Proceedings of the 3rd International Conference on Signal Processing and Machine Learning
DOI: 10.54254/2755-2721/5/20230604

Investigation of influence of additional convolutional and max-pooling layers in general adversarial network

Zhibin Deng

College of Letter and Science, University of California, Santa Barbara, CA 93107, United States of America

zhibin@ucsb.edu

Abstract. General Adversarial networks (GANs) are distinctively sensitive models to variation of parameters. Thus, a variable-control measurement is necessary to evaluate the performance of changing the parameters (layers) in GANs. This paper focuses on the performance of DCGANs. It reviews author's experiment about the performance of DCGAN when applying the model to small datasets (MNIST, fashion- MNIST, cifar10, and human faces). This paper aims to discuss the influence of additional convolutional layers, max-pooling layers, and MLP in the generator and discriminator when dealing with small datasets. Using variable controlling, it is noteworthy that more convolutional layers can bring more accuracy for RGB images but not for single-channel images. MLP generators or discriminators are prone to model crashes. The respective analysis of each dataset's result is provided later, which proposes that the layer's influence is unstable among single-channel pictures. However, due to the complexity it brings, more layers can potentially increase the stability and convergence of RGB images. Some discussions for improving current experiments are listed, and some future research directions utilizing more advanced and matured methods are proposed.

Keywords: DCGAN, CNN, Convolutional Layers, Max-pooling Layers, Image Generation.

1. Introduction

GAN has become an important research focus for machine learning in recent years. It is an artificial intelligence algorithm to generate more examples of specific datasets from their estimated probability distribution. GANs consists of generators and discriminators, two crucial structural networks. The generator can create examples from existing distribution of datasets using random input vectors, and the discriminator evaluates whether the generated examples are supposed to be "true" or "false". GANs' training involves the evolvement of generators and discriminators, which causes both sides to compete with each other until the loss between real and fake data is smaller than the threshold [1].

Convolutional neural networks (CNNs), composed of generators and discriminators, are usually implemented by networks consisting of convolutional and other kinds of fully connected layers, such as global-average or max-pooling layers [2]. Thus, the generator and the discriminator in a typical GAN can be considered as CNNs that contain multiple layers. The advantage of using CNNs is that CNNs offer options that enable decreasing the parameters within the network and simultaneously reconcile some negative side effects [3].

The structures of GANs have been extended to ensure that one can utilize them under various circumstances and adapt them to different applications. For instance, in terms of super-resolution

DOI: 10.54254/2755-2721/5/20230604

images, SRGAN super-resolves a photo image with a 4x upscaling factor [4]. The image-to-image translation, which enables gender transformation and face swapping, is also supported by GAN because an image encoder is trained efficiently [5]. Also, Conditional GAN (CGAN) is widely used in face aging, where the CGAN can aesthetically generate a human face image, and some parameters can support the manually aging and rejuvenating effects of images [6].

However, there are some inherent drawbacks of GAN. For instance, GANs are usually prone to model collapse and vanishing gradients (if the discriminator performs too well, the gradient will be very close to zero, and the training will become extremely slow). They also lack appropriate evaluation metrics, so it is not easy to evaluate the accuracy and efficiency of networks. Thus, Wasserstein Distance is utilized in the Wasserstein GAN (WGAN) to substitutes for the standard GAN loss function, so some potential problems from the original GAN are partly ameliorated [7].

Because a modern GAN is usually composed of various convolutional layers, judging the stability and efficiency of GAN when using various quantities and classes of convolutional layers is necessary and essential. This paper aims to analyze the influence of additional convolutional layers and max-pooling layers in the generator when dealing with four particular small-scale datasets: MINST, fashion-MINST, cifar10, and a human face dataset.

The experiment contains two parts: Controlling parameters for single-channel and three-channel (RGB) images. For single-channel images, additional convolutional or max-pooling layers are tested; for RGB images, additional convolutional layers or MLP generators/discriminators are tested. The results show that convolutional or max-pooling layers are ineffective for single-channel images. However, more convolutional layers do bring a positive impact on RGB images. Also, MLPs are prone to model crashes, so some further improvement should be considered.

