Generating Handwritten Digits using DCGAN

Jayesh Betala - 210712854

School of Electronic Engineering and Computer Science
Queen Mary University of London, UK

Github Link for Public Access: https://github.com/jbetala7/DCGAN-MNIST.git
ec21989@qmul.ac.uk

Abstract: The experiment in this paper showcases the importance of the generation of handwritten digits as a benchmark task in the field of computational creativity. GAN networks in the past have been known to produce real-like sound images given a set of target data distribution. Using the MNIST dataset, we first train the generator and discriminator models to build the Deep Convolutional Generative Adversarial Network which then generates new data of handwritten digits. It was observed that the proposed structure of the network achieves equivalent behavior compared to other state-of-the-art models like VAE.

1 Introduction

Generative Adversarial Network also known as GAN was first implemented and designed by Ian J. Goodfellow and his acquaintances in October 2014. They established and described the network for assessing different generative models by using adversarial processes. The key feature of the framework is to simultaneously train the generator that captures data from the defined dataset while the discriminator estimates the probability of the data. Moving ahead in process, the generator's task is to recreate the training data and confuse the discriminator between the generated output and the distribution of the training data. Therefore, GANs are highly used for the purpose of generating real-like fake images.

Inspired from various models of GAN the experiment described in this paper uses the MNIST dataset to process the real images from the database and compare it to the outputs generated by our proposed DCGAN framework that has been implemented using TensorFlow. The adversarial process allows the generator to learn to produce realistic images while the discriminator learns to differentiate between real and fake images, till the point the model can no longer distinguish between both and make confident decisions. At the end, we describe some more implementations for future work using the proposed DCGAN architecture.

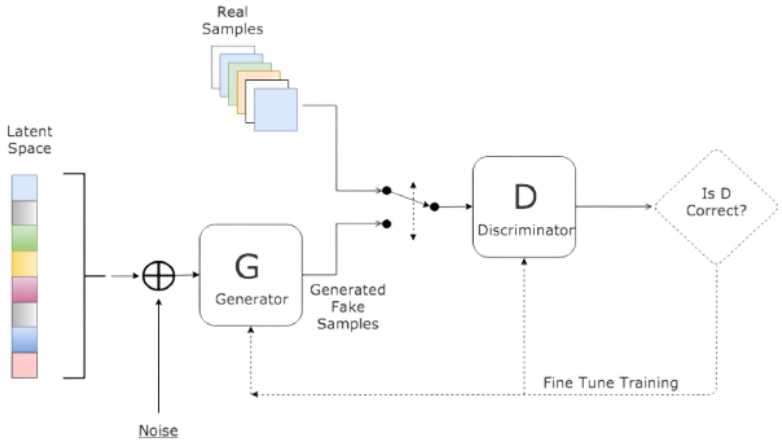

Fig. 1: GAN Structure

2 Background

Among variety of generative models for image generation, VAEs and GANs are the two main state-of-the-art models used in computer vision. The VAE system is an artificial neural network that belongs to the family tree of probabilistic graphical models and is used to optimize the lower bound. It was typically made for the purpose of unsupervised learning but because of its effectiveness it has been proven to be useful even in some of the machine learning domains. The GAN system as discussed above consists of a generator and a discriminator. Variants of GAN was created to improve the performance and the stability in different domains such as the decision tree controller GAN proposed by Takuhiro et al., the system is capable of learning hierarchically representations without the need of any supervision. Another GAN model proposed by Gregor et al. suggests using a spatial attention mechanism for the generation of images one section at a time and has been defined as a DRAW network as the model's workflow is like how a human would create a drawing. Various recurrent GANs have been used in the past to generate images that are almost identical to the real images used as input, however those models and networks are beyond the scope of our experiment and has been declared here just for the purpose of some background knowledge.

3 Dataset

The Mixed National Institute of Standards and Technology dataset, also know as MNIST contains a large collection of handwritten digits. The database consists of a training set with 60,000 examples and a testing set which contains 10,000 examples. The training set has a buffer size of 60,000 and a batch size of 256 which is required to calculate the shape of an image of size 28x28 to fit into a pixel bounding box.

Like this dataset, the US Postal also known as USPS handwritten dataset consists of a rather small number of samples. The training and the test set has respectively 7291 and 2007 samples.

Lack of samples in USPS dataset made the decision easier for us to choose between the two datasets, and therefore the MNIST dataset was chosen for the experiment mentioned in the paper.

