Computers, Materials & Continua CMC, vol.64, no.1, pp.217-231, 2020

A New Sequential Image Prediction Method Based on LSTM and DCGAN

Wei Fang[1, 2], Feihong Zhang[1, *], Yewen Ding[1] and Jack Sheng[3]

Abstract: Image recognition technology is an important field of artificial intelligence. Combined with the development of machine learning technology in recent years, it has great researches value and commercial value. As a matter of fact, a single recognition function can no longer meet people's needs, and accurate image prediction is the trend that people pursue. This paper is based on Long Short-Term Memory (LSTM) and Deep Convolution Generative Adversarial Networks (DCGAN), studies and implements a prediction model by using radar image data. We adopt a stack cascading strategy in designing network connection which can control of parameter convergence better. This new method enables effective learning of image features and makes predictive models to have greater generalization capabilities. Experiments demonstrate that our network model is more robust and efficient in terms of timing prediction than 3DCNN and traditional ConvLSTM. The sequential image prediction model architecture proposed in this paper is theoretically applicable to all sequential images.

Keywords: Image prediction, LSTM, DCGAN.

1 Introduction

Deep learning has achieved notable success in the fields of speech recognition, natural language processing, computer vision, image or video analysis, and multimedia. At the present, the combination of image recognition and deep learning has become a research hotspot in the field of computer vision, but there are still many limitations. This is manifested in that the identified objects are discrete, independent of each other, and primarily classified operations. In order to expand the relevant business needs, the recent development focus on image recognition has been placed on the timing of the images associated with each other, and to extend the traditional classification operation to the prediction operation. Research on time-series images can benefit from a variety of applications, such as short-term heavy

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

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

[3] Department of Economics, Finance, Insurance and Risk Management University of Central Arkansas, Conway, 72035, USA.

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

Received: 18 February 2019; Accepted: 08 April 2019.

rainfall forecasting, video classification, behavior recognition, etc. [Shi, Gao, Lausen et al. (2017); Fernando and Gould (2016); Chen and Forbus (2017)].

For the study of time series data, many algorithms for sequence modeling have been proposed in the past. The most famous of them is the HMM hidden Markov model [Rabiner (1989)]. However, the computational complexity of HMM is too large, even if it is implemented using the dynamic programming Viterbi algorithm [Lou (1995)], there will be incalculable problems when there are too many states. Fortunately, the idea of the Recurrent Neural Network (RNN) was put forward in the late 1980s. RNN is mainly used for the study of time series data. It has end-to-end guidance, specific framework and regularization methods such as weight attenuation, dropout mechanism, and degree of freedom to improve the over-fitting characteristics [Murugan and Durairaj (2017)]. RNN has achieved some success in speech recognition, language modeling, picture description and other issues in the past few years [Xu, Lu, Yang et al. (2017)]. But the traditional RNN has a Long-Term Dependencies problem. Later, in 1997, Hochreater et al. [Hochreiter and Schmidhuber (1997)] proposed an improved version of RNN and recently improved and promoted by Alex Graves. On numerous issues, the LSTM-based model effectively solves the problem of gradient disappearance or gradient explosion and long-term memory deficiency of RNN. Nowadays, LSTM has been applied and developed in various fields. For example, Cho et al. [Cho, Van, Gulcehre et al. (2014)] proposed Gated Recurrent Unit (GRU) in 2014. It combines Forgotten Gate and the Input Gate into a single update gate, which also mixes the cell state and the hidden state. The final model is simpler than the standard LSTM model. The concept of Depth Gated RNN was proposed by Yao et al. [Yao, Cohn, Vylomova et al. (2015)]. Shi et al. [Shi, Chen, Wang et al. (2015)] proposed a new network combining convolution operations with LSTM, which can simultaneously learn spatial and temporal features. Heryadi et al. [Heryadi and Warnars (2017)] utilized Stacked LSTM and CNN-LSTM to identify fraudulent transaction information. Han et al. [Han, Wu, Jiang et al. (2017)] used Bi-LSTM to help customers match the most suitable wearing style. Wang et al. [Wang, Li, Ding et al. (2018)] realized efficient LSTM utilizing structured compression technology on FPGA. Li et al. [Li and Shen (2017)] optimized image description is based on Bi-LSTM and sequence sampling.

In this paper, we refer to the advantages of ConvLSTM and conduct research on its theory. Porting DCGAN to the LSTM ontology solves the timing prediction and provides more robust technical support for image feature extraction and image restoration. In this paper, the radar image is used as the dataset, and the proposed technology is applied to the meteorological field which expands the application range of image recognition.

The major contributions of this paper are as follows:

1. Using DCGAN for image feature extraction and image restoration operations.

2. Single frame prediction and sequence prediction are completed by constructing the internal structure of LSTM.

3. A stack cascading strategy is proposed to solve the problem of unstable initial network training.

4. Optimized the loss function and the learning rate assignment problem, making the prediction result more stable.

2 Relation work

This section describes the basic principles of RNN and LSTM. The principle of a convolutional encoder for feature extraction and feature restoration will also be explained.

