ResearchGate

See discussions, stats, and author profiles for this publication at: https://www.researchgate.net/publication/312325842

Move prediction in Gomoku using deep learning

Conference Paper · November 2016

DOI: 10.1109/YAC.2016.7804906

CITATIONS
17

READS
11,926

4 authors:

Kun Shao
Chinese Academy of Sciences
13 PUBLICATIONS **552** CITATIONS
SEE PROFILE

Dongbin Zhao
Chinese Academy of Sciences
401 PUBLICATIONS **10,093** CITATIONS
SEE PROFILE

Zhentao Tang
Chinese Academy of Sciences
16 PUBLICATIONS **179** CITATIONS
SEE PROFILE

Yuanheng Zhu
Chinese Academy of Sciences
69 PUBLICATIONS **2,078** CITATIONS
SEE PROFILE

All content following this page was uploaded by Zhentao Tang on 16 January 2018.

The user has requested enhancement of the downloaded file.

31st Youth Academic Annual Conference of Chinese Association of Automation
Wuhan, China; November 11-13, 2016

Move Prediction in Gomoku Using Deep Learning

Kun Shao, Dongbin Zhao, Zhentao Tang, and Yuanheng Zhu
The State Key Laboratory of Management and Control for Complex Systems, Institute of Automation,
Chinese Academy of Sciences
Beijing 100190, China
Emails: shaokun2014@ia.ac.cn; dongbin.zhao@ia.ac.cn; tangzhentao2016@ia.ac.cn; yuanheng.zhu@ia.ac.cn

***Abstract*—The Gomoku board game is a longstanding challenge for artificial intelligence research. With the development of deep learning, move prediction can help to promote the intelligence of board game agents as proven in AlphaGo. Following this idea, we train deep convolutional neural networks by supervised learning to predict the moves made by expert Gomoku players from RenjuNet dataset. We put forward a number of deep neural networks with different architectures and different hyper-parameters to solve this problem. With only the board state as the input, the proposed deep convolutional neural networks are able to recognize some special features of Gomoku and select the most likely next move. The final neural network achieves the accuracy of move prediction of about 42% on the RenjuNet dataset, which reaches the level of expert Gomoku players. In addition, it is promising to generate strong Gomoku agents of human-level with the move prediction as a guide.**

***Keywords*—Gomoku; move prediction; deep learning; deep convolutional network**

I. Introduction

A. Gomoku

Gomoku[1], also called five in a row, is an abstract strategy board game originated from ancient China. Generally speaking, Gomoku is played with black and white stones on a board with 15×15 intersections as shown in Fig. 1. In its simplest form, black plays first, and players place a stone of their own color on an empty intersection alternately. To win the game, you have to get an row of five stones horizontally, vertically, or diagonally.

As a suitable test-bed for artificial intelligence research, a large number of approaches have been proposed for Gomoku. Among them, the most popular method is the game tree search. This method searches the game tree from a root node of the current board state. Once there exits a leaf board state with five-in-a-row, it returns the move sequence. However, a complete search requires evaluation of $p!/(p-n)!$ board states, where p is the number of legal moves under current board state and n is the depth [1]. With exhaustive search being infeasible, Schaeffer raises the history heuristic and alpha-beta search to speed up game tree searching [2].

On the other hand, researchers try to use neural networks to create high-lever Gomoku agent. Freisleben trains a network to play Gomoku against an opponent using reinforcement learning and evaluates the board states by rewarding good moves and penalizing bad moves [3]. Zhao applies adaptive dynamic programming with a three layers fully-connected neural network to learn to play Gomoku [4]. This method is able to improve Gomoku agent's performance by playing against itself. With the input to the neural network consisted of 40 selected patterns, the performance of such a agent heavily relies on the quality of the feature selection and available expert knowledge.

