Review and Re-thinking of significant GAN models

Zhizuo Zhang
UC San Diego
zzhizuo@eng.ucsd.edu

Chenhao Zhou
UC San Diego
c3zhou@eng.ucsd.edu

Abstract

In statistical learning, two main models are called the generative model and the discriminative approach. Generative Adversarial Networks, or GANs for short, is a method using neural networks to simulate the generative model and discriminative model. The powerful property of learning high-dimensional and complex data distribution allows GANs to be applied in various problems, such as image synthesis, semantic image editing, style transfer, image super-resolution and image classification. The aim of this paper is to review some creative GANs models, to distinguish the difference and motivation among those papers. On top of that, we use NAS(neural network architecture search) to find the optimal architecture for GANs to accomplish certain tasks.

1. Introduction

Generative adversarial networks(GANs) have become a hot research topic in recent years. Yann LeCun, a legend in deep learning, Turing Award winner, said that "GANs is the most interesting idea in the last 10 years in machine learning." There are a large number of papers related to GANs from 2014 until now.

GAN has two models: a generative model $\mathcal{G}$ and a discriminative model $\mathcal{D}$. On one hand, the aim of $\mathcal{G}$ is to make realistic images which have the same pattern in a certain dataset; On the other hand, the aim of $\mathcal{D}$ is to distinguish whether the given input is real or fake. Both $\mathcal{G}$ and $\mathcal{D}$ will be trained simultaneously and in competition with each other which will eventually lead to better performances for both of them.

Intuitively, how can we train a generative model to generate the image without knowing direct information like generative distribution, high-level features and etc? Crucially, the generator has no direct access to real images. $\mathcal{G}$ just uses the information while it interacts with the $\mathcal{D}$. At the same time, the discriminative model $\mathcal{D}$ receives the synthetic samples and samples which are drawn from the stack of real images. The error signal to the discriminator is provided through the simple ground truth of knowing whether the image came from the real stack or from the generator. And the same error signal, via the discriminator, can be used to train the generator, leading it towards being able to produce forgeries of better quality.

As for the architecture of the GAN, in the very beginning, the networks that represent the generator and discriminator are typically implemented by multi-layer networks(MLP). A few years later, all kinds of variant GANs showed up like CNN, RNN or more complicated architectures. However, whatever they changed, the generator and discriminator networks must be differentiable or at least can be defined gradient in order to get trainable, though, in some times, it is not necessary for them to be directly invertible. There comes up a question, how does the generator generate data. The generator can mainly be considered as a mapping from the representation space, called a latent space, which is normally a high-dimensional Gaussian distribution space, to the space of the data. We can express this process more formally as $\mathcal{G} : \mathcal{G} \to R^{|x|}$, where $z \in R^{|x|}$ is a sample from the latent space $\mathcal{Z}$, $x \in R^{|x|}$ is a sample from the real image and $| \cdot |$ denotes the number of dimensions.

In a basic GAN, the discriminator network $\mathcal{D}$ can be characterized as a function that maps from an image data into a probability that the given data is from the real data distribution or the generator distribution: $\mathcal{D} : \mathcal{D}(x) \to (0, 1)$. For a related generator network, $\mathcal{G}$, the discriminator, D, may be trained to classify images as either being from the real training data or the fake data generated from the generator network $\mathcal{G}$. When the discriminator is trained into optimal, it should be able to perfectly get the classify results which have the same ratio of the given data. If the generator distribution is able to match the real data distribution, then the discriminator will be confused, so the generator should predicting 0.5 for all inputs, which means that the discriminator $\mathcal{D}$ only can guess without any prior information.

