 CMC, vol.57, no.1, pp.167-178, 2018

A Method for Improving CNN-Based Image Recognition Using DCGAN

Wei Fang[1, 2], Feihong Zhang[1, *], Victor S. Sheng[3] and Yewen Ding[1]

Abstract: Image recognition has always been a hot research topic in the scientific community and industry. The emergence of convolutional neural networks(CNN) has made this technology turned into research focus on the field of computer vision, especially in image recognition. But it makes the recognition result largely dependent on the number and quality of training samples. Recently, DCGAN has become a frontier method for generating images, sounds, and videos. In this paper, DCGAN is used to generate sample that is difficult to collect and proposed an efficient design method of generating model. We combine DCGAN with CNN for the second time. Use DCGAN to generate samples and training in image recognition model, which based by CNN. This method can enhance the classification model and effectively improve the accuracy of image recognition. In the experiment, we used the radar profile as dataset for 4 categories and achieved satisfactory classification performance. This paper applies image recognition technology to the meteorological field.

Keywords: DCGAN, image recognition, CNN, samples.

1 Introduction

Nowadays, with the development of deep learning, people are increasingly pursuing the accuracy of image recognition. Deep learning neural networks that mimic human thinking also appear more and more in this field. The design of the architecture, the tuning of parameters, and the selection of samples directly influences the final recognition results of the neural network. At present, many studies have used Convolutional Neural Network(CNN) as an entry point to improve the accuracy of image recognition. As we all know, the CNN can use the original pixels of the image directly as input. It is no longer necessary to extract the features in advance using the traditional method. This has reported superior performance compared to earlier work relying on manual features [Dixit, Chen, Gao et al. (2015)]. In fact, CNN is successfully applied for the classification of handwritten characters and gesture recognition [Kim, Lee and Park (2008)] which is applied directly to the data flow without pre-processing or feature selection. CNN training model is invariant to distortions such as scaling, translation,

[1] School of Computer & Software, Jiangsu Engineering Center of Network Monitoring, Nanjing University of Information Science & Technology, Nanjing, 210044, China.

[2] State Key Lab. for Novel Software Technology, Nanjing University, Nanjing, 210023, China.

[3] Computer Science Department, University of Central Arkansas, Conway AR, 72035, USA.

[*] Corresponding Author: Feihong Zhang. Email: 20171211494@nuist.edu.cn.

rotation and has strong generalization ability. The biggest advantage of CNN is that can handle high-dimensional data onto the shared convolution kernel. Convolutional kernels deal with complex feature calculations through multi-layer training in end-to-end networks. This design can greatly reduce amounts of parameters of the neural network and at the same time reduce the complexity of the neural network model, giving an optimization space of large classification accuracy.

Most of the CNN image classification is based on supervised learning. A large amount of data is needed as a training sample to obtain more accurate classification in training process. However, some samples are hard to collection. For example, radar profiles of the specific climate. It is extremely difficult to collect samples due to the limitations of the conditions. Fortunately, lan Goodfellow [Goodfellow, Pouget-Abadie, Mirza et al. (2014)] proposed GAN which was a framework of generating models and inspired by game theory. GAN can generate images or image restoration. In pix2pix, change monochrome image to color image, line drawing image of texture, shadows and luster image etc. [Isola, Zhu, Zhou et al. (2017)]. For the dispersive phenomenon that the raw GAN training appeared, Conditional GAN [Mirza and Osindero (2014)] turns the original generation process into a generation process based on some additional information. The generator tries to generate labels and random noise together. The discriminator discriminates between the data source and the data label at the same time, providing the generator of a more efficient gradient which makes it easily extendable to semi-supervised learning. After that, due to the instability and intractability of neural network self-training, DCGAN [Radford, Metz and Chintala (2016)] extends the structure to convolutional neural networks. In this work, a set of convolutional neural networks were proposed, using Batch Normalization to achieve local normalization making it possible to train real large-scale datasets such as CelebA.

