Prediction of Future Appearances via Convolutional Recurrent Neural Networks Based on Image Time Series in Cloud Computing

Zao Zhang, Xiaohua Li

State University of New York at Binghamton, Binghamton, NY 13902, USA
zhangzao1003@163.com, xli@binghamton.edu

Abstract. In recent years, cloud computing has become a prevalent platform to run artificial intelligence (AI) and deep learning applications. With cloud services, AI models can be deployed easily for the convenience of users. However, although cloud service providers such as Amazon Web Services (AWS) have provided various services to support AI applications, the design of AI models is still the key in many specific applications such as forecasting or prediction. For example, how to forecast the future appearance of ornamental plants or pets? To deal with this problem, in this paper we develop a convolutional recurrent neural network (CRNN) model to forecast the future appearance according to their past appearance images. Specifically, we study the problem of using the pine tree's past appearance images to forecast its future appearance images. We use a plant simulation software to generate pine tree's growing images to train the model. As a result, our model can generate the future appearance image of the pine tree, and the generated images are very similar to the true images. This means our model can work well to forecast the future appearance based on the image series.

Keywords: Cloud Computing, CRNN, Image series, Forecasting.

1 Introduction

In today's computer science field, cloud computing and artificial intelligence are the two most potential research directions. Cloud computing is an important innovation in the information area. The core idea of cloud computing is to coordinate many computer resources together so as to allow users to obtain unlimited resources through the network without the limitation of time and space. Cloud computing is a kind of services related to information technology, software, and the Internet. Cloud computing brings together many computing resources and implements automatic management through software. Computing platforms such as Amazon web service (AWS) can provide this type of on-demand cloud computing services to individuals or companies, and users can build their project conveniently.

To build a project for a specific task, artificial intelligence is a popular choice, because artificial intelligence always leads to low human resource cost and high working speed. There are many subclasses in artificial intelligence. Among them, deep learning has become increasingly popular due to its wide applications in many areas such as computer vision, speech recognition, natural language processing, text processing, etc. Although the specific name of “ deep learning ” was coined just several years ago, deep learning has achieved profound progress and played important roles in the living and working of human beings. The quick and dramatic success of deep learning has motivated many more people to conduct further research in deep learning. Due to the importance of cloud computing and deep learning, it is not difficult to see that a combination of them has great potential.

A lot of artificial intelligence or deep learning models have been developed. Many of them have been applied successfully in practice to meet user’s requirement, and the applications usually rely on cloud computing because of the high computing resource and data resource requirement.

However, for many specific applications, existing artificial intelligence models are still unable to meet people’s requirements. For instance, with the continuous improvement of social productivity, people's living standards are also constantly improving, and people naturally begin to pursue more spiritual enjoyment. Raising ornamental plants and keeping pets have become very popular entertainment. However, when people buy ornamental plants or pets, they can't know if flowers or pets grow into their favorite appearance in the future. At this time, a scientific and accurate prediction method is needed to help buyers choose their flowers and pets.

As to future appearance forecasting, there are existing image-based aging models for human face aging forecasting. Although face aging has very interesting market applications, the models rely on special face patterns and can hardly extended directly to other forecasting applications. Existing face aging methods can be roughly categorized into prototype-based approaches and physical model-based approaches. Prototype-based approaches transfer feature of age according to the difference of face images in different age groups [1] [2]. Physical model-based approaches focus on the block physical details of the human face [3] [4]. But neither of them can be well promoted to the field of image aging outside the face aging because both of them focus on the unique structure of human face to improve their forecasting accuracy. For other applications that are different from human face forecasting, such as forecasting the future appearance of plants or animals, new deep learning models need to be developed.

