Hindawi
Journal of Robotics
Volume 2017, Article ID 2061827, 7 pages
https://doi.org/10.1155/2017/2061827

Research Article

Long Short-Term Memory Projection Recurrent Neural Network Architectures for Piano's Continuous Note Recognition

YuKang Jia,[1] Zhicheng Wu,[1] Yanyan Xu,[1] Dengfeng Ke,[2] and Kaile Su[3]

[1]*School of Information Science and Technology, Beijing Forestry University, No. 35 Qinghuadong Road, Haidian District, Beijing 100083, China*
[2]*National Laboratory of Pattern Recognition, Institute of Automation, Chinese Academy of Sciences, No. 95 Zhongguancundong Road, Haidian District, Beijing 100190, China*
[3]*College of Information Science and Technology, Jinan University, No. 601, West Huangpu Avenue, Guangzhou, Guangdong 510632, China*

Correspondence should be addressed to Yanyan Xu; xuyanyan@bjfu.edu.cn

Received 10 May 2017; Revised 30 July 2017; Accepted 6 August 2017; Published 12 September 2017

Academic Editor: Keigo Watanabe

Long Short-Term Memory (LSTM) is a kind of Recurrent Neural Networks (RNN) relating to time series, which has achieved good performance in speech recogniton and image recognition. Long Short-Term Memory Projection (LSTMP) is a variant of LSTM to further optimize speed and performance of LSTM by adding a projection layer. As LSTM and LSTMP have performed well in pattern recognition, in this paper, we combine them with Connectionist Temporal Classification (CTC) to study piano's continuous note recognition for robotics. Based on the Beijing Forestry University music library, we conduct experiments to show recognition rates and numbers of iterations of LSTM with a single layer, LSTMP with a single layer, and Deep LSTM (DLSTM, LSTM with multilayers). As a result, the single layer LSTMP proves performing much better than the single layer LSTM in both time and the recognition rate; that is, LSTMP has fewer parameters and therefore reduces the training time, and, moreover, benefiting from the projection layer, LSTMP has better performance, too. The best recognition rate of LSTMP is 99.8%. As for DLSTM, the recognition rate can reach 100% because of the effectiveness of the deep structure, but compared with the single layer LSTMP, DLSTM needs more training time.

1. Introduction

Piano's continuous note recognition is important for a robot, whether it is a bionic robot, a dance robot, or a music robot. There have been companies researching on music robots. For example, Vadi produced by Vatti is able to identify a voiceprint.

Most of the existing piano's note recognition techniques use Hidden Markov Model (HMM) and Radial Basis Function (RBF) to recognize musical notes with one musical note at a time and therefore are not suitable for continuous note recognition. Fortunately, in the field of pattern recognition, Deep Neural Networks (DNNs) have shown great advantages. DNNs are used to recognize features extracted from a large number of hidden nodes [1] and seek reverse partial guidance through the chain rule and at the same time make the neural network weight matrix convergence through training data iteratively and then achieve recognition [2]. RNN adds a time series based on DNN [3], which makes features have time continuity [4, 5]. However, in experiments, we find that RNN's time characteristics will disappear completely after four iterations [6], and a music note is generally longer than a frame [7], so RNN is not suitable for piano's continuous note recognition [8]. Fortunately, a variant of RNN, named LSTM, is proposed [9–12], in which an input gate, an output gate, and a forgotten gate are added to memorize a long-term cell state to maintain long-term memory [8, 9, 13–16]. Furthermore, LSTMP adds a projection layer to LSTM to increase its efficiency and effectiveness.

This paper studies LSTM and LSTMP for piano's continuous note recognition, and in order to solve the temporal classification problem, we combine LSTM and LSTMP with

