Effect Analysis based on GAN and CGAN Comparison

Xinyao Hao *

Department of software engineering, Taiyuan University of Technology, Taiyuan, China

* Corresponding author email: 1252753076@qq.com

Abstract. Image generation has traditionally been. a popular research direction in the vision community of computer, which aims to learn the distribution of a given data set, to generate realistic images obeying this distribution. Owing to the rapid growth of networks of convolutional neurons, image generation which is based on deep learning has made breakthroughs in both accuracy and speed. For different scenarios, however, the results of the same generation algorithm may differ dramatically. In order to probe the application limitations of various algorithms, in this paper, taking the Minst data set as the research object, we choose two representative generation algorithms, GAN and CGAN, to construct models and analyze their generation effects. We initially present the basic models of GAN and CGAN, which include their structures and differences, respectively. By comparing the GAN and CGAN models, we can see that the randomness of generated images is greatly decreased with the extra controls. Finally, the paper analyzes the effect of CGAN handwritten image generation, and different numbers have different effects under the same epoch.

Keywords: Image Generation; Deep Learning; GAN; VAE.

1. Introduction

Image generation is a fundamental problem in the areas of graphic design and machine vision, which aims to learn the distribution of a given data set to generate realistic images according with that distribution [1]. We know it is not really difficult for humans to describe the content of an image, but it is still a very tough and challenging task to make a computer understand the content of the image and output a complete image with certain details and changes. Owing to the quick development of deep learning and model recognition technology, the precision and speed of image generation have achieved breakthroughs in recent years. Nowadays, image generation has been broadly applied in many domains of our real life, including facial image editing, extended research data sets, image attribute conversion.

Initial research on image generation. was predominantly based on mathematical or statistical models, depending on handcrafted features to describe what the image contains. Since the convolutional neural network can get the high-order semantic features of the input dataset in an adaptive manner, it has become a basic part of many computer vision tasks including image generation. According to various network frameworks, the existing image generation algorithms based on deep learning mainly include the following categories:

(1) Image generation based on Variational AutoEncoder (VAE) [2], VAE can process all kinds of data, continuous or separate, sequential and non-sequential, and even labeled or unlabeled, making them very powerful generating tools. As Dosovitskiy and Brox point out, Vae models tend to produce unrealistic, fuzzy samples [3]. Due to the way data distribution recovery and loss function are calculated, a major drawback of VAE is that they generate fuzzy outputs. Flow-based models are conceptually friendly to modeling complex distributions, but they are limited by the performance problems of density estimation compared with the most advanced autoregressive models, however, there is a significant difference in the cost of training between them, and the flow-based model takes several times longer than GAN to generate the same resolution image.

(2) Image generation based on Generative Adversarial Network (GAN). With the development of in-depth learning, GAN can understand the feature content of images through learning, and generate realistic and diverse images [4]. There are some problems with existing GAN, for example, in image generation, the sample which is generated by the multi-network model is too random to generate the specified image. In 2014, Goodfellow put forward Generative Adversarial Networks [5]. At the end

of the paper, he points out the strengths and weaknesses of GAN, as well as the future direction and expansion of GAN Research. The first extension he mentioned is: a conditional generative model P(X|C) can be achieved by adding C A S input to both the generator and discriminator. In order to enhance the stability of training, another natural angle is to change the learning method, turning the pure unsupervised GAN into a semi-supervised or supervised GAN. This can add a little restraint to GAN's training, or add a little goal. Conditional Generative Adversarial Nets (CGAN) is a very straightforward model change that introduces Conditional variable Y, a label of data, into the modeling of both the Generative Model G and the discriminant model D. It is an improvement of transforming unsupervised GAN into the supervised model by CGAN.

Based on GAN model and CGAN, this paper focuses on the problem of image generation. The main work is as follows: Firstly, the basic model of the GAN is analyzed and derived, and the derivative model of GAN, CGAN, is introduced, and it can be seen that the randomness of the generated image is greatly reduced under the condition control. At the same time, the effect of CGAN handwritten image generation is analyzed in detail.

