Deep Learning and Block Go

Shi-Jim Yen[†1], Ching-Nung Lin [2], Guan-Lun Cheng[3] and Jr-Chang Chen[4]

***Abstract*:** Google Deepmind AlphaGo successfully defeated a professional nine dan Go player last March. One of the reasons is that they use deep learning to do a pure pattern-matching approach and predict the next move. In this paper, we use deep learning on the game of Block Go. Block Go is a variance of Go. In this paper, firstly we introduce the complexity of Block Go which is between checkers and Othello. Then we apply Deep Convolutional Neural Network (DCNN) on Block Go. Finally, we show that Block Go is a good research topic for deep learning.

Keywords: Block Go, Computer Go, Deep Learning, Deep Convolutional Neural Network.

1. Introduction

In 2009, Zhang Xu, who is one of the strongest Go players, invented this game. This game is a simplified version of Go. The complexity is 10^45, which is between checkers and Othello. Each move is to place a Block. The total number of move sequences is 18. The design of Block Go is for children. Block Go is popular in most Taiwanese children's Go colleges. We believe that Block Go is also suitable for a testbed for deep learning. In this paper, we will apply deep learning on Block Go. The rule of Block Go and the complexity of Block Go are described in section 2. Section 3 shows two deep learning models for Block Go and the experimental results. Finally, Section 4 makes the conclusion and states future works.

2. Rules and Complexity

Block Go is played on a 13x13 Go Board. Each side, named Blue and Red, has nine blocks. Seven blocks of them are the same as those in Tetris, named I, J, L, O, S, T and Z. Each of the seven blocks can be viewed as a composed of four stones. The other two same blocks are composed of a single stone named B. Figure 1 and 2 respectively show the initial board and a complete game. Figure 4 shows the number of possible placement styles for one legal position. The *legal position* is either the four star positions that are (4,4), (4,10), (10,4) and (10,10) in the first four moves, or on the positions adjacent to his/her own blocks. For example, in Figure 5, Blue (Red) can place a block on the white (green) points. Figure 6 illustrates illegal moves. The rules are stated as following:

1. In each move, each player can place one block on the legal positions.
2. When all the 18 blocks are placed, the game is finished. The counting of territory is the same as that of Go, and the player with more territory wins. The handicap is 3.5 points.
3. If one player's stones are within the opponent's territory and the group has no one space surrounded, the stones are dead and belong to the opponent. Figure 1 is an example of a Block Go game.

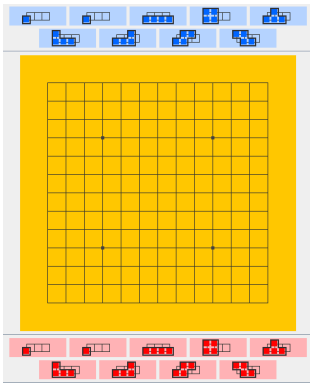

Figure 1. Initial board for Block Go.

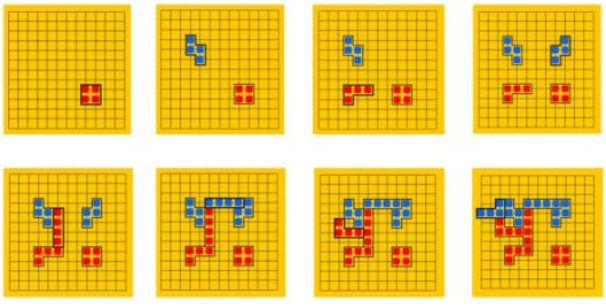

Figure 2. A Block Go game(1).

[1] Shi-Jim Yen[†1] , Ching-Nung Lin[2] , Guan-Lun Cheng [3] Lin are with the Department of Computer Science and Information Engineering, National Dong Hwa University, Hualien 97401, Taiwan (e-mail: sjyen@mail.ndhu.edu.tw, 810221001@gms.ndhu,edu.tw, 610421203@gms.ndhu.edu.tw).
[2] Jr-Chang Chen[4] is with the Department of Applied Mathematics, Chung Yuan Christian University, Taoyuan 32023, Taiwan. (e-mail: jcchen@cycu.edu.tw).

