Research on the Application of Double Conditional General Advantage Nets (DCGAN) to generate Simple Background Image with Specified Hue

Shuai Yang¹, Haicheng Bai²
¹Information Management Department, Chuxiong Normal University, Chuxiong, Yunnan, China
²Information Management Department, Yunnan Normal University, Kunming, Yunnan, China
* e-mail: ysh@ctc.edu.cn

Abstract. Since the General Advanced Nets (GAN) proposed by Ian Goodfellow in 2014, the research on GAN has become a hot spot of deep learning. People can use GAN to generate images, but there are many problems in GAN, such as the inability to generate images of specified categories, poor model stability, and high training difficulty. Wasserstein GAN improves the disadvantages of the original GAN. Meanwhile, Conditional General Advanced Nets proposes to add condition vectors on the basis of the original GAN, so that the GAN can generate specified pictures, making the original GAN more and more easy to use. Most of the research work of GAN focuses on the real face and real scene generation, but the generation of the commonly used background image in the design is less. In this paper, we use two conditional GAN (DCGAN) and WGAN model framework to generate a simple background image of specified hue and random graphics, which can provide designers with fast and available background images in some simple projects.

1. Introduction
In some designs, some simple background pictures are often used. These pictures may only be composed of colour blocks or simple geometric figures. At the same time, the hue and geometric composition of the pictures are specified to a certain extent. The hue of the picture generally includes warm hue and cold hue, but these two categories are relatively general. Many times, people will use a descriptive word to describe, such as autumn colour, whose hue is generally golden, the background picture with a sense of science and technology, the main hue tends to be blue, and the main hue of the picture with Chinese style may tend to be big red or blue, yellow, etc. If every designer designs these background pictures from scratch or gets them from the network, it will take a lot of time. It is not advisable for a design project with time constraints. What ’s worse, make designers can not focus on the main content of the design.

Fortunately, with the continuous development of deep learning research in recent years, people have been able to generate a variety of images using the adversary generation network, and it has reached the point of falsification, which has a lot of research work. The purpose of this article is to use DCGAN to generate simple background images.
2. Relevant concepts

2.1 Generative Adversarial Networks (GAN) [1]
GAN was proposed by Ian Goodfellow and others in 2014. GAN consists of two models: Generator model G and Discriminator model D. The generator maps the noise to the data space through the mapping function, and the output of the discriminator is a scalar, which represents the probability that the data comes from the real training data rather than the generated data of G (Generator). The task of generating model G is to receive a random noise Z, generate a picture and record it as G (z). D is a discriminator model. Its task is to determine whether a picture is "real". Like other binary neural networks, its input is a picture x, and its output D (x), that is, x is the probability of a real picture.

In model training, the goal of generating network G is to generate real pictures, and to deceive and distinguish network D, while the goal of D is to distinguish the generated false pictures from the real pictures. G and D use each other to improve themselves. In this process, G constantly improves its ability to generate real pictures, and D constantly improves its ability to distinguish false pictures. Finally, the ideal result is that G can generate enough fake pictures, and D can’t determine the true or false of the pictures generated by G. at this time, D (G (z))=1/2, the basic structure of GAN is shown in Figure 1 [2].

![GAN Diagram](image)

Figure 1. Generative Adversarial Nets

The GAN model has no loss function. The optimization process is a "mini max two-player game" problem. The value function of the model is as follows:

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

In the process of training, the training model D needs to maximize the log (D (x)) of real samples, while the generated model G needs to minimize the log (1-D (G(z))), that is, maximize the loss of D. G and D are trained at the same time, but in the training, one side should be fixed, the other side’s parameters should be updated, and the other sides’ errors should be maximized by alternating iterations. Finally, G can estimate the distribution of real samples.