The major contributions of this paper are as follows:

1. We designed a novel model structure to generate sample which hard to collect based on DCGAN's high scalability and excellent sample generation capabilities.

2. The learning rate decay strategy is used to speed up learning on generator optimization problems.

3. In our image recognition experiment, we built a recognition framework based on CNN and matched enough sample generation to strengthen the training recognition model. Finally improved the classification accuracy.

4. We used the radar profile as a dataset, applies the proposed technology to the meteorological field and extends the application of image recognition.

2 Relation work

2.1 GAN

There are two components in the GAN framework. One is the generate model G and the other is the discriminant model D. The G model is responsible for producing spurious data that is close to the real data. The D model is responsible for identifying the authenticity of the data produced by G. Competition between D and G made the two sides constantly to optimize the training until reaching a balanced state. GAN can learn

independently through such clever design. Similar to the two-player min-max game, where one plays the generator role and attempts to generate samples from random noise, and another plays the discriminator, attempts to discriminate synthetic samples and real ones. Its overall loss function is expressed as minimizing the distance between the generated data distribution and the real data distribution. Given a generator condition, the best discriminator D is shown as:

$$D(x) = \frac{P_{data}(x)}{P_{data}(x)+P_{model}(x)} \tag{1}$$

In this theory, Goodfellow et al. [Goodfellow, Pouget-Abadie, Mirza et al. (2014)] concludes that the final desired result is that the real distribution $P_{data}(x)$ is equal to the generated distribution $P_{model}(x)$ and the discriminant boundary is $1/2$. However, it is easy to lose the direction of convergence because of GAN's own degree of freedom in training is too large. In addition, the deep neural network of the GAN is not stable in training and prone to underfitting or overfitting. Therefore, it is difficult to adjust the parameters of the GAN itself during training.

2.2 DCGAN representation

OpenAI proposed Improved GAN [Salimans, Goodfellow, Zaremba et al. (2016)], which defined two training techniques: feature matching and minibatch discrimination. It enhances the diversity of samples generated by the generate network also increases the diversity of the discriminate network when discriminating samples. Inspired by this, DCGAN expanded GAN from multi-layer perceptron MLP structure of convolutional neural network structure [Radford, Metz and Chintala (2016)]. It provides a set of convolutional neural networks, removing the pooling layer and adding Batch Normalization between convolutional layer and activation function to achieve local normalization, which greatly improves the network model. DCGAN expands on GAN not only retains the ability to generate excellent data but also incorporates the advantages of CNN feature extraction, making it have an improvement in image analysis and processing capabilities. DCGAN has achieved satisfactory results from training in real large-scale datasets such as CelebA, LSUN and Google Image Net. Among the visual data, there are a lot of near-duplicate images, which cause a serious waste of limited storage, computing, and transmission resources of networks and a negative impact on recognition experience [Zhou, Wu, Huang et al. (2017)]. Therefore, in this paper, we designed sample generation experiments based on the DCGAN network structure.

2.3 CNN image recognition

Recently, CNN is widely used in image recognition applications [Oquab, Bottou, Laptev et al. (2014); Shin, Yamaguchi, Ohnishi et al. (2016)]. Different from existing methods, CNN can generate high-level semantic representations by learning and concatenating low-level edge and shape features from a large amount of labeled data. The upper layers of CNN are more sensitive to semantics, while the middle layers are particularly sensitive to underlying patterns such as colors and gradients, so using the upper layers or middle layers is a common and effective practice of CNN. Numerous practices and researches have made CNN have many variant forms. From the LeNet-5 [LeCun, Bottou, Bengio et

al. (1998)] to AlexNet which is the key to promote the development of CNN; From Googlenet to VGGNet and OverFeat, deep network extraction features can be used for image classification, detection and segmentation [Krizhevsky, Sutskever and Hinton (2012); Szegedy, Liu, Jia et al. (2015); Simonyan and Zisserman (2014)]. The aim of object detection is to find the location of all the targets and specify each target category on a given image or video. The radar profile that we want to recognize is different from the general object image. It describes categories based on spectral distribution and color similarity. So, this level of semantics can use CNN to perform feature extraction better [Dixit, Chen, Gao et al. (2015)]. Classifications operation need to be performed after the features were extracted from the recognition system. We directly connect feature extractors and classifiers in the network as the main structure of identification framework. This paper does not use MPM as a classifier, which can directly estimate the probabilistic accuracy bound by minimizing the maximum probability of misclassification [Gu, Sun and Sheng (2016)], but adopts the method of full connection layer and Softmax function classification.