For this purpose, this paper develops a new convolutional recurrent neural networks (CRNN) model [5] [6], which can effectively forecast the plants' or pets' future appearance according to the image series of their past appearance. The CRNN neural network model developed in this paper is a multi-level neural network model consisting of a convolutional neural network (CNN) portion and a recurrent neural network (RNN) portion. The CNN portion is used to extract features in each image, while the RNN portion is used to process the time correlation among images and generate simulated forecasting images. The desired training data of this CRNN model is time-series images, which means a sequence of images of one individual under different ages. For example, a sequence of images of a tree of one-year old, two-year old, three-year old, and so on. The output of the model is also a sequence of images of the same individual with older age labels. For example, a sequence of images of the same tree of four-year old, five-year old, six-year old and so on. Our experiments show that our model can successfully generate future images that are pretty similar to the label images.

There are two contributions in this paper. The first and the most important contribution is that we find a good model to forecast the future appearance of one object according to its past appearance. Plants and animals always look similar when they are young. But when they grow up, they will have significant differences. At this time, if we can accurately forecast their future appearance through our neural network model, the buyers will be able to choose satisfied plants or pets. This model can be applied in shops to help customers purchase plants and pets. The application is not limited to such an example, though. For example, it can be helpful to plant breeding. Existing plant breeding techniques, whether genetic techniques, chromosome techniques or molecular techniques, all need to select desired characteristics by long-time observation of the plants. If we can forecast plants' appearance features by our model, we can save a lot of time and costs.

The second contribution of this paper is that we demonstrate the CRNN structure can work very well in image series generation and forecasting. There are two main popular image generation methods: conditional generative adversarial networks (CGAN) [13] and conditional adversarial autoencoder (CAAE) [11]. This paper gives a new alternative method. In addition, we also show that the CRNN can learn time dependency among images well. The majority of existing researches use CRNN structure to learn the spatial dependency of an image, while we use the CRNN differently.

The rest of this paper is organized as follows. In Section II, a brief review of related work will be given, including several existing forecasting methods and some application cases of CRNN. Then our CRNN structure will be described in Section III. In Section IV, results of our experiments and evaluation will be given. Finally, a conclusion of our work and a discussion about some possible future research directions will be given.

2 Related Work

2.1 Image Forecasting Networks Model

Some effective imaging forecasting and face aging generative networks model include Variational Autoencoder (VAE), Adversarial Autoencoder (AAE), Conditional Adversarial Autoencoder (CAAE) and Conditional Generative Adversarial Networks (CGAN), which will be reviewed below.

Variational Autoencoder is a useful approach to unsupervised learning of complicated distributions, VAE is appealing because they are built on top of standard function approximates (neural networks), and can be trained with stochastic gradient descent [7]. There have been a lot of improvements based on the original VAE defined in 2013 by Kingma and Rezende. For instance, in 2016, Y. Pu et al. developed a new Variational Autoencoder setup to analyze images, this VAE uses

Deep Generative Deconvolutional Networks (DGDN) as the decoder and uses a Convolutional Neural Networks (CNN) based encoder for the distribution of DGDN parameters [8]. The main advantage of VAE is that we can compare the pixel values of generated images and input images more easily because VAE uses encoding-decoding. The main disadvantage of VAE is that the generated images are always blurry, which means VAE may not be suitable in some situations with high precision requirements.

Adversarial Autoencoder is a probabilistic autoencoder that uses the recently proposed GAN to perform variational inference by matching the aggregated posterior of the hidden code vector of the autoencoder with an arbitrary prior distribution [9]. AAE combines the strengths of VAE and GAN. AAE discarded KL-divergence loss and choose to use adversarial networks. But AAE can still extract and learn the feature of images by the encoder part, which replaces the random noise-based generative networks of GAN and leads to more precise generative networks. These characteristics make AAE works well in generative networks and semi-classification problem.

In the structure of CAAE, a convolutional neural network is adopted as the encoder. The convolution is employed instead of pooling and allows the network to learn its spatial downsampling [10]. Two discriminator networks are imposed and this makes CAAE can generate more photo-realistic images. Compared to AAE, the main difference is that the CAAE uses discriminator in both the encoder and the generator. This structure guarantees a smooth transition in the latent space, and the discriminator on generator assists to generate photo-realistic images [11].