2.1 RNN structure principle

As a neural network to cope with sequence problems, RNN introduces the concept of "memory". It regards generating sequences as the essence of dealing with problems. For example, in the application of the simulated writing thesis, when dealing with set operations $R_1 \cup \alpha$, the placeholder α must be a set, and the RNN can learn such sequence features. But before that, we need to convert the processing object to a hard format sequence in order to make RNN train. The structure of the RNN model is presented in Fig. 1, where X_t is the input at time t, symbol U, V, W are weights. Unlike the Convolutional Neural Network (CNN), the entire network of RNNs shares a set of parameters, greatly reducing the amounts of parameters that need to be trained and estimated. S_t is the "memory" state at time t, and the calculation is indicated in Eq. (1). The function f is an activation function and can be tanh or sigmoid. We can think of the hidden state S_t as memory and capture the information on the past time. O_t is the output at time t, the candidate word selected by the function $softmax$, as showed in formula 2. O_t is calculated based on the current time and all the "memory" of the past. The cross entropy loss function is introduced as the objective function of model learning at the output position, and then the parameters are optimized by BPTT algorithm.

$$S_t = f(UX_t + WS_{t-1} + b) \tag{1}$$

$$O_t = softmax(VS_t + a) \tag{2}$$

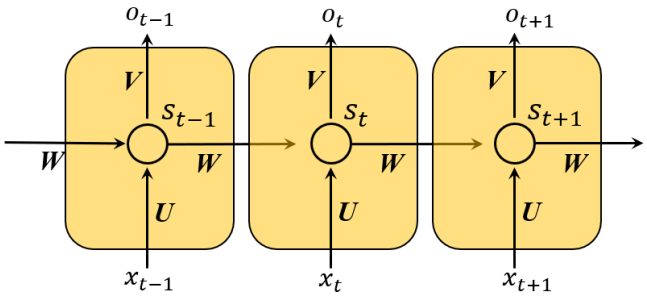


Figure 1: Internal structure diagram of the RNN

In theory, the RNN can handle long-term dependency problems. But in reality, as time goes on, RNN will gradually lose the ability to learn that information. On the one hand, although the RNN can map the input accepted by the neuron node at the current time to between -1 and 1 through the tanh function during training, polynomial multiplication can cause a serious gradient disappearance problem when the parameter is updated by the chain rule. On the other hand, the activation function and network parameter value used by the RNN model may produce a gradient explosion.

2.2 LSTM gate operation

To solve the long-term dependency problem, the LSTM network introduces some sub-

networks of memory blocks to replace the implicit nodes in the cyclic neural network. Compared to traditional RNN, the Forgotten Gate unit is introduced into the standard LSTM network, allowing the network to learn to "forget" and stay away from saturation. The structure of the LSTM networks memory block is illustrated in Fig. 2.

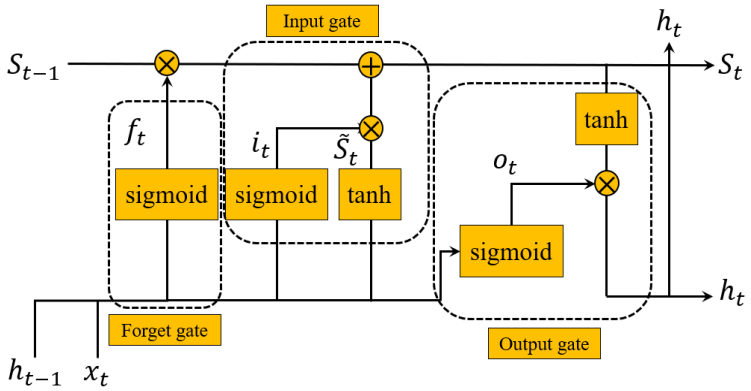


Figure 2: LSTM memory module structure

At heart of LSTM is the transportable cell state, which spreads throughout the memory chain with only a small amount of linear computation. The LSTM is implemented by a "gate" mechanism, such as Fig. 2, when performing operations to remove or add information on the cellular state. It contains a sigmoid network layer and a pointwise multiplication operation. The specific process of gate operation is shown in Tab. 1.

Table 1: Gate operation process

Gate operation	
Forget gate f_t:	It performs a forgetting operation of the cell state. Execute by formula 3.
Input gate:	This layer mainly updates the old state S_{t-1} to the new state S_t. The internal $sigmoid$ layer determines the value i_t that needs to be updated, and the $tanh$ layer is responsible for creating a new candidate value vector $\tilde{S}_t$. According to formula 4 and formula 5 can determine the update content. First multiplied S_{t-1} by f_t and discard the information that needs to be forgotten. After that, we can get the latest status by adding $i_i * \tilde{S}_t$. Its mathematical expression is shown in formula 6.
Output gate o_t:	The output is obtained based on the cell state. First run a $sigmoid$ layer to determine which parts from the cell state will be output. Then use tanh to process the cell state and multiplied it with the output of the $tanh$ gate to obtain our output result. This operation follows formulas 7 and 8.

$$f_t = sigmoid\left(W_f \cdot [h_{t-1}, x_t] + b_f\right) \quad (3)$$

$$i_t = sigmoid(W_i \cdot [h_{t-1}, x_t] + b_i) \quad (4)$$

$$\tilde{S}_t = tanh(W_c \cdot [h_{t-1}, x_t] + b_c) \quad (5)$$

$$S_t = f_t * S_{t-1} + i_t * \tilde{S}_t \quad (6)$$

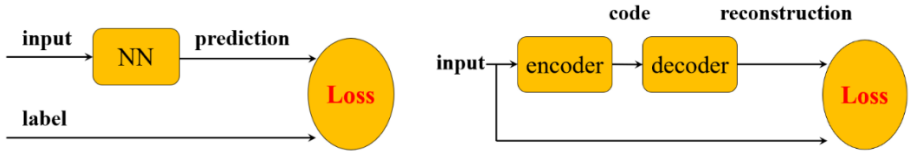
$$o_t = sigmoid(W_o \cdot [h_{t-1}, x_t] + b_o) \quad (7)$$

$$h_t = o_t * tanh(S_t) \quad (8)$$

2.3 Feature extraction and image restoration

The focal point of this paper is to accurately predict the trend of radar image changes in the specified time period in the future. Since the input and output are images, it is necessary to capture the spatial structure of these image data. In this process, feature extraction and image restoration are involved, so the concept of an encoder is introduced.