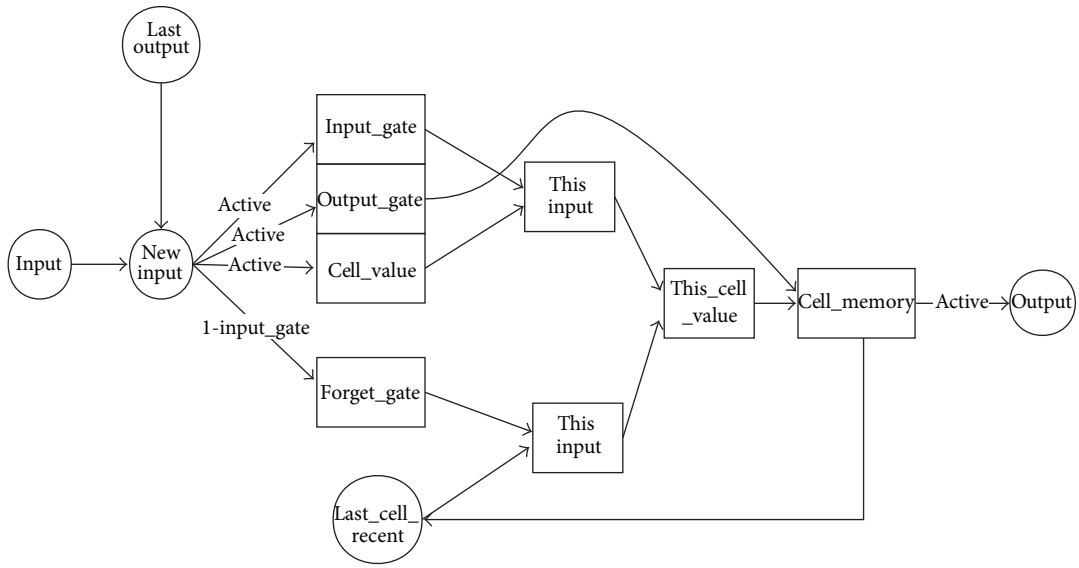

Figure 1: The LSTM network structure.

a method named CTC [17]. In experiments, we test the performance of a single layer LSTM, Deep LSTM, and a single layer LSTMP with different parameters. Compared with the traditional piano's note recognition methods, LSTM and LSTMP can recognize continuous notes, that is, some simple piano music. The experimental results show that a single layer LSTMP can attain a recognition rate of 99.8% and Deep LSTM can reach 100%, which proves that our methods are quite effective.

The rest of this paper is organised as follows. In Section 2, we first introduce the LSTM network architecture, and then Deep LSTM. LSTMP is illustrated in Section 3. In Section 4, we discuss CTC. The experimental results are presented in Section 5, and finally, in Section 6, we draw conclusions and give our future work.

2. LSTM

2.1. The LSTM Network Architecture. LSTM is a kind of RNN which succeeds to keep memory for a period of time by adding a "memory cell." The memory cell is mainly controlled by "the input gate," "the forgetting gate," and "the output gate." The input gate activates the input of information to the memory cell, and the forgetting gate selectively obliterates some information in the memory cell and activates the storage to the next input [18]. Finally, the output gate decides what information will be outputted by the memory cell [19].

The LSTM network structure is illustrated in Figure 1. Each box represents different data, and the lines with arrows mean data flow among these data. From Figure 1, we can understand how LSTM stores memory for a long period of time.

The recognition procedure of LSTM begins with a set of input sequences $x = (x_1, x_2, \ldots, x_t)$ (x_i is a vector) and finally outputs a set of $y = (y_1, y_2, \ldots, y_t)$ (y_i is also a vector), which is calculated according to the following equations:

$$i_t = \sigma\left(W_{ix}x_t + W_{im}m_{t-1} + b_i\right) \quad (1)$$

$$o_t = \sigma\left(W_{ox}x_t + W_{om}m_{t-1} + b_o\right) \quad (2)$$

$$f_t = 1 - i_t \quad (3)$$

$$tc_t = g\left(W_{cx}x_t + W_{cm}m_{t-1} + b_c\right) \quad (4)$$

$$c_t = f_t \odot c_{t-1} + i_t \odot tc_t \quad (5)$$

$$m_t = o_t \odot hc_t \quad (6)$$

$$y_t = \phi\left(W_{ym}m_t + b_y\right). \quad (7)$$

In these equations, i means the input gate, and o and f are the output gate and the forget gate, respectively. tc is the information input to the memory cell, and c includes cell activation vectors, and m is the information the memory cell outputs. W represents weight matrices (e.g., W_{ix} represents the weight matrix from input x to the input gate i). b is the bias (b_i is the input gate bias vector), and g and h are the activation function of cell inputting and cell outputting, respectively, regarded as *tanh* and *linear* in most of the models and also in this paper. $\odot$ is the point multiplication in a matrix. ϕ is the activation function of the neural network output, and we use $softmax$ in this paper.