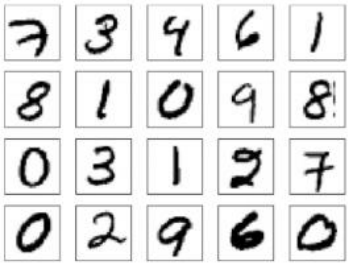

Fig. 2: Sample Image from MNIST training dataset

4 System Description

Our DCGAN network has a well-defined CNN architecture which is basically based on few design rules:

- The network structure includes both pooling and convolutional layers. Just like the default convolutional network, DCGAN too adopts the nature to remove all the pooling layers and is replaced by the convolutional layers. The pooling layers in the discriminator are replaced by a strided convolutional while the pooling layers existing in the generator are taken over by fractional-strided convolutions.
- The network has a deep neural network which slows down the operation speed. To avoid this issue a proposed solution adapted by the fully connected layer existing in the model is to directly connect the random input of the generator and the out layer of the discriminator with the input feature and the output feature of the convolutional layer, respectively.
- The network using batch normalization can efficiently solve the issue of having many layers which results the output data to change its distribution and gradually as the layers in the neural network increase the divergence of the output data becomes larger too.
- The network's generator and discriminator both uses separate activation functions. While ReLU function along with Tanh activation function for the output layer are used in the generator, the discriminator uses just LeakyReLU for all the layers.

The improvements mentioned above therefore allows DCGAN to give comparable performance like GAN or simple CNN.

5 Experiments and Results

The experiments were conducted using the handwritten digit database existing in the MNIST dataset. The network was trained for 3000 epochs which took about 6 hours on Google Colab's Pro version. After each epoch 16 images of handwritten digits are generated by the generator model. Based on the feedback given by the discriminator model the generator model keep improvising the outputs. The generator model learns around 2.3 million parameters to produce a fake image.

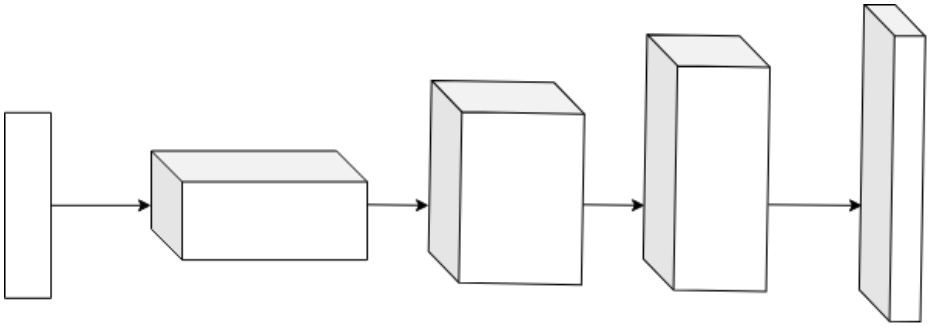

Fig. 3: Generator network

Using TensorFlow Keras the discriminator model is created which takes images of size 28x28x1 as input from the MNIST dataset as well the generator model and then performs discrimination to decide whether the generated image is real or fake.

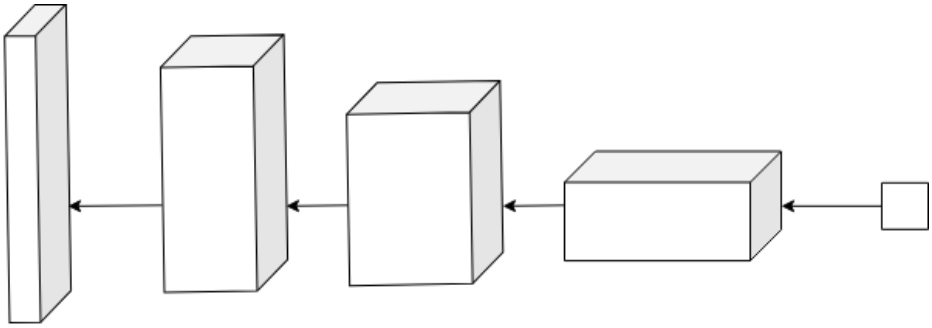

Fig. 4: Discriminator network

Adam optimizer and two different functions are used separately by both generator and discriminator model to calculate the loss. From the images below we can see the progress and the performance of our model which produces spectacular real-like images barely indistinguishable from the original dataset. Therefore, it proves that the framework of DCGAN if not the best, but still relatively stands among and can be compared with other state-of-the-art performing models.