Conditional Generative Adversarial Networks (CGAN) is developed from the traditional Generative Adversarial Network (GAN). CGAN adds a condition in both the generator and the discriminator, and such condition could be added based on the label or some part of data [12]. The condition is feeding with input noise together, and will combine into joint hidden representation. The adversarial training framework allows for considerable flexibility in how this hidden representation is composed [13]. As a result, CGAN can generate images with specific attributes more conveniently.

2.2 Applications of CRNN

Convolutional Neural Networks (CNN) and Recurrent Neural Networks (RNN) are two widely used neural network structures. Convolutional neural networks are special neural network architectures that are especially suitable for processing image data. Besides their superior performance in image processing, they can also be used to process other types of data, such as text, speech, audio, etc. Convolutional neural network architectures are usually built with the following layers: convolution layer, activation function layer, pooling layer, fully connected layer and loss layer [14]. Recurrent neural networks (RNN) are developed specifically for processing sequential data with correlations among data samples. They have the nice capability of processing sequential data, and can be designed to model both long and short term data correlations. By combining the CNN and RNN, the CRNN not only utilizes the representation power of CNN, but also employs the context modeling ability of RNN.

The CNN layers can learn good middle-level features, and help the RNN layer to learn effective spatial dependencies between image region features. Meanwhile, the context information encoded by RNN can lead to better image representation, and transmit more accurate supervisions to CNN layers during backpropagation (BP) [15].

In a single image, the distribution of features always relies on each other, and CRNN can work very well in this task. Because CNN can extract the embedded features and RNN can process their spatial dependency, CRNN has been used in single-image distribution learning tasks [15]. Another task, i.e., learning spatial dependency of the image, is more complicated. For example, if images are highly occluded, the recover the original image including the occluded portion is very difficult. Some researchers are still working in this area. But if the occluded images are image series with some inherent context information, this problem can be processed with the CRNN model. In the paper [16], the CRNN structure works very well and gets good performance.

CRNN structure has also been applied to the text recognition problems, where CNN can be used to recognize a single character while RNN can be used to extract text dependency according to the context. Especially, if the edge feature of the text is strong, then a Max-Feature-Map (MFM) layer can be added into the CRNN model to enhance the contrast [17].

CRNN also shows pretty good performance in music classification tasks, where CNN can be used to extract local feature and RNN can be used to extract temporal summarization of the extracted features [18].

3 CRNN Model for Prediction of Future Appearances

This section focuses on developing our CRNN model to forecast the future appearance of plants. The illustration of the convolutional recurrent neural network model is shown in Fig. 1.

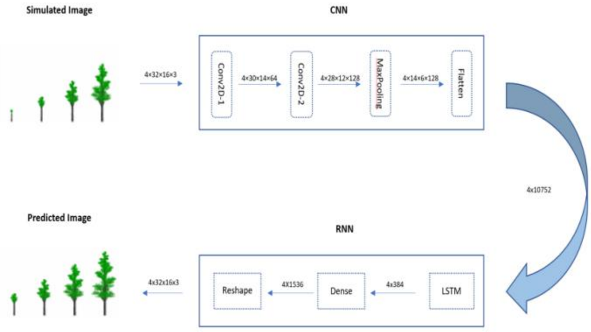

Fig. 1. Outline of CRNN Model

As shown clearly in Fig. 1, our training data are pine tree image series with series length 4, which includes images of a pine tree at 10, 15, 20 and 25 years old. Then we apply a CNN to process each image. The CNN mainly consists of four layers: two convolution layers, one pooling layer and one flatten layer. After the CNN, there is an RNN with LSTM structure, which mainly consists of two layers: an LSTM layer and a Dense layer. The output will be a generated image series of length 4. This will be compared with the label, which is a real image series with series length 4 as well. The label is the images of this pine tree at ages 15, 20, 25 and 30 years. We focus on analyzing the performance of predicting the pine tree at 30 years of age from this pine tree's images at 10, 15, 20 and 25 years of age. After training, this CRNN model can be used to predict the future appearance of pine trees from their past appearances.