3 Method

In this section, we will explain the main ideas and methods of this paper, including the establishment of network models, the generation of image samples, the performance of the sample test, and specific image recognition programs. The overall process is shown in Fig. 1.

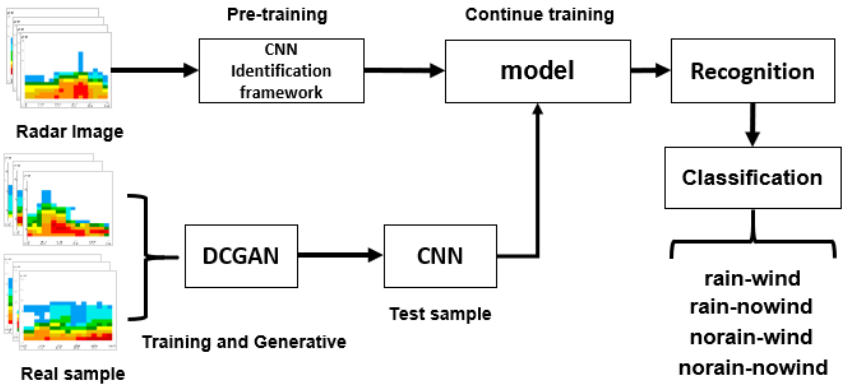

Figure 1: Image recognition system process framework

3.1 Build DCGAN network

DCGAN is implemented in convolutional neural networks. It can also be understood as the application of convolutional neural networks in GAN. In traditional CNN, feature extraction and down sampling are performed through the convolutional layer and the pooling layer respectively. But in DCGAN, the discriminative model and the generative model cancelled the pooling layer [Gong, Wang and Lazebnik (2014)], leaving only the convolutional layer and allowing the network to learn spatially up and down sampling by itself. The discriminative model is a convolutional neural network which removes the full-connected layer. All activation functions using LeakyReLu, and Sigmoid or SoftMax function is used as a binary problem. The essence of the discriminant model is to compress a picture into a feature vector. Generative model is a deconvolution process.

All activation functions except the output layer uses ReLu, and the output layer uses tanh function. In this way, it is possible to turn several random feature vectors into pictures. Upsampling and down sampling are achieved by defining stride when writing code. In the model structure of this paper, we build our own sample generation network and discriminant network by referring to the DCGAN network structure.

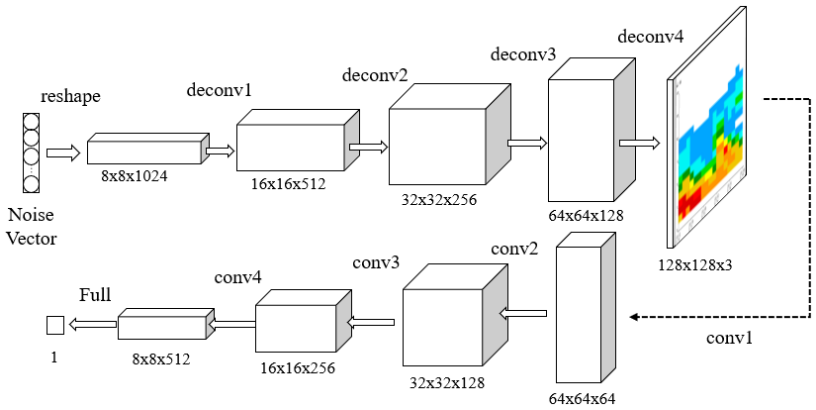

Figure 2: The structural association of generative model and discriminative model in DCGAN

