Syntax Literate: Jurnal Ilmiah Indonesia p–ISSN: 2541-0849
e-ISSN: 2548-1398
Vol. 7, No. 10, Oktober 2022

COMPARATIVE ANALYSIS OF DCGAN AND WGAN

Syaiful Haq Al Furuqi, Handri Santoso
Teknik Informatika, Universitas Pradita, Indonesia
Email: syaiful.haq@student.pradita.ac.id, handri.santoso@pradita.ac.id

Abstract

In current years, image recognition is increasingly being used, however, it becomes much less correct if there is not plenty data available while training data. Generative Adversarial Networks (GAN) can assist to create new data that is nearly similar to the original data to help the training process while the original data isn't always much in order that the training process might be more accurate. GAN is currently growing and there are an increasing number of types, along with the Deep Convolutional GAN (DCGAN) and Wasserstein GAN (WGAN) algorithms. This study analyzes the comparison among DCGAN and WGAN which objectives to offer a decision approximately which algorithm is better to use. Based on the research results, DCGAN is simpler however there are still drawbacks, particularly the Mode collapse and vanishing gradient, at the same time as WGAN can remedy those shortcomings however the process is slower.

Keywords: analysis; GAN; DCGAN; WGAN

Introduction

Image recognition is now increasingly being used in various fields, but it can become less accurate if there is not much data available for training. To anticipate this, there is an algorithm to create new data or images that are almost similar to the original, namely Generative Adversarial Networks (GAN).

GAN was first introduced in 2014 by Goodfellow in the field of machine learning and is included in the unsupervised learning algorithm (Kaveri, Meenakshi, Deepan, Dharnish, & Haarish, 2021). GAN uses 2 neural networks or artificial neural networks consisting of Discriminator and Generator (Praramadhan & Saputra, 2021).

How to cite:	Syaiful Haq Al Furuqi. et al (2022) Comparative Analysis of DCGAN and WGAN, Syntax Literate: Jurnal Ilmiah Indonesia, *(7)* 10,
E-ISSN:	2548-1398
Published by:	Ridwan Institute

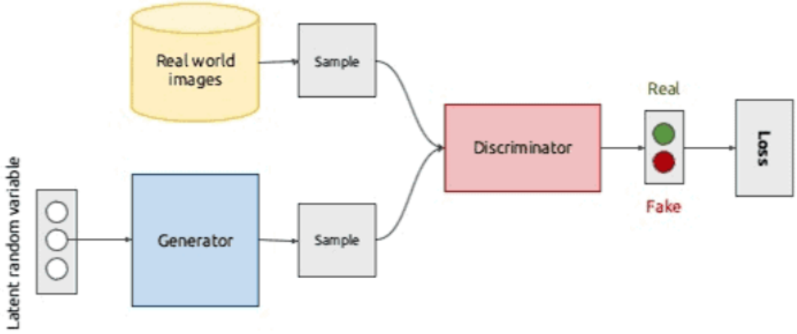

Figure 1
GAN Architecture

The discriminator takes into account the possibilities of the available dataset, which optimizes work to distinguish fake and genuine data.

The generator can sample a noise variable, and is trained to capture the original distributed data so that it can produce data that is as close as possible to the original and can fool the discriminator (Weng, 2019).

There are many types of GAN today, including Deep Convolutional GAN (DCGAN), and Wasserstein GAN (WGAN).

DCGAN is one of the GAN algorithms that was originally introduced in 2016 which is an improvement from the previous algorithm, namely Convolutional GAN (Santosa, Rachmawati, Agung, & Wirayuda, 2021). The difference between DCGAN and the original GAN is the addition of a Convolutional Neural Network (CNN) layer on the discriminator which acts as a classifier of the data generated by the generator. In DCGAN architecture, the pooling layers contained in the discriminator are replaced with strided convolutions and the generator is replaced with fractional-strided convolutions, then use batch normalization applied to the discriminator and generator, then remove the fully connected hidden layers in deeper architectures, and use ReLU activation for all generator layers except for the output layer using Tanh, and using Leaky ReLU activation for all discriminator layers (Widjojo, Palit, & Tjondrowiguno, 2020).