2. Theory review and methodology

2.1. The basic theory behind DCGAN

The training of Convolutional GANs is often unstable. Thus, some particular constraints is implemented on the architectural topology of Convolutional GANs to ensure that the training is relatively stable under various circumstances, transferring them into Deep Convolutional GANs (DCGANs) [8]. As referred to in the introduction, the intrinsic and most essential parts of GANs are their generators and discriminators. The underlying principles for the adversarial game between generators and discriminators of DCGAN refer to the minimax problem. It is known that the GAN loss function is separated into two individual parts: generator loss and discriminator loss (1).

$$E_x\left[\log\left(D(x)\right)\right] + E_z\left[\log\left(1 - D\left(G(z)\right)\right)\right] \tag{1}$$

Training the generator minimizes the generator loss value, so the smaller the value can lead to the more vital generation ability. The discriminator cannot distinguish between the actual dataset samples and outputs (2). Whether a generator is penalized or not is determined by whether it causes the discriminator to fail to distinguish between the actual dataset samples and outputs. The generator loss formula determines the possibility that the discriminator labels the output as fake or not.

$$\nabla_{\theta_g}\frac{1}{m}\sum_{i=1}^{m} log\left(1 - D\left(G\left(z^{(i)}\right)\right)\right) \tag{2}$$

Maximizing the discriminator loss value helps to punish the situation that the discriminator classifies the input wrong (3). The function Log(D(x)) represents the statistical probability in which the generator creates a standard fake image.

$$\nabla_{\theta_d}\frac{1}{m}\sum_{i=1}^{m}\left[logD\left(x^{(i)}\right) + \log(1 - D\left(G\left(z^{(i)}\right)\right))\right] \tag{3}$$

DOI: 10.54254/2755-2721/5/20230604

2.2. Convolutional and max-pooling layers

In this experiment, there are some primary parameters including strides and padding inside the convolutional layer. It is hypothesised that the next layer's node has many overlaps in correspondence of its neighbors by observing the nearby regions. The overlap can be interchanged through the manipulation of strides. Meanwhile, one potential problem is that the convolution step is the loss of information that might exist on the image's border. Because they are only captured when the filter slides, they never have the opportunity to be observed during the training [3]. When "padding" is set to be "same," zero-padding is applied. The problem is efficiently resolved, and the output size can also be managed through padding. Another critical parameter in the CNN is the max-pooling layer, which is able to divide the image into a certain number of sub-region rectangles and only returns the maximum value among the values of that sub-region [3].

2.3. Methodology, platform and code/data sources

The methodology for all series of testing in the experiment is relatively simple: controlling variable. Three variables are considered in the experiment: the number of epochs, datasets, and additional convolutional / max-pooling layers. When performing variable controls, one variable is set at different levels while other variables remain unchanged. Table 1 and Table 2 represent the specific levels of each variable. When all experiments are done, the results are compared for each class of variable, and an analysis of the consequence and validity of changing each variable is provided.

Table 1. MNIST, fashion-MNIST, cifar10.

Variable	Range
Number of training epochs	15, 30, 45, 60
Number of Conv2DTranspose Layers in the generator	0-1 additional layers
Max-pooling Layers in the generator	0-1 additional layers

Table 2. Human face.

Variable	Range
Number of training epochs	10, 20, 30, 40
Number of Conv2DTranspose Layers in the generator	3 or 5
MLP generator	Yes or no
Number of Conv2DTranspose Layers in the discriminator	3 or 5
MLP discriminator	Yes or no

Four datasets were utilized during the training process. The first dataset is the MNIST, consisting of 60,000 training images and 10,000 testing images taken from public hand-written records. The second dataset is the fashion-MNIST. Although it's similar to MNIST, it contains the same number of single-channel pictures of fashion products on Zalando's official website [9]. Cifar-10, an RGB datasets being used, is a multi-class dataset that contains 60,000 32 × 32 colored images, and the images are categorized into ten different classes [10]. In addition, a dataset containing 2,300 human faces was implemented.