On top of the interesting academic problems related to training and constructing GANs, the generalization ability of discriminator $\mathcal{D}$ and generator $\mathcal{G}$ is hard to determined. First of all, from the preceding statement, we can easily observe that the evaluation of discriminator $\mathcal{D}$ is related to the quality of training data which partly generated by genera-

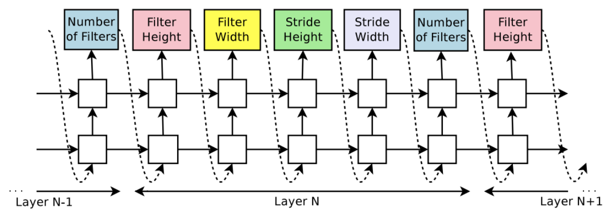

Figure 1. neural network architecture search

tor $\mathcal{G}$. At the same time, the generator $\mathcal{G}$ needs the error signal from discriminator $\mathcal{D}$ to get better result. It is really hard to separately define whether an architecture for discriminator $\mathcal{D}$ and for generator $\mathcal{G}$ is good or bad. We use the method mentioned in the paper[12] which use RNN to search the hyper-parameter and architecture for a given task and a given neural network, as we can see from the figure. After that, we will use cross-validation method to evaluate certain classical GAN models to have an other view to evaluate their performance.

2. Related Work

2.1. Generative algorithms

Generative algorithms can be classified into two classes: explicit density model and implicit density model.

2.1.1 Explicit density model

An explicit density model assumes the distribution and utilizes true data to train the model containing the distribution or fit the distribution parameters. When finished, new examples are produced utilizing the learned model or distribution. The explicit density models include maximum likelihood estimation (MLE), approximate inference[5], and Markov chain method[4]. These explicit density models have an explicit distribution, but have some limitations.

2.1.2 Implicit density model

An implicit density model does not directly estimate or fit the data distribution. It produces data instances from the distribution without an explicit hypothesis and utilizes the produced examples to modify the model. Prior to GANs, the implicit density model generally needs to be trained to utilize Markov chain-based sampling. GANs belong to the directed implicit density model category.

2.2. Adversarial idea

The adversarial idea has been successfully applied to many areas such as machine learning, artificial intelligence, computer vision, and natural language processing. The recent event that AlphaGo[10] defeats the world's top human player engages public interest in artificial intelligence. The intermediate version of AlphaGo utilizes two networks competing with each other.

Adversarial machine learning is a minimax problem. The defender, who builds the classifier that wee want to work correctly, who builds the classifier that we want to work correctly, is searching over the parameter space to find the parameters that reduce the cost of the classifier as much as possible. Simultaneously, the attacker is searching over the inputs of the model to maximize the cost.

The adversarial idea exists in adversarial networks, adversarial learning, and adversarial examples. However, the may have different objectives.

2.3. Algorithms

In this section, we will analyze the mathematical knowledge and background of GANs.

2.3.1 Generative Adversarial Networks (GANs)

The GANs' main idea has been discussed in the previous part. Briefly, GANs represent a mapping from noise space or latent space to data space by discriminator $\mathcal{G}$ and classify an input image by discriminator $\mathcal{D}$. The discriminator $\mathcal{D}$ is trained to maximize the probability of giving the correct label to both training data and fake samples generated from the generator $\mathcal{G}$ .The $\mathcal{G}$ is trained to minimize $log(1 - D(G(z)))$ simultaneously.

2.3.2 mathematical statement

The loss function of GANs[2] is

$$\begin{aligned}\min_G \max_D V(D,G) = E_{x\sim p_{data}(x)}[logD(x)]+ \\ E_{z\sim p_z(z)}[log(1-D(G(z))]\end{aligned} \tag{1}$$

As we can see from the loss function. The first part is the loss designed for discriminator $\mathcal{D}$ whose goal is to get the correct label. $log[D(x)$ is the cross-entropy loss between $[0,1]^T$ and $[1-D(x), D(x)]^T$. Similarly, the second term is the loss for generator $\mathcal{G}$, which is also a cross-entropy loss between $[0,1]^T$ and $[1-D(G(z)), D(G(z))]^T$. If it is a little bit confusing, we could separate the second term into two parts which first contain the successful classification result from $\mathcal{D}$ and unsuccessful classification result from $\mathcal{D}$, then it helps to understand the loss function's meaning.