We assume that the output and input of an artificial neural network are the same ones. Several different feature representations of the input can be achieved by training, and this process is the implementation step of the encoder. The essence of the encoder is a neural network that reproduces the input signal as much as possible, capturing the essential features of the input data, similar to PCA. It has two ways of learning: supervised learning and unsupervised learning. Supervised learning is shown on the left side of Fig. 3. We use the difference between the predicted result and the standard label as a loss function, and adjust the weight parameters of each layer by backpropagation until the function converges. But in reality, more samples are unlabeled. We input these unlabeled data into the encoder to get the input representation. Then obtain a message through the decoding function in the encoder. If the output information is very close to the input information, it is obvious that we have reasons to believe that the encoder is reliable. We can adjust the encoder and decoder parameters shown on the right side of Fig. 3 to minimize the reconstruction loss. This is the way of unsupervised learning.



(a) Supervised learning (b) Unsupervised learning

Figure 3: Feature learning schematic

A convolutional encoder can learn local structural information in a 2D image. It utilizes an unsupervised learning approach that combines convolution and pooling operations to achieve feature extraction. Convolution encoder shares weight and can reconstruct the input using a linear combination of implicitly encoded image blocks. Adopt a single channel image as an example. Take a single channel image x as an example, its calculation process is shown in formula 9. Where σ represents the activation function, h^k represents the kth convolution surface, o represents the reconstruction result and symbol $*$ represents the convolution operation. The kth weights and bias values are represented by w^k and b^k, respectively. The range of the number of convolution surfaces is represented by H. The matrix w^k is flipped through two dimensions to get the matrix $\widetilde{w}^k$. Symbol c is the bias of the input channel. The tolerance between the input sample and the reconstructed result is taken as the loss function as shown in formula 10. A complete convolutional encoder can be obtained by optimized by the BP algorithm. In this article, part of DCGAN is inspired by convolutional encoders.

$$\begin{cases} h^k = \sigma\left(x * w^k + b^k\right) \\ o = \sigma\left(\sum_{k\in H} h^k * \widetilde{w}^k + c\right) \end{cases} \tag{9}$$

$$Loss = \frac{1}{2n}\sum(x_i - y_i)^2 \tag{10}$$

3 Proposed method

This section will elaborate on the radar image prediction method, including radar image feature extraction and reconstruction, prediction network model establishment and stack cascading strategy. The overall process is shown in Fig. 4. Traditional ConvLSTM uses images directly as input to achieve rainfall nowcasting [Salman, Heryadi, Abdurahman et al. (2018)]. This can lead to more complex data processing by the network. We believe that the extraction of effective features of DCGAN can reduce the complexity of LSTM learning.

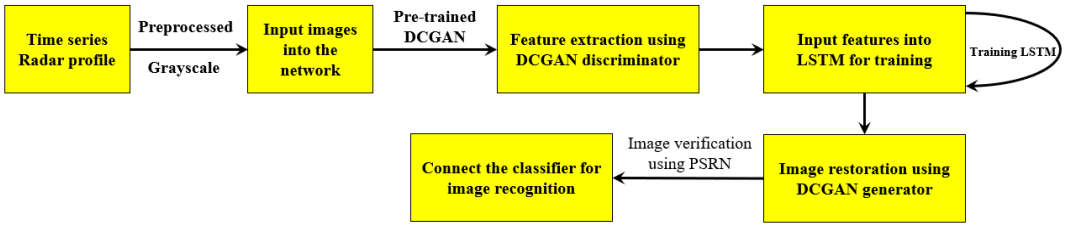


Figure 4: Flow diagram of image prediction

3.1 DCGAN encoder

In order to make the advantages of the two-dimensional feature of the convolutional encoder learning image fully exploit, we have to overwrite the generator and discriminator in the DCGAN structure. DCGAN is developed on CNN. It cannot only learn the semantic features of images, but also remember the distribution of training samples, which are more flexible than CNN [Fang, Zhang, Sheng et al. (2018)]. DCGAN has excellent feature capture capabilities that can be utilized to achieve facial image generation, emotional language expression and digital signal modulation classification [Sagawa and Hagiwara (2018); Chang and Scherer (2017)]. The DCGAN encoder consists of two parts: an encoding module and a decoding module. The coding module realizes feature extraction of the radar image and the decoding module realizes reconstruction of the radar image feature. Based on this idea, this article sets up specific experiments and conducted related verification.

We designed a four-layer convolution coding module structure and a four-layer transposed convolution decoding module structure. The LSTM network is connected between the two modules to learn the feature distribution of the time series image. The overall process is shown in Fig. 5. The images are first input into the encoding module to extract spatial features. These features are subsequently input into the LSTM for learning prediction. Finally, the predicted image can be obtained by decoding module.