As shown in Fig. 2. The overall network structure mainly includes two kinds of networks, a discriminative network and a generated network. Their layers are unified into 4 layers. Our goal is to train a generator G that can transform the noise vector z into sample x. The training target of the generator G is defined by a discriminator D which distinguished between the real sample data $p_{data}(x)$ and the generated data $p_z(z)$. The generator G will confuse the discriminator D to think that the generated data is true. Through training will guide G and D eventually find a balance of non-convex game. We use gradient descent method of optimization and do without any assumptions or model requirements on the distribution of data in advance. The network loss function is defined by:

$$V(D,G) = E_{x\sim P_{data}(x)}[log\, D(x)] + E_{z\sim P_z(z)}\left[log\left(1 - D\big(G(z)\big)\right)\right] \quad (2)$$

The convergence direction of the network is $\min_G \max_D V(D,G)$. We decompose the loss function of formula (2) into two parts, where the discriminative model loss functions and the generative model loss function are as follows:

$$LOSS_{(D)} = -(log(D_1(x)) + log(1 - D_2(G(z)))) \quad (3)$$

$$LOSS_{(G)} = -(log(D_2(G(z)))) \quad (4)$$

where D represents a discriminator, G represents a generator. $G(z)$ is a sample generated by a random vector and x is a real sample data. We obtain the optimal weight value by minimizing the loss function. Then to make the generative model generate the sample what we need. We have adopted a strategy with a constantly decreasing learning rate in order to speed up learning in training process. The reason why the strategy with a constantly decreasing learning rate is adopted because of DCGAN uses mini-batch gradient descent to optimize the network parameters in this paper. Appearing of noise let

the descending process do not to converge accurately on the iterative process. Whenever training a certain number of times, the learning rate will be decreasing once. In the beginning, a larger learning rate can achieve very fast convergence. As the learning rate getting smaller, the stride of convergence also decreasing. It will not cause much error even if it swings around the minimum value. The learning rate decay strategy can be expressed as:

$$\alpha_i = \frac{1}{1+decay\_rate*epoch_i} * \alpha_0 \tag{5}$$

The $decay\_rate$ will set to 0.95 in subsequent experiments, $epoch_i$ is represented as the ith training. α_0 is the initial learning rate. The decayed learning rate needs to be combined with a optimizer in order to achieve a goal of quickly obtaining an optimal solution and making the later training more stable. An optimizer that operates on parameters, such as Momentum which makes the gradient steeper. Although it can converge faster, makes the training very difficult. Another optimizer, such as AdaGrad, which adds punish patterns based on modifying the learning rate so that each parameter has its own learning efficiency but it's inefficient. In this paper, we combine these two optimizers and use Adam to accelerate the training of neural networks. Its mathematical expression is as follows:

$$m = b1 * m + (1 - b1) * dx \tag{6}$$

$$v = b2 * v + (1 - b2) * dx^2 \tag{7}$$

$$W += -learn\_rate * m/\sqrt{v} \tag{8}$$

In the formula, b represents bias and W represents weight. The updating of weight parameters depends on two variables m and v and the amount of change, dx. Formula (6) contains the Momentum gradient attribute, and formula (7) contains AdaGrad's resistance attribute. Therefore, taking both m and v into account by formula (8) weight parameters can be updated.

