RES MILITARIS
REVUE EUROPEENNE D ETUDES EUROPEAN JOURNAL OF MILITARY STUDIES
Social Science Journal

Generation of "Hand-Drawn" Images Using Deep Convolutional Generative Adversarial Networks

By

Dr. Kiran Mayee Adavala
Professor in CSE Vidya Jyothi Institute of Technology Hyderabad
Email: Kiranmayee@research.iiit.ac.in

Abstract

Generative Adversarial Networks are nowadays being used for generating new and life-like images from datasets of old images. This paper studies the usefulness of a more improved algorithm called Deep Convoluted Generative Adversarial Networks for the generation of hand-drawn images using the case study of cartoon images. A dataset of cartoon images from a cartoon dataset is used for testing the model and the quality of the generated synthetic images are found to be good.

Introduction

Hand-drawn art images exist everywhere, in the form of abstract art or drawings, as well as cartoon characters. Generative Adversarial Networks (GANs) are a type of artificial intelligence model that can generate new data that is similar to a given dataset. The new images created look exactly like the class of images that they are derived from, fooling even experts at times. Thus, GANs can be used to create synthetic data that mimic the actual almost to perfection. GANs are today applied in Text, music and Image generation, Data Augmentation, Video prediction and Fashion design. Cartoons are considered as hand-drawn versions of realistic objects with very small variations that determine that they are cartoons and not real-life things. They are drawn in 2 Dimensions and this is the major reason for their non-conformance to photographed images. Most work on GAN has so far focused on photographs of actual things as data sets. Take the example of the MNIST Fashion dataset. The dataset is a set of monochrome photographs of clothes of different types such as jerkins, t-shirts, jeans and so on, all with image size 28x28x1. The GAN model generates new type of clothes from these old ones that look realistic. This paper focuses on a specific case of hand-drawn figures – cartoons, to study the applicability of DCGANs in the field of hand-drawn images. We discuss GAN and DCGAN in sections 3 and 4. Some of the research done so far in the field of synthetic image generation is discussed in section 2. Section 5 focuses on the dataset used to test the DCGAN model. Preprocessing and the remaining steps are discussed in methodology in section 6. Section 7 discusses the results.

Literature Survey

Goodfellow et al. [5] propose the concept of GANs for the first time describing two sub-networks, one that generates new data from old and another that tries to discover the synthetic data. Radfold et al. [6] present a stronger version of GAN that uses Deep Convolutional Neural Networks called DCGAN. Han Xiao et al. [1] introduce Fashion MNIST dataset, which consists of 70,000 grayscale images of clothing items from 10 different categories. The authors train several different machine learning models on the dataset, including GANs, and find that it is a useful benchmark for evaluating their performance. Han Zhang et al. [2] propose a new GAN architecture called FashionGAN, which is specifically

designed for generating images of fashion models wearing clothing. The authors train their model on the Fashion MNIST dataset and achieve state-of-the-art results in terms of image quality. Yunsheng Li et al. [3] present an improved version of the FashionGAN architecture, called FashionGAN++, which incorporates multi-branch attention mechanisms to improve image quality. The authors evaluate their model on the Fashion MNIST dataset and achieve state-of-the-art results. Shuang Li et al. [4] propose a novel adversarial training approach for generating high-quality images of clothing items using GANs. The authors apply their method to the Fashion MNIST dataset and achieve state-of-the-art results in terms of image quality and diversity.

