Performance Comparison between Generative Adversarial Networks (GAN) Variants in Generating Anime/Comic Character Images - A Preliminary Result

Nur Qamarina Mohd Noor
Faculty of Computing
Universiti Malaysia Pahang Al Sultan Abdullah
Pekan, Pahang Malaysia

Azlee Zabidi
Faculty of Computing
Universiti Malaysia Pahang Al Sultan Abdullah
Pekan, Pahang Malaysia
azlee@umpsa.edu.my

Mohd Izham Bin Mohd Jaya
Faculty of Computing
Universiti Malaysia Pahang Al Sultan Abdullah
Pekan, Pahang Malaysia

Tan Jia Ler
Faculty of Information and Communcation Technology
Universiti Tunku Abdul Rahman
Kampar, Perak, Malaysia

2024 IEEE Symposium on Industrial Electronics & Applications (ISIEA) | 979-8-3503-8686-8/24/$31.00 ©2024 IEEE | DOI: 10.1109/ISIEA61920.2024.10607225

***Abstract*— In recent years, the popularity and demand for digital animation, specifically anime characters, has brought significant challenges and opportunities for the world of computer graphics and artificial intelligence. This research dives deep into the comprehensive exploration of two well-known Generative Adversarial Networks (GANs)— Deep Convolutional Generative Adversarial Network (DCGAN) and CycleGAN, — with a specific focus on anime character generation. GANs, consisting of a generator and discriminator, operate in a feedback loop to create and evaluate synthetic data. This research will identify challenges within each GAN model and develop objectives to address these challenges. Both of GAN models were mainly executed in the Google Colab environment to optimize the GPU-accelerated runtime. The dataset is sourced from publicly accessible anime image repositories and both GAN models will be evaluated using Fréchet Inception Distance (FID) and Inception Score (IS). FID compares the distribution of generated images with the distribution of a set of real images ("ground truth") while IS only evaluates the distribution of generated images. Together, they provide a comprehensive evaluation of a GAN's performance. In the realm of anime character generation, achieving authenticity and diversity is crucial.**

Keywords—GAN, FID, IS, DCGAN, CycleGAN

I. INTRODUCTION

GAN comprises of a generator (G) and a discriminator (D), operate in a feedback loop to make and evaluate data. The main purpose of a generator is to create fake data that is as realistic as possible. It does this by trying to generate realistic images from a noise input. In contrast, the discriminator's job is to distinguish between real and generated data. As the generator creates increasingly convincing data, the discriminator continually refines its ability to differentiate between real and fake data. In the rapidly evolving world of digital design and animation, there exists a pressing issue that both amateurs and professionals facing with - creating unique, high-resolution anime/comic characters that captivate the audience. As industries, especially the gaming and entertainment sectors, constantly evolve, so do their demands for complex, novel characters. GANs [1-3] have become torchbearers in this field, providing automated solutions that are both unique and thematically resonant. Unique anime/comic character design is not just a matter of artistic achievement. It's an economic necessity. Successful character design can lead to brand recognition, merchandise opportunities, and sequels or spin-offs. Conversely, poorly designed anime/comic characters can make content unrelatable or forgettable, leading to losses and missed opportunities. Though GAN is considered as the pioneer in realistic synthetic anime/comic character generation, there are still set of challenges that need to be addressed.

DCGAN, despite its advancements, faces a significant challenge in consistently producing high-quality fake images. The inherent difficulty lies in achieving a level of image realism that matches real images. This limitation hinders the model's performance in applications where generating visually convincing and realistic images is critical. Moreover, the sensitivity of DCGAN to hyperparameters further complicates the optimization process, adding a layer of complexity to the generation of high-quality images. As for CycleGAN, the translation from real images to anime-style images is a huge challenge for CycleGAN. The complexity of anime aesthetics, including unique features and artistic nuances, make it challenging for the model to faithfully transform real-world images into a convincing anime style. This limitation restricts CycleGAN's effectiveness in applications that require precise and high-fidelity translation, hindering its usefulness in tasks where maintaining the essence of anime artistry is important.

This paper aims to conduct performance comparison between DCGAN and CycleGAN to understand unique features of each model and identify scenarios where one model might outperform the others. The goal is to understand how each model would perform in anime/comic character generation task based on FID and IS values.