In the previous tests, it was often found that there were barely noticeable difference in the generated images because the sample parameters almost converged on one point. During the sample generation experiment, we used mini-batch execution to improve training efficiency. This strategy can make reasonable use of the computer's memory while also saving the training time. But at the same time, batch training also brings about competition between the gradients. Through the study of neural network, it is found that when the parameters of a certain layer of neural network change with the gradient training, the distribution of the output data of the layer may change. For the each layer of the neural network, the output distribution of each layer will be different from the corresponding input signal distribution after the operation within the layer. This difference will increase with the increase of the network depth, which resulting in the covariate shift [Ioffe and Szegedy (2015)] problem that the trained model cannot be generalization well. Gradient may gradually vanish when it spreads. For this reason, we add Batch Normalization between the convolution and activation functions to solve the vanishing gradient problem and help the gradient propagated to each layer. Batch Normalization can overcome the difficulty of deep neural network training well. In the process of normalization, the batch training samples can be expressed as: $X =$

$\{x_1, x_2, \ldots, x_m\}$; Where x_i represents the sample at index i. The mean and variance can be calculated by these samples. Mathematical expression is as follows:

$$mean_x = \frac{1}{m}\sum_{i=1}^{m} x_i \tag{9}$$

$$\sigma_x^2 = \frac{1}{m}\sum_{i=1}^{m}(x_i - mean_x)^2 \tag{10}$$

Formula (9) shows calculating the average value through sample points. Using the average can be calculated variance by formula (10). According to the calculated results, the normalization operation can be executed. The range of $\hat{x}_i$ can be constrained by the element ε in conjunction with formula (11). ε is an indefinite number within a certain range.

$$\hat{x}_i = \frac{x_i - mean_x}{\sqrt{\sigma_x^2 + \varepsilon}} \tag{11}$$

$$y_i = \gamma \hat{x}_i + \beta \tag{12}$$

To enable DCGAN to learn the appropriate input, γ and β are used to transform $\hat{x}_i$, and the process is represented by $BN_{\gamma,\beta(x_i)}$. Both γ and β are learned autonomously by the network.

3.2 Generate samples by DCGAN

We trained 2000 radar profiles with rain-wind and rain-nowind categories on DCGAN, respectively. In order to train more efficiently and prevent all images of being read into memory at one time, we used a mini-batch training method. Each batch trains 64 images. For every 100 training batch, a sample plot will be generated locally for visual inspection. We set the number of training for 300 times to make the network fully learn from generating features. After the end of the training, the generated model was saved for the next training and generate sample. With DCGAN, rain-wind and rain-nowind samples were generated for later experiments. Fig. 3 shows the generated samples in different epoch cases. The first half (raw) is the training results by original DCGAN, and the last half (add norm) is the training results after adding Batch Normalization. It can be seen that the sample convergence is more realistic after adding Batch Normalization.

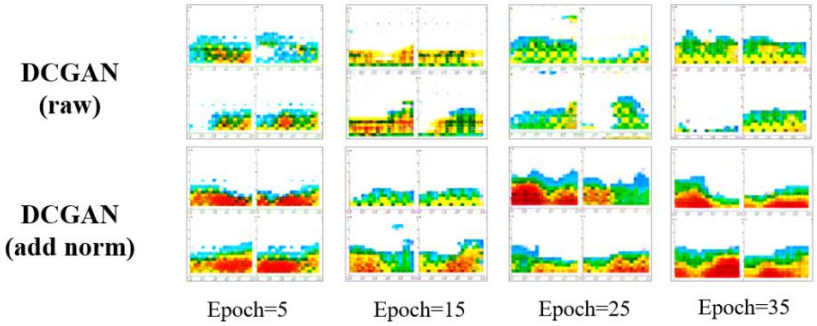

Figure 3: Radar profile dataset

3.3 Establishing image recognition network

We established a recognition framework based on deep learning convolutional neural networks to identify the radar profiles.

As shown in Fig. 4, according to the scale and number of test data, referring to the VGG19 network form, the model for identification is constructed as a neural network having 4 layers of convolution layers and 3 layers of full connection layers. The result of the output is the four classifications of the radar profile image.