For MNIST, fashion-MNIST, and cifar-10, TensorFlow is utilized in constructing the DCGAN model. TensorFlow is a machine learning system that operates at large scales and in various kinds of environments [11]. The code sources are collected from the TensorFlow official website, demonstrating how to build a typical DCGAN that can be used to generate hand-written digits in the MNIST dataset. For Human faces, a dataset containing above 2,300 human faces was utilized. Pytorch was utilized to construct GAN for generating fake human faces [12]. The Pytorch code for GAN training was downloaded from the official website of the Pytorch online tutorial. For simplicity, only three generators and three discriminators in the code were adapted during the experiment.

DOI: 10.54254/2755-2721/5/20230604

3. Results, analysis, and future improvements

To ensure that the results for each GAN are relatively stable, each training was repetitive, so the training was done twice, and the results were compared. The results for different datasets and the analysis for the potential reasons behind the difference in those results are discussed below.

3.1. Results for MNIST

The original GAN has its generated image converging after approximately 40 epochs of training. Generally, no significant oscillation of digit shapes was detected after 40 epochs. However, it could be observed that current errors between the generated image and the actual image were not negligible. When adding one additional Conv2DTranspose layer with strides equal to (2, 2) and revising the subsequent layers, some of the generated digits become closer to the actual samples in the MNIST datasets. The converging epochs (the number of epochs when the generated images became convergent) are approximately the same as the original GAN. When adding one additional Max-pooling layer with no strides and good padding, it was noticeable that the generated images became convergent slightly (several epochs) later than the previous two GANs. While some generated images became less distinguishable from the actual images, others were even less recognizable as hand-written digits, as shown in Figure 1.

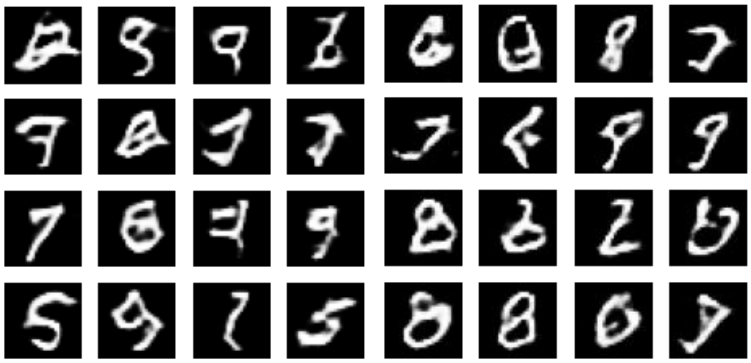

Figure 1. Generated image with one additional convolutional layer or one max-pooling layer.

3.2. Results for fashion-MNIST

Similar to the MNIST dataset's result, the original GAN's generated image converged to a specific range after approximately 40 epochs. However, since the fashion_MNIST contains various images of luxury items, it would be hard to observe whether the digits in the fake images were similar to the real ones or not. The shape of each image after 40 epochs was usually stable. When adding one additional Conv2DTranspose layer with strides equal to (2, 2) and revising the subsequent layers, the converging periods were significantly larger (around the 50th epochs). The oscillations of image shapes were still apparent. When adding one additional Max-pooling layer with no strides and valid padding, one exciting discovery was that the convergence of images was not balanced between different kinds of images. For instance, the generated images of shirts eventually converged. In contrast, the images of shoes were still unstable even after 60 epochs of training, as shown in Figure 2.