3.1 CRNN Forecasting Model

As shown in Fig. 1. Our input data is an image series $\mathbf{x}_{i,t}$ with size T×H×W×D, which i denotes the index number of images sequence and t donates the time step label in time-series images sequence. H means the height of each image, W means the width of each image and D means the dimensions of each image. Input data is sent into our CNN portion and the output of CNN portion is a tensor $\mathbf{z}_{i,t}$, which equals to:

$$\mathbf{z}_{i,t} = f(\mathbf{x}_{i,t}; \{\mathbf{w}_x\}) \tag{1}$$

where $\mathbf{w}_x$ denotes the weighting coefficients in our CNN portion. Two CNN layer extracts the spatial feature in each image and adds one max-pooling layer and one flatten layer. Our CNN model can learn spatial dependency in each image individually. The CNN portion can map our input data $\mathbf{x}_{i,t}$ to tensor $\mathbf{z}_{i,t}$, and $\mathbf{z}_{i,t}$ is the input of the RNN portion.

In our RNN portion, the LSTM layer is the core structure to learn time dependence in time series images sequence, and the LSTM layer maps the tensor $\mathbf{z}_{i,t}$ to a representation series $\mathbf{h}_{i,t}$ which equals to:

$$\mathbf{h}_{i,t} = f(\mathbf{h}_{i,t-1}, \mathbf{z}_{i,t}; \{\mathbf{w}_z\}) \tag{2}$$

where $\mathbf{w}_z$ denotes the weighting coefficients in the LSTM layer. Then the output of the LSTM layer $\mathbf{H}_i$ is sent to a dense layer. Actually, the dense layer is a classifier to classify the pixel value of every pixel, the value internal is [0,255], which means the classification class is 256. The number of pixels is equal to our input time-series images sequence which is T×H×W×D. The output of the dense layer equals to:

$$\hat{\mathbf{y}}_i = f(\mathbf{H}_i; \{\mathbf{w}_h\}) \tag{3}$$

In the last, a reshape layer is added to transfer our prediction value to generated forecasting images. Our model can generate forecasting future images according to the past time-series images of the same individual.

3.2 Data processing in CRNN Model

In order to understand our CRNN model better, it is helpful to describe the procedure of data processing in detail, including the dimensions and values of important parameters and tensors. The values of the CRNN parameters are also selected carefully with many repeated experiments.

From Fig. 1, we can see that the dimension of the input tensor is 4 × 32 × 16 × 3, which means the input data is a series of color images with series length 4 and the size of each image is 32 rows and 16 columns. Because the kernel size of the first convolution layer is 3 × 3 and the number of filters is 64, the output of the first convolution layer is a tensor of dimension 4×30×14×64.

The output of the first convolution layer becomes the input of the second convolution layer. In the second convolution layer, the number of filters is 128 and the kernel size is still 3×3. Therefore, the dimension of the output tensor of the second convolution layer is 4×28×12×128. After the two convolution layers, a pooling layer is applied, which reduces the dimension of the data tensor to 4 × 14 × 6 × 128 with a stride (or decimation ratio) 2.

Then, a Flatten layer is used in order to connect the CNN with the RNN. As the layer name suggests, the function of this layer is to flatten each 4 × 14 × 6 × 128 data tensor into a two-dimensional data array with size 4 × (14 × 6 × 128) = 4 × 10752. This finishes the CNN portion of the CRNN model.

Note that the CNN portion processes each image individually. Next, we apply RNN to learn the information embedded in the image series. The first layer of the RNN portion is an LSTM layer. The LSTM layer has 4 time steps, which consists of 4 LSTM cells. We set the dimensions of both the LSTM states and outputs to be 384. Therefore, the output of the LSTM layer is a data array with dimension 4×384.

To generate the predicted images, we use a Dense layer to generate output data tensors with the same dimension as the targeting pine tree image series. Specifically, the dimension is 4×1536. Note that 1536 equals to 32×16×3, the size of a color image. We apply a reshape step at the end to obtain 4 predicted images with size 32×16×3. This will be compared to the label image series for loss function calculation during training.