For fixed $\mathcal{G}$, the optimal discriminator $\mathcal{D}$ is given by[2]:

$$D^*_G(x) = \frac{p_{data}}{p_{data}(x) + p_g(x)} \tag{2}$$

which basically means that if discriminator is trained into perfect, then $\mathcal{D}$ can classify each label into the correct class that the ratio of real is $D^*_g(x)$

The minimax formula can be formulated as:

$$\begin{aligned} C(G) &= \max_D V(D,G) \\ &= E_{x\sim p_{data}}[logD^*_G(x)] \\ &+ E_{z\sim p_z}[log(1-D^*_G(G(z)))] \\ &= E_{x\sim p_{data}}[logD^*_G(x)] \\ &+ E_{x\sim p_G}[log(1-D^*G(x))] \end{aligned} \tag{3}$$

2.4. GAN's variants

InfoGAN[1] proposes to decompose the input noise vector into two parts: z, which is seen as incompressible noise; c, which is called the latent code and will target the significant structured semantic features of the real data distribution. InfoGAN aims to solve

$$\min_G \max_D V_1(D,G) = V(D,G) - \lambda I(c; G(z,c)) \tag{4}$$

where $V(D,G)$ is the objective function of original GAN, $G(z,c$ is the generated sample, I is the mutual information, and λ is the tunable regularization parameter.

Conditional GANs(cGAN)[8] is another variant of GAN whose main difference from the original model is that both the input for discriminator $\mathcal{D}$ and generator $\mathcal{G}$ have a conditional variable y. The objective function of conditional GANs is:

$$\begin{aligned} \min_G \max_D V(D,G) = E_{x\sim p_{data}(x)}[logD(x|y)]+ \\ E_{z\sim p_z(z)}[log(1-D(G(z|y))] \end{aligned} \tag{5}$$

The benefit of the conditional label of y can be numerous. For instance, we can extract wanted data with the help of a conditional label. In GAN[2], the data generated by generator $\mathcal{G}$ is extracted or determined by the noise space $z \sim \mathcal{N}(0,I)$ which is chosen randomly. The mapping between the noise space and the distribution of training data is also learned while training. It means that during the training process, if a certain label of data occupies the most likelihood of noise space, then we may be more likely to get that certain label data, but not able to get the same density distribution of generator $\mathcal{G}$.

Based on cGANs, [3] gives many variants of GANs that samples conditioning on class labels, text, bounding box and keypoints. In [11][7], text to photo-realistic image synthesis is connected with stacked generative adversarial networks (StackGAN). cGANs have also been used for convolutional face generation, face aging, image translation, synthesizing outdoor images having specific scenery attributes, natural image description, and 3D-aware scene manipulation.

As we can see, all the preceding variants are seeking to modify the objective function which leads to better performance for GAN. DCGAN[9] provides significant contributions to GAN in that its suggested Convolutional Neural Network[6] architecture greatly stabilizes GAN training.

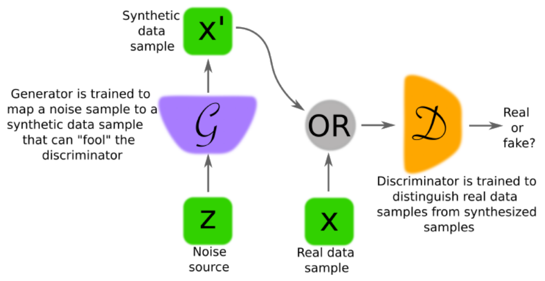
Synthetic data sample
X'
Generator is trained to map a noise sample to a synthetic data sample that can "fool" the discriminator
$\mathcal{G}$
OR
$\mathcal{D}$
Real or fake?
Discriminator is trained to distinguish real data samples from synthesized samples
Z
Noise source
X
Real data sample

Figure 2. illustration of GAN

By contrary, we can see that the architecture of previous GAN models is using multi-layer networks (MLP) to build generator $\mathcal{G}$ or to build discriminator $\mathcal{D}$ However, in [6], CNN kernels are able to extract information or pattern for the local area. Such ability can tremendously boost the performance of both generator and discriminator. Of course, DCGAN suggests that GAN models should follow the basic convolutional layer, batch norm layer and non-linear activation layer in the sequence which will get better results. It solves the instability of training GAN only through architecture, it becomes a baseline for modeling various GANs proposed later.

CycleGAN is a type of GAN which is designed to solve image-to-image translation. When paired training data is available, references can be used for these image-to-image translation tasks. However, reference can not be used for unpaired data (no input/output pairs), which was well solved by Cycle-consistent GANs (CycleGAN). CycleGAN is important progress for unpaired data. It is proved that cycle-consistency is an upper bound of the conditional entropy. CycleGAN can be derived as a special case within the proposed variational inference framework, naturally establishing its relationship with approximate Bayesian inference methods.