After conducting some experiments, we find that, compared with the f_t standard equation, (3) is more simple and easier to converge. Not only does the training time become less but also the number of iterations becomes smaller. Therefore, in the neural networks in this paper, we use (3) to calculate f_t instead of the f_t standard equation.

2.2. Deep LSTM. In piano's continuous note recognition, we also build a multilayer neural network to further increase the

recognition rate. Deep LSTM adds an LSTM after another and so on [10]. The added LSTMs have the same structure as the original one. Each layer regards the output from the last layer as the input of the next layer. We hope that the neural networks in different LSTM layers will learn different characteristics, so as to learn the various features of musical notes from different aspects and therefore improve the recognition rate.

3. LSTMP-LSTM with a Projection Layer

In LSTM, there are a large number of calculations in the various gates, calculating the number of parameters N in the neural network. The weight matrix dimension input by the input gate, the output gate, and the cell state at this time is $ni * nc$, and the weight matrix dimension at the last time is $nc * nc$, and the output matrix dimension connected to the output of the neural network is $no * nc$, where ni and no are the dimensions of the input and the output, respectively, and nc is the number of memory cells. We can easily get the following formula:

$$N_{\text{LSTM}} = 3 * ni * nc + 3 * nc * nc + no * nc; \tag{8}$$

that is,

$$N_{\text{LSTM}} = 3 * ni * nc + nc * (3 * nc + no). \tag{9}$$

As we increase nc, N_{LSTM} grows in a square pattern. Therefore, increasing the number of memory cells to increase the amount of memory costs a lot, but a smaller cell number will bring a lower recognition rate, so we propose an architecture named LSTMP, which can not only improve the accuracy, but also effectively reduce the computations.

In the output layer of the neural network, LSTM outputs a matrix of $nc^2 * m_t$. Then, m_t is sent into the output matrix to be outputted and also serves as the input to the neural network at the next time. We add a *projection* layer to the LSTM architecture, and after passing this layer, m_t becomes an $nc * nr$ matrix called r_t, which replaces m_t as the input of the next neural network. When the memory cell number of the neural network increases, the number of parameters in the neural network is

$$N_{\text{LSTMP}} = 3 * ni * nc + 3 * nc * nr + no * nr + nc * nr; \tag{10}$$

that is,

$$N_{\text{LSTMP}} = 3 * ni * nc + nr * (4 * nc + no). \tag{11}$$

Calculating $N_{\text{LSTM}} - N_{\text{LSTMP}}$, we have

$$N_{\text{LSTM}} - N_{\text{LSTMP}} = nc * (3 * nc + no) - nr * (4 * nc + no). \tag{12}$$

Therefore, in LSTMP, the factor that affects the total number of parameters changes from $nc * nc$ to $nc * nr$. We can change the value of nc/nr to reduce the computational complexity. When $3 * nc > 4 * nr$, LSTMP can speed up the training model. Moreover, with the projection layer, LSTMP can converge faster to ensure the convergence of the model. The mathematical formulae of LSTMP are as follows:

$$\begin{aligned} i_t &= \sigma\left(W_{ix}x_t + W_{ir}r_{t-1} + b_i\right) \\ o_t &= \sigma\left(W_{ox}x_t + W_{or}r_{t-1} + b_o\right) \\ f_t &= 1 - i_t \\ tc_t &= g\left(W_{cx}x_t + W_{cr}r_{t-1} + b_c\right) \\ c_t &= f_t \odot c_{t-1} + i_t \odot tc_t \\ m_t &= o_t \odot hc_t \\ r_t &= W_{rm}m_t \\ y_t &= \phi\left(W_{yr}r_t + b_y\right). \end{aligned} \tag{13}$$

In these formulae, r_t represents the *projection* layer, and the other equations are the same as LSTM.

Figure 2 is the structure of LSTMP, and the part marked with red dashed lines is the projection. By comparing Figure 1 with Figure 2, we can see that LSTMP is LSTM with a projection layer.

Algorithm 1 is the pseudocode of LSTMP. w is the input weight matrix, and u is the weight matrix of the last result. b is bias and r is the projection matrix. We put the extracted musical notes features into the neural network and the algorithm executes until we get an acceptable recognition rate.

4. CTC

The output layer of our LSTM and LSTMP is called CTC [20]. We use CTC because it does not need presegmented training data or external postprocessing to extract label sequences from the network outputs.