Initialization weights are taken as random values of a normal distribution standard deviation from 0.01, and the initialization bias value is 0. The convolution operation stride uniformly set to 1. The processing mode is SAME for the exceeding boundary portion. The stride of the pooling operation set to 2, its boundary processing method is VALID. The initialization operations for weights and bias, convolution kernel, and pooling in the remaining convolution layer are same as the first layer. Since the picture pixels are used as direct inputs, the data dimension needs to be changed to obtain the final one-dimensional classification result. Therefore, we define 1024 kernels of the first fully connected layer. Image recognition network to add dropout mechanism between fully connected layers in order to prevent too many unnecessary kernels of participating in calculations. The second layer of full connectivity define 512 kernels, and the last layer uses 4 kernels as output, representing 4 types of probability results.

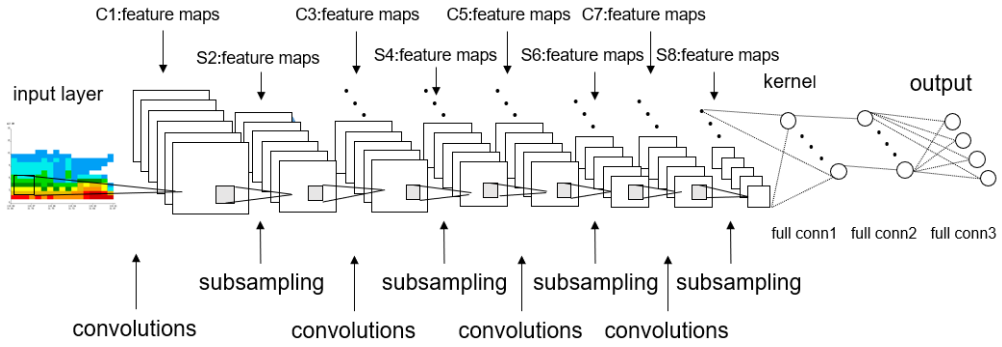

Figure 4: Image recognition framework: We defined 32 convolution kernels of 5x5 dimensions in the first convolutional layer. The second convolution layer sets 64 convolution kernels of 5x5 dimensions; the third convolution layer sets 128 convolution kernels of 3x3 dimensions; The fourth convolution layer also sets 128 convolutions of 3x3 dimensions. Behind the link, there are 3 fully connected layers

3.4 Sample performance test

Although the samples generated by DCGAN have been affirmed visually, there is still necessary to carry out a test to prove whether the sample really has the attributes of real data [Theis, Oord and Bethge (2015)]. We use the pre-trained CNN identification network of Fig. 4 as the detection basis, randomly input part of the generated samples of the network and verify the quality of the generated samples based on the classification results.

4 Experiments

In this section, we will first introduce the dataset and then statistically the pre-training results of the recognition framework as the basis for subsequent optimization. According to the verification result of the generated sample, we have selected part of the generated sample and the real sample to participate retraining based on the pre-training. The final result includes the accuracy in training and the final test accuracy.

4.1 Dataset preparation

Our pre-training dataset has 10,000 radar profiles, including four classification categories: Rain-wind, rain-nowind, norain-wind, norain-nowind. There are 2500 radar profiles in each category, and the pixel size of each image is 540*440. Radar profiles are from radar observation stations in Nanjing and Anhui in 2016 and 2017. We prepared two categories generated samples for quality verification: 200 rain-wind images and 200 rain-nowind images. In the final mixing training, samples generated by DCGAN were expanded to 1000 in each category. In the final testing, we collected the latest radar profiles for classification.

4.2 Deep learning model

All models of image recognition are trained in the deep network framework shown in Fig. 4. The first training starts from the initial state, because the objects we are training are special, and using an off-the-shelf model such as the ImageNet Champion model do not work well.

4.3 Radar profile recognition

We conduct a pre-training experiment firstly, which prove that our CNN network is better than the original CNN network. After the pre-training, it was found that the accuracy of the raw CNN basically converged to about 51%, while the accuracy of our custom CNN structure basically converged to about 84% in Fig. 5. The experimental results show that our designed network has more advantages.