3. Method

3.1. Architecture

In this part, we will discuss more about the architecture of our models, like figure 1. Firstly, we start with the original version of GAN[2]. Like most multi-layer neural networks(MLP), both discriminator $\mathcal{D}$ and generator $\mathcal{G}$ are implemented by MLP. On top of that, the number of perceptrons is gradually going larger while the layers are deeper. For instance, in the original version, the generator $\mathcal{G}$ has 100, 128, 256, 512, 28*28 perceptrons each layer. The first 100 layer represents the dimension of the latent space and the following 128, 256, 512 are a common designation for MLP neural network. Then, in our NAS experiment, we will change not only the number of perceptrons for each MLP layer, but also add or reduce the total layer.

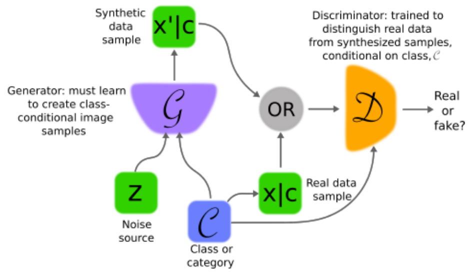
Synthetic data sample
x'|c
Discriminator: trained to distinguish real data from synthesized samples, conditional on class, $\mathcal{C}$
Generator: must learn to create class-conditional image samples
$\mathcal{G}$
OR
$\mathcal{D}$
Real or fake?
z
Noise source
x|c
Real data sample
$\mathcal{C}$
Class or category

Figure 3. illustration of cGAN

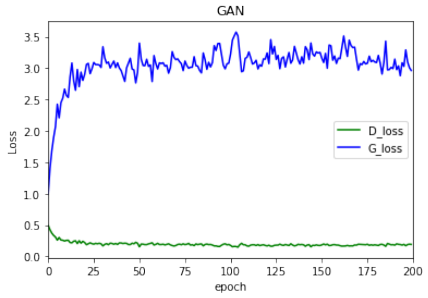

Figure 4. GAN loss curve: original version

As for cGANs, we can see from figure 2. The original version of cGAN has the following architecture: The generator $\mathcal{G}$ has 100+10, 128, 256, 512, 1024, 28*28 number of perceptrons for each MLP layer. The conditional variable will go through an embedding layer (a variant of MLP) which help the model to encode the conditional label. Each MLP layer consists of batch norm layer and leaky relu layer to get better non-linear representation for GAN. As for the discriminator $\mathcal{D}$, it has 10+28*28, 512, 512, 512, 1 number of perceptrons for each MLP layer. And during doing our experiments, we will also change all the parameters of channels.

As for DCGAN, its architecture is like the GAN[2]. However, all the MLP layers are substituted into a convolutional layer: The generator $\mathcal{G}$ has the architecture as 100, 128, 128, 64, 1. The first 100 is MLP layer for encoding the latent space and the following are all channels for the convolutional layer. Among the related layer, batch norm layer and upsample transmition are added. The discriminator $\mathcal{D}$ has 1, 16, 32, 64, 128 channels for each convolutional layer.

4. Experiment

In our experiment, we use the NAS to search hyper-parameters for our GAN models and check their performance and loss curve. We do all the experiments on the MNIST dataset.

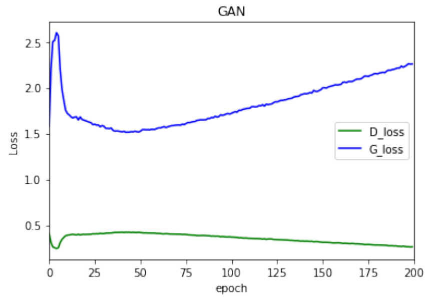

Figure 5. GAN loss curve: $\mathcal{G}$: 128, 256, 512, 1024; $\mathcal{D}$: 512, 512, 256