DOI: 10.54254/2755-2721/5/20230604

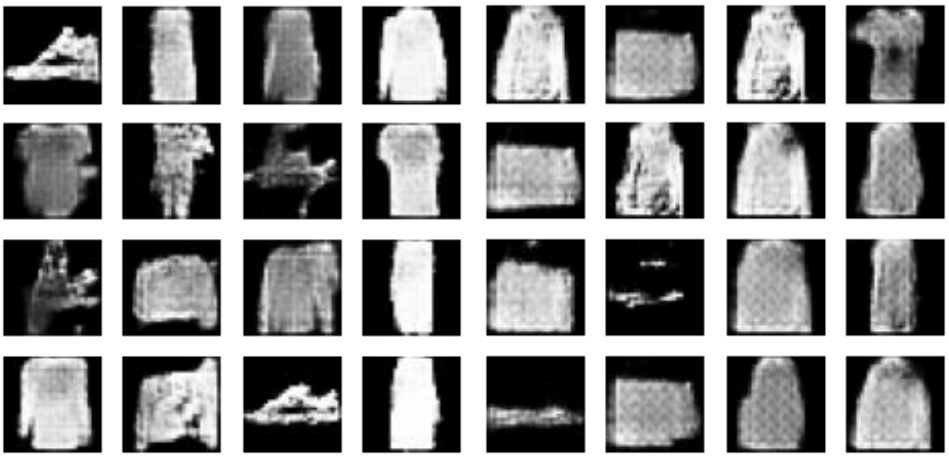

Figure 2. Generated image with one additional convolutional layer or one max-pooling layer.

3.3. Results for Human faces

When evaluating the loss function for human face generation, it turned out that generators containing more convolutional layers are more likely to produce images with fewer generator and discriminator errors. When looking closely at the image generated in the cycles, it was noticeable that if the discriminator contains more convolutional layers, then the GAN would create less blurred images (here, "blurred" implies when the distinction between different colors is not clear). Similarly, generators containing more convolutional layers could generate images with more appropriate shapes and colors. However, it was also discovered that the MLP discriminator created images that were not convergent to the real images. When MLP generators were implemented with any discriminator, those combined GANs quickly failed to generate more images. They kept generating the same image over time, a "model crash" situation. One could see that the discriminator loss skyrocketed to 100 when MLP generators were implemented with a five-convolutional-layer generator or MLP discriminators, as shown in Figure 3.

Figure 3. Generated Image with 3 convolutional layers in the generator, 3 convolutional layers in the discriminator.

DOI: 10.54254/2755-2721/5/20230604

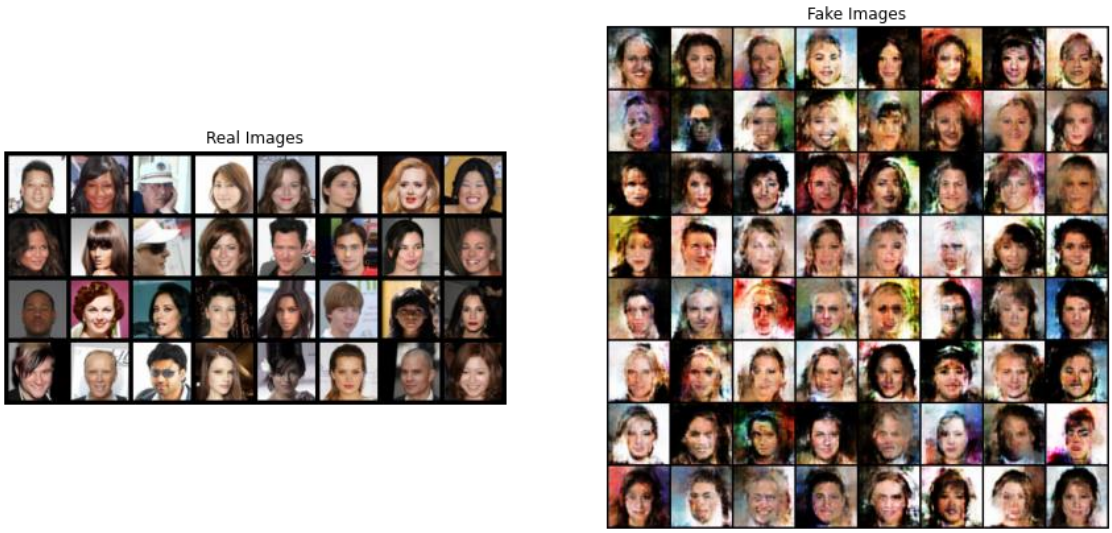

Figure 4. Generated Image with 3 convolutional layers in the generator, 5 convolutional layers in the discriminator.

