Life-and-death Problem Prediction using Deep Convolutional Neural Network in the Game of Go

Shi-Jim Yen[†1], Guan-Lun Chen[†1], Jr-Chang Chen[†2]

Abstract: The life-and-death problem in Go is regarded as a very important research topic because it requires a completely correct response and often occurs during the game. The research on the life-and-death problem includes methods such as search trees and feature simulation. Deep Convolutional Neural Network (DCNN), which has been proven to be applied to the game of Go, can achieve a higher accuracy rate of predicting the next move and increase the strength of the Go program. This paper trains a DCNN model to predict correct moves for life-and-death problems in the game of Go. We use the classic life-and-death problems of Go, including 5925 questions in the training data and 96 questions in the test data. From these questions, 281444 game boards and 3496 game boards can be generated for training and testing. The experimental results show that the accuracy of DCNN answering the life-and-death problem is 96.90% for the first move, and the accuracy for the consecutive moves is 90.25%, and the predicted move has a high accuracy. This paper also studies adding the model of the life-and-death problem to the model for predicting the next move, trying to improve the accuracy of predicting the next move.

Keywords: computer games, the game of Go, life-and-death problems, Deep Convolutional Neural Network.

1. Introduction

Since 2007, the Go program using the Monte Carlo Tree Search (MCTS) method has used the Monte Carlo method for statistics, combined with Upper Confidence Bound Applied to Trees, to greatly enhance its Go strength [1-4], so that the computer Go program has begun to have the ability to play against professional players. At 2016, AlphaGo developed by Google DeepMind, with the assistance of Deep Convolutional Neural Network (DCNN) [1] and combined with MCTS to calculate Go moves, finally defeated the top human Go players.

However, there are still many issues worthy of study in Go, one of which is the issue of life-and-death. [5,6] The ultimate goal of Go is to enable yourself to encircle more land than your opponent. In order to encircle more land, you need to kill the opponent's pieces, or try your best to keep yourself alive. Therefore, in the process of playing go, it is often necessary to make judgments and further actions on the life-and-death status of some Go shapes. In the life-and-death problem, one must ensure that every move is correct, in order to successfully kill the opponent's pieces, or keep one's own pieces alive, otherwise it is easy to be cracked by the opponent and reverse the situation.

DCNN can be efficiently applied to image recognition, including computer game problems [7-9], and has quite good results. In this paper, it is hoped that DCNN can be used to predict the moves of life-and-death problems. It is expected that while achieving accurate predictions, it can also be faster than the search method and reduce the cost of writing rules.

2. Methods

AlphaGo and DarkforestGo have proven that DCNN applications have very good results in Go[1][11], with short reaction times and no need to write rules. The DCNN model in this paper, which contains 12 layers of Full Convolutional Layer, uses 25 feature planes. The first layer contains 92 channels with a kernel size of 5x5, and the remaining Convolutional Layer contains 384 channels with a kernel size of 3x3. Figure 1 shows the model framework. The 25 feature planes include liberties, Ko, stone position, move history, rank, border and territory. Table 1 shows the feature planes.

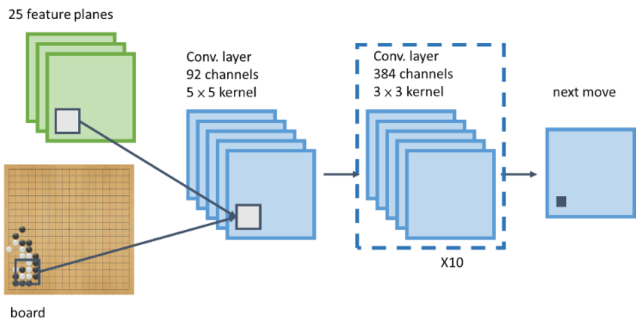

Fig. 1. The model framework.

Table 1. The 25 feature planes.

feature	Data type	Number
our/opponent liberties	binary	6
Ko	binary	1
our/opponent stones/empty	binary	3
our/opponent history	real	2
opponent rank	binary	9
border	binary	1
position mask	real	1
our/opponent territory	binary	2

We use the classic Go life-and-death problems [10], including 5925 questions in the training data and 96 questions in the test data. These datasets are rotated, flipped, and black-and-white swapped to increase the variability and quantity of the data. And also filter out the questions in the training data that are repeated with the test data. Finally, there are 281,444 training game boards and 3,496 testing game boards.