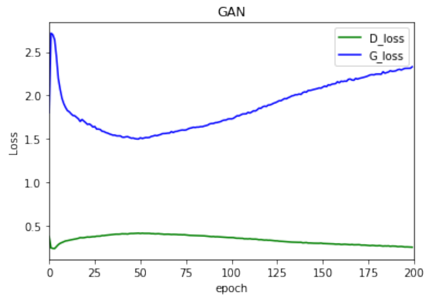

Figure 6. GAN loss curve: $\mathcal{G}$: 128, 256, 512, 1024; $\mathcal{D}$: 512, 1024, 512

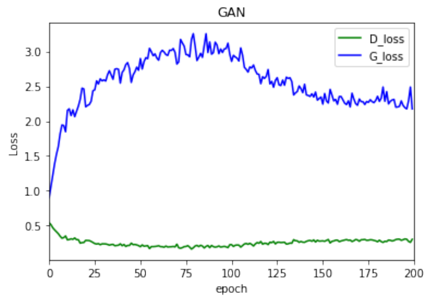

Figure 7. GAN loss curve: $\mathcal{G}$: 128, 256, 512, 1024, 1536; $\mathcal{D}$: 512, 1024, 512

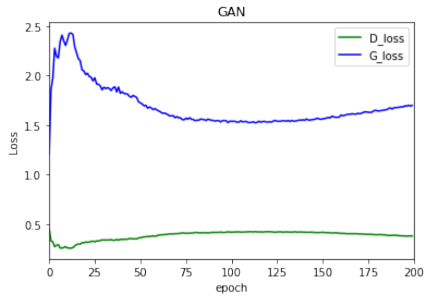

Figure 8. GAN loss curve: $\mathcal{G}$: 128, 256, 512, 1024, 1536; $\mathcal{D}$: 512, 512, 256

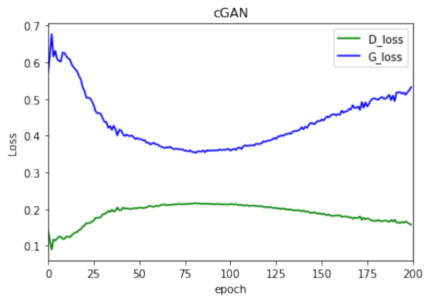

Figure 9. cGAN loss curve: $\mathcal{G}$: 128, 256, 512, 1024; $\mathcal{D}$: 512, 512, 512

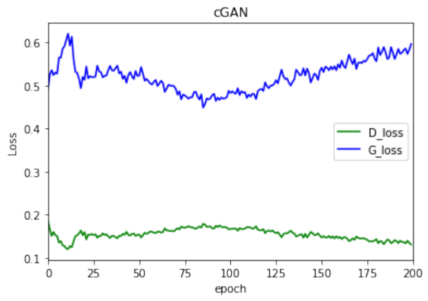

Figure 10. cGAN loss curve: $\mathcal{G}$: 128, 256, 512, 1024; $\mathcal{D}$: 512, 512, 512

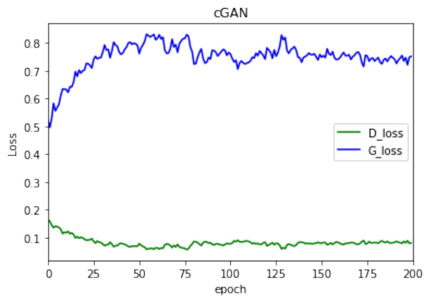

Figure 11. cGAN loss curve: $\mathcal{G}$: 128, 256, 512, 1024, 1536; $\mathcal{D}$: 512, 512, 512

4.1. Loss curve

As we can see from the results of the results, the number behind generator $\mathcal{G}$ and $\mathcal{D}$ is the number of perceptrons of MLP layer or the channel of convolutional layer. We present the loss curve for the original version of GAN[2] as figure4. Then we change the generator $\mathcal{G}$'s number of perceptrons of MLP as 128, 256, 512, 1024, and discriminator $\mathcal{D}$'s number of perceptrons of MLP as 512, 512, 256. Then figure 6, 7, 8 are all NAS results for GAN.

As contrast experiment, we set the same parameters in the cGAN as showed in figure 9, 10, 11, 12.