To be the same as many latest neural networks, CTC has forward and backward algorithms. When it comes to the forward algorithm, the key point is to estimate the distribution through probabilities. Given the length T, the input sequence x, and the training set S at time t, the activation ${y^t}_k$ of the output unit k at time t is interpreted as the probability of observing label k ($or\ blank\ if\ k = |L| + 1$):

$$p(\pi \mid x, S) = \prod_{t=1}^{T} y_{\pi t}^{t}. \tag{14}$$

We refer to the elements $\pi \in {L'}^T$ as paths, where ${L'}^T$ is the set of the length T sequences over the alphabet $L' = L \cup blank$. Then we define a many-to-one map ϕ to remove first the repeated labels and then the blanks from the paths. With glance at the paths, will find they are mutually exclusive. According to the characteristic, the conditional probability of some labelling $l \in L^{\leqslant T}$ can be calculated by summing the probabilities of all the paths mapped onto it by ϕ:

$$p(l \mid x) = \sum_{\pi \in \phi^{-1}(l)} p(\pi \mid x). \tag{15}$$

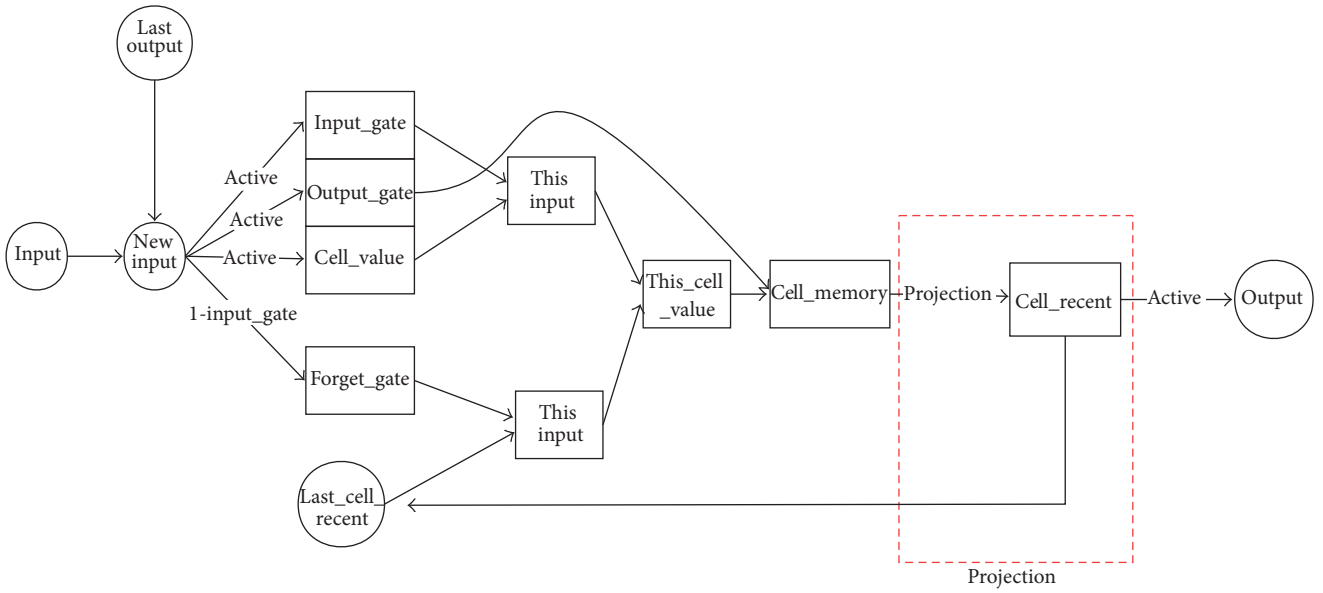

Figure 2: The LSTMP structure.

Require: $input\ \ Result_{last}\ \ Cell_{last}\ \ w\ u\ b\ r$
Ensure: $Cell_{value}\ \ Result$
  while *When having input* **do**
    $temp_{input} = (Result_{last} * u_{input}) + (input * w_{input}) + b$
    $temp_{output} = (Result_{last} * u_{output}) + (input * w_{output}) + b$
    $temp_{cell} = (Result_{last} * u_{cell}) + (input * w_{cell}) + b$
    $input_{gate} = sigmoid(temp_{input})$
    $output_{gate} = sigmoid(temp_{output})$
    $forget_{gate} = 1 - input_{gate}$
    $Cell_{value} = activate(temp_{cell}) * input_{gate} + Cell_{last} * forget_{gate}$
    $Result = output_{gate} * activate(cell_{value})$
    $Cell_{value} = (Result, r)$
  end while

Algorithm 1: LSTM-projection.