II. Related Papers

A. GAN Models

In [4], DCGAN represents a significant advancement in the realm of GANs by incorporating Convolutional Neural Networks (CNNs). In contrast to traditional GANs, both the discriminator and generator in DCGAN employ CNNs instead of the typical multilayer perceptrons. As the name suggests, DCGAN introduces a deep convolutional architecture into the GAN framework as shown in Fig. 1. The key distinguishing feature is the utilization of convolutional layers without the inclusion of maximum pooling or fully connected layers. This architecture relies on the synergy between convolutional strides and transpose operations for down-sampling and up-sampling, respectively.

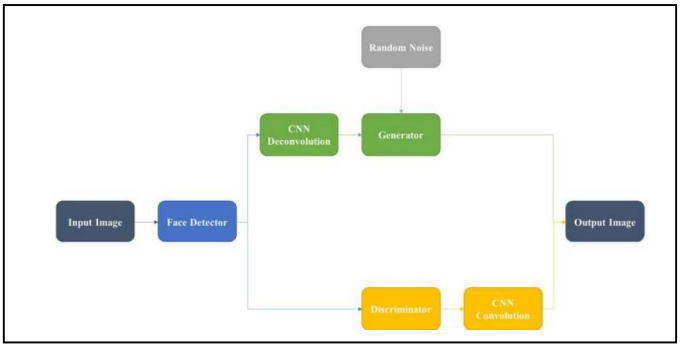

Fig. 1. DCGAN model's framework

By integrating CNNs, DCGAN enhances the ability of the generator and discriminator to understand spatial hierarchies and capture complex patterns in the data. This not only contributes to the generation of more realistic and detailed images but also improves feature extraction in the discriminator. The absence of dense layers and the emphasis on convolutional operations make DCGAN particularly suitable for image-related tasks, providing a powerful framework for generating high-quality images through adversarial training.In the PyTorch implementation, ReLU activation is used in the generator, employing Tanh for output, while the discriminator utilizes LeakyReLU activation for all layers[4-6]. The DCGAN framework utilizes TensorFlow's convolution function to process data into a low-dimensional matrix for the discriminator's authenticity comparison. The decov function expands this matrix to a high-dimensional form, helping the generator in image generation. Defined functions like conv2D and deconv2d encapsulate these convolutional neural network processes.

During DCGAN training, the generative model aims to fool the discriminant model, while the discriminative model attempts to accurately distinguish between generated and real images. This adversarial training promotes a dynamic interplay between the two models, with the ideal result being that the generated images achieve a mixture of realism and falseness (D(G(z)) = 0.5). The discriminator is constructed as a forward convolutional neural network, while the generator maps the latent space vector (z) to the data space through a series of 2D convolutional transposed layers. The DCGAN emphasizes the use of stride convolution instead of pooling for down-sampling to improve the network's ability to learn its pooling function. Batch normalization and the LeakyReLU functions promote healthy gradient flow, which is crucial for effective learning in both the generator and discriminator. The generator's structure is similar to the discriminator's but uses a deconvolution function to reverse the convolution results, generating images from randomly generated noise (Z). The symmetry in the number of convolution and deconvolution layers ensures the similarity of the resulting images.

As for CycleGAN, it is an unsupervised learning method that incorporates a cycle consistency loss function within the framework of GANs. Fig. 2 shows the CycleGAN's model framework. The main objective of CycleGAN is image-to-image translation between two sample spaces, A and B, with an underlying relationship. [7-9] The generator (denoted as G) transforms samples from X to Y, while a discriminator, D_u, distinguishes between generated and real images in domain Y. Similarly, there is a mapping function F that converts samples in Y back to X, and a discriminator D_γ for domain X.

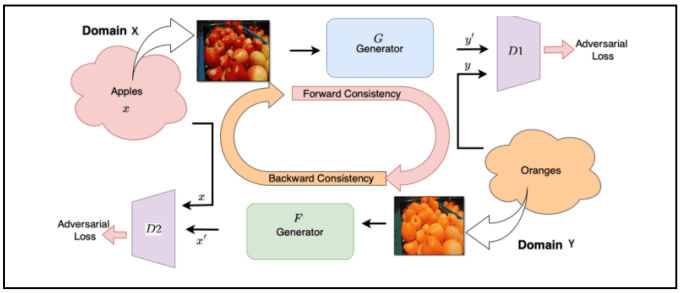

Fig. 2. CycleGAN model's framework

Applying adversarial loss can ensure that the translated image belongs to the distribution of images from Domain Y but cannot guarantee that we get the exact corresponding sample in Domain Y. A translation of an image from Domain X to a corresponding image from Domain Y was easily possible since we had paired samples from both domains. Thus, a simple supervised loss like the mean absolute error on the output of the generator and the ground-truth image in Domain Y was enough to ensure that the translation of an image from Domain X results in the corresponding image in Domain X.However, since there is no access to corresponding paired samples in the task of unpaired image translation, supervised loss cannot be used, which makes it difficult to ensure correspondence at the sample level in the two domains.