As the curve figure can show the hyper-parameter settings for DCGAN. From the training curve, we can see some very interesting truth. Firstly, the loss curve is not always going down or has the temptation to go down. It may

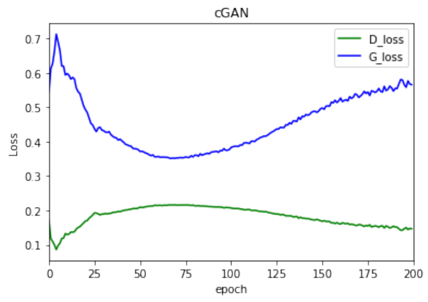

Figure 12. cGAN loss curve: $\mathcal{G}$: 128, 256, 512, 1024; $\mathcal{D}$: 512, 512, 1024

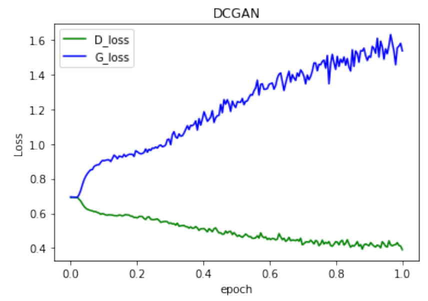

Figure 13. DCGAN loss curve (original version): $\mathcal{G}$: 128, 128, 128, 64; $\mathcal{D}$: 16, 32, 64, 128

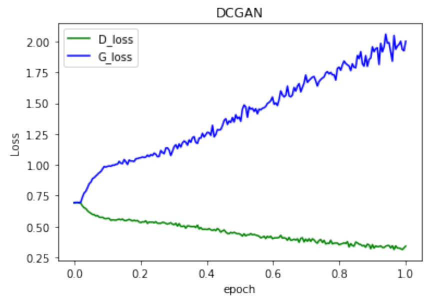

Figure 14. DCGAN loss curve (original version): $\mathcal{G}$: 128, 128, 128, 64; $\mathcal{D}$: 16, 32, 64, 128, 256

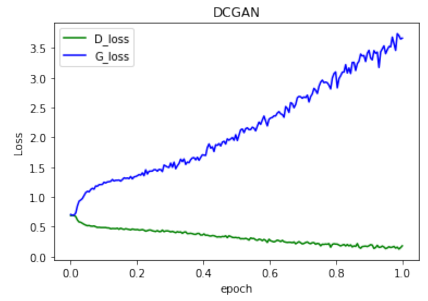

Figure 15. DCGAN loss curve (original version): $\mathcal{G}$: 128, 128, 64, 32; $\mathcal{D}$: 16, 32, 64, 128, 256

go upper for a while then go down. The reason why such a phenomenon happens may because that the loss function of $\mathcal{G}$ has relationship to $\mathcal{D}$ and vice versa. So the loss curve of

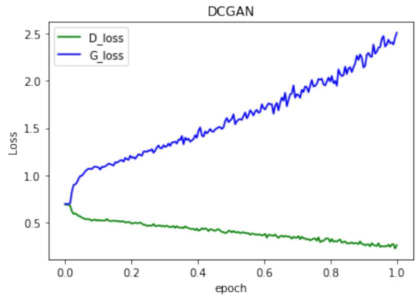

Figure 16. DCGAN loss curve (original version): $\mathcal{G}$: 256, 256, 64, 32; $\mathcal{D}$: 16, 32, 64, 128, 256

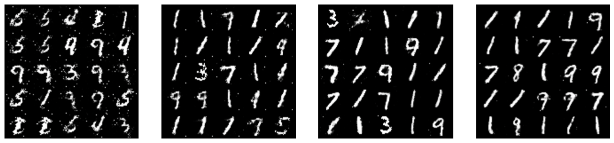

Figure 17. GAN (original version): $\mathcal{G}$: 128, 256, 512, 1024; $\mathcal{D}$: 512, 256. Epoch 50, 100, 150, 200

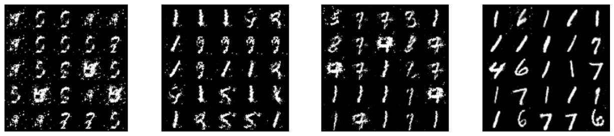

Figure 18. GAN : $\mathcal{G}$: 128, 256, 512, 1024; $\mathcal{D}$: 512, 512, 256. Epoch 50, 100, 150, 200