After all these procedures, CTC will complete its classification task.

5. Experiments

We conduct all our experiments on a server with 4 Intel Xeon E5-2620 CPUs and 512 GB memories. A NVIDIA Tesla M2070-Q graphics card is used to train all the models. The programming language we use is python 3.5.

We choose the piano as our instrument. We record 445 note sequences as our dataset and the length of each sequence is around 8 seconds.

In the extraction of features, we carry out Hamming window processing and then take Fast Fourier Transform (FFT) for the real part and the imaginary part of each window. Then we let the FFT result to be orthogonal by adding the square of the real part and that of the imaginary part together. Apart from that, we gain the log of the quadratic sum. Finally, the normalization of the input data is performed.

In the experiments, the number of kinds of notes is 8, and the number of input nodes is 9. We try different numbers of cell units in our models, from 20 to 320. The initial value of the neural network is set as a random value within $[-0.2, 0.2]$, and the learning rate is 0.001. In terms of the structures, all the neural networks are connected to a single layer CTC. As for the dataset, we choose 80% of the samples as the development set and 20% as the test set.

5.1. Experimental Results. Table 1 shows the recognition rates and how many times LSTM, DLSTM, and LSTMP with different parameters need to iterate until their recognition rates are stable, and the best results are in bold.

In Table 1, "LSTMP-80 to 20" means the LSTMP model projecting 80 cell states to 20 cell states. From Table 1, we see

Table 1: The number of iterations and the recognition rates of different models.

Model	Number of iterations	Recognition rate
LSTM-20	300	77.2%
LSTM-40	300	87.1%
LSTM-80	300	**90.7**%
LSTM-160	400	82.5%
DLSTM-two layers	61	85.7%
DLSTM-three layers	71	99.5%
DLSTM-four layers	109	**100.0**%
DLSTM-five layers	76	97.5%
DLSTM-six layers	103	**100.0**%
LSTMP-10 to 20	43	56%
LSTMP-30 to 20	35	94.2%
LSTMP-40 to 20	45	96.0%
LSTMP-40 to 30	30	94.0%
LSTMP-60 to 30	**22**	98.0%
LSTMP-80 to 20	58	**99.8**%
LSTMP-80 to 30	27	97.3%
LSTMP-80 to 40	29	97.3%
LSTMP-160 to 40	39	99.0%
LSTMP-160 to 80	50	95.0%
LSTMP-320 to 160	93	93.4%

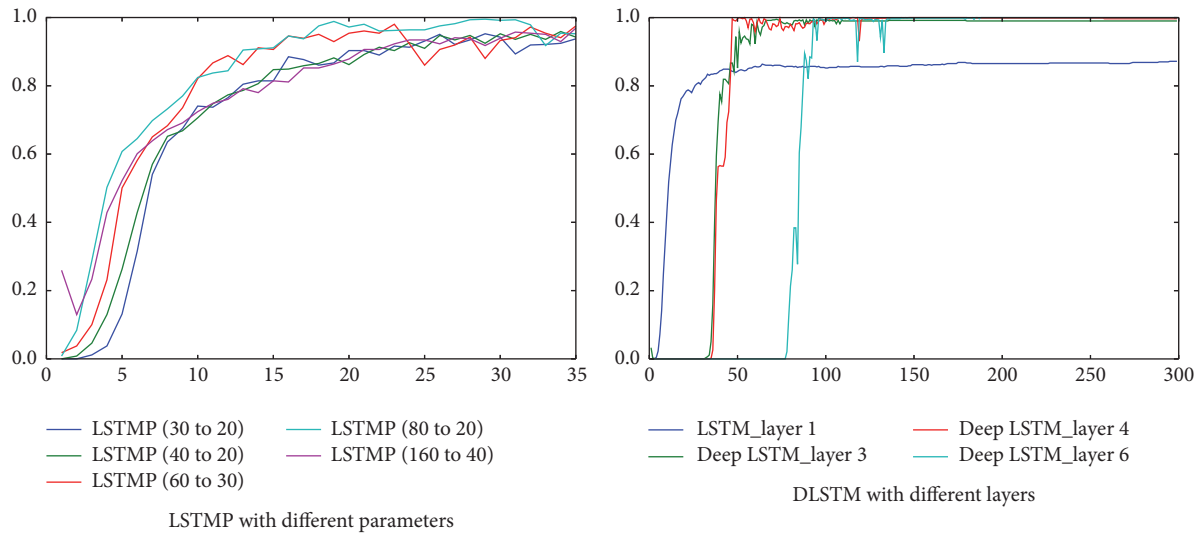

Figure 3: The recognition rates of LSTMP with different parameters and DLSTM with different layers.