2. Method

2.1 GAN

2.1.1 Network Structure

Generating antagonism network is a powerful model of generating discrimination, which considers the problem of generating modeling as a play between two opposing networks, and the generator network generates synthetic data of a given noise source, and the recognizer network differentiates between the export of the generator and the real data, as shown in Figure 1.

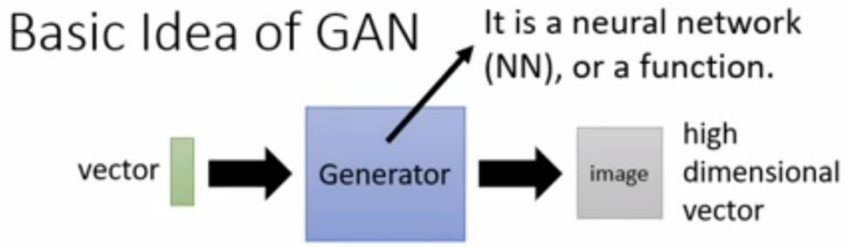


Fig 1. The structure of basic GAN neweork

2.1.2 Loss Function

D is the discriminator and G is the generator. Since the calculation of loss is made at the discriminator and the output of the generator is generally true or false, the binary cross-entropy function is employed as a whole, as:

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

Here, the max(D) part is to make D (x) closer to 1, D (g (z)) closer to 0, to train the discriminator by keeping the discriminator unchanged, and to train the discriminator by keeping the discriminator unchanged by the Ming part, it's better to expect D (g (Z)) output to be close to 1, which means that the smaller the term, the better, which is Ming.

2.1.3 Data Processing

We use the MNIST dataset to build the pattern, the test set contains a total of 10,000 images and labels while the training set train holds a total of 60,000 images and labels. Each image is a 28×28 pixel 0~9 gray matter handwritten digital image, black background white characters, the GAN generator its input is a length of n one-dimensional vector, outputs a $28 \times 28 \times 1$ dimensional picture, the discriminator inputs a $28 \times 28 \times 1$ dimensional image, and the output is a number between 0 and 1, where 1 means that the image is true and 0 means that the image is false.

2.1.4 Training Details

The training of GAN is founded on the idea of minimum and maximum game, which can be divided into the following steps: (1) Randomly select a batch of real pictures; (2) Randomly generate batch N-dimensional vectors and pass them into the Generator to generate batch fake images. (3) The real picture has a label of 1 and the false picture has a label of 0. The real picture and the false picture are passed into Discriminator as a training set for training. (4) The Generator is trained as a loss by comparing the false image's Discriminator prediction to 1(which means that if Discriminator judges the false image to be 1, the generated image is “Real”).

2.2 CGAN

2.2.1 Network Structure

From the structure of the network model displayed in Figure 2, additional information Y is added to the discriminator and generator. Y can be a category tag or other type of data, and Y can be used as A, additional input layers are thrown into the discriminator and generator.

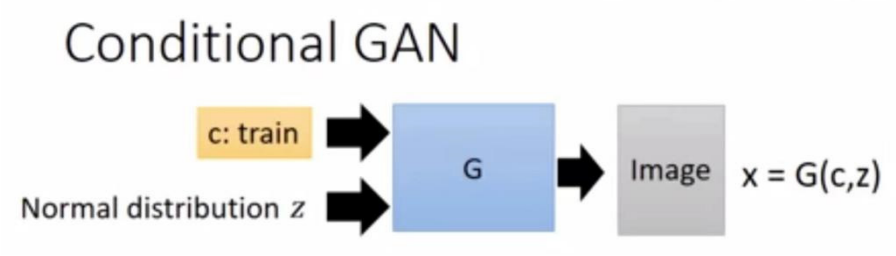


Fig 2. The structure of CGAN neweork

The input of the generating network is a random number with a label. The operation is to generate an N-dimensional normal distribution random number, and then use the embedding layer to convert the positive integer (index value) into an N-dimensional dense vector. This dense vector is multiplied by the N-dimensional normal distribution random number. The aim of the discriminant model of common GAN is to judge the false or true according to the input picture. In CGAN, it is not only to determine the authenticity, but also to determine the kind. So, it inputs a $28 \times 28 \times 1$ dimensional picture, and the output has two: one is a number between 0 and 1. Here, 1 is to determine that the picture is true, and 0 is to determine that the picture is false. The other is a vector that determines what class the image belongs to.