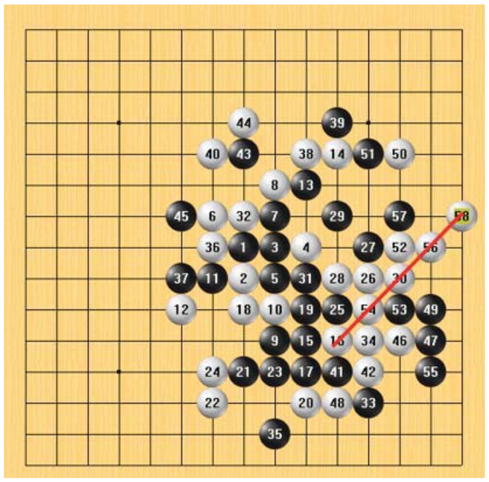

Fig. 1. Example of positions from a game of Gomoku after 58 moves have passed. It can be seen that the white win the game at last.

B. Deep Learning

Deep learning is a new branch of machine learning with multiple neural network layers to model high-level abstractions in massive data [5]. As a form of representation learning methods, deep learning has multiple levels of representation. By composing a large number of simple but non-linear modules, deep network can transform the representation with raw input into a higher representation, and learn very complex functions.

Deep learning is making major advances in solving artificial intelligence problems. It has turned out to be very good at discovering complicate features in high dimensional data and help to solve the hand-craft feature extraction problem [6]. In recent years, various deep learning architectures such as deep full connected neural networks, deep convolutional neural networks, deep belief networks and recurrent neural networks

Dongbin Zhao is also with the University of Chinese Academy of Sciences, Beijing, China.
This work is supported by National Natural Science Foundation of China (NSFC) under Grants No. 61273136, No.61573353 and No. 61533017.
[1]https://en.wikipedia.org/wiki/Gomoku

978-1-5090-4423-8/16/$31.00 ©2016 IEEE

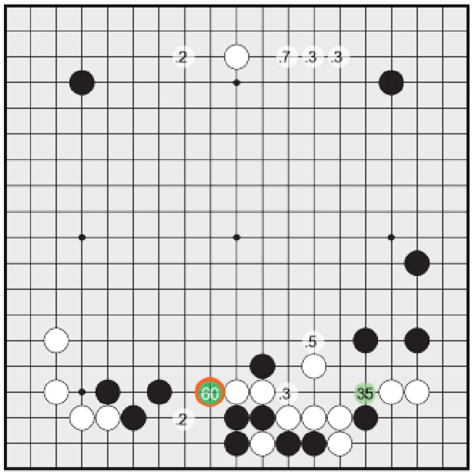

Fig. 2. Example of SL policy network of AlphaGo [10]. Move probabilities directly come from the network and are represented as a percentage.

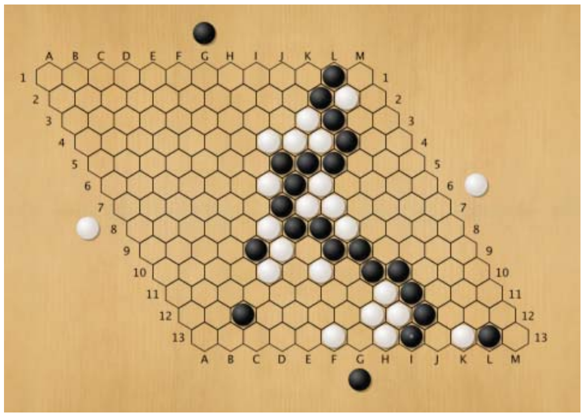
Fig. 3. Example of a Hex game [15]. Black wants to join top and bottom, white wants to join left and right, and black wins the game at last.

have been applied to fields like computer vision [7], speech recognition [8] and natural language processing [9].

With AlphaGo making a hit, deep learning becomes a promising approach for board game artificial intelligence [10]. As the first stage of AlphaGo, a policy network is trained to predict the moves of expert players using supervised learning, which reduces the width of search to a large extend. Inspired by this, we try move prediction using deep learning to improve Gomoku agent intelligence in this paper.