4 Experiments

4.1 Data Collection

We apply the software tool L-studio to generate images of pine trees. L-studio is a standard software that has been used widely in the botanical research community to create plant simulation models [19]. With L-studio we can conveniently generate various plant images with desired labels. The generated time-series images of a kind of pine tree can be shown in Fig. 2.

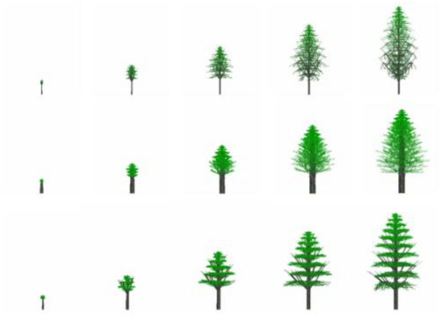

Fig. 2. Time-series images of pine tree generated by L-Studio

We chose this kind of pine because there are many variable parameters to guarantee the diversity of simulated images. As shown in Fig. 1. We choose years of pine to be 10, 15, 20, 25, and 30, which means to model pine that is 10 years old, 15 years old, 20 years old, 25 years old, and 30 years old, respectively. Each image of each pine tree thus forms an image series. Each series consists of 5 images, which are images of a pine tree at 10, 15, 20, 25 and 30 years old.

Besides the parameter of the age of the tree, there are many other parameters that can be adjusted in order to model different pine trees and to generate different images. Some important parameters include the growth rate of tree's height, the growth rate of tree's diameter, the shrinkage rate of tree's growth, the slope of needle leaf of pine, the angle between branch, the slope of branch, the shrinkage rate of tree's height growth, the shrinkage rate of tree's diameter growth, the shrinkage rate of branch, the shrinkage rate of second-level branch, the diameter of branch, the diameter of crown. There are about fourteen parameters that can be used to generate different trees. Each of these parameters is adjustable within a certain interval, such as [0, 1] or [0, 180] degree. Therefore, L-studio is very powerful for generating pine tree images with great differences in appearance. We can be confident our training data is fully diverse.

We use such simulated image series as training data to train our CRNN model. We use the 3463 series (about nine-tenths of the total data) as training data and 384 series (about one-tenth of the total data) as test data. Our CRNN is expected to predict the fifth image, which means the image of the 30 years-old pine tree, based on the first four images. This means we predict the future appearance images of the pine tree after several years of growth. We will compare the real fifth image with the predicted image to analyze the prediction performance.

4.2 Parameter Optimization

Several parameters in the CNN portion need to be set after test, such as the number of filters, size of the kernel, stride value, padding method, dilation rate, activation functions, etc. The difference in the activation function brings the biggest difference in results. The learning curves of different activation functions is shown in Fig. 3.

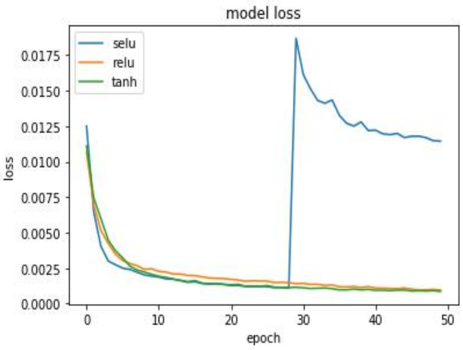

Fig. 3. Learning curves of different activation functions.

From the learning curves of these activation functions shown in Fig. 3, we can see that *tanh* and *relu* has better performance than *selu*. The performances of *tanh* and *relu* are almost similar. However, because of the instability issue we encountered with the *relu* function, we adopt *tanh* as the activation function in the convolution layers of our CRNN model. The rest parameter values used in CNN portion can be shown in Table 1.

Table 1. Adopted parameters value in CNN portion.