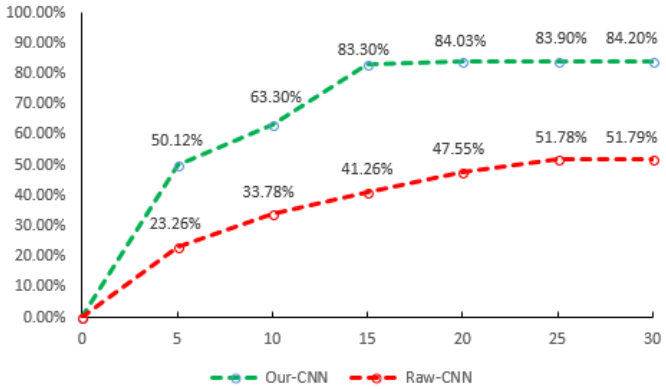

Figure 5: Comparison of customized CNN and raw CNN pre-training result

According to the experimental results, we abandon the raw CNN network structure and run our self-designed model for the following experiments. After testing radar profiles of 4 categories, the results are shown in Fig. 6.

category	rain-wind	rain-nowind	norain-wind	norain-nowind
rain-wind	0.890	0.080	0.030	0.000
rain-nowind	0.040	0.915	0.045	0.000
norain-wind	0.150	0.045	0.770	0.035
norain-nowind	0.000	0.025	0.070	0.905

Figure 6: 4-classification result of radar profiles tested by pre-training models

Before carrying out the mixing training, we verified the authenticity of the DCGAN generated samples. Here, we only need to verify both rain-wind and rain-nowind because these two kinds of samples are relatively difficult to obtain. The generated samples were input into the pre-trained model, and the correct rate of classification was counted. As shown in Fig. 7, the generated samples are already very close to the real samples.

category	rain-wind	rain-nowind	norain-wind	norain-nowind
rain-wind	0.900	0.070	0.030	0.000
rain-nowind	0.060	0.880	0.060	0.000

Figure 7: Generated samples test result

We trained the data generated by DCGAN together with real data and found that the accuracy of the mixed training has improved. In Fig. 8, the accuracy rate after mixing training has converged to 89.37%, and the training process is more stable.

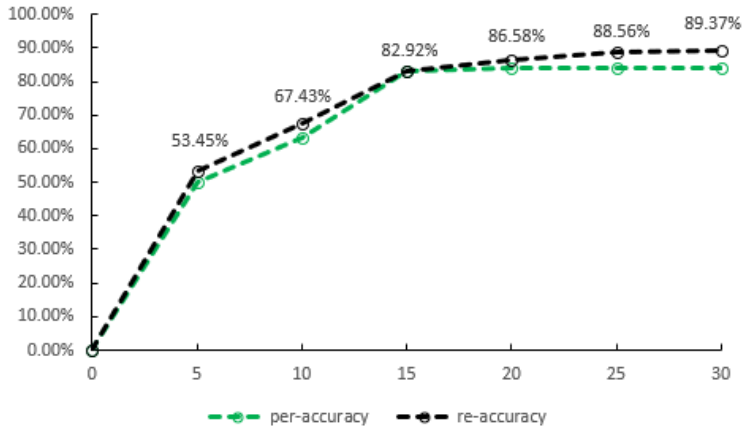

Figure 8: Comparison of pre-training convergence result

Finally, we collected the latest radar data onto testing. The results show that the accuracy of model recognition after mixed training has improved, as shown in Fig. 9.

category	rain-wind	rain-nowind	norain-wind	norain-nowind
rain-wind	0.910	0.040	0.050	0.000
rain-nowind	0.025	0.940	0.035	0.000
norain-wind	0.120	0.040	0.810	0.030
norain-nowind	0.000	0.025	0.055	0.920

Figure 9: The results of test the improved recognition model

5 Conclusion

In this paper, we have combined DCGAN with CNN for the second time. The experimental results show that accelerating learning through the learning rate decayed strategy can make the sample generated by DCGAN more valuable for training. Not only can participate in training with real data onto the CNN network that we design, but also improve the recognition accuracy of radar profile images. At the same time, we also solved the problem of difficult convergence of parameters caused by the difficulty of collecting samples and excessively similar features. In terms of details of recognition, DCGAN can be optimized by adjusting the number of trainings and learning rate to obtain more realistic samples. The CNN image recognition framework can also make the results more accurate by setting network depth and convolution kernel parameters. Combining image recognition of meteorological applications is an extension of deep learning in applications. Later, we can automate the recognition task, detect weather conditions in real time, and use more optimized deep learning algorithms to achieve more accurate weather forecasts.