The organization of the remaining paper is arranged as follows. In Section II we describe the move prediction method used in this paper. Section III presents the deep convolutional neural network approach for Gomoku move prediction in detail. And then Section IV presents the experiment results of different network architectures and hyper-parameters, as well as a brief discussion. In the end, we draw a conclusion of the paper.

II. Move Prediction

Generally speaking, human experts heavily rely on pattern recognition to select moves when playing games. Expert players can gain accurate judgement about current board state at a glance, without estimating possible future positions in mind. This mechanism is different from classical computer algorithms, which simulate thousands of possible future moves. Hence, move prediction for games using pattern recognition might be the key component to improve the performance of game agent. Until now, move prediction has successfully applied in board games like Go and Hex.

A. Move Prediction in Go

Go[2] is an ancient board game for two players. Apart from a larger board than Gomoku, Go's playing rule is also much more complicated. Owing to the enormous search space and difficulty of evaluating board positions and moves, Go has long been viewed as the most challenging classic games for artificial intelligence.

In recent years, move prediction with deep learning is gradually applied to play Go. In 2008, Sutskever applied a two-layer convolutional network to predict moves of Go [11]. Using the current board state and previous moves as input, it achieved an accuracy of 34%. In 2014, Clark adopted the similar approach using a 8-layer convolutional neural network to achieve 44% move prediction accuracy [12]. In 2015, Maddision trained a 12-layer convolutional neural network by supervised learning and the network correctly predicts 55% of the expert moves, equalling a 6 dan human player [13]. Also in 2015, Tian achieved almost the same prediction result with his Darkforest Go program [14].

In the end of 2015, Google successfully solved the Go problem and published a paper in the Nature [10]. Later in March 2016, AlphaGo defeated Lee Sedol by 4:1 and helped the computer Go achieve a top human lever at the first time. In AlphaGo, the policy network consists of 13 convolutional layers and rectifier nonlinearities, with a softmax layer outputting a probability distribution over all legal moves, as shown in Fig. 2. The network is trained with 30 million positions from the KGS Go Server to maximize the likelihood of the human moves selected in a certain state. The test accuracy is 57.0% using all 48 input features, and 55.7% using only raw board state and move history as inputs, both matching the state-of-the-art prediction performance. With the help of policy network's move prediction, AlphaGo improves its search efficiency and achieves the top Go expert lever.

B. Move Prediction in Hex

Hex[3] is a classic connection board game played by two players on an 13×13 rhombus of hexagonal cells. Each player is assigned two opposite sides of the board and a stone color. The winner is the first player that joins their two sides with a contiguous chain of their stones, as shown in Fig. 3. Although

[2]https://en.wikipedia.org/wiki/Go_(game)

[3]https://en.wikipedia.org/wiki/Hex_(board_game)

the rule is simple, Hex has deep tactics and complicated strategies, serving as a test bed for artificial intelligent algorithms. Previous approach for computer Hex focuses on search tree method, but now move prediction with deep learning is also adopted. Young trains a network by supervised learning to predict the moves produced over a database of positions [15]. The network consists of 10 convolutional layers and one fully connected output layer. The move prediction works as an initialization stage of a deep Q network [16] and helps the neuroHex agent achieve a high lever.

C. Move Prediction with Deep Learning

Move prediction has deep relationship with deep learning. The target function of board games used to be highly complex, since human experts usually think in a complicated, non-linear way. Besides, the target function is always non-smooth because a minor change to a position could dramatically alter the most likely next move. These properties make this learning task very challenging and motivate us to attempt a deep learning approach, since deep learning is well suited to learn complex, non-smooth functions [5]. Move prediction also provides an opportunity to test the abilities of deep learning on the artificial intelligence of games.