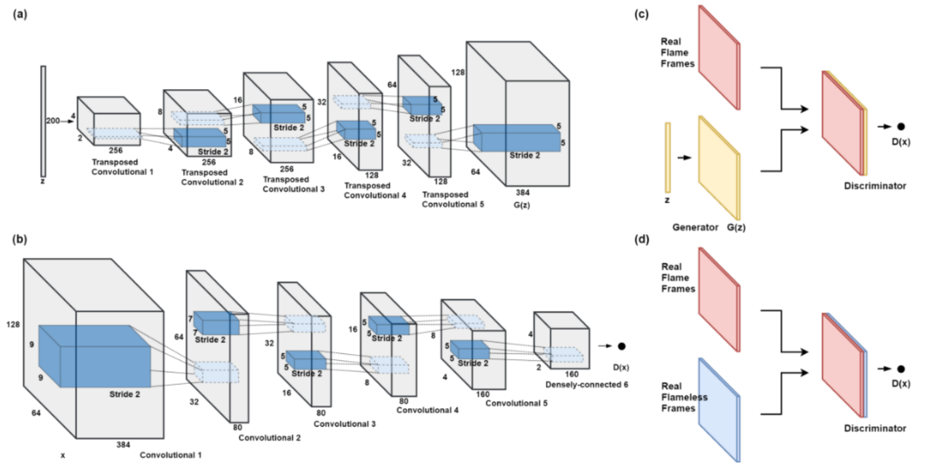

Figure 2
DCGAN Architecture (Dutta, 2020)

The downside to DCGAN is the Collapse Mode and vanishing gradients. Collapse mode is a term in GAN to describe the inability of GAN to produce different images, for example when training dog data, GAN can only produce 1 dog. Vanishing gradients are caused because the confidence-values of the discriminator only contain a single value, namely "b" or "w" and 0 or 1 (Suran, 2020).

WGAN was first introduced in 2017. WGAN comes with increased stability when training the model and there is a function called wasserstein loss which correlates with the quality of the resulting image. WGAN does not use a discriminator in classifying or predicting the resulting image as genuine or fake, but replaces it with critics to judge the resulting image is genuine or fake.

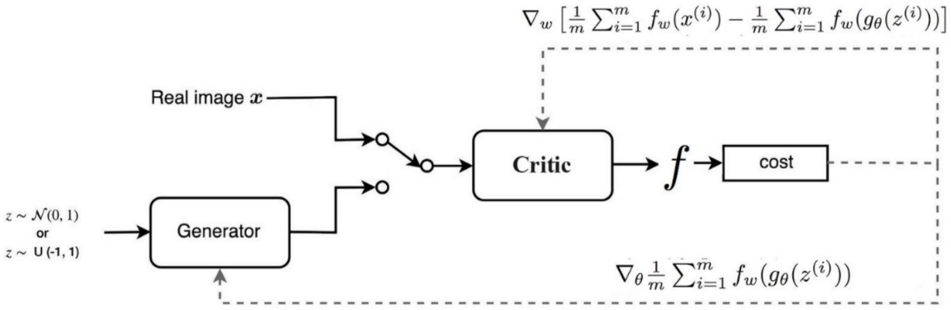
$\nabla_w \left[\frac{1}{m}\sum_{i=1}^{m} f_w(x^{(i)}) - \frac{1}{m}\sum_{i=1}^{m} f_w(g_\theta(z^{(i)}))\right]$
Real image x
Critic
f
cost
$z \sim \mathcal{N}(0,1)$
or
$z \sim$ U (-1, 1)
Generator
$\nabla_\theta \frac{1}{m}\sum_{i=1}^{m} f_w(g_\theta(z^{(i)}))$

Figure 3
WGAN Architecture and Computation (Hui, 2018)

The advantage of WGAN is that it is more stable in training models and is not sensitive to model architecture and hyperparameter configuration, then the loss of discriminators affects the image quality created by the generator (Brownlee, 2019), and also WGAN uses a gradient penalty so that discriminators can train more complex data and reduce the problem of vanishing gradients (Lee, Kim, Kim, & Kim, 2020).