Acknowledgement: This work was supported in part by the Priority Academic Program Development of Jiangsu Higher Education Institutions.

References

Dixit, M.; Chen, S.; Gao, D.; Rasiwasia, N.; Vasconcelos, N. (2015): Scene classification with semantic fisher vectors. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 2974-2983.

Gong, Y.; Wang, L.; Guo, R.; Lazebnik, S. (2014): Multi-scale orderless pooling of deep convolutional activation features. *European Conference on Computer Vision*, pp. 392-407.

Goodfellow, I.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D. et al. (2014): Generative adversarial nets. *Advances in Neural Information Processing Systems*, pp. 2672-2680.

Gu, B.; Sun, X.; Sheng V. S. (2011): Structural minimax probability machine. *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, no. 7, pp. 1646-1656.

Ioffe, S.; Szegedy, C. (2015): Batch normalization: Accelerating deep network training by reducing internal covariate shift. *Machine Learning*, pp. 1-11.

Isola, P.; Zhu, J. Y.; Zhou, T.; Efros, A. A. (2017): Image-to-Image translation with conditional adversarial networks. *Computer Vision and Pattern Recognition*, pp. 1-17.

Kim, H. J.; Lee, J. S.; Park, J. H. (2008): Dynamic hand gesture recognition using a CNN model with 3D receptive fields. *IEEE Conference on Neural Networks and Signal Processing*, pp. 14-19.

Krizhevsky, A.; Sutskever, I.; Hinton, G. E. (2012): Imagenet classification with deep convolutional neural networks. *Advances in Neural Information Processing Systems*, pp. 1097-1105.

LeCun, Y.; Bottou, L.; Bengio, Y.; Haffner, P. (1998): Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278-2324.

Mirza, M.; Osindero, S. (2014): Conditional generative adversarial nets. *Machine Learning*, pp. 1-7.

Oquab, M.; Bottou, L.; Laptev, I.; Sivic, J. (2014): Learning and transferring mid-level image representations using convolutional neural networks. *IEEE International Conference on Computer Vision and Pattern Recognition*, pp. 1717-1724.

Radford, A.; Metz, L.; Chintala, S. (2015): Unsupervised representation learning with deep convolutional generative adversarial networks. *Computer Science*, pp. 1-15.

Salimans, T.; Goodfellow, I.; Zaremba, W.; Cheung, V.; Radford, A. et al. (2016): Improved techniques for training gans. *Advances in Neural Information Processing Systems*, pp. 2234-2242.

Simonyan, K.; Zisserman, A. (2014): Very deep convolutional networks for large-scale image recognition. *Computer Vision and Pattern Recognition*, pp. 1-14.

Shin, A.; Yamaguchi, M.; Ohnishi, K.; Harada, T. (2016): Dense image representation with spatial pyramid VLAD coding of CNN for locally robust captioning. *Computer Vision and Pattern Recognition*, pp. 1-18.

Szegedy, C.; Liu, W.; Jia, Y.; Sermanet, P.; Reed, S. et al. (2015): Going deeper with convolutions. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1-9.

Theis, L.; Oord, A.; Bethge, M. (2015): A note on the evaluation of generative models. *Machine Learning*, pp. 1-10.

Zhou, Z.; Wu, Q. J.; Huang, F.; Sun, X. (2017): Fast and accurate near-duplicate image elimination for visual sensor networks. *International Journal of Distributed Sensor Networks*, vol. 13, no. 2.