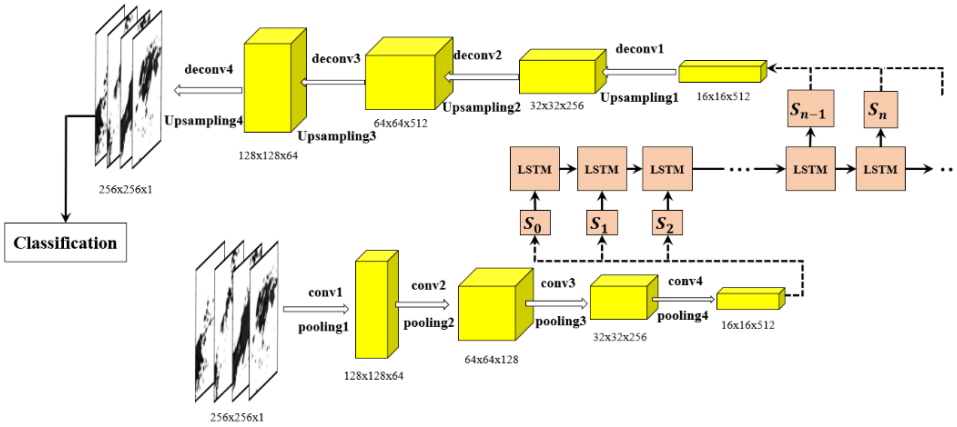


Figure 5: DCGAN encoder structure diagram

Training the encoder requires the following two specific operations: feedforward propagation and overall parameter tuning. Feedforward propagation is the process of an image from input to output in an encoder. The entire encoder should be fine-tuned by the error back propagation algorithm when the feedforward propagation is complete. Assume that there are training samples $N(x^\sigma, y^\sigma)$, $\sigma\epsilon[1, N]$. $x^\sigma = (x_1^\sigma, x_2^\sigma, \dots, x_m^\sigma)^T$ is the input and $y^\sigma = \left(y_1^\sigma, y_2^\sigma, \dots, y_m^\sigma\right)^T$ is the standard output. The predicted output is expressed as $o^\sigma = (o_1^\sigma, o_2^\sigma, \dots, o_m^\sigma)^T$. Although the input x^σ is exactly the same as the standard output y^σ, different symbols are used to distinguish concept. The loss function of the predicted output is defined by the Euclidean distance, as showed in formula 11.

$$Loss = \frac{1}{2}\sum_{\sigma=1}^{N}\sum_{i=1}^{N}(o_i^\sigma - y_i^\sigma)^2 \tag{11}$$

3.2 Stack connection and algorithm optimization

In this paper, the stack structure is adopted in the network structure design. This method enables each layer of the network to be trained separately, which is equivalent to initializing a reasonable value. Of course, its benefits are not confined to this one. We use the encoder with three hidden layers shown in Fig. 6 to explain the principle of stacking connection.

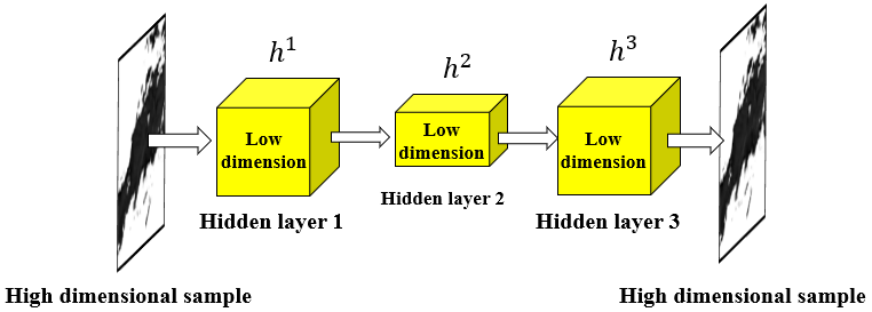


Figure 6: Three-layer encoder structure

We perform independent training for the first hidden layer, and get the first-order feature

representation h^1 of the original input. Taking h^1 as input, it performs independent coding training again, and simultaneously acquires second-order features representation h^2. The third layer uses h^2 as input to generate high-dimensional data. Finally, three hidden layers are combined to form a stack-connected encoder network. The specific structure is shown in Fig. 7. Except for the final layer, the loss function by which each layer of training is based is defined by the square of the difference between the output and input of each layer of the network. For high-dimensional data, it is very difficult to quickly come up with a complete set of available models. Due to too many node parameters, blindly increasing the depth will only make the result more and more uncontrollable. The stack cascading method enables each layer of the network to be individually trained to ensure dimensional controllability. This kind of practice can be understood as the layer-by-layer dimensionality reduction of complex problems, and the second-level training is directly performed using the feature values after dimensionality reduction, so that the network depth can be controlled. After independent training, we can easily get the value of each intermediate layer. After linking up in series, the overall network tuning is performed. In this paper, the stack cascading strategy is adopted both inside the DCGAN encoder and at the LSTM connection.