III. Deep Convolutional Neural Network

In this paper, we train a deep convolutional neural network to predict the next move given the current board state as input. In this section we describe the data preprocessing skills, different network architectures and the details of the training procedure.

A. Experimental Platform

In the following experiments, we choose Tensorflow and Keras as the experimental platform to do the move prediction research. As the most popular deep learning library at present, TensorFlow[4] has deeply promoted the development of deep learning. It is an open source software originally for expressing machine learning algorithms, especially deep learning [17]. Since the release, more and more people use TensorFlow for the purposes of conducting machine learning and deep neural networks research. Besides, in order to conduct deep learning experiments in an efficient way, many researches come to use TensorFlow combined with Keras. Keras[5] is a highly modular neural networks library, capable of running on top of TensorFlow. It is developed with the focus on enabling fast experimentation with the least possible delay from idea to result in deep learning.

B. Data and Preprocessing

The dataset for training comes from RenjuNet[6]. RenjuNet is an online database for Gomoku games, providing the download service of all the game records in the library for playing and research. All the game records and their related information are stored in XML format, with a renjunet_v10_yyyymmdd.rif file.

In this paper, we use the renjunet_v10_20160517.rif dataset. The dataset contains as many as 66000 Gomoku games. For each game, there are a different number of moves from a couple dozens to more than one hundred. Clearly, it will produce repeated data when extracting (state, action) pairs from different games. To remove duplicate datas in an efficient way, we iterate through each board image with different hash value, which is based on their image gray value. In this way, we can compare the hash values instead of comparing images. Thus, every image only need to be accessed once, resulting in an considerable time saving. After that, we get 1247582 total (state, action) pairs. Each board state of the dataset is treated as an image with the label being the next move under that specific state. That is, each board state s is preprocessed into a set of 15×15 feature planes. The feature value is split into multiple planes of binary values with one-hot encoding, representing whether an intersection is black, white or empty. Ground truth labels are stored in a separate array matching the indices for its related state. The labels are also processed with one-hot encoding. In this way, the Gomoku move prediction problem becomes an image classification problem. Given a specific board configuration, the deep neural network architecture tries to find the most possible next move.

C. Architecture

To gain the best performance for our Gomoku move prediction problem, we have tried a number of different network architectures. We use a multi-layer full convolutional neural network, as shown in Fig.4. The input of the network consists of $15\times15\times3$ feature planes, followed by a 5×5 filters convolutional layer. Because the winning strategy in Gomoku forms a pattern of 5 stones in a row, we choose the 5×5 first filters to capture features that are less than 5 consecutive stones. The following convolutional hidden layer has a kernel size of 3×3 and this layer repeats N times. All the convolutional hidden layers come with a stride of 1, and outputs are zero-padded back up to 15×15. Each convolutional layer is followed by a rectified linear units(ReLU) nonlinearity,

$$f(x) = \max(0, x), \tag{1}$$

where x is the input of ReLU and $f(x)$ is the output. The final hidden layer convolves 1 filter of kernel size 1×1, and a softmax function is applied to select the next move. Instead of using two softmax output layers to predict black and white moves respectively [13], we only use one softmax layer to predict the next move. Since in a Gomoku game, the next move color can be obtained just according to the board state. It is noted that we don't use pooling in the deep convolutional network model because they negatively affect the performance as proven before [12].

Similar to image categorization tasks, the softmax classifier interprets the output from the network layer as the log prob-

[4]https://www.tensorflow.org/
[5]https://keras.io
[6]http://www.renju.net/downloads/downloads.php