2.2 Wasserstein GAN [3]
The original GAN training is unstable and difficult to train. If the discriminator D is trained too well, the gradient of the generator will disappear easily, and the generator loss will not decline; if the discriminator is not trained well, the gradient of the generator will not be correct. Only when the discriminator training is not good or bad can it be, but it is difficult to grasp when it is good or bad, even in different stages before and after the same round of training. Martin Arjovsky and others have done a series of work. In the "Wasserstein GAN" paper, two different forms of GAN proposed by Ian Goodfellow are analyzed in detail. The first form is equivalent to minimizing the JS divergence...
between the generated distribution and the real distribution under the optimal discriminator. Because the random generated distribution is difficult to overlap with the real distribution and JS divergence cannot be ignored. The second form is equivalent to minimizing the direct KL divergence of the generated distribution and the real distribution, and maximizing its JS divergence, which contradicts each other, leading to the instability of the gradient. Moreover, the asymmetry of KL divergence makes the generator prefer to lose diversity rather than accuracy, leading to the Collapse Mode phenomenon. Lapse mode phenomenon. WGAN introduces Wasserstein distance on the basis of this analysis. Because it has superior smooth characteristics compared with KL divergence and JS divergence, it can theoretically solve the problem of gradient disappearance. Finally, it makes some improvements on GAN, solves the problem of unstable training, and provides a reliable index of training process, which is highly related to the quality of generated samples[3].

2.3 conditional GAN [4]
In the early days, since GAN did not need to be modelled in advance, if we want to generate high-pixel pictures and controllable output, the use of GAN will be uncontrollable. For example, if the input real pictures are some pictures of cats or dogs, the final generation are also some random low pixel images of cats or dogs. The random low pixel image cannot be specified to generate only the cat or dog. The basic idea of conditional GAN is to add some constraints to the GAN model to control the output of GAN.

Conditional GAN introduces conditional constraint y into generator model G and discriminator model D to guide the data generation process. Conditions can be any supplementary information, such as class labels, data of other modes, etc., so that GAN can be better applied to cross modal problems, such as automatic image annotation. The relationship between the discriminator and generator of CGAN and vector y is shown in Figure 2.

![Figure 2. Conditional Generative Adversarial Nets](image)

The noise z and condition y are sent to the generator as input at the same time to generate cross domain vectors, which are then mapped to the data space by nonlinear functions. The data x and the condition y are sent to the discriminator as inputs at the same time to generate cross domain vectors and further judge the probability that x is the real training data [3,4].The value function of the model is changed as follow:

$$\min_{G} \max_{D} V(D,G) = \mathbb{E}_{x \sim p_{data}(x)} \left[ \log D(x \mid y) \right] + \mathbb{E}_{z \sim p_{z}(z)} \left[ \log(1 - D(G(z \mid y))) \right]$$

3. Method
Double conditional GAN (DCGAN) is composed of two conditional GANs, which are recorded as GAN1 and GAN2 (similar to StackGAN). The first GAN generates rough images and the second
GAN generates high-resolution images [2]. However, unlike StackGAN, the two conditional vectors \( y \) are different. The composition of DCGAN is shown in Figure 3.

![Figure 3. The composition of DCGAN](image)

### 3.1 Composition of real picture data

The real picture data uses the crawler program written by python, and selects all kinds of presentation backgrounds to provide website crawling access. These websites are selected to obtain the real image, because these websites not only easily obtain the pictures, but also easily obtain the style description of the pictures. The final data is 36000 pictures, 9 categories are selected, each of which is 4000. The 9 categories are: fresh, business, watercolor, Chinese style, simplicity, cuteness, retro, education, science and technology.

Each classified image has certain hue characteristics. All images are scaled to 256 * 256 as a whole, and packed into numpy array by image processing module of keras, so as to facilitate later input into the model for training.

### 3.2 Condition vector \( y \) processing

In the conditional GAN, the condition vector \( y \) is generally encoded in the way of one-hot. In DCGAN, the \( y \) vector in GAN1 is encoded in the way of one-hot. Simply speaking, the 9 classifications of real data are transformed into one-hot encoding, which turns into 36000 * 9 vectors, which are put into the generator together with \( z \). Although the one-hot coding method will produce the problem of high dimension and sparse data, the problem is not large here, because there are only nine classification items in total. In addition, as a condition \( y \), it is very convenient to control the network to obtain the semantic information of picture description. In GAN2, the condition vector \( y \) is randomly sampled from the independent Gaussian distribution \( N(\mu(\varphi),\Sigma(\varphi)) \) to obtain the implied variables. Where \( \mu(\varphi) \) and \( \Sigma(\varphi) \) are about a function of \( \varphi \). For the objective function of the generator in the training process, add \( D_{KL}(N(\mu(\varphi),\Sigma(\varphi)))\|N(0, I) \) to get the condition vector \( y \) [6,7,8].