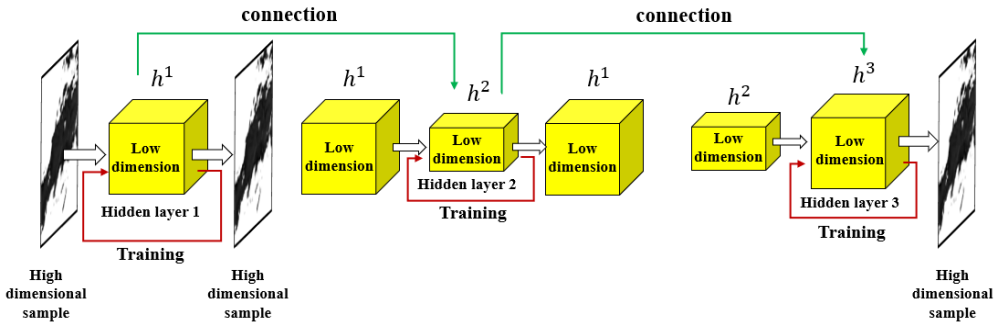


Figure 7: Stack structure diagram

In order to make the model have better generalization ability and avoid over-fitting to achieve good balance, we introduce the concept of regularization. The L1 and L2 regularities can be expressed as formulas 12 and 13, respectively:

$$L1(\theta) = \alpha \sum_i |\theta_i| \tag{12}$$

$$L2(\theta) = \frac{1}{2}\alpha \sum_i |\theta_i|^2 \tag{13}$$

Regularization can be seen as a penalty for the loss function. L1 regularization can generate a sparse weight matrix for feature selection, L2 regularization [Radford, Metz and Chintala (2015)] prevents model overfitting. The regular term α is a coefficient, and θ_i can be expressed as the reciprocal of the weight of each layer, indicating that the level of update should be reduced for the layer that learns too high weight. On the contrary, for the nodes in the layer that learn too low weights, the degree of update is to be increased, so as to achieve the purpose of equalizing the ownership value in the layer. Therefore, in this paper, the L2 regularization is added after the loss function of the encoder, as showed in the formula 14.

$$Loss = \frac{1}{2}\sum_{\sigma=1}^{N}\sum_{i=1}^{N}(o_i^{\sigma} - y_i^{\sigma})^2 + \frac{1}{2}\alpha\sum_{i}^{N}(\theta)^2 \quad (14)$$

3.3 Predictive feature

We use the DCGAN coding module to obtain the input time series, and then convert it into a fixed-size state vector to complete the time series feature extraction. At this time, the information volume of the entire input sequence will be stored in the cell state S_t of the LSTM neurons. The LSTM prediction module will use the cell state of the above neurons as the initial state of the cell, and output a predicted sequence of future time periods. The cell contains three gate units, which use the BPTT algorithm to calculate the gradient to achieve weight update. In the construction of the LSTM model, this article focuses on ConvLSTM. The calculation method in the gates is converted into convolution operation to realize the convolution operation of radar image features and weights. This paper is divided into single frame prediction and sequence prediction in predicting radar images. Single frame prediction only completes prediction of the next frame image. Sequence prediction outputs multiple consecutive frame images of a fixed time period.

First, we consider single-step prediction. Suppose we have a data set $X = \{x_1, x_2, \dots, x_n\}$. The purpose is to generate the nth data using the former n-1 time series data. As shown in Fig. 2, each cell was found to have a fixed output. We can understand this output as a prediction of the next moment, so that for the data set X that removes the last item, the network output will be a new prediction sequence $\hat{X} = \{\hat{x}_2, \hat{x}_3, \dots, \hat{x}_n\}$. We only need to extract the last output of $\hat{X}$ to get the predicted value, and the overall loss function of the network will be defined as $(\{x_1, x_2, \dots, x_n\} - \{\hat{x}_2, \hat{x}_3, \dots, \hat{x}_n\})^2$. The principle of single frame prediction is shown in Fig. 8.

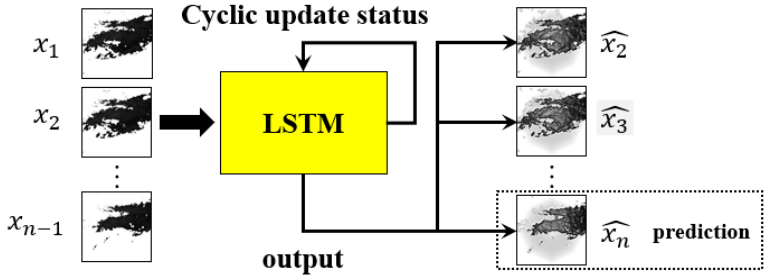


Figure 8: Single frame prediction schematic

Sequence prediction is based on a single frame prediction, but the structure is more complicated, as showed in Fig. 9. We consider that after each picture passes through a LSTM network, its output is the next frame prediction. Theoretically, when the image at time t_1 is updated n times, the state at time t_{n+1} will be output. Suppose we predict the subsequent time series image according to the data set $\{x_1, x_2, \dots, x_{n-t}\}$, then we need the original data to update the t-level state to get the prediction sequence $\{\hat{x}_{n-t+1}, \hat{x}_{n-t+2}, \dots, \hat{x}_n\}$. Therefore, we need to increase the depth of the LSTM network in the single-step prediction model vertically, and the specific depth is determined

according to requirements.