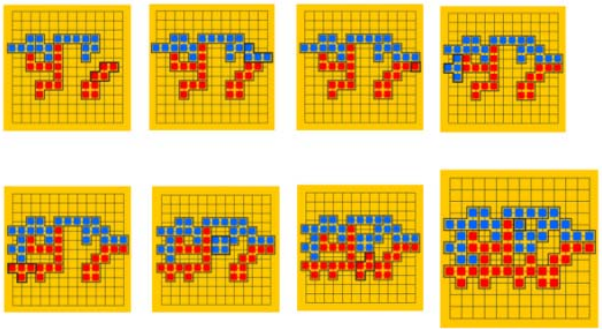

Figure 3. A Block Go game(2). Red won by 0.5 point.

Tetris name	I	J	L	O
possibility	8	16	16	4
Block				

S	T	Z	B	B
16	16	16	1	1

Figure 4. The nine blocks and the numbers of their possible placement for one legal position.

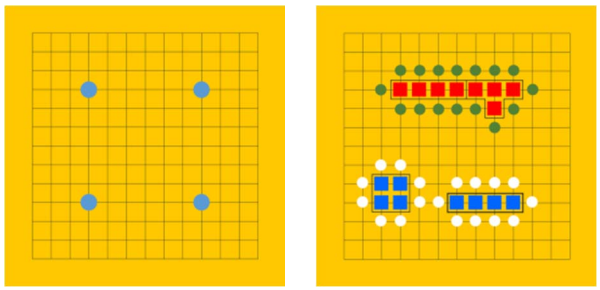

Figure 5. Legal positions.

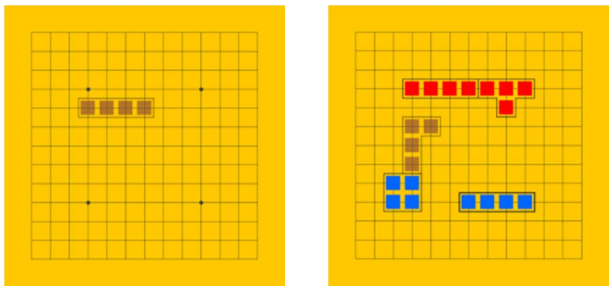

Figure 6. Illegal moves.

The complexity of Block Go is estimated as follows:

$$(77 \times 4) \times (77 \times 3) \times ((66 \times 10)^2) \times ((55 \times 17)^2) \times ((44 \times 22)^2) \times ((33 \times 26)^2) \times ((22 \times 30)^2) \times ((11 \times 33)^2) \times ((34)^2) \times ((35)^2) = 10^{45}.$$

For the first ply, the number of possible moves is 77, which is the possible block-placement method on one legal position for the nine blocks. Initially, there are 4 legal positions. Thus, the possibility of the first play is 77×4. In the following plies, when a block is placed, the number of legal positions increases, but the number of the remaining blocks decreases. The complexity of Block Go is 10^{45} that is close to Nine Men's Morris (10^{50}) and Othello (10^{58}) [3].

3. Deep Learning

Deep learning can do a pure pattern-matching approach and predict the next move for Block Go. This is useful in a MCTS Block Go program. MCTS (Monte Carlo Tree Search) has four stages that are *selection*, *expansion*, *simulation* and *backpropagation* [1], as in Figure 7. The goal of *selection* is to find an action which maximizes UCB formula [12] in every level of the game tree, and obtain the best sequence from the root to the leaf node. The second stage *expansion* is to *create* a new child node, corresponding to one of the legal moves of the parent node. In this stage, we often give prior knowledge to those children nodes of the parent node, and give each child node a weight. If we can expand the most important child node each time, the search will be very efficiently. As the state-of-the-art Go programs [8][9][10][11], we use deep learning to predict the next move and give each child node a weight.