WGAN seems to be here to perfect the DCGAN, but behind the advantages of the WGAN, there is a drawback, namely the training process is longer than DCGAN, but comparable to the resulting image.

Research Methodology

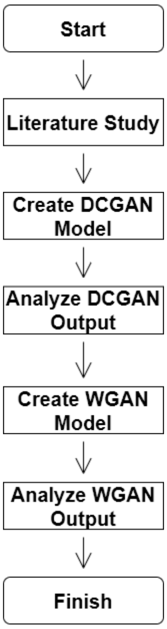

Figure 4
Research Methodology

The first thing to do in this research is to study the literature related to this research, after conducting a literature study, create a DCGAN model, then analyze the results of the model, then create a WGAN model, then analyze the output generated by the model.

Result and Discussion

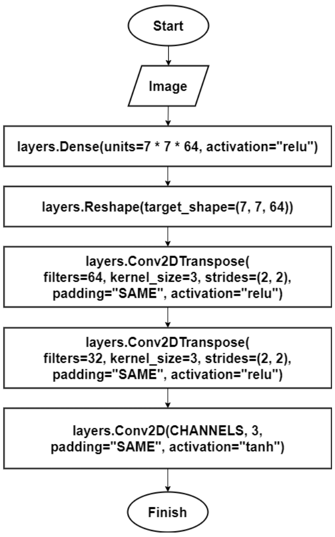
Start
Image
layers.Dense(units=7 * 7 * 64, activation="relu")
layers.Reshape(target_shape=(7, 7, 64))
layers.Conv2DTranspose(
filters=64, kernel_size=3, strides=(2, 2),
padding="SAME", activation="relu")
layers.Conv2DTranspose(
filters=32, kernel_size=3, strides=(2, 2),
padding="SAME", activation="relu")
layers.Conv2D(CHANNELS, 3,
padding="SAME", activation="tanh")
Finish

Figure 5
DCGAN Generator Model

The DCGAN generator model uses a dense layer with 7x7x64 units with relu activation, a reshape layer with a 7x7x64 target shape, 2 Conv2DTranspose layers with 64 and 32 filters, and ends with Conv2D with channel 1 because the image is a grayscale image with tanh activation.

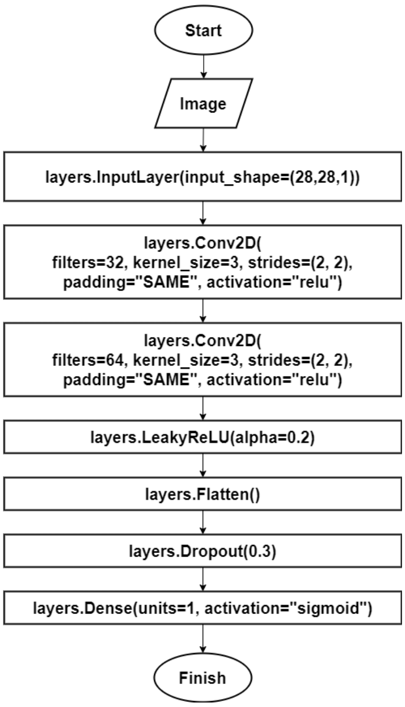
Start
Image
layers.InputLayer(input_shape=(28,28,1))
layers.Conv2D(
filters=32, kernel_size=3, strides=(2, 2),
padding="SAME", activation="relu")
layers.Conv2D(
filters=64, kernel_size=3, strides=(2, 2),
padding="SAME", activation="relu")
layers.LeakyReLU(alpha=0.2)
layers.Flatten()
layers.Dropout(0.3)
layers.Dense(units=1, activation="sigmoid")
Finish

Figure 6
DCGAN Discriminator Model

DCGAN discriminator model, using input layer with input_shape 28,28,1, 2 Conv2D layers with filters 32 and 64, LeakyReLU layer with alpha 0.2, Flatten layer, Dropout layer, Dense layer with unit 1 and activation sigmoid.