that DLSTM and LSTMP perform much better than LSTM, and their best recognition rates are almost the same, which are 100% and 99.8%, respectively. As for the numbers of iterations, LSTMP needs much less iterations than LSTM and DLSTM, which makes LSTMP more suitable for piano's continuous note recognition for robotics considering the efficiency.

5.2. LSTMP and DLSTM with Different Parameters. Figure 3 illustrates LSTMP with different parameters and DLSTM with different layers. The x axis means the number of iterations and the y axis means the recognition rate. We see that for LSTMP the model projecting 80 cell states to 20 cell states has the best result, but all LSTMP results are very close. As for DLSTM, we see clearly that Deep LSTM is much better than LSTM with only one layer.

5.3. Comparisons of LSTM, LSTMP, and DLSTM. We compare LSTM, LSTMP, and DLSTM in Figure 4. Given the same parameters, LSTMP performs much better than LSTM. As for LSTMP and DLSTM, we find that when the number of

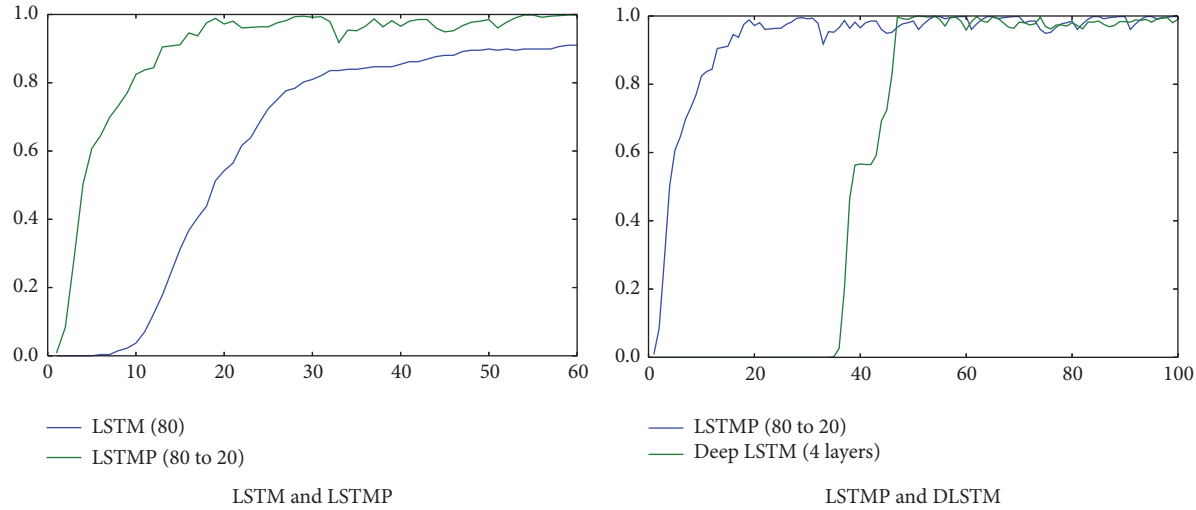

Figure 4: Comparisons of LSTM, LSTMP, and DLSTM.

iterations is small, LSTMP has great advantages, but as the number of iterations increases, DLSTM becomes better.

6. Conclusions and Future Work

In this paper, we have used neural network structures called LSTM with CTC to recognize continuous musical notes. On the basis of LSTM, we have also tried LSTMP and DLSTM. Among them, LSTMP worked best when projecting 80 cell states to 20 cell states, which needed much less iterations than LSTM and DLSTM, making it most suitable for piano's continuous note recognition.

In the future, we will use LSTM, LSTMP, and DLSTM to recognize more complex continuous chord music, such as piano music, violin pieces, or even symphony, which will greatly improve the development of music robots.

Conflicts of Interest

The authors declare that they have no conflicts of interest.