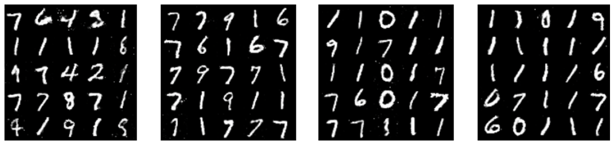

Figure 19. GAN : $\mathcal{G}$: 128, 256, 512, 1024; $\mathcal{D}$: 512, 512, 512. Epoch 50, 100, 150, 200

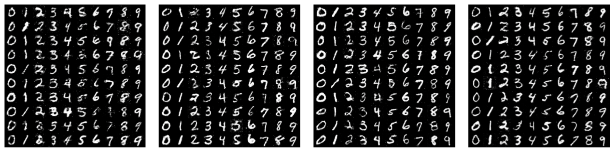

Figure 20. cGAN (original version): $\mathcal{G}$: 128, 256, 512, 1024; $\mathcal{D}$: 512, 512, 512. Epoch 50, 100, 150, 200

$\mathcal{G}$ has 2 factors that the $\mathcal{G}$ itself and thee $\mathcal{D}$. After analysis of such justification, we can better understand why such a strange curve occurs.

The loss curves for DCGAN only have one pattern: the loss for generator $\mathcal{G}$ is always being larger while the loss for discriminator $\mathcal{D}$ is always being smaller. We think that this may due to the power of the convolution neural network to fit computer vision tasks that help make the discriminator converge more quickly.

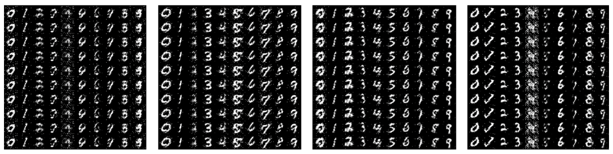

Figure 21. cGAN : $\mathcal{G}$: 128, 256, 512, 1024, 1536; $\mathcal{D}$: 512, 512, 512. Epoch 50, 100, 150, 200

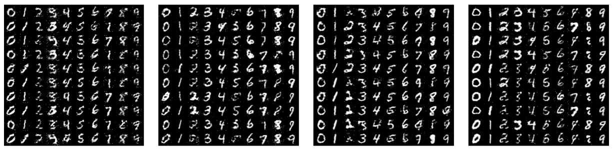

Figure 22. cGAN : $\mathcal{G}$: 128, 256, 512, 1024; $\mathcal{D}$: 512, 512, 256. Epoch 50, 100, 150, 200

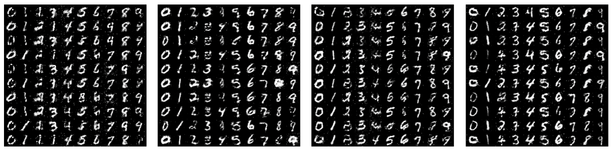

Figure 23. cGAN : $\mathcal{G}$: 128, 256, 512, 1024, 2048; $\mathcal{D}$: 512, 512, 512, 1024. Epoch 50, 100, 150, 200

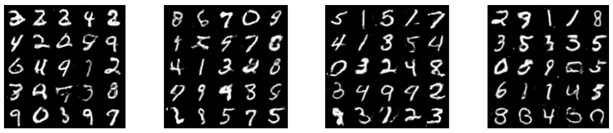

Figure 24. DCGAN (original version): $\mathcal{G}$: 128, 128, 64, 32; $\mathcal{D}$: 16, 32, 64, 128, 256. Epoch 50, 100, 150, 200

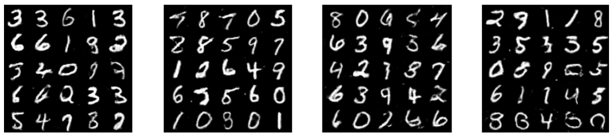

Figure 25. DCGAN : $\mathcal{G}$: 128, 128, 128, 64; $\mathcal{D}$: 16, 32, 64, 128, 256. Epoch 50, 100, 150, 200

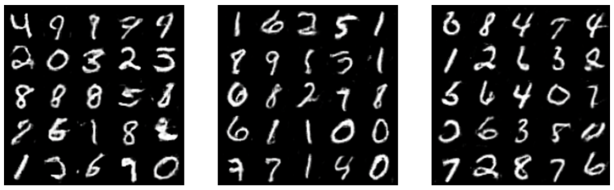

Figure 26. DCGAN : $\mathcal{G}$: 256, 128, 64, 32; $\mathcal{D}$: 16, 32, 64, 128, 256. Epoch 50, 100, 150 with early stopping method.