DCGAN is applied in various diverse fields. Fang et al. [7] use DCGAN for developing a new gesture recognition algorithm. Dewi et al. [8] generate synthetic traffic signals for prevention of accidents. Wu et al. [9] propose a method using DCGAN augmentation to identify leaf disease in tomato. Bushra et al. [10] demonstrate the use of DCGAN in converting hand-drawn sketch images to face transformations. Speaker identification based on DCGAN is discussed in [11]. Kim et al. [13] generate pedestrian dataset using DCGAN. An et al. [14] perform encryption in end-to-end communication systems using GAN for generating the key. Paper [15] uses a linear weighted fusion method to reduce the interference of defects in detection of fabric defects. An attempt is made in [16] to generate synthetic medical images using DCGAN. Shi et al. [17] determine the quality of corn ear using DCGAN. Xu et al. [18] use augmentation to detect Parkinson's disease. Pradhan et al. [19] have applied DCGAN to re-create traditional cloth designs. Li et al. [20] create new avatars from the old using DCGAN.

Generative Adversarial Networks (GAN)

The concept of GANs [5] was first introduced by Ian Goodfellow and his colleagues who published a paper titled "Generative Adversarial Nets" in which they proposed a novel approach to generative modeling. The idea behind GANs is to have two neural networks, a generator network, and a discriminator network, that compete with each other. The generator network learns to generate new data that is similar to the training data, while the discriminator network learns to distinguish between real and fake data. The two networks are trained simultaneously, with the generator trying to fool the discriminator by generating data that is similar to the training data, and the discriminator trying to correctly identify the real data from the fake data.

3.1 Principles

The principles behind GANs are based on game theory, specifically on the concept of a minimax game. In a minimax game, there are two players, a maximizer, and a minimizer. The maximizer's goal is to maximize a certain value, while the minimizer's goal is to minimize the same value. In the case of GANs, the generator network acts as the maximizer, trying to generate data that maximizes the discriminator's error rate, while the discriminator network acts as the minimizer, trying to minimize its error rate. The two networks are trained simultaneously, with the generator trying to generate data that is similar to the training data, while the discriminator tries to distinguish between real and fake data.

3.2 Training Process

The training process of GANs involves the following steps -

1) Randomly sample a noise vector from a normal distribution. This vector is the input to the generator network.
2) Feed the noise vector into the generator network, which generates a new data point.

3) Feed the real data point and the generated data point into the discriminator network.
4) Calculate the error rate of the discriminator network for both the real and generated data points.
5) Back-propagate the error from the discriminator network to both the generator and discriminator networks.
6) Update the weights of the generator and discriminator networks using gradient descent.
7) Repeat steps 1-6 until the generator network is able to generate data that is similar to the training data, and the discriminator network is able to distinguish between real and fake data with a high degree of accuracy.

Challenges

Despite their success in various applications, GANs still face some challenges such as Mode Collapse (generator network only generates a limited subset of the possible data points, Vanishing Gradient (the gradients of the loss function become too small, leading to slow convergence or stagnation of the training process), Oscillation (generator and discriminator networks oscillate between producing fake data that is too similar to the real data and producing fake data that is too different from the real data), Evaluation (difficult to evaluate the quality of generated data objectively), limitations of proposed metrics and Bias (biased towards the training data, leading to the generation of data that is similar to the training data but may not be representative of the true data distribution).

Deep Convolutional Generative Adversarial Networks (DC GAN)

Deep Convolutional GAN (DCGAN) was proposed by a researcher from MIT and Facebook AI research [6]. It is widely used in many convolution-based generation-based techniques. The focus of this paper was to make training GANs stable. Hence, they proposed some architectural changes in the computer vision problems. A visualization of DCGAn is presented in Figure 1. In this paper, we will be using DCGAN on the Simpsons dataset with the aim of generating images related to cartoons.

4.1 Need for DCGANs

DCGANs are introduced to reduce the problem of mode collapse. Mode collapse occurs when the generator got biased towards a few outputs and can't able to produce outputs of every variation from the dataset. For example- take the case of "mnist" digits dataset (digits from 0 to 9), we want the generator should generate all type of digits but sometimes our generator got biased towards two to three digits and produce them only. Because of that the discriminator also got optimized towards that particular digits only, and this state is known as mode collapse. But this problem can be overcome by using DCGANs.