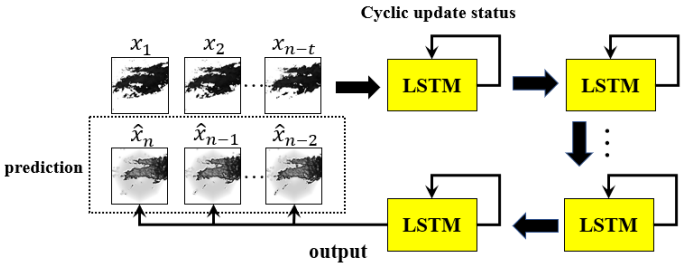


Figure 9: Sequence prediction schematic

4 Experiment

In this section, we first briefly present the radar dataset. Then perform training verification on the DCGAN used for image feature extraction and image restoration. Finally link the LSTM and DCGAN encoder to compare 3DCNN and traditional ConvLSTM with error analysis to demonstrate our superiority. The experimental result includes an assessment of the loss of the predicted results and an image showing the predictions.

4.1 Radar dataset

The standard radar data set used in this article is SRAD2018, the data format is gray scale images, which can be downloaded from the designated official website. (https://tianchi.aliyun.com/competition/entrance/231662/information). We randomly selected 200 sets of data as the training data set and 10 sets of data as the test data set. The data of each set of training are radar sample data covering 6 hours and 6 minutes interval. Each set of data in the test provides only first five hours six minutes apart. In single-frame prediction, it is only obtained a radar image which for five hours six minutes. Time series prediction is to obtain the radar image result in the next hour.

4.2 Encoder comparison

According to the frame structure shown in Fig. 6, we temporarily train the neural network of image feature extraction and image restoration without considering LSTM. The DCGAN network module that encodes and decodes the image is set to a depth of four layers and formula 14 is trained as a loss function. Due to the overall stack cascading strategy, once a more accurate model is generated, the model parameters will be persisted, avoiding the recovery of poorly performing images. Training the DCGAN dataset does not require much image data. After 50 batch training, the model approaches convergence. In order to reflect the advantages of this article using DCGAN for encoding and decoding operations, we compared other methods: sparse encoder and stack encoder. The abscissa indicates the number of iterations and the ordinate indicates the loss. We can see that the DCGAN encoder produce less error than the other two encoders. We compare the training processes of these three encoders together, as showed in Fig. 10. Red indicates sparse encoder, green line indicates stack encoder, and the blue line indicates DCGAN encoder. The black line indicates that there is no optimized DCGAN

encoder, and the training is very unstable. Through this comparison chart, we can clearly see the superiority of the DCGAN encoder combined with the stack cascading strategy, and its loss is the lowest.

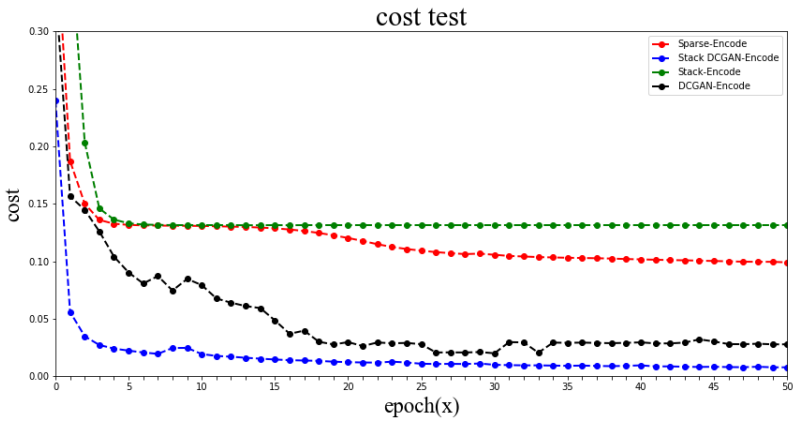


Figure 10: Sequence prediction schematic

For the images generated by each encoder, we perform a comprehensive comparison by PSNR, RMSE, SSIM and Cost factor. As showed in Tab. 2, the DCGAN encoder is the most superior under the comprehensive comparison.

Table 2: Comprehensive evaluation

Method	PSNR	RMSE	SSIM	Cost
DCGAN encoder	28.4944	46.4323	0.8208	0.0079
Sparse encoder	20.5621	79.4781	0.6245	0.0992
Stack encoder	18.0277	83.9036	0.4992	0.1316

At the time, the encoder already has a good function of feature extraction and image restoration. We put a radar image into the encoder and the reconstructed image has a relatively high degree of resilience, as showed in Fig. 11.

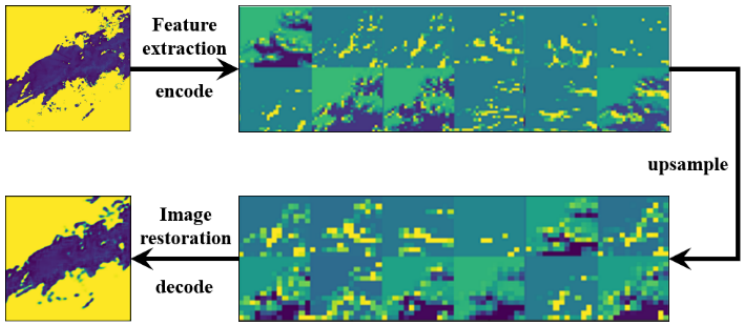
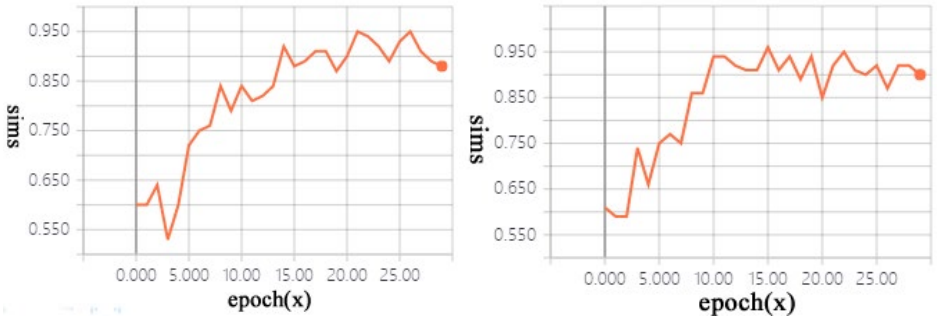