4.2. Generated image

From the data generated by each model: GAN, cGAN, DCGAN. We find some interesting facts. Firstly, the distribution of number is very biased in generator $\mathcal{G}$ of GAN[2]. The number 1, 7 and 9 are more likely to show up in the latent space. During the training process, the generated data from GAN at the very beginning is very poor but is getting

Method:$\mathcal{D}$ and $\mathcal{G}$	GAN	cGAN	DCGAN	CycleGAN
GAN	86.15	84.88	82.25	83.44
cGAN	89.71	90.21	89.98	89.23
DCGAN	93.81	93.51	91.19	93.01
CycleGAN	88.74	90.29	91.26	90.48

Table 1. The cross validation results table. (correct rate)

better. In the figure generated by cGAN, we can see that the distribution has a normal density because of the conditional label. However, if we go through the details of the images, we can see that the noise or the white spots are wide-spread among those images. As for the DCGAN's generated images, in the beginning, the generated images have already been very similar to the real images. After 200 epochs training, some digits showed in the results seem a little bit overfit.

4.3. Cross validation

The table shows us the four GAN models, GAN, cGAN, DCGAN, CycleGAN's cross validation results.

5. conclusion

In order to find GAN, cGAN, DCGAN, CycleGAN's ability of generator $\mathcal{G}$ and discriminator $\mathcal{D}$, we did amount of experiments. We first change the hyper-parameters of each MLP layers and convolution layers. Than we present all the loss curve results. After that, we show the generator image samples. On top of that, we use the optimal architectures for each model to cross validate each discriminator $\mathcal{D}$'s ability to classify the image and the generator $\mathcal{G}$'s ability to generate the image. We get the following conclusion: aimlessly adding the channels of convolution layer or the number of perceptrons of MLP layer will not increase the network's performance. The convolution neural network has a better ability to do computer vision tasks. More complicated architecture can really help the generalization ability of neural network.

References

[1] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel. Infogan: Interpretable representation learning by information maximizing generative adversarial nets. In *Advances in neural information processing systems*, pages 2172–2180, 2016.

[2] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Advances in neural information processing systems*, pages 2672–2680, 2014.

[3] J. Gui, Z. Sun, Y. Wen, D. Tao, and J. Ye. A review on generative adversarial networks: Algorithms, theory, and applications. *arXiv preprint arXiv:2001.06937*, 2020.

[4] G. E. Hinton, T. J. Sejnowski, and D. H. Ackley. *Boltzmann machines: Constraint satisfaction networks that learn*. Carnegie-Mellon University, Department of Computer Science Pittsburgh, 1984.

[5] D. P. Kingma and M. Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

[6] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012.

[7] T. Kurutach, A. Tamar, G. Yang, S. J. Russell, and P. Abbeel. Learning plannable representations with causal infogan. In *Advances in Neural Information Processing Systems*, pages 8733–8744, 2018.

[8] M. Mirza and S. Osindero. Conditional generative adversarial nets. *arXiv preprint arXiv:1411.1784*, 2014.

[9] A. Radford, L. Metz, and S. Chintala. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*, 2015.

[10] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot, et al. Mastering the game of go with deep neural networks and tree search. *nature*, 529(7587):484, 2016.

[11] H. Zhang, T. Xu, H. Li, S. Zhang, X. Wang, X. Huang, and D. N. Metaxas. Stackgan: Text to photo-realistic image synthesis with stacked generative adversarial networks. In *Proceedings of the IEEE international conference on computer vision*, pages 5907–5915, 2017.

[12] B. Zoph and Q. V. Le. Neural architecture search with reinforcement learning. *arXiv preprint arXiv:1611.01578*, 2016.