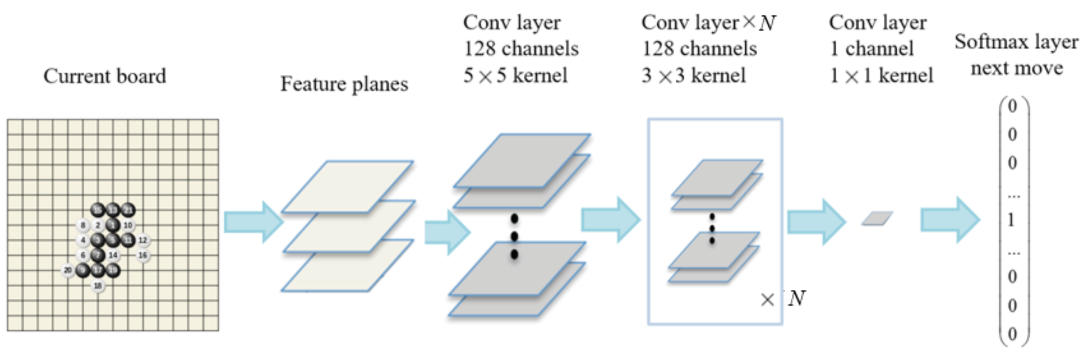

Fig. 4. Structure of networks used in this paper. N represents the number of the convolutional layer with 3×3 filters.

ability for each specific move. The loss function of ith board position move is

$$Loss_i = -log(exp(z_i)/\sum_{j=1}^{m} exp(z_j)), \tag{2}$$

where z_i is the input value of softmax layer for the ith board position, m equals to 225 in the Gomoku game.

D. Training

The RenjuNet dataset is grouped into training set, validation set and test set. We have 147582 data for test, the remaining 1100000 data for training and evaluation, using a 8:2 split. Different from AlphaGo's asynchronous stochastic gradient descent optimizer, we use the Adam optimizer to help with convergence. Adam is straightforward to implement, computationally efficient, and little memory requirement [19]. The Adam update rule is shown below.

$$m = \beta_1 \times m + (1 - \beta_1) \times dx, \tag{3}$$

$$v = \beta_2 \times v + (1 - \beta_2) \times (dx)^2, \tag{4}$$

$$x = x + (-\alpha \times m \,/\, (\sqrt{(v)} + eps)). \tag{5}$$

The aim is to update x based on its gradient dx, with the hyper-parameters β_1 β_2 eps and α. We adopt the default values of Adam hyper-parameters that β_1 equals to 0.9, β_2 equals to 0.999 and eps equals to $1e{-}8$. The learning rate α is initialized to 0.003 and the specific initialization method of m and v can be found in Kingma's paper [19].

Before training, we randomly shuffle the training data into different stages of games. This ensures that each batch contains diversified board states, preventing the network quickly overfitting and getting trapped into poor local optimum. For each training step, we randomly select a mini-batch number of samples from the RenjuNet dataset, and apply Adam update rule to maximize the log likelihood of the action. We use the glorot_uniform function to initialize weights of convolutional layers and their biases. The training process is applied on a computer with a 4GHz Intel i7-6700k CPU, Nvidia TitanX GPU and 32GB of memory.

TABLE I
RESULTS FOR DIFFERENT TOP LAYER NETWORK.

Depth	Full Connected Top Layer	1×1 Convolutional Top Layer
N=2	acc: 36.85%, params: 6785121	acc: 40.59%, params: 305025
N=4	acc: 40.24%, params: 7080289	acc: 41.52%, params: 600193
N=6	acc: 40.15%, params: 7375457	acc: 41.46%, params: 895361

IV. RESULTS AND DISCUSSION

To evaluate the performance of our move prediction network, we test different top layers, batch normalization method, and a variety of hyper-parameters. In the end, we have a discussion of the move prediction method and the future research direction.

A. 1×1 Top Convolutional Layer

In the first stage, we evaluate how a 1×1 top convolutional layer impact the prediction accuracy. In Clark's view, using a fully connected top layer was more effective than a convolutional top layer [12]. To compare the performance of these two choices in our Gomoku problem, we train different top layers and the results are shown in Table I.