Parameter	Value
Number of Filters	61/128
Size of Kernel	3×3
Stride	1
Padding Method	Valid Padding
Pooling Size	(2,2)

A long short-term memory (LSTM) structure is used in the RNN portion of our model to solve the vanishing gradient problem [20]. The cause of this problem is that gradients used in learning the weighting parameters are obtained by multiplying the error gradients of each time step. Because the time step of RNN may be big, if the error gradients of each time step are mostly smaller than one, the overall multiplication results will approach zero. On the other hand, if the error gradients are mostly bigger than one, then the overall gradients may explode to infinity. And this vanishing gradients will prevent the convergence of training. Although some other structures such as Gate Recurrent Unit (GRU) can achieve similar performance in many tasks with a simpler structure and lower computational complexity [21]. Our task is an image series problem, LSTM can achieve better performance in image processing are. As a result, one LSTM layer is added into the RNN portion.

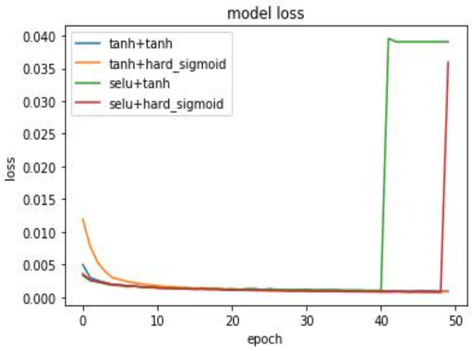

Fig. 4. Learning curves under different activation functions in LSTM.

Some other parameters also affect the difference of results, such as the number of LSTM neurons, the initialize method of the recurrent kernel, dropout value, implementation mode, activation function, etc. The main difference also comes from the different choices of activation functions. In LSTM, there are two activation functions, one for initializing the LSTM and the other for the recurrent steps in the LSTM. Considering that the *relu* function may lead to too big outputs in RNN [22] and *tanh* or hard *sigmoid* is an improvement over *sigmoid*, we try to choose the initializing activation function from *tanh* and *selu*, and to choose the recurrent activation function from *tanh* and *hard-sigmoid*. Therefore, we need to compare four combinations in total. The result is shown in Fig. 4.

From Fig. 4, we can see that the *selu* activation function will saturate much earlier than *tanh*, and the hard *sigmoid* has better performance in the recurrent step. Therefore, we choose *tanh* as the initial activation function and hard *sigmoid* function as the recurrent step activation function. The rest parameter used in RNN portion is shown in Table 2.

Table 2. Adopted parameters values in RNN portion.

Parameter	Value
Number of LSTM neurons	384
Initializer of recurrent kernel	orthogonal recurrent kernel
Dropout value	0.3/0.3
implementation mode	Products of a large number of small dots

During training, we need to choose appropriate loss function, optimizer, batch size and number of epochs. Because the training data and predict results are all images, mean squared error (MSE) can be a good choice of the loss function. The result of the different optimizer is shown in Fig. 5. And the result of the different number of batch size and epoch is shown in Fig. 6.

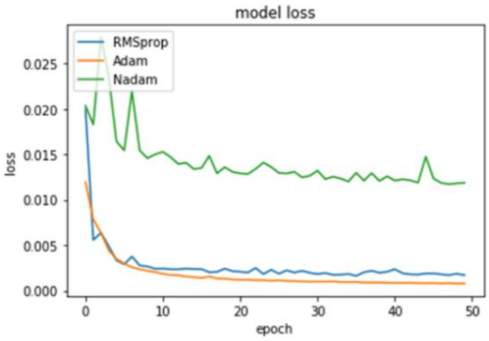

Fig. 5. Learning curves with different optimizers.

From Fig. 5, we can see that the loss of *Nadam* is not stable and not smooth. This means that *Nadam* may lead to learning too fast and finally lead to over-fitting. In contrast, the performance of *Adam* is stable and outstanding. Therefore, we use *Adam* as the optimizer in our CRNN training.

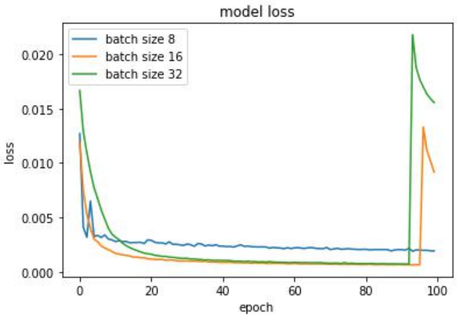

Fig. 6. The learning curve with different batch sizes.