2.2.2 Loss Function

According to the formula, CGAN is equivalent to adding a condition to both the generator part and the discriminator part on the basis of the original GAN. The additional information replenished by CGAN only needs combining with X and Z, as input of G and D [6], the loss function of CGAN is obtained as follows:

$$\min_G \max_D V(D,G) = E_{x\sim p_{data}(x)}[\log D(x|y)] + E_{z\sim p_z(z)}[\log(1 - D(G(z|y)))] \quad (2)$$

2.2.3 Data Processing

Different from the MNIST data set used in the GAN model, the input of the CGAN generating network is a random number with a label, and the operation is to generate an N-dimensional normal distribution random number, then the Embedding layer is used to convert the positive integer (index value) into a dense vector of N dimension, and multiply the dense vector with an N-dimensional random number that complies with the normal distribution. There are two outputs: one is a number between 0 and 1, of which 1 means that the picture is true and 0 means that the picture is false. The other is a vector that determines what class the image belongs to.

2.2.4 Training Details

The training of CGAN is different from that of GAN, which can be divided into the following steps: (1) Randomly selecting batch of real pictures and their labels. (2) Randomly generate batch N-dimension vectors and label labels, and combine them with the embedding layer and pass them into Generator to generate batch false images. (3) Discriminator's loss function is composed of two parts, one is the comparison between the true result and the false result, the other is the result of the label which the picture belongs to. (4) Generator's loss function also consists of two parts, one is whether the generated image is judged by Discriminator as 1, and the other is whether the generated image is divided into the correct classes.

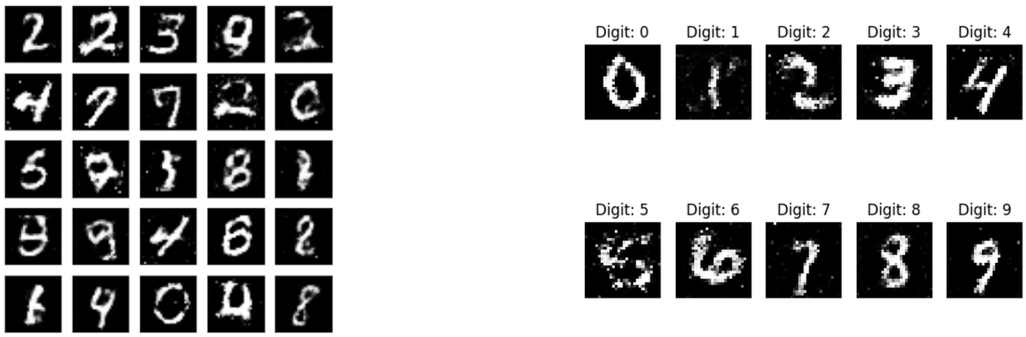
3. Experiment

3.1 Experimental Environment and Setting

The experiment of this paper is founded on the open-source deep learning system TensorFlow under the Windows operating system, using Python as the programming language. The data set which is used in the experiment is the MNIST data set, which is the most common handwritten font data set in the area of machine learning, and contains 10 varieties of handwritten digital images from 0 to 9, each set of numbers contains 10,000 test samples and 60,000 training samples, and each image is a 28 x 28-pixel gray-scale image. GAN, CGAN network input image size of 28 * 28, the data set for gray-scale handwritten digital images, features relatively few, training iterative 10000 Epoch. In the training phase, the network uses Adam Optimizer with the learning rate set to 0.0002 and the weight attenuation set to 0.5[7].

3.2 Performance Analysis

GAN goes into an N-dimensional random number with a normal distribution, and CGAN uses an Embedding layer to convert the positive integer (index value) into a dense vector of a fixed size to label the random number, and the dense vector is multiplied by the N-dimensional normal distribution random number to obtain a labeled random number. Figure 3 shows the effect of generating the GAN model and CGAN model. CGAN not only generates handwritten data, but also can judge whether the generated tags correspond to their conditions, so as to achieve the effect of classification generation.