In order to tackle this issue, the CycleGAN framework proposes to use cyclic consistency loss. The cyclic consistency loss ensures that the two functions learned are inverse functions of each other. Since the requirement for functions to have an inverse is that they have to be bijective (i.e., one-one and onto), it implicitly ensures a one-to-one correspondence at the sample level in the case of unpaired image translation.

The architecture of the generator network in CycleGAN consists of several components:

- Convolution with ReLU Activation: The network starts with a convolutional layer using Rectified Linear Unit (ReLU) activation. ReLU is a popular activation function known for introducing non-linearity in neural networks.
- Two Down-sampling Blocks: These blocks are responsible for reducing the spatial dimensions of the input images from 128 x 128 to 256 x 256. Down-sampling usually involves operations such as convolution and pooling to decrease the resolution of the image.

- Residual Blocks: Residual blocks are a key component in deep neural networks, contribute to the learning of identity mappings and aiding in the convergence of the training process
- Two Up-sampling Blocks: These blocks are designed to increase the spatial dimensions of the images, taking them from 128 x 128 back to 64 x 64. Up-sampling typically involves operations like transposed convolution to increase the resolution of the image.
- Convolution with Hyperbolic Tangent (tanh) Activation: The last layer of the generator applies a convolution operation with a hyperbolic tangent (tanh) activation function. Tanh is commonly used for the final layer of a generator in GANs to produce pixel values in the range [-1, 1].

For the discriminator in CycleGAN, a 70 x 70 PatchGAN architecture is used. PatchGAN is a variant of the discriminator that classifies images by dividing them into smaller patches. The output of a regular GAN discriminator is a single scalar, while PatchGAN outputs an N x N array of values, where each value corresponds to a patch of the input image. This approach allows the discriminator to assess the realism of local patches, providing more detailed feedback to the generator during training.

B. IS and FID Metrics

Two common performance metrics are used to measure the performance of the GAN models. The IS formula involves calculating the Kullback-Leibler (KL) divergence of conditional probability distributions [10]. Specifically, it measures the distance between the predicted distribution and the true distribution of the main objects in the generated image. The IS equation is given by:

$$IS(G) = \exp(EI_{X\sim Pg} D_{KL}(P(y|x)||p(y))) \quad (1)$$

where (x) represents a generated picture, (y) represents the main object in the picture, (Pg) is the generator's distribution, and (DKL) is the Kullback-Leibler divergence. A higher IS indicates greater diversity and clarity in the generated images.

For FID, it is a metric used to evaluate the quality of generated images produced by a GAN or other generative models by evaluating the feature vectors of generated and ground truth image extracted by passing through a pre-trained neural network [11]. FID measures the difference of two Gaussians (synthetic and real-world images) where the Gaussian is the maximum entropy distribution for given mean and covariance. The distance between the Gaussian with mean (m, C) obtained from p(.) and the Gaussian with mean (mw, Cw) obtained from pw(.) is measured by the following equation:

$$d^2\big((m, C), (m_w, C_w)\big) = ||m - m_w||_2^2 + Tr(C + C_w - 2(CC_w)^{1/2} \quad (2)$$

A lower FID score indicates that the generated images are closer in terms of features to the real images.

III. Methodology

Both DCGAN and CycleGAN are executed using Google Colab to utilize the provided GPU resources. The deep learning framework used for both GANs are Pytorch with the Ignite as the high-level library to train the neural networks. The training in Ignite is based on three core components:

- Engine: It is equivalent to training loop. It takes a 'train_step' as an argument and runs it over each batch of the dataset, triggering events as it goes.
- Events: They are emitted by the Engine when it reaches when it reaches a specific point in the run/training.
- Handlers: Functions that can be triggered when a certain Event is emitted by the Engine. Ignite has a long list of pre-defined Handlers such as checkpoint, early stopping, logging and built-in metrics.

The FID and IS metrics are provided by the *ignite.metric* submodule which provides a way to compute various quantities of interest in an online fashion without having to store the entire output history of a model. The high-level diagram of the workflow for both GANs are shown in Fig. 3. The details of each process in the workflow are discussed separately for each GAN.