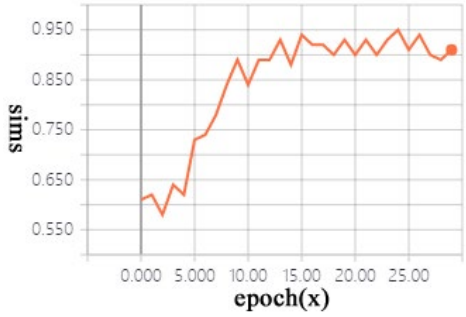
Figure 11: Feature extraction and image restoration

4.3 Image prediction based on LSTM

After obtaining a robust coding model, we will predict the radar image through LSTM. The training set is put into the DCGAN encoder and the features of the image are then transfer to the LSTM. We need to compare with 3DCNN and traditional ConvLSTM to show the robustness of the proposed method. We first perform single-frame prediction, which is a feasible verification of the subsequent sequence prediction. As showed in Fig. 12, when the training batch is 30 times, all three enter the convergence oscillating zone. Through observation, DCGANLSTM has an advantage in the single-frame prediction training, but it has not much different from the other two. On the left is the 3DCNN training process, with ConvLSTM in the middle and DCGANLSTM on the right. The abscissa represents the iteration batch and the ordinate represents the similarity. The horizontal axis represents the number of training and the vertical axis represents the similarity.



(a) 3DCNN (b) ConvLSTM



(c) DCGANLSTM

Figure 12: Comparison of image restoration

After completing the single frame prediction, it is time to enter the sequence prediction experiment. We reset the neural network in the same way as mentioned above, increasing the network depth while preserving the stack cascading strategy. After 3000 iterations by batch, the error distribution images of the three methods are obtained. In order to facilitate the observation of the final results of the three methods, we merge the error distribution maps as showed in Fig. 13. We find that 3DCNN and ConvLSTM training are not stable. Although the latter basically converges to a range, the oscillation of the range is not good for an accurate prediction. It is observed that DCGANLSTM

outperform 3DCNN and ConvLSTM in accuracy and stability.

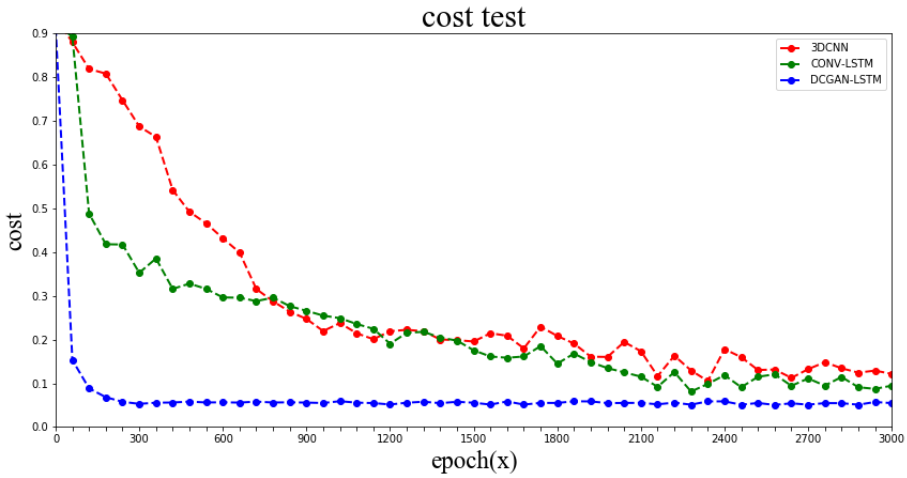


Figure 13: Comparison of sequence prediction methods

Some prediction results are shown in Fig. 14. This illustration includes the implementation of two cases corresponding to the three methods. From top to bottom: prediction by 3DCNN, ConvLSTM and DCGAN-LSTM. We can find that DCGAN-LSTM can predict the image more sharply especially in the boundary. Since the network model is not very accurate, it is almost impossible to give sharp and accurate predictions of the whole radar images in longer-term predictions.