3.4. Results for Cifar10

Using cifar10 as the dataset, it was observed that the training failed to generate the required images close to the real images in cifar10. Most generated images have simple RGB structures containing only pure green, blue and yellow pixel points. The oscillation between every two generated images was often apparent, implying that the DCGAN failed to find a particular pattern among the cifar10 datasets, as shown in Figure 5.

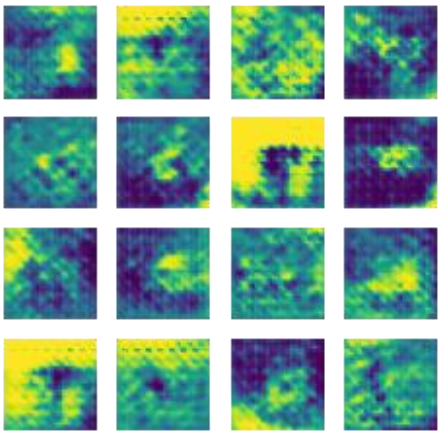

Figure 5. Abnormal results from cifar10.

3.5. Analysis

For MNIST and fashion-MNIST, since the hand-written digits are rather simple, more convolutional layers in the generator cannot contribute to the stability. When viewing a randomly generated image of max-pooling layers, it can be seen that the pixels with larger gray levels are excluded from the edge of the image, which is determined by the innate process of max-pooling. Thus, max-pooling may work better when dealing with hand-written digits (since the digit is usually placed in the middle of the image), but better performance is not guaranteed. For fashion-MNIST, the implementation of the max-pooling layer, in contrast, causes the generation of low-quality images.

For cifar10, the abnormal generation of RGB images can be partly explained by reviewing the categorization of cifar10's structure. Cifar10 consists of 10 classes of images, and each class consists of different images in comparison to other classes. Thus, the generator cannot easily learn from the

DOI: 10.54254/2755-2721/5/20230604

discriminator how to make a distinction between real and fake images due to high variation of images in cifar10.

For human faces, more convolutional layers in the generator imply that the generator is able to generate more sophisticated images, and more layers in the discriminators in the discriminator means that it becomes even stricter about the distinction standard between images, so the generator has to produce images that are closer the real ones. For MLP, it turned out that it generates much more images at one time, so the gradient loss becomes dangerously small and leads to model crashes.

3.6. Future improvement

This experiment utilizes generators and discriminators with 3-7 convolutional layers and, at most, one max-pooling layer. However, more layers can be implemented to further test the question of whether more layers can be helpful in increasing the stability and accuracy of GAN. For MNIST and fashion_MNIST, a loss function visualization should be added later so that the loss variation can be presented better during the training. For cifar10, it will probably be helpful to first divide the dataset into ten groups and use different GANs to train them respectively. Also, it will be more efficient to utilize only one platform, either Pytorch or TensorFlow, to do the training. The repetition of the experiment could be further strengthened (for instance, doing the same experiment at least 15 times) to ensure validity. Some advanced methods, such as, can be implemented instead [13]. Some other ways of application may also be the future goal of GAN experiments. Other kinds of GAN, such as style-based GAN, a more advanced generator architecture that leads to an automatically learned, unsupervised division of high-level attributes, can also be considered. The least-square GAN, implemented with the least squares loss function inside the discriminator, owns a more stable training and generates a higher-quality image [14, 15]. In terms of generating the image, it will be promising to utilize this GAN in future experiments.

4. Conclusion

Four datasets were utilized in this experiment. Single-channel and RGB images are treated, respectively, and the results indicated that additional convolutional layers, or max-pooling layers, cannot guarantee the increase of stability or accuracy of GAN for single-channel images, but additional convolutional layers do have some positive effect in enhancing the performance of RGB image generation. It is also noticeable that the MLP generator or discriminator is not efficient in the production of RGB image: model crash is frequent during the training. The experiment itself has some intrinsic problems, and it can be further conducted and improved.