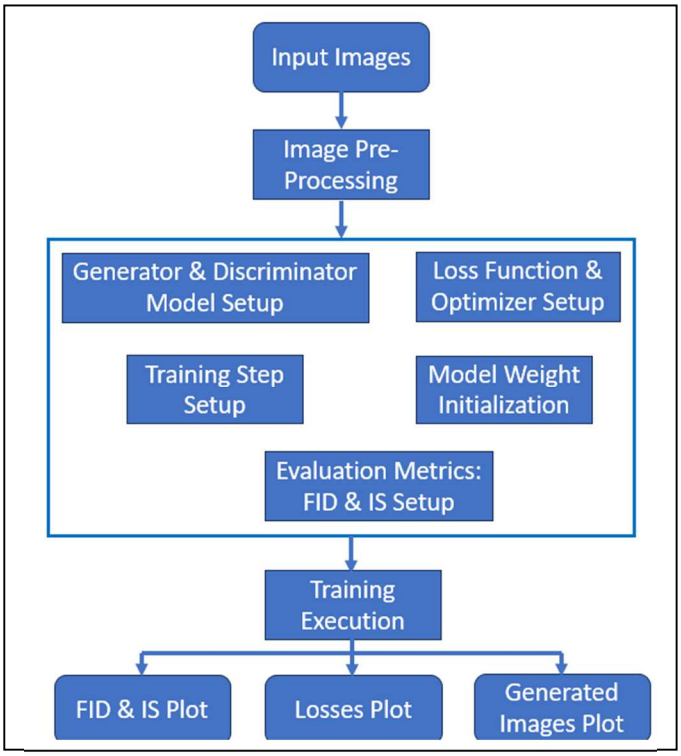

Fig. 3. High-level diagram of the GAN workflow

A. DCGAN

For DCGAN, the input images are in form of single-domain images. Those images will be pre-processed where they will be resized, cropped and normalized. The generator (G) and discriminator(D) for the DCGAN is setup as described in Section II. The loss function used for the DCGAN is Binary Cross Entropy loss and the optimizer is Adam optimizer. The basic training steps starts with running the G and D model and the loss are propagated backward to each model. Both models' weights must be randomly initialized from a Normal Distribution with mean=0, stdev=0.02.

The training execution starts with the *init_weights* handler is triggered at the start of the engine run and it is responsible for applying the *initialize_fn* function to randomly generate weights for the generator and discriminator. The *store_losses* handler is responsible for storing the generator and discriminator losses in *G_losses* and *D_losses* respectively. It

is triggered at the end of every iteration. The *store_images* handler is responsible for storing the images generated by the generator model during training. It provides us with a visual progress of the training. It is triggered every 500 iterations. The *log_training_results* handler is triggered every epoch where the evaluator engine is run to evaluate the IS and FID metrics. The results are finally stored per epoch.

B. CycleGAN

For CycleGAN, the input images are in form of dual-domain images. Images in both domains will be pre-processed where they will be cropped, flipped horizontally and normalized. The brightness, contrast, saturation and hue of those images will also be changed. The generator (G) and discriminator(D) for the CycleGAN is setup as described in Section II. The loss function is Binary Cross Entropy Loss and the optimizer is Adam optimizer. The basic training steps starts with running the G and D model and three losses namely G loss, D loss and cycle-consistency loss are propagated forward and backward to each model. Both G and D models' weights must be randomly initialized from a Normal Distribution with mean=0, stdev=0.02.

The training execution starts with the trainer engine runs the *update_fn* function to update models at every iteration while running over the input data. This function comprises of getting the predictions, calculating the loss, calculating the gradients using backpropagation and applying the gradients to the optimizer. At the beginning of each epoch, the *run_evaluation* handler produces the fake image and reconstruct the real image from the fake image. The *log_generated_images* handler logs all three real images, fake images and reconstructed images. The *log_metrics_results* handler attached to the trainer engine is triggered every 20 epochs to evaluate the IS and FID metrics.

IV. Results

Once the execution of both DCGAN and CycleGAN completed, their FID (blue-color), IS (red-color) and generated images are plotted. Details of the FID, IS and generated images are discussed separately for each GAN.