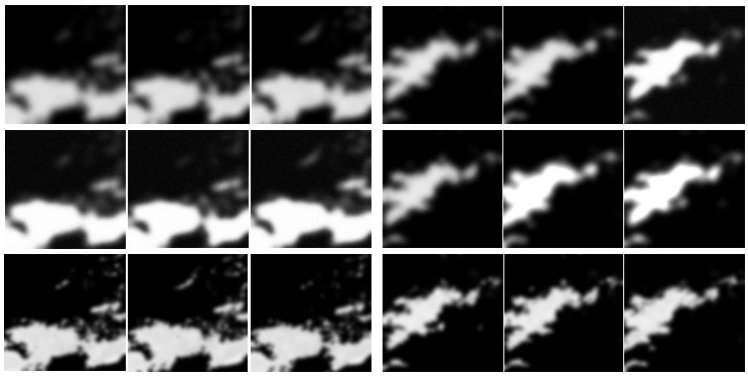


Figure 14: Examples for the image prediction result

5 Conclusion

In this paper, we propose a radar image prediction method based on LSTM and DCGAN. The experimental results show that the encoder modified by DCGAN is more robust than other encoders, and the LSTM with stack cascade strategy is also lower in the prediction error rate than 3DCNN and ConvLSTM. By analyzing the internal characteristics of the network, we modified the appropriate activation function and adopted a learning rate

attenuation strategy to optimize the resulting oscillations during the training process. At the same time, we use L2 regularization to rewrite the loss function in the encoder, avoiding over-fitting. Combining DCGAN and LSTM is an extension of the deep learning algorithm in the meteorological field. In the later stages, we will refine the model and add classification capabilities. Classification is given at the same time as the predicted image is output to achieve a complete meteorological prediction requirement.

Acknowledgement: This work was supported in part by the Open Research Project of State Key Laboratory of Novel Software Technology under Grant KFKT2018B23, the Priority Academic Program Development of Jiangsu Higher Education Institutions, and the Open Project Program of the State Key Lab of CAD\&CG (Grant No. A1916), Zhejiang University.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

Chang, J.; Scherer, S. (2017): Learning representations of emotional speech with deep convolutional generative adversarial networks. *IEEE International Conference on Acoustics, Speech and Signal Processing*, pp. 2746-2750.

Chen, K.; Forbus, K. (2018): Action recognition from skeleton data via analogical generalization over qualitative representations. *Thirty-Second AAAI Conference on Artificial Intelligence*.

Cho, K.; Van Merrienboer, B.; Gulcehre, C.; Bahdanau, D.; Bougares, F. et al. (2014): Learning phrase representations using RNN encoder-decoder for statistical machine translation. arXiv:1406.1078.

Fang, W.; Zhang, F.; Sheng, V. S.; Ding, Y. (2018): A method for improving CNN-based image recognition using DCGAN. *Computers, Materials & Continua*, vol. 57, no. 1, pp. 167-178.

Fernando, B.; Gould, S. (2016): Learning end-to-end video classification with rank-pooling. *International Conference on Machine Learning*, pp. 1187-1196.

Han, X.; Wu, Z.; Jiang, Y. G.; Davis, L. S. (2017): Learning fashion compatibility with bidirectional LSTMs. *Proceedings of the ACM on Multimedia Conference*, pp. 1078-1086.

Heryadi, Y.; Warnars, H. L. H. S. (2017): Learning temporal representation of transaction amount for fraudulent transaction recognition using CNN, stacked lSTM, and CNN-LSTM. *IEEE International Conference on Cybernetics and Computational Intelligence*, pp. 84-89.

Hochreiter, S.; Schmidhuber, J. (1997): Long short-term memory. *Neural Computation*, vol. 9, no. 8, pp. 1735-1780.

Li, J.; Shen, Y. (2017): Image describing based on bidirectional LSTM and improved sequence sampling. *IEEE 2nd International Conference on Big Data Analysis*, pp. 735-739.

Lou, H. L. (1995): Implementing the viterbi algorithm. *IEEE Signal Processing*

Magazine, vol. 12, no. 5, pp. 42-52.

Murugan, P.; Durairaj, S. (2017): Regularization and optimization strategies in deep convolutional neural network. arXiv:1712.04711.

Rabiner, L. R. (1989): A tutorial on hidden markov models and selected applications in speech recognition. *Proceedings of the IEEE*, vol. 77, no. 2, pp. 257-286.

Radford, A.; Metz, L.; Chintala, S. (2015): Unsupervised representation learning with deep convolutional generative adversarial networks. arXiv:1511.06434.

Sagawa, Y.; Hagiwara, M. (2018): Face image generation system using attribute information with DCGANs. *Proceedings of the 2nd International Conference on Machine Learning and Soft Computing*, pp. 109-113.

Salman, A. G.; Heryadi, Y.; Abdurahman, E.; Suparta, W. (2018): Single layer & multi-layer long short-term memory (LSTM) model with intermediate variables for weather forecasting. *Procedia Computer Science*, vol. 135, pp. 89-98.

Shi, X.; Chen, Z.; Wang, H.; Yeung, D. Y.; Wong, W. K. et al. (2015): Convolutional lstm network: a machine learning approach for precipitation nowcasting. *Advances in Neural Information Processing Systems*, pp. 802-810.

Shi, X.; Gao, Z.; Lausen, L.; Wang, H.; Yeung, D. Y. et al. (2017): Deep learning for precipitation nowcasting: a benchmark and a new model. *Advances in Neural Information Processing Systems*, pp. 5617-5627.

Wang, S.; Li, Z.; Ding, C.; Yuan, B.; Qiu, Q. et al. (2018): C-LSTM: enabling efficient LSTM using structured compression techniques on FPGAs. *Proceedings of the ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, pp. 11-20.

Xu, H.; Lu, H.; Yang, G.; Zhang, C. (2017): Sentiment analysis of Chinese version using SVM & RNN. *Proceedings of the 6th International Conference on Information Engineering*, pp. 5.

Yao, K.; Cohn, T.; Vylomova, K.; Duh, K.; Dyer, C. (2015): Depth-gated recurrent neural networks. arXiv:1508.037909.