4.2 Architecture

The input for the generator in DCGAN architecture is 100 uniform generated values with normal distribution. In every stride iteration, number of output channels are decreased. Also, the image dimension is increased. The dimension after a 4-iteration ½ stride with fractional convolution becomes 4x4x1024. The final output is of size (64,64,3). The generator uses Batch Normalization for stability. Also, fully-connected layers of GAN are done away with. In this paper, ReLU activation function is used in all layers of the generator, except for output layers.

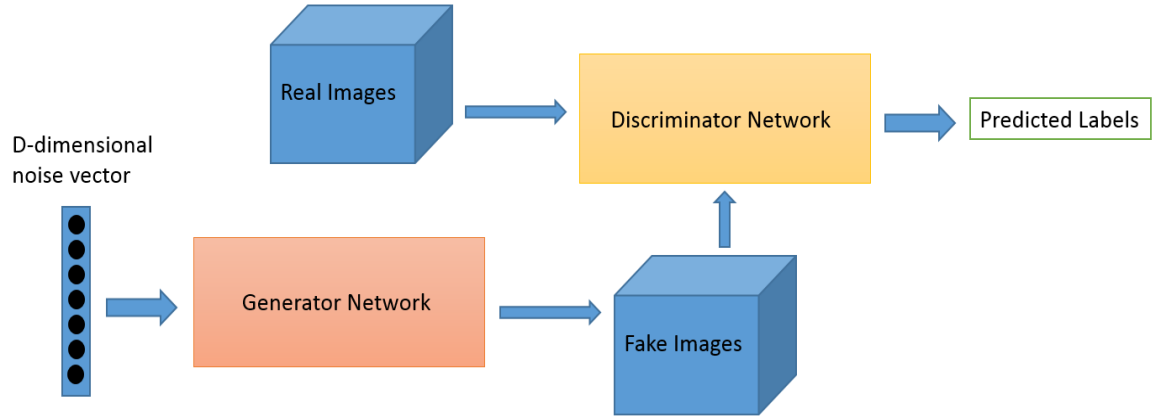

Figure 1. *DCGAN visualization. Image source: "Generative Adversarial Networks for Beginners," O'Reilly.*

Discriminator tries to distinguish a real image from a synthesized image. It mimics a convolutional neural network used for classification. The role of the discriminator here is to determine that the image comes from either a real dataset or a generator. The discriminator can be simply designed similar to a convolution neural network that performs an image classification task.

Implementation

The Simpsons DataSet

The Simpson's DataSet [2] is a folder containing the images of the cartoon characters from the famous cartoon show – "The Simpsons". These images are placed in subfolders – one per character in the show. The number of images varies from a few tens to more than a thousand for some characters. There are thirty-nine characters represented in the folders.

Methodology

The dataset is pre-processed to obtain fifteen folders containing eight hundred and more images of characters. These images are in color and have sizes such as 320 * 420 pixels. This dataset is then reduced to 28*28 sizes. These images are then converted to data frames of sizes 785 columns *1 row. The first column is reserved for a class label, which is a number from 1 to 39 representing the cartoon character. The remaining rows represent the corresponding image. A datasheet is thus created for every folder in the dataset. All these datasheets are then merged to form a single excel sheet. This is converted to a csv file. Finally, the first column is extracted and placed in a second file and called "y" or label file. The file containing image data is named "x" or data file.

The Simpson dataset contains 13,399 training images and 990 test images for each dimension (28, 28, 1). Since the value of each pixel is in the range (0, 255), we divide these values by 255 to normalize it. We plot first 25 images of training dataset. Now, we define the generator architecture, this generator architecture takes a vector of size 100 and first re-shape that into (7, 7, 128) vector and then, it applies transpose convolution on that reshaped image in combination with batch normalization. The output of this generator is a trained image of dimension (28, 28, 1). The discriminator takes an image of size 28*28 with 1 color channel and outputs a scalar value representing an image from either dataset or generated image.