References

[1] Goodfellow, I., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., Courville, A., & Bengio, Y. (2020). Generative Adversarial Networks. Communications of the ACM, 63(11), pp.1–4. https://doi.org/10.1145/3422622

[2] Creswell, A., White, T., Dumoulin, V., Arulkumaran, K., Sengupta, B., & Bharath, A. A. (2018). Generative Adversarial Networks: An overview. IEEE Signal Processing Magazine, 35(1), pp.1. https://doi.org/10.1109/msp.2017.2765202

[3] Albawi, S., Mohammed, T. A., & Al-Zawi, S. (2017). Understanding of a convolutional neural network. 2017 International Conference on Engineering and Technology (ICET), pp.2-6. https://doi.org/10.1109/icengtechnol.2017.8308186

[4] Ledig, C., Theis, L., Huszar, F., Caballero, J., Cunningham, A., Acosta, A., Aitken, A., Tejani, A., Totz, J., Wang, Z., & Shi, W. (2017). Photo-realistic single image super-resolution using a generative adversarial network. 2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), pp.4681. https://doi.org/10.1109/cvpr.2017.19

[5] Dong, H., Neekhara, P., Wu, C., & Guo, Y. (2017). Unsupervised Image-to-Image Translation with Generative Adversarial Networks. Arxiv, pp.1–4. https://doi.org/https://doi.org/10.48550/arXiv.1701.02676

[6] Antipov, G., Baccouche, M., & Dugelay, J.-L. (2017). Face aging with conditional generative adversarial networks. 2017 IEEE International Conference on Image Processing (ICIP), pp.1.

DOI: 10.54254/2755-2721/5/20230604

https://doi.org/10.1109/icip.2017.8296650

[7] Weng, L. (2019). From GAN to WGAN. Arxiv, pp.5–11. https://doi.org/https://arxiv.org/abs/1904.08994

[8] Radford, A., Metz, L., & Chintala, S. (2016). Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks. Arxiv, pp.1. https://doi.org/https://doi.org/10.48550/arXiv.1511.06434

[9] Xiao, H., Rasul, K., & Vollgraf, R. (2017). Fashion-MNIST: a Novel Image Dataset for Benchmarking Machine Learning Algorithms. Arxiv, pp.2. https://doi.org/https://doi.org/10.48550/arXiv.1708.07747

[10] Abouelnaga, Y., Ali, O. S., Rady, H., & Moustafa, M. (2016). CIFAR-10: Knn-based ensemble of classifiers. 2016 International Conference on Computational Science and Computational Intelligence (CSCI), pp.1. https://doi.org/10.1109/csci.2016.0225

[11] Abadi, M., Barham, P., Chen, J., Chen, Z., Davis, A., Dean, J., Devin, M., Ghemawat, S., Irving, G., Isard, M., Kudlur, M., Levenberg, J., Monga, R., Moore, S., Murray, D. G., Steiner, B., Tucker, P., Vasudevan, V., Warden, P., … Zheng, X. (2016, November). {tensorflow}: A system for {large-scale} machine learning. USENIX. Retrieved September 19, 2022, from https://www.usenix.org/conference/osdi16/technical-sessions/presentation/abadi

[12] Imambi, S., Prakash, K. B., & Kanagachidambaresan, G. R. (2021, January 23). Pytorch. SpringerLink. Retrieved September 19, 2022, from https://link.springer.com/chapter/10.1007/978-3-030-57077-4_10

[13] Liu, M.-Y., Huang, X., Yu, J., Wang, T.-C., & Mallya, A. (2021). Generative adversarial networks for image and video synthesis: Algorithms and applications. Proceedings of the IEEE, 109(5), pp.839. https://doi.org/10.1109/jproc.2021.3049196

[14] Karras, T., Laine, S., & Aila, T. (2019). A Style-Based Generator Architecture for Generative Adversarial Networks. CVPR 2019 Open Access, pp.4401.

[15] Mao, X., Li, Q., Xie, H., Lau, R. Y. K., Wang, Z., & Smolley, S. P. (2017). Least Squares Generative Adversarial Networks. ICCV 2017 Open Access, pp.2794.