The training process uses Adam's optimizer on the model generator and discriminator with a value of 0.0001 and beta_1 0.5 and epoch 150 with each epoch of 60 seconds produces discriminator loss -0.17738741636276245 and generator loss 0.6740452647209167, then the resulting image is as follows

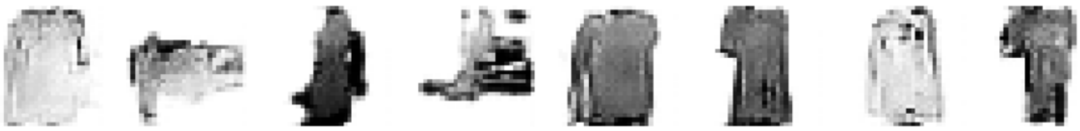

Figure 7
DCGAN Image Result

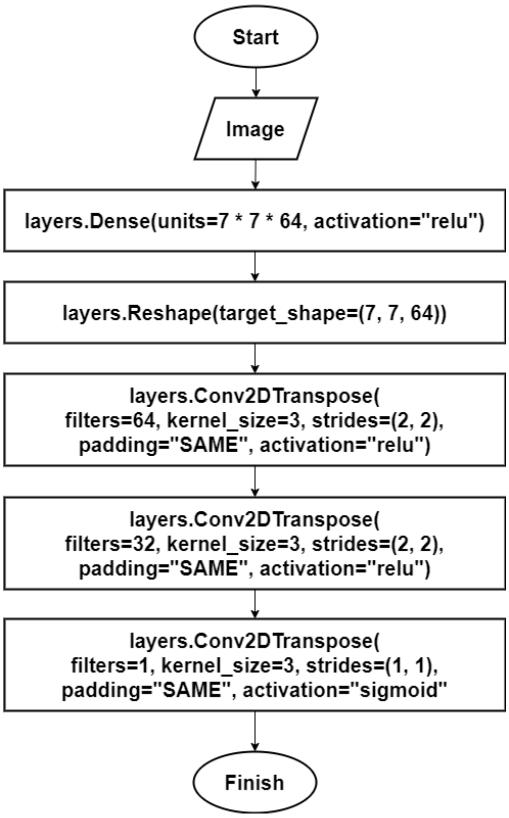
Start
Image
layers.Dense(units=7 * 7 * 64, activation="relu")
layers.Reshape(target_shape=(7, 7, 64))
layers.Conv2DTranspose(
filters=64, kernel_size=3, strides=(2, 2),
padding="SAME", activation="relu")
layers.Conv2DTranspose(
filters=32, kernel_size=3, strides=(2, 2),
padding="SAME", activation="relu")
layers.Conv2DTranspose(
filters=1, kernel_size=3, strides=(1, 1),
padding="SAME", activation="sigmoid"
Finish

Figure 8
WGAN Generator Model

The WGAN generator model uses a dense layer with 7*7*64 units with relu activation, a reshape layer with target_shape 7,7,64, 2 Conv2DTranspose layers with 64 and 32 filters and relu activation, and a Conv2DTranspose layer with 1 filter and activation sigmoid.

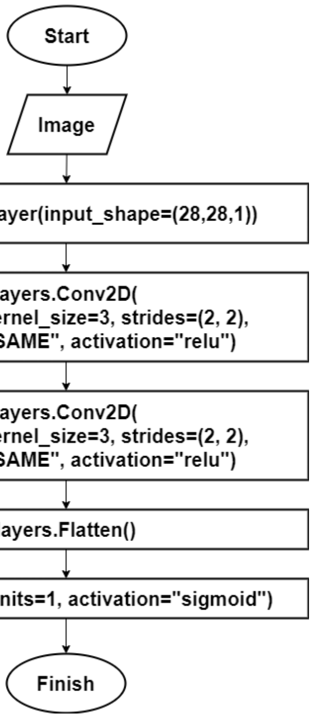
Start
Image
layers.InputLayer(input_shape=(28,28,1))
layers.Conv2D(
filters=32, kernel_size=3, strides=(2, 2),
padding="SAME", activation="relu")
layers.Conv2D(
filters=64, kernel_size=3, strides=(2, 2),
padding="SAME", activation="relu")
layers.Flatten()
layers.Dense(units=1, activation="sigmoid")
Finish

Figure 9
WGAN Critic Model

The critical WGAN model uses an input layer with an input shape of 28,28,1, 2 Conv2D layers with 32 and 64 filters respectively, a flatten layer, and a Dense layer with unit 1 and sigmoid activation.