From Table I, we can see that different from Clark's conclusion, the full connected top layer performs a little worse than the 1×1 top convolutional layer. In the RenjuNet dataset, leaving the 1×1 top convolutional layer out drops the test accuracy by almost 1%-2%. In addition, the 1×1 top convolutional network has much fewer parameters and the training time is also reduced a lot. Generally speaking, the 1×1 top convolutional network architecture is extremely effective and efficient for move prediction problem. In the following experiments, we all use the full convolutional network architecture with a 1 × 1 top convolutional layer.

B. Batch Normalization

Batch Normalization is used to normalize the activations of previous layers at each batch [18]. It potentially helps deep learning in two ways: faster learning and higher overall

TABLE II
RESULTS OF USING BATCH NORMALIZATION OR NOT.

Depth	Without Batch Normalization	With Batch Normalization
N=2	acc: 40.59%	acc: 40.84%
N=4	acc: 41.52%	acc: 41.68%
N=6	acc: 41.46%	acc: 41.53%

accuracy. In our work, we test how the batch normalization affect the training performance.

From Table II, we can conclude that batch normalization really increases the move prediction accuracy, but very limited. Probably since the training data is much different from the ImageNet dataset where batch normalization generate remarkable performance boost. To keep our model simple and easy to train, we don't use batch normalization in the following experiments.

C. Different Hyper-parameters and Stages

In this stage, we test a number of hyper-parameters and compare the accuracy and loss of different networks. All the results are presented in Table III.

1) Depth, filter numbers and batch size: From the results, it is apparent to see that larger and deeper networks have qualitatively better performance than shallow networks with N increasing from 0 to 10. Further more, as the depth of network increase to more than four convolutional layers, the accuracy change becomes unobvious. For different filter numbers per convolutional layer except the last one, the results show that when the number equals to 128, we have the best accuracy and loss. As for different batch sizes, 32 seem to be the best choice. In general, it is encouraging to note that the prediction accuracy can reach about 42%. To the best of our knowledge, it is an accessible accuracy for Gomoku move prediction at the lever of expert players. Trading off runtime and accuracy, the best network has one convolutional layer with 128 5×5 filters, four convolutional layers with 128 3×3 filters and one layer with a 1×1 filter.

2) Accuracy and loss figures: Training accuracy and validation accuracy of our best model are presented in Fig. 5. In the first epoch, the accuracy increases very fast. In the following, the accuracy increases slowly and remains almost unchanged after 3 epoches. Training loss and validation loss are also presented in Fig. 6. We can see that both the loss decrease quickly on the first epoch and decrease slowly after then. Eventually, the loss converges to be about 2 in the following epoches.

3) Accuracy of different stages: To test the prediction accuracy of different game stages, we divide the test set into different groups by every 10 moves and leave moves more than 100 as a separate group, all the result is presented in Fig. 7. We can see that in the opening, the accuracy is lower and it increases to a high lever in the middle stage. When the moves being more than 70, the accuracy begins to decrease until the end. Possible reasons may be that there are too many empty positions in the opening, resulting a lower prediction

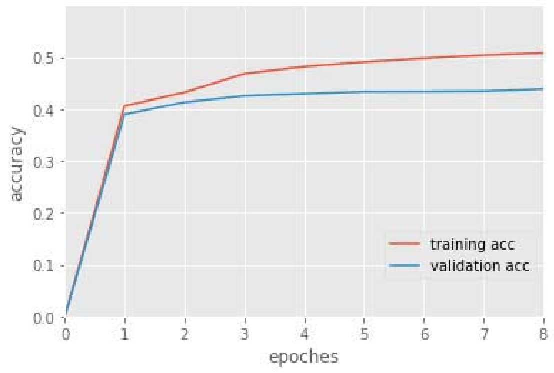

Fig. 5. Training accuracy and validation accuracy of our best model.