The third stage *simulation* is to perform a simulation from the position represented by the new created node. When the simulation is completed, the final position is scored and used to give the position value. This value is propagated along with the path in the selection stage to the root node. Each node updates its own statistical data by the value. One of the differences between Block Go and Go is that only 18 plies in Block Go. Thus, the quality of the simulation is important. A bad simulation may not make any territory, because there are not enough moves to form the boundary.

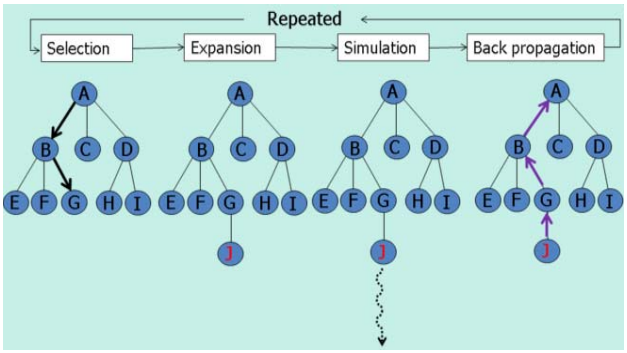

Figure 7. Scheme of MCTS. [1]

This section describes two deep learning experiments on Block Go. Figure 8 is the basic DCNN(deep convolutional neural nework) for Block Go. The network architecture is similar to [8]. The training data is from a Block Go APP company.

The purpose of the first experiment is to predict the "*fully-match*" move which is correct when matching the full block. The training data includes 3680010 positions and testing data includes 129609 positions. The data augmentation is with rotation at 90-degree intervals and horizontal/vertical flipping. We also remove all AI play moves for the row data, and only keep human moves. The input is 45 planes, as in

Table 1.

Table 1. 45 planes for the fully-match experiment.

1	Self-moves
2	Opponent moves
3	Empty
4	Boarder
5-21	1^{st}-17^{th} move
22	All 1 plane
23	All 0 plane
24	The four star position
25-45	All current legal moves considering rotation.

The batch size is 18 because there are 18 positions in one game. Each batch can include all stages of a game. There are 19 layers. Each follows a ReLU layer. The Channel size is 512. The feature size in the first 12 layers is 7x7; the feature size in layer 13 to layer 19 is 3x3. The channel size in the last layer is 4, it is similar to multi label classification. One block is treated as 4 labels. For the block with the single stone, it is treated as four same label. The batch size 10 is used and an epoch is 368,000 mini-batches. After 24 epochs, the full match accuracy is 3.00%.

In the first experiment, the accuracy is low. We design the second experiment. It only partially predicts of the next move. This "*partially-match*" model only predicts the legal position of next move. If there are multiple legal positions, we use the most left-up legal position. There are 4 planes as shown in Table 2.

Table 2. The 4 planes for the partially-match experiment.

1	Self moves
2	Opponent moves
3	Empty
4	Boarder

The training data includes 2668864 positions and testing data includes 22576 positions. The data augmentation is with rotation at 90-degree intervals and horizontal/vertical flipping. We didn't remove the AI play moves for the row data. There are 19 layers. Each follows a ReLU layer. The channel size in the first layer is 30; the channel size in layer 2 to layer 18 is 228; The channel size for the last layer is 1. The feature size in the first layer is 5x5. The feature size in other layers is 3x3. The batch size 256 is used and an epoch is 10,000 mini-batches. After 150 epoch, the accuracy is 70.4%.

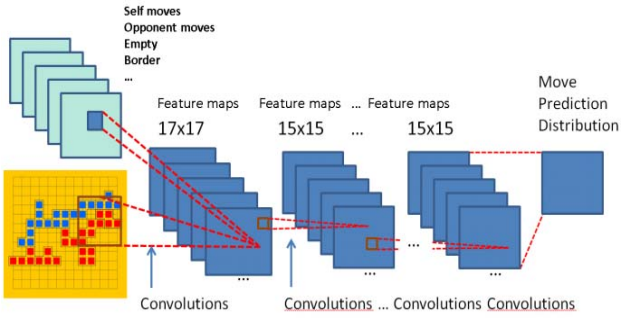

Figure 8. Deep convolutional neural nework for Block Go

4. Conclusion

In this paper, we describe the rule of Block Go and introduce the complexity of Block Go. Then we design two deep learning experiments for Block Go. In practice, we can give the fully-match model more weight, and also consider the partially-match model's result. The two models will be helpful in the MCTS Block Go program.