From the experiment results shown in Fig. 6, we can find that higher batch sizes will lead to over-fitting earlier. However, when the batch size decreases to 8, the loss suffers from a higher error floor compared with batch size 16. This might be due to gradient vanishing. As a result, we set batch size as 16 and set the number of epochs to 95 because this gives the lowest loss in Fig. 6. All parameter optimization work has been displayed.

5 Result & Evaluation

The performance of our CRNN for pine tree image prediction is listed in Table 3. The result is evaluated according to four criterions: Mean Squared Error (MSE), Mean Average Error (MSE), Signal to Noise Ratio (SNR) and minimum validation loss. All results are calculated between the predicted image and real image. Some typical examples of the real pine tree images (which are generated by L-studio) and predicted pine tree images (which are the output of CRNN) are shown in Fig. 7. As can be seen, our CRNN model can successfully predict the pine tree images.

Table 3. Performance of CRNN.

Criterion	Value
MAE	8.57
MSE	377.35
SNR	322.55
Min Val-Loss	6.24e-04

From Fig. 7, we can see from the perspective of human eyes that the predicted image is very close to the original real image. From Table. 3, all four criterion values are very low, which means that our predictions are also successful from a data perspective.

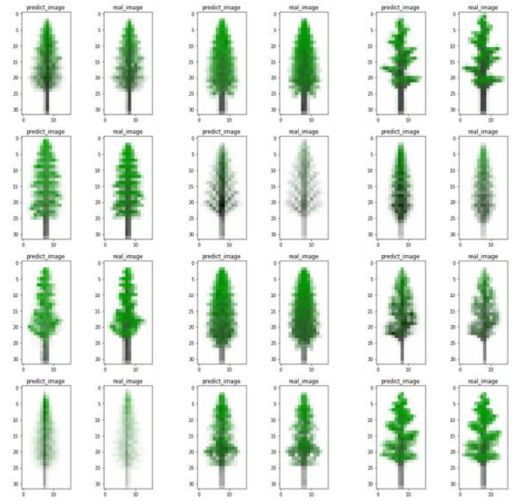

Fig. 7. Comparison between real pine tree images and predicted pine tree images.

6 Conclusion & Future Work

In this paper, we have developed a deep learning model that uses the convolutional recurrent neural network (CRNN) for image prediction. Specifically, we train the CRNN model with a pine tree image data set generated by L-studio and demonstrate that this model can successfully predict the future appearance of pine trees according to their past appearances. The predicted result of the developed CRNN is very good.

There are some issues that can be addressed to further improve performance. First, the data set is generated by the simulation software L-studio, which is still far away from real pine tree photos. Although L-studio has many adjustable parameters for us to generate many different images, it still cannot be fully compared with the diversity of natural images. As future work, we should try to collect enough images of natural pine trees or other objects and use them to test the prediction capability of the proposed CRNN model.

Second, we will try to extend the model developed in this paper to more application scenarios, such as medical image processing. Organs of the same disease usually have a similar development process. If the disease process of the diseased organ is regularly recorded as training data, we can train a suitable model to predict the development of the disease in other patients, to prevent it with some advance treatments. Of course, how to collect enough training data is still a core issue.