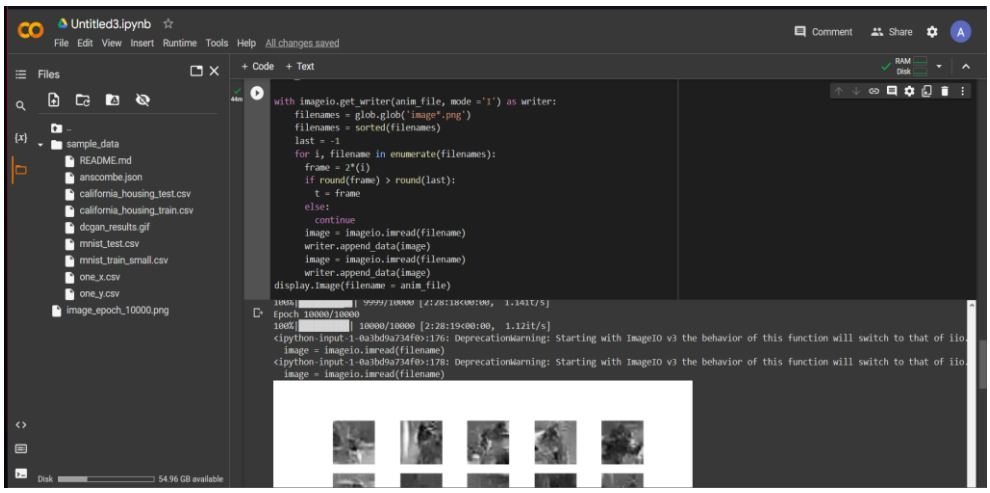

Figure 2. *Screenshot of 10000 epoch run of DCGAN on Simpson data*

DCGAN code is run on Python on Tensorflow with GPU in CoLab environment. DCGAN is implemented in Keras. The input is Simpsons dataset contains images of size (28, 28) of 1 color channel instead of (64, 64) of 3 color channels. Therefore, some changes are made to the architecture. Some modules such as Keras, TensorFlow (version 2.11), Pandas, matplotlib (for plots) are used in the code. Simpson dataset is uploaded to the sample_data folder. The discriminator (and later on, the generator) is compiled with binary cross entropy loss and adam optimizer. The discriminator is rendered untrainable before combining it with generator. Then, the generator is compiled.

At first the entire dataset is run to test the efficiency of the run. It is found that the dataset is too huge and gives around 23 runs before getting interrupted. This is because the GPU processor tends to restart every six hours. Hence, a smaller dataset with lesser number of labels is tried and the same result is observed. Thus, dataset belonging to the class "Abraham Grandpa Simpson" is input to the processor with 10, 100, 250, 500, 1000, 2000 and 10000 epochs, all running within three hours. Figure 2 shows a screenshot of the 10,000 epoch run.