### 3.3 Network architectures

For each GAN network, using the GAN network model provided by Erik Linder norén [9]. While retaining the overall network structure and the number of parameters of GAN, some modifications are made according to WGAN principles, making the model more stable and easy to train. The main changes are as follows: (1) Remove sigmoid from the last layer of GAN2 discriminator \( D \) to activate function; (2) Do not take log for loss of all generators and discriminators; (3) Truncate their absolute value to no more than a fixed constant \( c \) after each parameter update of discriminator; (5) Use RMSProp optimization algorithm for GAN1, while use SGD optimization algorithm for GAN2, rather than the traditional Adam or momentum.
3.4 Training process
In the first stage, a) first train the first GAN (GAN1) discriminator D1 with real data (batch = 40) to prevent the discriminator from being too poor; b) set Noise $z$ is input to generator and training generator together with condition $y$; c) the process of fixed generator and training discriminator alternately until GAN1 training is completed.

In the second stage, the output of the first stage and $y'$ are input into the generator of GAN2, and the generator and discriminator are trained alternately. Finally, an image of 1024 * 1024 is generated.

4. Application results
The model has been trained in batch_size = 400 rounds. In Figure 4 below, four classifications are selected to show the results of images generated in the first batch_size of 100, 200, 300, 400 rounds. For the convenience of display, the images are all scaled and the original image input is 1024 * 1024.

| classification | 100  | 200  | 300  | 400  |
|----------------|------|------|------|------|
| Simplicity     | ![Image](image1) | ![Image](image2) | ![Image](image3) | ![Image](image4) |
| Technology     | ![Image](image5) | ![Image](image6) | ![Image](image7) | ![Image](image8) |
| Refreshing     | ![Image](image9) | ![Image](image10) | ![Image](image11) | ![Image](image12) |
| Chinese style  | ![Image](image13) | ![Image](image14) | ![Image](image15) | ![Image](image16) |

Figure 4. The composition of DCGAN

From Figure 4, we can see that when the times of batch is 100, the image is rough and noisy. When the times of batch is 200, the model has started to generate a rough outline of the image, but it is fuzzy. When the times of batch is 300, the image has a better effect, and when the times of batch is 400, the content and color of image has been basically formed.

5. Conclusion
Using two conditional GANs and the conditional vector $y$ can generate simple background image, and can specify the corresponding type artificially, which provides a method for generating some simple background images. As long as the model is trained, it can quickly generate the specified image many times. It can provide images quickly in some less demanding designs.

But there are also some problems in this model. First, the description line of classification is not very clear, and there will be overlaps between different classifications. Therefore, the final generated
image categories will still be affected by the real image features, and sometimes the generated images will be uncontrollable. Second, the resolution of the generated images is relatively small. It is difficult to produce large images. Forcing the size of the output image to increase will make the image blur and increase the calculation amount of the model, while too small images will have many limitations in use.

How to make the classification more accurate and diverse, make the image input more controllable, and improve the size of the output image while not reducing the image details and not multiplying the calculation is the focus of the next step.

References
[1] Ian J. Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David WardeFarley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. In Advances in Neural Information Processing Systems 27, pages 2672–2680. Curran Associates, Inc., 2014.
[2] Cheng Cheng.(2018).Popular understanding GAN,https://zhuanlan.zhihu.com/p/33752313
[3] Martin Arjovsky, Soumith Chintala, Léon Bottou.(2017).Wasserstein GAN. https://arxiv.org/abs/1701.07875.
[4] Mehdi Mirza,Simon Osindero.(2014).Conditional Generative Adversarial Nets. https://arxiv.org/abs/1411.1784.
[5] A. Dosovitskiy and T. Brox. Generating images with perceptual similarity metrics based on deep networks. In NIPS, 2016. 2 .
[6] Han Zhang, Tao Xu, Hongsheng Li, Shaoting Zhang, XiaoGANg Wang, Xiaolei Huang, Dimitris Metaxas.(2016).StackGAN: Text to Photo-realistic Image Synthesis with Stacked Generative Adversarial Networks. https://arxiv.org/abs/1612.03242.
[7] J. Bruna, P. Sprechmann, and Y. LeCun. Super-resolution with deep convolutional sufficient statistics. In International Conference on Learning Representations (ICLR), 2016. 2, 3, 5 .
[8] A. Mahendran and A. Vedaldi. Visualizing deep convolutional neural networks using natural pre-images. International Journal of Computer Vision, pages 1–23, 2016. 7, 8.
[9] Erik Linder-Norén.(2018).GAN.https://github.com/eriklindernoren/Keras-GAN/commit/1a0428a1ec441adbac9df7cc18a6fccf75b83bc.