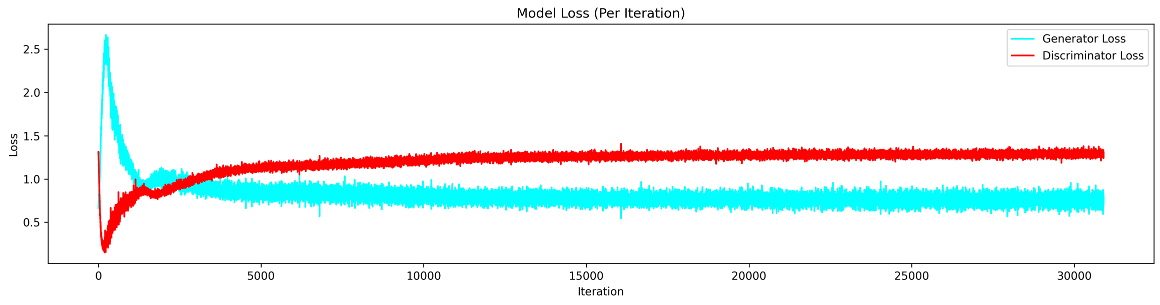

Figure 8: Loss vs Iteration Graph for BanglaLekha-Isolated Dataset

4.3.3 Comparison with OnkoGAN

OnkoGAN paper output, our OnkoGAN model-revisited architecture's output and our modified Onko-GAN model architecture's output is shown in the following:

For Ekush Dataset:

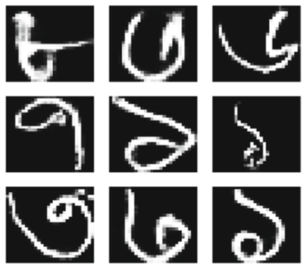

Figure 9: Output of paper

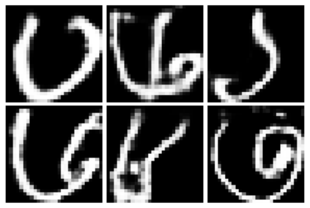

Figure 10: Output of OnkoGAN-like model

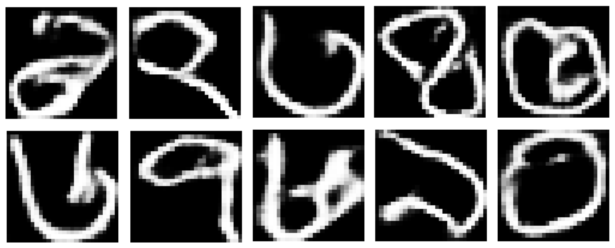

Figure 11: Output of revised OnkoGAN model

For BanglaLekha-Isolated Dataset:

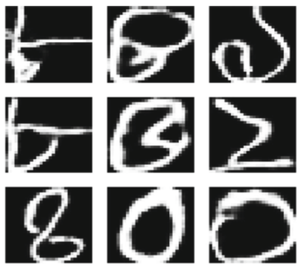

Figure 12: Output of paper

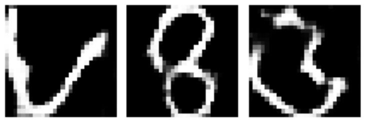

Figure 13: Output of OnkoGAN-like model

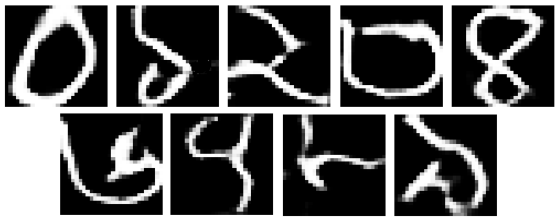

Figure 14: Output of revised OnkoGAN model

Comparison table for the discriminator loss obtained is given below:

Table 1: Comparison of Loss for Each Dataset

OnkoGAN model	**Discriminator(min) Loss**		
	Paper	***Revisited***	***Modified***
Ekush	0.56648755	**0.53098259**	0.59937197
BanglaLekha-Isolated	0.624247	0.578476	**0.4447885**

4.3.4 Discussion

Even though our images don't look as sharp as OnkoGAN paper's, we however reduced the minimum discriminator loss and also removed mode collapse. Furthermore we illustrated in figure 5 and 6 that the generator successfully persuaded the discriminator. We are intending in future to introduce encoders which will encode original images, which we will utilize as noise for our GAN model, resulting in better output inclining towards the reality.

References

[1] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio, "Generative adversarial nets," *Advances in neural information processing systems*, vol. 27, 2014.

[2] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," *arXiv preprint arXiv:1511.06434*, 2015.

[3] T. Miyato, T. Kataoka, M. Koyama, and Y. Yoshida, "Spectral normalization for generative adversarial networks," *CoRR*, vol. abs/1802.05957, 2018.

[4] S. Haque, S. A. Shahinoor, A. Rabby, S. Abujar, and S. A. Hossain, "Onkogan: Bangla handwritten digit generation with deep convolutional generative adversarial networks," in *International Conference on Recent Trends in Image Processing and Pattern Recognition*, pp. 108–117, Springer, 2018.