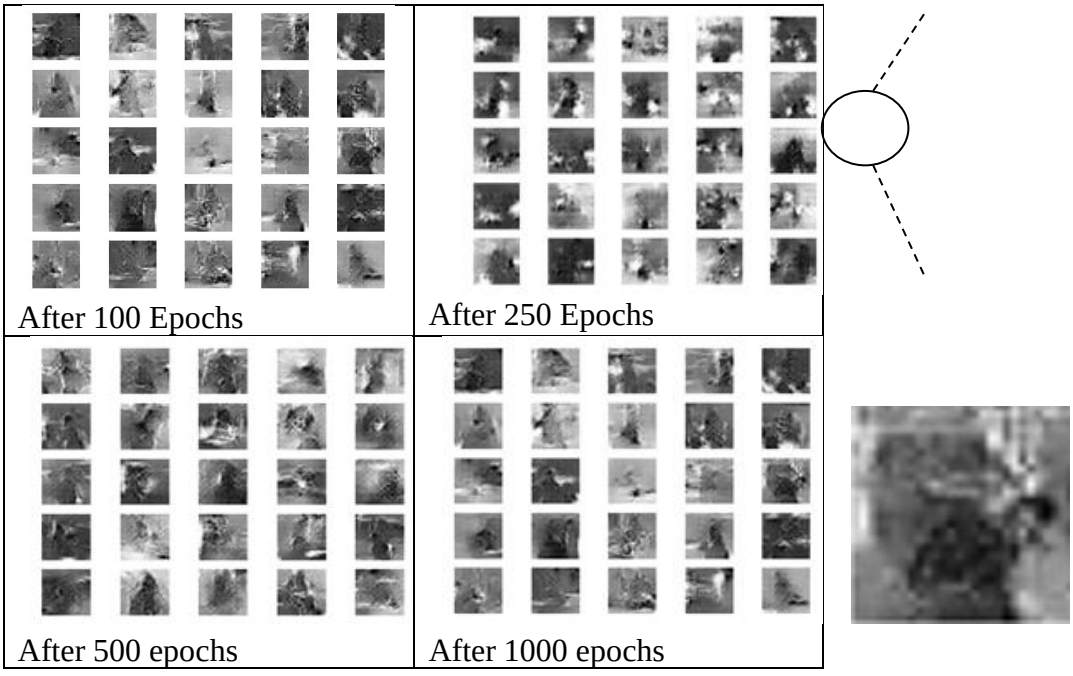

Figure 3 *DCGAN run with 500 and 1000 epochs repectively*

Results

The image results for the various runs of DCGAN algorithm using Python show that a few interesting images appear at the end of 1000 epochs (Figure 3). After this number, there is no significant improvement in the images. In fact, the images tend to get foggy as the number of epochs tends toward 2000. Figure 3 shows the evolution of twenty five images of the character "Abraham Grandpa Simpson" through the epoch. Image number fifteen that is placed at third row fifth column shows an image close to a "goblin" after 1000 epochs. A similar result is also observed at 10,000 epochs – a "Gimpson" image – in Figure 5. The generated synthetic images are checked manually for quality.

The output of the 10000 epoch run is presented in Figure 4. The parameters for output shape and Param value are given along with the Layer type under consideration. In the discriminator summary, it is observed that the model "Sequential" takes all parameters (212, 865) as trainable. On the other hand, the generator summary shows that 384 parameters are untrainable. The discriminator outputs of LeakyRelU with input at 0.2 and Dropout at 0.3 are also presented.