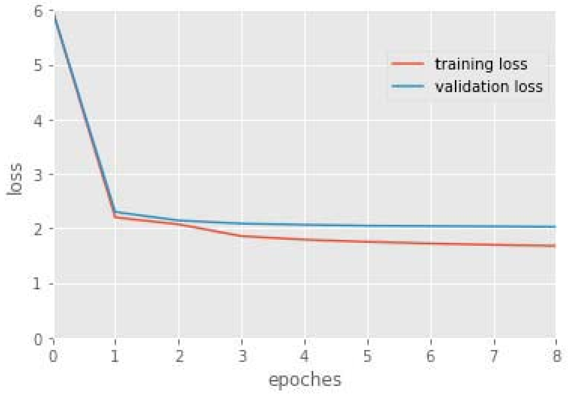

Fig. 6. Training loss and validation loss of our best model.

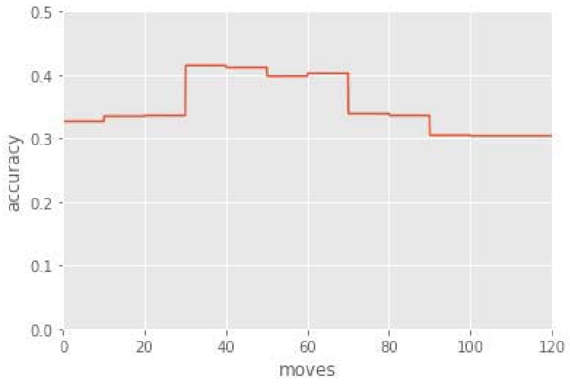

Fig. 7. Test accuracy of different game stages.

accuracy. With the moves increasing, the board state becomes clear and easy to select next move. However, with the board occupied with more and more moves, it is difficult for network to identify the board patterns and to make the next move, leading the accuracy going down.

D. Discussion

According to the results, our move prediction network has proved to generalize well from predicting expert Gomoku player's moves, reaching a 42% accuracy. This is surely

TABLE III
RESULTS FOR DIFFERENT NETWORK HYPER-PARAMETERS.

Depth	Filter Number	Batch Size	Training Accuracy	Validation Accuracy	Test Accuracy	Training Loss	Validation Loss	Test Loss
N=0	128	32	24.76%	20.45%	20.21%	2.918	3.079	3.133
N=1	128	32	38.40%	32.16%	31.17%	2.253	2.526	2.612
N=2	128	32	49.92%	42.24%	40.59%	1.778	2.081	2.227
N=4	128	16	47.32%	41.97%	40.94%	1.889	2.107	2.268
N=4	128	32	51.51%	43.47%	41.52%	1.691	2.279	2.193
N=4	128	64	49.64%	42.35%	41.63%	1.764	2.113	2.246
N=4	128	128	48.59%	41.96%	41.47%	1.809	2.945	2.252
N=4	64	32	47.43%	41.45%	41.66%	1.875	2.141	2.239
N=4	192	32	48.39%	42.37%	41.41%	1.808	2.072	2.186
N=4	256	32	48.27%	42.18%	41.56%	1.834	2.103	2.183
N=6	128	32	50.94%	43.69%	41.46%	1.705	2.023	2.301
N=8	128	32	48.97%	42.44%	41.92%	1.808	2.062	2.239
N=10	128	32	48.81%	42.78%	42.04%	1.825	2.081	2.275

attributed to our deep learning based approach. On the other side, our work also has its limitation, compared with other Go move prediction results. Although we have more than a million (station action) pairs in total, it is still a small dataset compared with AlphaGo's 30 million. In addition, the dataset quality is also much lower level compared with KGS Go Server data. To perform well, it is necessary to build a board evaluation function and combined with tree search. On the other hand, integrating with reinforcement learning techniques and self-playing is considered as a powerful technique to improve the performance. Among them, actor-critics algorithms that train two models simultaneously, one to predict the next move and the other to evaluate the current board situation, seems very suitable for Gomoku.