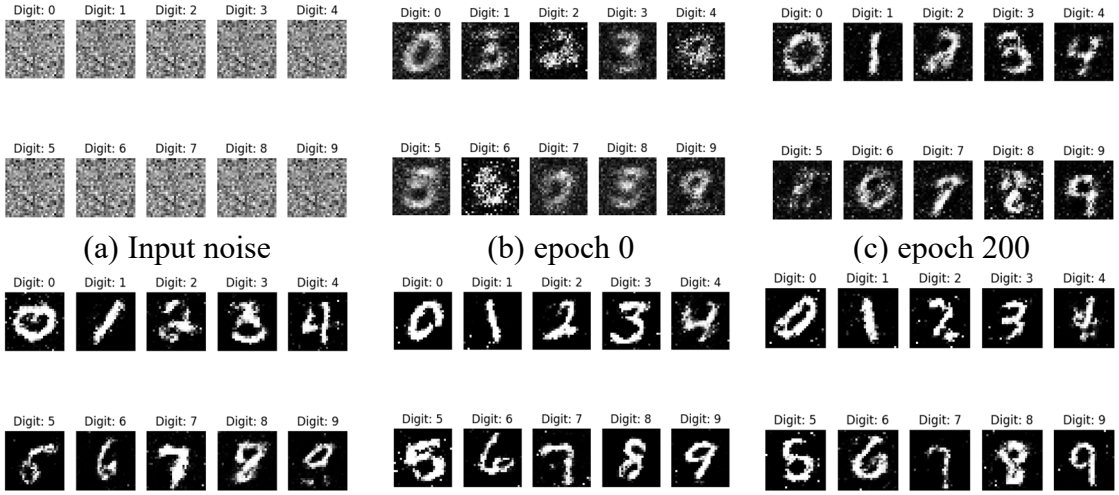


(a) GAN (b) CGAN

Fig 3. Comparison of generation results between GAN and CGAN

We also visualize the generation results of different algorithms at different epochs for the purpose of estimating the effectiveness of the intermediate generation process of different methods. Taking CGAN as an example, Figure 4 displays the generation effect of CGAN from epoch 0 to 10000. It can be seen the number in the generator and discriminator in the confrontation of constantly clear. At epoch 0, all of the generator results are very similar, and we personally think the generator may have obtained a few features and accidentally tricked the discriminator because it learned so little, all of the generated results will then converge to this feature, which results in the first generated result is

very similar. As the number of trainings ascends, the generator learns more and more features, and the resulting results slowly begin to show a remarkable difference. The visual results of the two patterns show that the effect of the training for different numbers is different and the effects of the generation of different handwritten numbers are different in the same epoch. At epoch 200, the numbers 4 and 6 were the most blurred, 2 and 8 were the fuzziest, and all the numbers did not show obvious numerical features due to the low number of training sessions. At epoch 1000, the numbers 0,1,3,4, and 9 are the first to make a complete shape, while the numbers 5 are the worst, and the shapes are scattered; the numbers 2,6,7, and 8 are generally visible, but it is still relatively difficult to distinguish them without a category label. When epoch is 4000, most of the numbers can be directly identified and produced well. The numbers 0,1,4,6,7,8 is clear, but the lines are too thick and the shape of the numbers is abstract. The number 5 is close to the number 6, the number 9 is close to the number 0, and the strokes of 2 and 3 are interrupted. At 8000 epochs, all the numbers were clearly identified due to a certain number of times of training, compared to 4 and 5, which produced the most general results and had the whitest spots around 4 and 5.