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d (Conv2D)             (None, 14, 14, 64)        1664
 leaky_re_lu (LeakyReLU)     (None, 14, 14, 64)        0
 dropout (Dropout)           (None, 14, 14, 64)        0
 conv2d_1 (Conv2D)           (None, 7, 7, 128)         204928
 leaky_re_lu_1 (LeakyReLU)   (None, 7, 7, 128)         0
 dropout_1 (Dropout)         (None, 7, 7, 128)         0
 flatten (Flatten)           (None, 6272)              0
 dense (Dense)               (None, 1)                 6273
=================================================================
Total params: 212,865
Trainable params: 212,865
Non-trainable params: 0
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_1 (Dense)             (None, 6272)              633472
 reshape (Reshape)           (None, 7, 7, 128)         0
 batch_normalization (BatchN  (None, 7, 7, 128)        512
 ormalization)
 conv2d_transpose (Conv2DTra  (None, 14, 14, 64)       204864
 nspose)
 batch_normalization_1 (Batc  (None, 14, 14, 64)       256
 hNormalization)
 conv2d_transpose_1 (Conv2DT  (None, 28, 28, 1)        1601
 ranspose)
=================================================================
Total params: 840,705
Trainable params: 840,321
Non-trainable params: 384
_________________________________________________________________
  0%|          | 0/10000 [00:00<?, ?it/s]
Epoch 1/10000
  0%|          | 1/10000 [00:13<38:45:16, 13.95s/it]
Epoch 2/10000
  0%|          | 2/10000 [00:15<18:01:45,  6.49s/it]
Epoch 3/10000
```

Figure 4. *Output of DCGAN run*

The output of the first twenty-five images after a 10,000 epoch run is presented in Figure 5. The images show marked improvement from the previous epochs. Some of the images are clearer than the rest. The image on the first row, fifth column, for example, is a new cartoon character – say, Gimpson. There may be more such revelations if the number of epochs is further increased.

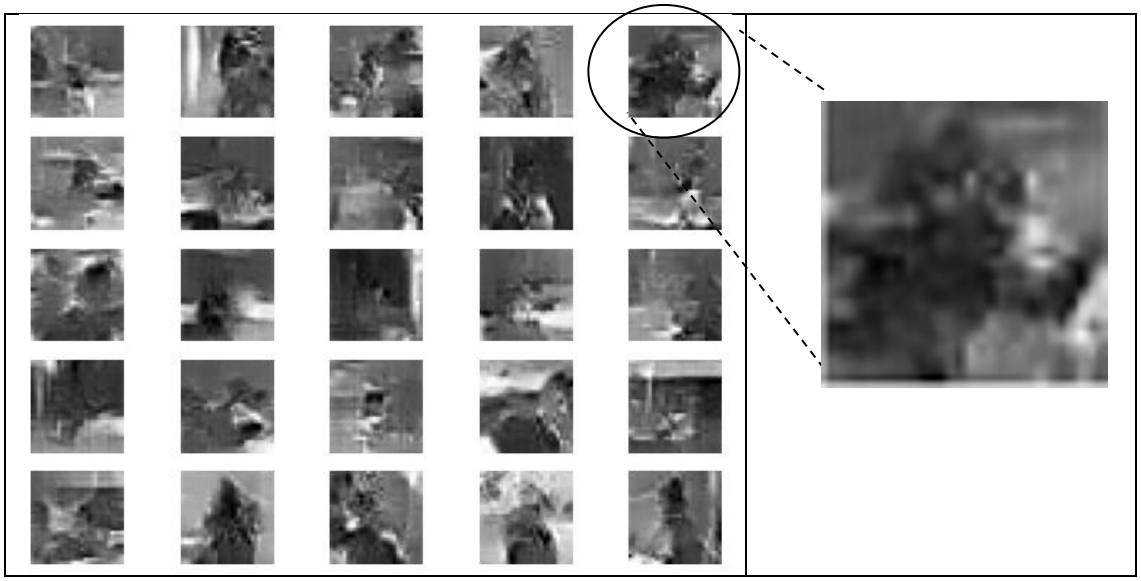

Figure 5. *Final DCGAN image after 10000 epochs*

Conclusion

This paper studies the usefulness of a Deep Convolutional Generative Adversarial Network in the generation of images of new cartoon characters from old ones. The case study of Simpsons Dataset is taken to analyze the same. The model is run through 10,000 epochs and some interesting and realistic new cartoon images are discovered in the process from 1000 epoch onwards. These results show that synthetic images can be generated using DCGANs from sources of hand-drawn images such as cartoons as well.

References

Xiao, H., Rasul, K., & Vollgraf, R. (2017). Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms. arXiv preprint arXiv:1708.07747.

Zhang, H., Xu, T., Li, H., Zhang, S., Huang, X., & Wang, X. (2018). FashionGAN: Generating High Fidelity Images of Fashion Models wearing Garments. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops (pp. 2187-2196).

Li, Y., Li, X., Li, Y., Tao, D., & Li, J. (2021). FashionGAN++: Improved Baseline and Multi-Branch Attention Model for Fashion Image Synthesis. IEEE Transactions on Industrial Informatics, 17(1), 193-202.

Li, S., Xu, H., Liu, T., & Cao, X. (2019). Adversarial Training for Fashion Image Generation. IEEE Access, 7, 10255-10263.