The experiments in this paper are based on the hardware and software environment of Intel Xeon E5-2630, 64 GB RAM,

†1 Dept. Of Computer Science and Information Engineering, National Dong Hwa University, Taiwan

†2 Dept. Of Computer Science and Information Engineering, National Taipei University, Taiwan

NVidia GTX 1080 and Ubuntu 14.04. The experiment in this paper uses Torch Framework as the training and testing environment, and writes mini-batch for training. Here, each Epoch is defined as 10,000 mini-batches, and each mini-batch contains 256 game boards. The learning rate used in Epoch 20 and before is 0.02, and in Epoch 21 and later it is 0.01. In the end, 30 Epochs were trained. The reason is that after the 15th Epoch, the prediction accuracy of the trained model tends to be stable for the test data.

In addition, this paper also studies adding the model of the life-and-death problem to the model for predicting the next move. It is expected that after the information is sent to the trained life-and-death model (hereinafter referred to as LD Model), the predicted life-and-death information can be merged into the DarkforestGo model (hereinafter referred to as Df Model) [11], in an attempt to enhance its prediction accuracy. Therefore, the experiment uses the data set of Tygem as the source of training and testing data to establish an original Df Model and a Df Model (hereinafter referred to as Df-LD Model) containing life-and-death information to compare the prediction accuracy.

The process is to send the Tygem data set into the LD Model, make it return a matrix containing the probability of the next move, and then integrate the matrix into the Df-LD Model as a feature plane. Therefore, Df-LD Model has 26 feature planes, and other structures are consistent with Df Model. Both Df Model and Df-LD Model train for 25 epochs to obtain a higher accuracy rate. Each Epoch contains 10,000 mini-batches, and the Learning rate is 0.05 between Epoch 10 and before, 0.02 between Epoch 20 and before, and 0.01 after that.

3. Experiments and results

Use the test data to check the prediction accuracy for the 30th Epoch of the LD Model, and the results are shown in Table 2 below.

Table 2. LD model prediction accuracy.

First move	All Black moves (i.e. the 1, 3, 5, 7,…moves)	All Black moves with random White moves
96.90%	90.25%	78.49%

Judging from the accuracy of all boards, LD Model has a very good effect on solving life-and-death problems. It can correctly predict how to proceed in the next step of the board in most cases, so as to correctly attack the opponent as much as possible. When inputting questions on the aforementioned operating environment, the predicted answer can be sent out in an average of 0.0044 seconds.

However, in the life-and-death problem, it is necessary to make correct moves in succession in order to solve the opponent's offensive, or to really defend one's position. As long as one wrong step is made, it will lead to the failure of the attack or defense, that is, the final state of the whole life-and-death problem changes. Therefore, the experiment also checks the accuracy of all questions. The setting here is: if all the moves in a question are correctly predicted by the LD Model, it is considered that the question has been answered correctly. The results show that 90.25% of the questions can be solved completely and correctly.

In addition, the experiment of randomly dropping Go pieces on other places on the board is also carried out, in order to simulate in the process of the game, in addition to the single life-and-death problem, there are other situations that start to exist. In this experiment, 30 black or white Go pieces will be randomly added, and they will be separated from the original life-and-death problem by at least two columns and two rows, so as not to affect the original life-and-death result. It can be seen from the experimental results that the accuracy is lower than that of the situation without random moves. It means that if you want to solve the life-and-death problem in the actual game, you must clearly mark the range of the life-and-death problem in order to have a high accuracy rate.

Table 3. LD Model of each move in all Black moves.

Step	1	3	5	7
Accuracy	97.66%	97.66%	96.81%	95.59%

In addition, the accuracy of each question is also analyzed to check the accuracy of LD-Model answers in different starting sequences. Here we only choose black (with an odd number of moves) for testing, and the results are shown in Table 3. According to the experience of solving life-and-death problems, it is speculated that there are more choices in the first step, and there should be a lower accuracy. However, the tests showed that the first moves had a higher predictive accuracy, which was slightly different than expected. When guessing right and wrong move, it is easier to have questions with multiple answers, but in this experiment, only the standard answers are compared, so when a question has multiple answers, such as Figure 2, there is a relatively high probability of errors.