V. CONCLUSION

In this work we have introduced the application of deep learning to the problem of predicting moves made by expert Gomoku players. Our contributions include a number of techniques that are helpful for this task, including the RenjuNet data preprocessing skills, efficient deep network architecture and effective training method. It is remarkable that a deep convolutional network architecture can predict the moves of the game to such a degree without any expert knowledge. In addition, we also have a lot of work to do to improve the method and to create strong Gomoku agent.

ACKNOWLEDGMENT

We thank Zhen Zhang, Le Lv and Linghan Zhang for their enthusiastic help. And we thank Rongxiao Zhang and Jian Yao for their constructive suggestions.

REFERENCES

[1] W. T. Katz and S. Pham, "Experience-based learning experiments using Gomoku," *IEEE International Conference on Systems, Man, and Cybernetics, 'Decision Aiding for Complex Systems, Conference Proceedings*, pp. 1405–1410, 1991.
[2] J. Schaeffer, "The history heuristic and alpha-beta search enhancements in practice," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 11, no. 11, pp. 1203C-1212, Nov. 1989.
[3] B. Freisleben, "A neural network that learns to play Five-in-a-Row," in *Proceeding of the 2nd New Zealand International Two-Stream Conference on Artificial Neural Networks and Expert Systems*, pp. 87–90, 1995.
[4] D. Zhao, Z. Zhang, and Y. Dai, "Self-teaching adaptive dynamic programming for Gomoku," *Neurocomputing*, vol. 78, no. 1, pp. 23C-29, Feb. 2012.
[5] Y. Lecun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.
[6] J. Schmidhuber, "Deep learning in neural networks: An overview," *Neural Netw.*, vol. 61, pp. 85–117, 2015.
[7] A. Krizhevsky, I. Sutskever, and G. E.Hinton, "Imagenet classification with deep convolutional neural networks," in *Proceeding of the Neural Information and Processing Systems*, pp. 1097–1105, 2012.
[8] G. Dahl, D. Yu, L. Deng, and A. Acero, "Context-dependent pre-trained deep neural networks for large-vocabulary speech recognition," *IEEE Trans. Audio Speech Lang. Processing*, vol. 20, no. 1, pp. 30–42, Jan. 2012.
[9] I. Sutskever, O. Vinyals, and Q. V. Le, "Sequence to Sequence Learning with Neural Networks," in *Proceeding of the Neural Information and Processing Systems*, 2014.
[10] D. Silver, A. Huang, C. J. Maddison, A. Guez, and L. Sifre, "Mastering the game of Go with deep neural networks and tree search," *Nature*, vol. 529, no. 7587, pp. 484–489, 2016.
[11] I. Sutskever and V. Nair, "Mimicking Go experts with convolutional neural networks," in *International Conference on Artificial Neural Networks*, pp. 101–110, 2008.
[12] A. Storkey, "Training deep convolutional neural networks to play go," in *Proceedings of the 32 nd International Conference on Machine Learning*, 2015.
[13] C. J. Maddison, A. Huang, I. Sutskever, and D. Silver, "Move evaluation in go using deep convolutional neural networks," in *Proceedings of the International Conference on Learning Representations*, 2015.
[14] Y. Tian and Y. Zhu, "Better computer Go player with neural network and long-term pediction," in *Proceedings of the International Conference on Learning Representations*, 2016.
[15] K Young, R Hayward, and G Vasan, "Neurohex: A Deep Q-learning Hex Agent," *arXiv preprint* arXiv: : 1604. 07097,2016.
[16] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, and J. Veness, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.
[17] M. Abadi, A. Agarwal, P. Barham, and E. Brevdo, "Tensorflow: Large-scale machine learning on heterogeneous distributed systems," *arXiv preprint* arXiv: 1603. 04467, 2016.
[18] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International Conference on Machine Learning*, 2015.
[19] D. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proceedings of the International Conference on Learning Representations*, 2015.

View publication stats