Goodfellow I, Pouget-Abadie J, Mirza M, Xu B, Warde-Farley D, Ozair S, et al. (2014). Generative adversarial nets. In: Advances in neural information processing systems.

Radford, Alec & Metz, Luke & Chintala, Soumith. (2016). Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks.

W. Fang, Y. Ding, F. Zhang and J. Sheng. (2019). "Gesture Recognition Based on CNN and DCGAN for Calculation and Text Output" in IEEE Access, vol. 7, pp. 28230-28237, doi: 10.1109/ACCESS.2019.2901930.

Dewi, C., Chen, RC., Liu, YT. et al. (2022).Synthetic Data generation using DCGAN for improved traffic sign recognition. Neural Comput & Applic 34, 21465–21480. https://doi.org/10.1007/s00521-021-05982-z.

Q. Wu, Y. Chen and J. Meng, "DCGAN-Based Data Augmentation for Tomato Leaf Disease Identification," in IEEE Access, vol. 8, pp. 98716-98728, 2020, doi: 10.1109/ACCESS.2020.2997001.

Bushra, S. N., & Maheswari, K. U. (2021). Crime Investigation using DCGAN by Forensic Sketch-to-Face Transformation (STF)-A Review. In 2021 5th International Conference on Computing Methodologies and Communication (ICCMC) (pp. 1343-1348). IEEE.

Zhang, Y., Wei, G., Min, H., Xu, Y. (2022). Text-Independent Speaker Identification Using a Single-Scale SincNet-DCGAN Model. In: Tan, Y., Shi, Y. (eds) Data Mining and Big Data. DMBD 2022. Communications in Computer and Information Science, vol 1745. Springer, Singapore. https://doi.org/10.1007/978-981-19-8991-9_2.

Puttagunta, M., & Subban, R. (2022). A Novel COVID-19 Detection Model Based on DCGAN and Deep Transfer Learning. Procedia computer science, 204, 65-72.

Kim, D. D., Shahid, M. T., Kim, Y., Lee, W. J., Song, H. C., Piccialli, F., & Choi, K. N. (2019, November). Generating pedestrian training dataset using DCGAN. In Proceedings of the 2019 3rd International Conference on Advances in Image Processing (pp. 1-4).

An, Y., Wang, M., Chen, L., & Ji, Z. (2022). DCGAN-based symmetric encryption end-to-end communication systems. AEU-International Journal of Electronics and Communications, 154, 154297.

Wei, C., Liang, J., Liu, H., Hou, Z., & Huan, Z. (2022). Multi-stage unsupervised fabric defect detection based on DCGAN. The Visual Computer, 1-17.

Smaida, M., Yaroshchak, S., & El Barg, Y. (2021). DCGAN for Enhancing Eye Diseases Classification. In CMIS (pp. 22-33).

Shi, B., Zhou, X., Qin, Z., Sun, L., & Xu, Y. (2021, April). Corn ear quality recognition based on DCGAN data enhancement and transfer learning. In The 4th International Conference on Electronics, Communications and Control Engineering (pp. 62-68).

Xu, Z. J., Wang, R. F., Wang, J., & Yu, D. H. (2020). Parkinson's disease detection based on spectrogram-deep convolutional generative adversarial network sample augmentation. IEEE Access, 8, 206888-206900.

Pradhan, A., Buragohain, A., Pathak, U., Ansari, S., & Baruah, M. (2020). Implementation of DCGAN to Generate Gamocha Design Patterns. In Electronic Systems and Intelligent Computing: Proceedings of ESIC 2020 (pp. 831-838). Springer Singapore.

Li, Z., & Wan, Q. (2021, November). Generating Anime Characters and Experimental Analysis Based on DCGAN Model. In 2021 2nd International Conference on Intelligent Computing and Human-Computer Interaction (ICHCI) (pp. 27-31). IEEE.