(a) Input noise (b) epoch 0 (c) epoch 200

(d) epoch 1000 (e) epoch 4000 (f) epoch 8000

Fig 4. Generation results of CGAN at different epochs

4. Discussion

It can be seen from the generated images that most handwritten numerals have blurred edges after being generated, and the images generated by CGAN have many defects, such as blurred edges, the generated image resolution is too low and so on [8]. In 2017, the Catholic University of Ukraine and ELEKS, a Czech Technical University in Prague and solutions provider, published a paper titled "DeblurGAN: Blind Motion Deblurring Using Conditional Adversarial Networks." [9]. In this paper, the researchers propose an end-to-end learning method called DeblurGAN, which is founded on the conditional antagonism generation network and content loss, to remove the blurring generated by the relative motion of the camera and the object in the image. I think that for handwritten numerals with blurred edges, we can treat the blurred edges of the numerals approximately as motion-generated blur, and the combination of DeblurGAN can solve the problem of image blurring to some extent.

VAE image is not as clear as GAN image, but VAE image is more accurate, if the two fusions, can make the image with the accuracy of VAE generated and the clarity of GAN generated [10]. In the paper: Autoencoding beyond pixels applyinging a learned similarity metric, Vae decoder is connected to GAN as the Geneator of GAN, making full use of the advantages of both, learning the prior distribution of data, and then sampling, making confrontation learning more effective.

The traditional image generation is simply by learning to simulate the distribution of the real image, and then after optimization processing to generate similar to the real image, it is equivalent to a discrimination task (generating an image can be classified into the same category as a real image),

while generating a realistic image based on a description is much more difficult and requires more training. In machine learning, this is a generation task, which is much more difficult than a discrimination task, because the generation model has to be based on a smaller seed input and a richer output of information, and I think in future work, generating conditions can be improved for more specific text descriptions [11], such as “Thick (thin)” and “Lean left (right)” to MNIST handwriting. In general, Conditional Generative Adversarial Networks is a very direct and effective improvement on the original GAN, enabling more efficient use of GAN for data generation, and in many places, there will be very important use.

5. Conclusion

In the paper, the problem of image generation under GAN and CGAN models is studied with MNIAT data set as the research object. Firstly, we introduce the two models of GAN and CGAN respectively, and introduce the derivative model of GAN, CGAN, by comparing the results of GAN and CGAN models generated from MNIST handwritten data set, we can see that the randomness of the generated images is greatly decreased under the condition of control. In the analysis of CGAN generation effect, different numbers generation effect is different under the same epoch, I think it is difficult to generate numbers with enclosed space inside.

References

[1] LeCun Y, Bengio Y, Hinton G. Deep learning[J]. nature, 2015, 521(7553):436-444.

[2] Kingma DP, WellingM. Auto-Encoding Variational Bayes[J/OL]. arXivpreprint arXiv:1312.6114.2013.

[3] Dosovitskiy, A., & Brox, T. (2016). Generating images with perceptual similarity metrics based on deep networks. arXiv preprint arXiv:1602.02644.

[4] Krizhevsky A, Sutskever I, Hinton G E. Imagenet classification with deep convolutional neural networks [J]. Advances in neural information processing systems, 2012,25:1097-1105.

[5] Goodfellow I JPouget-AbadieJMirza Met al. Generative adversarial nets[C]// Proceedings of the International Conference on Neural Information Processing Systems Montreal. Canada. 2014:2672-2680.

[6] Mirza M, Osindero S. Conditional Generative Adversarial Nets, 10.48550/arXiv.1411.1784[P]. 2014.

[7] Rumelhart D E, Hinton G E, Williams R J. Learning representations by back-propagating errors[J] nature. 1986. 323(6088):533.

[8] OordA, KalchbrennerN, VinyalsO, et al. Conditional image generation with PixelCNN decoders [C]// Proceedings of the 30th International Conference on Neural Information Processing Systems. 2016:4797-4805.

[9] Kupyn O, Budzan V, Mykhailych M, et al. DeblurGAN: Blind Motion Deblurring Using Conditional Adversarial Networks[C]// 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, 2018.

[10] Larsen, A. B.L., Sønderby, S. K., &Winther, O.(2015). Autoencoding beyond pixels using a learned similarity metric. arXiv preprint arXiv:1512.09300.

[11] Yan X, Yang J, SohnK, etal. Attribute2 image: Conditional image generation from visual attributes[J/OL]. arXiv preprint arXiv:1512.00570, 2015.