The training process uses the Adam optimizer on the generator model with a value of 0.0001 and beta_1 0.5 and RMSprop on the critical model with a value of 0.0005 and applies a gradient penalty and epoch 150 with each 90 second epoch resulting in a discriminator loss -0.05456067994236946 and a generator loss of 0.5148118138313293, the resulting image as follows.

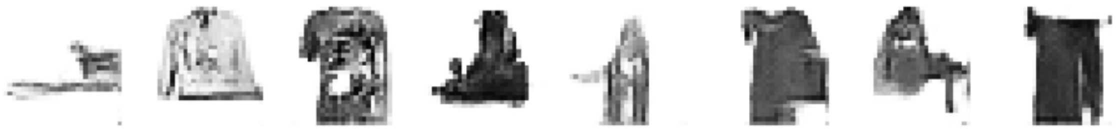

Fig 10
WGAN Image Result

Conclusion

After testing the DCGAN and WGAN models using almost similar models, it can be concluded that the image quality produced by WGAN is better, but with a longer model training process of 90 seconds per epoch because WGAN uses a gradient penalty, while the DCGAN training process faster, namely 60 seconds but with poor image quality.

BIBLIOGRAFI

Brownlee, Jason. (2019). How to Develop a Wasserstein Generative Adversarial Network (WGAN) From Scratch. Retrieved April 20, 2022, from Machine Learning Mastery website: https://machinelearningmastery.com/how-to-code-a-wasserstein-generative-adversarial-network-wgan-from-scratch/#:~:text=The Wasserstein Generative Adversarial Network,the quality of generated images. Google Scholar

Dutta, Himanshu. (2020). DCGAN Under 100 Lines of Code. Retrieved April 20, 2020, from Medium website: https://medium.com/swlh/dcgan-under-100-lines-of-code-fc7fe22c391. Google Scholar

Hui, Jonathan. (2018). GAN — Wasserstein GAN & WGAN-GP. Retrieved April 20, 2020, from Medium website: https://jonathan-hui.medium.com/gan-wasserstein-gan-wgan-gp-6a1a2aa1b490. Google Scholar

Kaveri, V. Vijeya, Meenakshi, V., Deepan, T., Dharnish, C. M., & Haarish, S. L. (2021). Image Generation for Real Time Application Using DCGAN (Deep Convolutional Generative Adversarial Neural Network). *Turkish Journal of Computer and Mathematics Education*, *12*(11), 617–621. Google Scholar

Lee, Hansoo, Kim, Jonggeun, Kim, Eun Kyeong, & Kim, Sungshin. (2020). Wasserstein generative adversarial networks based data augmentation for radar data analysis. *Applied Sciences (Switzerland)*, *10*(4). https://doi.org/10.3390/app10041449. Google Scholar

Praramadhan, Anugrah Akbar, & Saputra, Guntur Eka. (2021). *Cycle Generative Adversarial Networks Algorithm With Style Transfer For Image Generation*. 1–12. Retrieved from http://arxiv.org/abs/2101.03921. Google Scholar

Santosa, Pratama Yoga, Rachmawati, Ema, Agung, Tjokorda, & Wirayuda, Budi. (2021). Translasi Citra Malam Menjadi Siang Menggunakan Deep Convolutional Generative Adverserial Network. *EProceedings of Engineering*, *8*(1). Google Scholar

Suran, Abhisek. (2020). Deep Convolutional Vs Wasserstein Generative Adversarial Network. Retrieved April 20, 2022, from Towards Data Science website: https://towardsdatascience.com/deep-convolutional-vs-wasserstein-generative-adversarial-network-183fbcfdce1f. Google Scholar

Weng, Lilian. (2019). *From GAN to WGAN*. Retrieved from http://arxiv.org/abs/1904.08994. Google Scholar

Widjojo, Daniel, Palit, Henry Novianus, & Tjondrowiguno, Alvin Nathaniel. (2020). *Menghasilkan Background Game Music dengan Menggunakan Deep Convolutional Generative Adversarial Network*. *XIII*. Google Scholar