Acknowledgments

Thanks are due to Yanlin Yang for collecting the recording materials. This work is supported by the Fundamental Research Funds for the Central Universities (no. 2016JX06) and the National Natural Science Foundation of China (no. 61472369).

References

[1] A. K. Jain, J. Mao, and K. M. Mohiuddin, "Artificial neural networks: a tutorial," *IEEE Computational Science & Engineering*, vol. 29, no. 3, pp. 31–44, 1996.

[2] J. R. Zhang, T. M. Lok, and M. R. Lyu, "A hybrid particle swarm optimization-back-propagation algorithm for feedforward neural network training," *Applied Mathematics and Computation*, vol. 185, no. 2, pp. 1026–1037, 2007.

[3] L. Liang, Y. Tang, L. Bin, and W. Xiaohua, Artificial neural network based universal time series, May 11 2004. US Patent 6,735,580.

[4] R. J. Williams and D. Zipser, "A learning algorithm for continually running fully recurrent neural networks," *Neural Computation*, vol. 1, no. 2, pp. 270–280, 1989.

[5] L. R. Medsker and L. C. Jain, "Recurrent neural networks," *Design and Applications*, vol. 5, 2001.

[6] G. Shalini, O. Vinyals, B. Strope, R. Scott, T. Dean, and L. Heck, "Contextual lstm (clstm) models for large scale nlp tasks," https://arxiv.org/abs/1602.06291.

[7] A. Karpathy, The unreasonable effectiveness of recurrent neural networks. http://karpathy.github.io/2015/05/21/rnn-effectiveness/, 2015.

[8] D. Turnbull and C. Elkan, "Fast recognition of musical genres using RBF networks," *IEEE Transactions on Knowledge and Data Engineering*, vol. 17, no. 4, pp. 580–584, 2005.

[9] H. Sak, A. Senior, and F. Beaufays, "Long short-term memory recurrent neural network architectures for large scale acoustic modeling," in *Proceedings of the 15th Annual Conference of the International Speech Communication Association: Celebrating the Diversity of Spoken Languages, INTERSPEECH 2014*, pp. 338–342, September 2014.

[10] A. Graves, A.-R. Mohamed, and G. Hinton, "Speech recognition with deep recurrent neural networks," in *Proceedings of the 38th IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '13)*, pp. 6645–6649, May 2013.

[11] J. Schmidhuber, "Deep learning in neural networks: an overview," *Neural Networks*, vol. 61, pp. 85–117, 2015.

[12] K. M. Hermann, T. Kočiský, E. Grefenstette et al., "Teaching machines to read and comprehend," in *Proceedings of the 29th Annual Conference on Neural Information Processing Systems, NIPS 2015*, pp. 1693–1701, December 2015.

[13] B. Bakker, "Reinforcement learning with long short-term memory," in *In Advances in Neural Information Processing Systems*, pp. 1475–1482.

[14] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[15] P. Goelet, V. F. Castellucci, S. Schacher, and E. R. Kandel, "The long and the short of long-term memory - A molecular framework," *Nature*, vol. 322, no. 6078, pp. 419–422, 1986.

[16] Q. Lyu, Z. Wu, and J. Zhu, "Polyphonic music modelling with LSTM-RTRBM," in *Proceedings of the 23rd ACM International Conference on Multimedia, MM 2015*, pp. 991–994, aus, October 2015.

[17] S. Haim, A. Senior, R. Kanishka, and F. Beaufays, "Fast and accurate recurrent neural network acoustic models for speech recognition," https://arxiv.org/abs/1507.06947.

[18] F. A. Gers, J. Schmidhuber, and F. Cummins, "Learning to forget: Continual prediction with LSTM," *Neural Computation*, vol. 12, no. 10, pp. 2451–2471, 2000.

[19] F. A. Gers, N. N. Schraudolph, and J. Schmidhuber, "Learning precise timing with LSTM recurrent networks," *Journal of Machine Learning Research (JMLR)*, vol. 3, no. 1, pp. 115–143, 2003.

[20] A. Graves, S. Fernández, F. Gomez, and J. Schmidhuber, "Connectionist temporal classification: Labelling unsegmented sequence data with recurrent neural networks," in *Proceedings of the 23rd International Conference on Machine Learning, ICML 2006*, pp. 369–376, Pittsburgh, Pa, USA, June 2006.