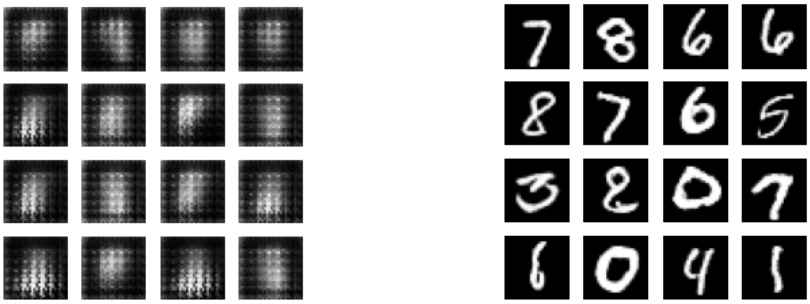

Fig. 5: Epoch 0 (left) & Epoch 3000 (right)

The experiment at the end showcases the results by creating 10 random samples and merges them together to create a gif of the generated images produced by our DCGAN framework.

6 Discussion and Higher-Level Issue

Discussing the results, it was observed that by 400 epochs the network starts producing images displaying digits which shows quick progress in the training process. The digits obtained produce lesser noise and the network even learns variants of the same number. In the figure below we can see three different outputs of the same number '4'. The discriminator loses confidence as the training continues because the generator gradually starts producing more indistinguishable images.

Fig. 6: Generated image at epoch 400

On the contrary, only few experiments have been implemented to produce handwritten digits or characters in different languages. This is because there exists only one dataset called MNIST-MIX which was made available to public in 2020. Therefore, most of the experiments showcasing the use of DCGAN for generating handwritten artifacts is based on the default MNIST database consisting only English

language. This might discourage many individuals from different ethnic backgrounds to perform and generate handwritten artifacts because of lack of resources available for the language they wish to use for their experiments.

While observing the results produced by some of the experiments using the MNIST-MIX dataset, it can be concluded that those experiments are definitely more enjoyable to perform because of the variety of results produced in multiple languages and would even lead to better research methods and stronger DCGAN networks having additional parameters and functions.

7 Conclusions and Future Work

The experiment shows that even small size of training data is enough to achieve state-of-the-art performance in the recognition of handwritten digits. The efficient and effective performance is a result of DCGAN using upgraded parameters and convolutional layers which improves the performance of the GAN.

The code cells in the Colab notebook have been properly commented to explain each step included in th process of generating images. The Colab notebook has been uploaded to GitHub and everything has been made open source for public access and feedbacks for improvements.

For future work there are multiple ways that can proposed to improve the network and generate even better artifacts. Them being:

- LTSMs have already proven to outperform DCGAN for image synthesis tasks. RNNs and LTSMs are known for using selective attention mechanism and generate images in small pieces,
- Using a more complicated network than proposed in the experiment performed in this paper, it is possible to generate doodles using the Sketchy database.
- Using the MNIST-MIX dataset for training DCGAN, interesting artifacts in different languages consisting not just handwritten digits but even handwritten characters can be generated.

References

1. Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A. and Bengio, Y., 2014. Generative Adversarial Networks. [online] arXiv.org. Available at: <https://doi.org/10.48550/arXiv.1406.2661> [Accessed 16 April 2022].
2. Hitawala, S., 2018. Comparative Study on Generative Adversarial Networks. [online] arXiv.org. Available at: <https://doi.org/10.48550/arXiv.1801.04271> [Accessed 16 April 2022].
3. Park, H., Yoo, Y. and Kwak, N., 2018. MC-GAN: Multi-conditional Generative Adversarial Network for Image Synthesis. [online] arXiv.org. Available at: <https://doi.org/10.48550/arXiv.1805.01123> [Accessed 16 April 2022].

4. Radford, A., Metz, L. and Chintala, S., 2016. Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. [online] arXiv.org. Available at: <https://doi.org/10.48550/arXiv.1511.06434> [Accessed 16 April 2022].
5. Reed, S., Akata, Z., Yan, X., Logeswaran, L., Schiele, B. and Lee, H., 2016. Generative Adversarial Text to Image Synthesis. [online] arXiv.org. Available at: <https://doi.org/10.48550/arXiv.1605.05396> [Accessed 16 April 2022].
6. Doi.org. 2020. ShieldSquare Captcha. [online] Available at: <https://doi.org/10.1088/2633-1357/abad0e> [Accessed 16 April 2022].