A. DCGAN

For DCGAN, the number of epochs is 50 and the FID and IS metrics are captured every epoch. Fig. 4 shows the FID and IS result for DCGAN. The FID metric is at around 0.08 at epoch = 0 and it keeps decreasing to around 0.03 at epoch = 50 while for the IS metric is around 1.3 at epoch = 0 and it keeps increasing to around 1.8 at epoch = 50. From this figure, both FID and IS metrics for DCGAN at the last epoch indicates that the generated images are closer in term of feature to the input images.

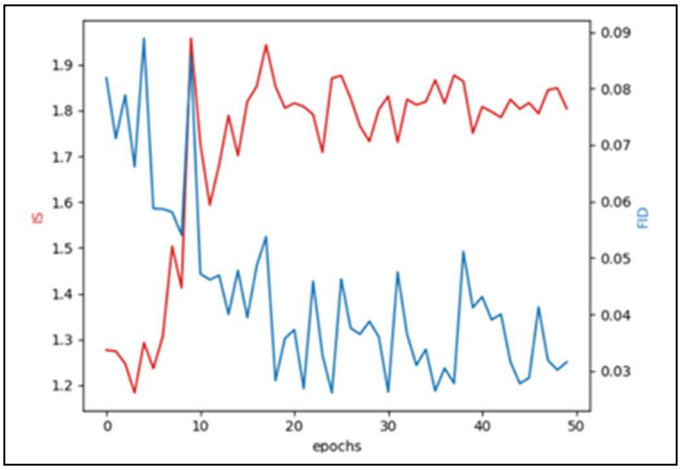

Fig. 4. FID and IS metrics for DCGAN

Fig. 5 shows the side-by-side comparison between input images (left-side) and generated images (right-side) for DCGAN. On the surface, the generated images seem similar to the input images. However, if look closer, there are some irregularities to the generated images such the unsymmetrical shapes between left eye and right eye and missing nose shape and lips line.

Fig. 5. Side by comparison between input images and generated images for DCGAN

B. CycleGAN

For CycleGAN, the number of epochs is 200 and the FID and IS metrics are supposed to be captured every 20 epochs. Fig. 6 shows the FID and IS result for CycleGAN. The FID metric is at around 0.4 at epoch = 0 and it fluctuates and reaches 0.9 at epoch = 10 (x20) while for the IS metric is around 2.5 at epoch = 0 and it fluctuates and reaches around 4.8 at epoch = 10 (x20). From this figure, the FID metric for CycleGAN at the epoch =10 indicates that the generated images are not closer in term of feature to the input images. As for the IS metric, it indicates greater diversity and clarity in the generated images although at only epoch = 10 (x20).

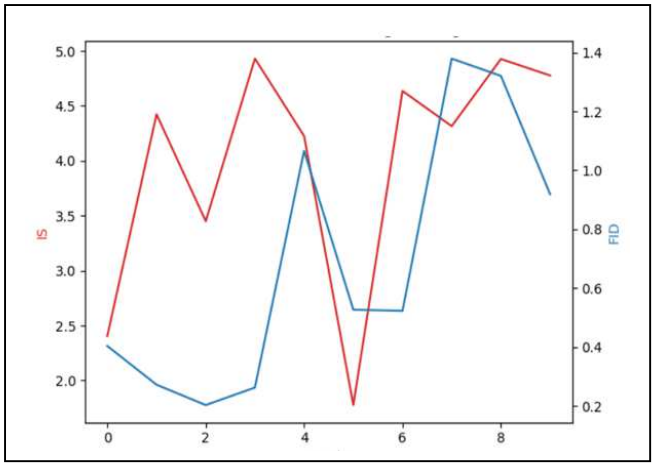

Fig. 6. FID and IS metrics for CycleGAN

Fig. 7 shows the side-by-side comparison between input images (left-side) and generated images (right-side) for CycleGAN. It can be seen in Fig. 7(a) the generated images almost similar to the input images except for minor missing nose shape and lips line. However, for Fig. 7(b), there are significant irregularities to the generated image. Although all the elements in the image are converted to anime-like texture where the facial features are not anime-ized but copy their real facial features. This is due to the trained dataset only comprises of real and anime female facial features, thus, it cannot produce anime features for male subjects.