The rule of Block Go is very simple, even simpler than the game of Go. The complexity is around 10^{45}, which is between checkers and Othello. Thus, Block Go may be a good testbed for computer games since it is easier to implement the policy network and the value network[9].

Because there are not many high quality Block Go game records, in the future, we will apply transfer learning to improve the DCNN of Block Go, based on the numerous 19×19 pro Go game records. Then it is possible to develop a strong Block Go program.

ACKNOWLEDGMENT

The authors would like to thank anonymous referees for their valuable comments in improving the overall quality of this paper, and Ministry of Science and Technology of Taiwan for financial support of this research under the contract numbers 104-2221-E-259 -009 -MY2 and 105-2218-E-259 -001.

References

[1] Chaslot, G.M.J.-B., Winands, M.H.M., Uiterwijk, J.W.H.M., van den Herik, H.J., and Bouzy, B., "Progressive strategies for Monte-Carlo Tree Search," *New Mathematics and Natural Computation*, vol.4, no. 3, pp. 343–357, 2008.

[2] R. Coulom, "Computing Elo Ratings of Move Patterns in the Game of Go," *ICGA Journal*, vol. 30, 2007

[3] H. Jaap van den Herik, Jos W.H.M. Uiterwijk, Jack van Rinswijck, "Games solved: Now and in the future," *Artificial Intelligence*, vol. 134, pp. 277-311, 2002.

[4] Edward J. Powley, Peter I. Cowling, Daniel Whitehouse, "Information capture and reuse strategies in Monte Carlo Tree Search with applications to games of hidden information," *Artificial Intelligence*, vol. 217, pp. 92-116, 2014.

[5] Shi-Jim Yen, Cheng-Wei Chou, Jr-Chang Chen, I-Chen Wu, and Kuo-Yuan Kao, "Design and Implementation of Chinese Dark Chess Programs," *IEEE Transactions on Computational Intelligence and AI in Games*, vol. 7, issue 1, pp 66-74, 2015.

[6] Shi-Jim Yen, Tsan-Cheng Su and I-Chen Wu, "The TCGA 2011 Computer-Games Tournament," *ICGA Journal*, vol. 34, no. 2, 2011, pp. 108–110.

[7] Russell, S. and Norvig, P., *Artificial Intelligence: A Modern Approach* 2/e, Prentice Hall, 2003.

[8] Yuandong Tian and Yan Zhu, "Better Computer Go Player with Neural Network and Long-term Prediction", ICLR, 2016.

[9] David Silver, Aja Huang, Christopher J. Maddison, Arthur Guez, Laurent Sifre, George van den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, Sander Dieleman, Dominik Grewe, John Nham, Nal Kalchbrenner, Ilya Sutskever, Timothy Lillicrap, Madeleine Leach, Koray Kavukcuoglu, Thore Graepel and Demis Hassabis, "Mastering the game of Go with deep neural networks and tree search", Nature, 2016, Pages 484-503, Vol. 529.

[10] Chang-Shing Lee, Mei-Hui Wang, Shi-Jim Yen, Ting-Han Wei, I-Chen Wu, Ping-Chiang Chou, Chun-Hsun Chou, Ming-Wan Wang, and Tai-Hsiung Yang, "Human vs. Computer Go: Review and Prospect", IEEE Computational Intelligence Magazine (IEEE CIM), Vol. 11, No. 3, pp. :67-72, August 2016.

[11] Chris J Maddison, Aja Huang, Ilya Sutskever, and David Silver, "Move evaluation in go using deep convolutional neural networks", In International Conference on Learning Representations, 2015.

[12] Gelly, S., Wang, Y., Munos, R., and Teytaud, O., Modification of UCT with patterns in Monte-Carlo Go, *Technical Report* 6062, INRIA, 2006.