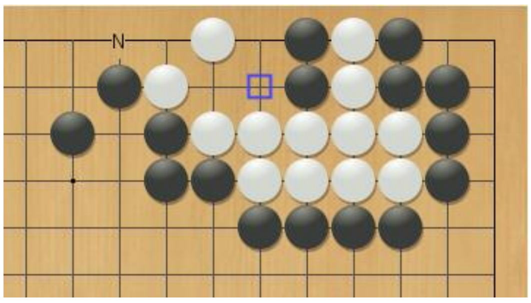

Fig. 2. An example question with multiple answers. Both square and N position are the correct answers.

Finally, in the research of trying to add the model of the life-and-death problem to the model of predicting the first move, after 25 epochs of training, the Df Model can obtain the highest accuracy rate of 51.91%, while the Df-LD Model slightly increases the accuracy rate, reaching 52.21%. The results show that adding the life-and-death probability information generated by the LD Model is not very helpful to the prediction accuracy, but it takes extra time to add, which is not conducive to the actual real-time game calculation. There are two reasons:

1. In the case of extra Go pieces, the accuracy of LD-Model is lower than that of the board with only simple questions.
2. When playing Go, high-level players have already predicted how the opponent will develop after the move, so they will not continue to waste pieces, forming a complete life-and-death problem, which also makes life-and-death information less useful.

4. Conclusion

The model trained in this paper to solve the life-and-death problem in the experiment can predict the best moves for most problems without writing a large number of rules. This model also demonstrates the applicability of DCNN to Go. If its information can help humans or AI Go players get answers, it will be beneficial to the Go world.

At the same time, the author also believes that it is expected to implement the model with a simpler neural network-like structure, such as fewer channels, or reduce the size of the input disk, in exchange for faster calculation time, replacing the Rollout Policy of MCTS. There are other Go pieces on the board, whether it can still be uninterrupted, and find out the emergency of the life-and-death problem, which can also be improved. The final goal is to analyze the questions currently used for training, find out the strength distribution of the questions, and adjust the proportion of the questions appropriately, in order to improve the accuracy of solving the life-and-death problem.

ACKNOWLEDGEMENT

The authors would like to thank anonymous referees for their valuable comments in improving the overall quality of this paper. This work was supported in part by the National Science and Technology Council under the contract NSC 110-2221-E-259 -007 -MY3, NSC 111-2634-F-A49 -013. We thank to National Center for High-performance Computing (NCHC) for providing computational and storage resources.

REFERENCES

[1] Silver, D., Huang, A., Maddison, C. et al, “Mastering the game of Go with deep neural networks and tree search,” Nature 529, 484–489 (March, 2016).

[2] Silver, D., Schrittwieser, J., Simonyan, K. et al., “Mastering the game of Go without human knowledge,” Nature 550, 354–359 (October, 2017).

[3] Silver, D., “Mastering Go and Shogi by Self-Play with a General Reinforcement Learning Algorithm,” Science, vol 362, Issue 6419, pp. 1140-1144, 2018, doi: 10.1126/science.aar6404.

[4] Danihelka, I., Guez, A., Schrittwieser, J., and Silver, D., “Policy improvement by planning with gumbel,” International Conference on Learning Representations, 2022.

[5] K. Chen, Z.X. Chen, “Static analysis of life-and-death in the game of Go”, *Information Sciences 121*, 1999.

[6] T. Wolf, L. Shen, “Checking Life-and-Death Problems in Go I: The Program ScanLD”, *ICGA journal 30(2)*, 2007.

[7] Jarrett, Kevin, et al. "What is the best multi-stage architecture for object recognition?." 2009 IEEE 12th international conference on computer vision. IEEE, 2009.

[8] Khan, A., Sohail, A., Zahoora, U. et al. "A survey of the recent architectures of deep convolutional neural networks," Artif Intell Rev 53, 5455–5516 (2020).

[9] Liskowski, Paweł, Wojciech Jaśkowski, and Krzysztof Krawiec. "Learning to play othello with deep neural networks." IEEE Transactions on Games 10.4 (2018): 354-364.

[10] Y.D. Tian, Y. Zhu, “Better Computer Go Player with Neural Network and Long Term Prediction”, *arXiv preprint*, 2015.

[11] 101weiqi.com,　Basic life-and-death problems, https://www.101weiqi.com/questionlib/