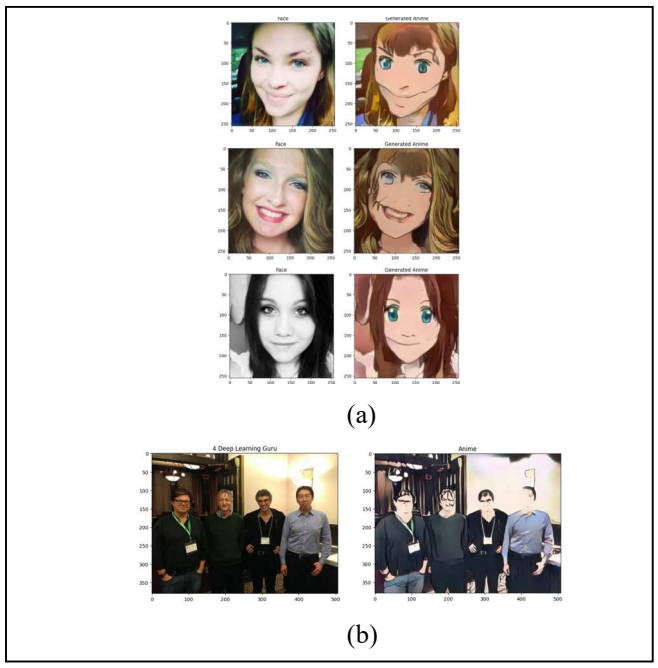

Fig. 7. Side by comparison between input images and generated images in CycleGAN

V. Conclusion and Future Works

In conclusion, the anime characters can be synthetically generated using GAN where in this paper, the attempt was made on DCGAN (single domain) and CycleGAN (dual domain). The performance for both GANs were evaluated using FID and IS where for DCGAN, both metrics indicates that generated images are closer to the real images. However, for CycleGAN, only IS metric indicates that the generated images are closer to the real images while for FID metric, it did not produce the desired result. Thus, further improvement on the method of capturing FID metric must be made to the code.

Acknowledgement

The authors would like to thank the Universiti Malaysia Pahang Al-Sultan Abdullah for additional financial support under Internal Research Grant RDU220374. This support is gratefully acknowledged.

References

[1] S. Ruan, "Anime Characters Generation with Generative Adversarial Networks," 2022 IEEE International Conference on Advances in Electrical Engineering and Computer Applications (AEECA), Dalian, China, 2022, pp. 1332-1335

[2] D. E. V, V. V. H. Kumar, R. D. Kumar and V. M. Sahithi, "Generation of Hilarious Animated Characters using GAN," 2023 7th International Conference on Trends in Electronics and Informatics (ICOEI), Tirunelveli, India, 2023, pp. 63-66

[3] Y. Zeng and D. Xue, "An Overview of Generative Adversarial Networks," 2022 IEEE 2nd International Conference on Electronic Technology, Communication and Information (ICETCI), Changchun, China, 2022, pp. 550-552,

[4] Z. Li and Q. Wan, "Generating Anime Characters and Experimental Analysis Based on DCGAN Model," 2021 2nd International Conference on Intelligent Computing and Human-Computer Interaction (ICHCI), Shenyang, China, 2021, pp. 27-31

[5] D. Sankalpa, J. Ramesh and I. Zualkernan, "Using Generative Adversarial Networks for Conditional Creation of Anime Posters," 2022 IEEE International Conference on Industry 4.0, Artificial Intelligence, and Communications Technology (IAICT), BALI, Indonesia, 2022, pp. 197-203

[6] Y. Jiang, "Performance Analysis Anime Character Generation Based on DCGAN Model," 2021 2nd International Conference on Intelligent Computing and Human-Computer Interaction (ICHCI), Shenyang, China, 2021, pp. 82-85

[7] L. Quan and H. Zhang, "Facial Animation Using CycleGAN," 2021 International Conference on Electrical, Computer, Communications and Mechatronics Engineering (ICECCME), Mauritius, Mauritius, 2021

[8] Xiao Xu and Xianlan Wang "Human face cartoon image generation based on CycleGAN", Proc. SPIE 12451, 5th International Conference on Computer Information Science and Application Technology (CISAT 2022)

[9] E. O. Rodrigues, E. Clua and G. B. Vitor, "Line Art Colorization of Fakemon using Generative Adversarial Neural Networks," 2022 21st Brazilian Symposium on Computer Games and Digital Entertainment (SBGames), Natal, Brazil, 2022, pp. 1-6

[10] M. J. Chong and D. Forsyth, "Effectively Unbiased FID and Inception Score and Where to Find Them," 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR), Seattle, WA, USA, 2020

[11] S. Guan and M. Loew, "Evaluation of Generative Adversarial Network Performance Based on Direct Analysis of Generated Images," 2019 IEEE Applied Imagery Pattern Recognition Workshop (AIPR), Washington, DC, USA, 2019, pp. 1-5