References

1. Y. Fu, G. Guo, and T. S. Huang. Age synthesis and estimation via faces: A survey. IEEE transactions on pattern analysis and machine intelligence, vol.32(11), pp. 1955 – 1976. IEEE (2010)
2. I. Kemelmacher-Shlizerman, S. Suwajanakorn, and S. M.Seitz. Illumination-aware age progression. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pp. 3334–3341. IEEE, Columbus, OH, USA (2014)
3. J. Suo, S.-C. Zhu, S. Shan, and X. Chen. A compositional and dynamic model for face aging. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol.32(3), pp. 385–401. (2010).
4. Y. Tazoe, H. Gohara, A. Maejima, and S. Morishima. Facial aging simulator considering geometry and patch-tiled texture. In: ACM Special Interest Group on Computer Graphics and Interactive Techniques Conference, pp. 90. Los Angeles, CA, USA (2012).
5. P. H. O. Pinheiro and R. Collobert, "Recurrent Convolutional Neural Networks for Scene Labeling," ICML (4), pp. 82-90, 2014.
6. M. Zihlmann, D. Perekrestenko, and M. Tschannen, " Convolutional Recurrent Neural Networks for Electrocardiogram Classification, " In: Computing in Cardiology(CinC), IEEE, Rennes, France (2017).
7. C. Doersch, " Tutorial on Variational Autoencoders, " Carnegie Mellon/ UC Berkeley, https://arxiv.org/abs/1606.05908, last accessed 2019/07/13
8. Y. Pu, Z. Gan, R. Henao, X. Yuan, C. Li, A. Stevens, L. Carin, "Variational Autoencoder for Deep Learning of Images, Labels and Captions, " In: 30th Conference on Neural Information Processing Systems 29, NIPS, Barcelona, Spain (2016)
9. A. Makhzani, J. Shlen, N. Jaitly, I. Goodfellow, B. Frey, "Adversarial Autoencoders," In: International Conference on Learning Representations (2016)
10. A. Radford, L. Metz, and S. Chintala. "Unsupervised representation learning with deep convolutional generative adversarial networks," In: International Conference on Learning Representations (2016).
11. Z. Zhang, Y. Song, H. Qi, " Age Progression/Regression by Conditional Adversarial Autoencoder," In: IEEE Conference on Computer Vision and Pattern Recognition, IEEE, Honolulu, HI, USA, (2017)

12. Goodfellow, I., Mirza, M., Courville, A., and Bengio, Y. (2013a). “Multi-prediction deep Boltzmann machines,” In: Advances in Neural Information Processing Systems, pages 548–556. NIPS (2013)
13. M. Mirza, S. Osindero, “Conditional Generative Adversarial Nets,” https://arxiv.org/, last accessed 2019/06/24
14. F. A. Gers and J. Schmidhuber, “LSTM Recurrent Networks Learn Simple Context Free and Context Sensitive Languages,” In: IEEE Transactions on Neural Networks, vol.12(6), IEEE, Switzerland 2001.
15. Z. Zuo, B. Shuai, G. Wang, X. Liu, X. Wang, B. Wang, Y. Chen, “ Convolutional Recurrent Neural Networks: Learning Spatial Dependencies for Image Representation” In: IEEE Conference on Computer Vision and Pattern Recognition Workshops, pp. 18-26. IEEE, Boston, MA, USA (2015)
16. J. Zheng, Y. Wang, X. Zhang, X. Li, “ Classification of Severely Occluded Image Sequences via Convolutional Recurrent Neural Networks,” In: IEEE Global Conference on Signal and Information Processing, Anaheim, CA, USA (2018)
17. L. Chen, S. Li, “Improvement Research and Application of Text Recognition Algorithm Based on CRNN,” In: 2018 International Conference on Signal Processing and Machine Learning, pp. 166-170. IEEE, New York, NY, USA, (2018)
18. K. Choi, G. Fazekas, M. Sandler, K. Cho, “Convolutional Recurrent Neural Networks for Music Classification,” In: IEEE International Conference on Acoustics, Speech and Signal Processing, IEEE, New Orleans, LA, USA, (2017)
19. “L-studio 4.0 User’s Guide,” http://algorithmicbotany.org/lstudio/
20. X. Glorot and Y. Bengio, “Understanding the difficulty of training deep forward neural networks, ” In: Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics, vol.9, pp.249-256.(2010)
21. K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, Y. Bengio, “ Learning Phrase Representations using RNN Encoder-Decoder for Statistical Machine Translation”, In: Proceeding of the 2014 Conference on Empirical Methods in Natural Language Processing, pp.1724-1734. (2014)
22. Y. Bengio, P. Simard, and P. Frasconi, “Learning long-term dependencies with gradient descent is difficult,” In: IEEE Transactions on Neural Networks, vol.5(2), pp.157-166. IEEE (2014)