International Journal of Hybrid Information Technology
Vol.9, No.4 (2016), pp. 51-58
http://dx.doi.org/10.14257/ijhit.2016.9.4.05

Suggesting Moving Positions in Go-Game with Convolutional Neural Networks Trained Data

*Hoang Huu Duc[1], Lee Jihoon[2] and *Jung Keechul[3]

Media department, Soongsil University, Seoul, Korea
[1]duc93t@gmail.com, [2]leejh3928@naver.com, [3]jungkeechul@gmail.com

Abstract

Nowadays, machine learning and deep learning has been becoming popular and useful in applying to resolve people's problem. Specially, in HCI (Human Computer Interaction) field like robots or automatic game programs. Go-Game (the game of Go) is still a challenge in coding to get the wisest moves each turn to achieve the winner at the end of a game. In our work, we suggest the next move based on Convolutional Neural Networks (CNNs) and make evaluations and comparisons to gamers separate in 3 ranks (levels). We train 5-layers CNNs by supervised learning from a database of human games using the board-states. The network suggests the move of the selected player and the others player can be helped or not- depend on playing option. The program can also play the game automatically without human interactions during all the game progress (Machine-Machine game). In the other way, our program can interact with a human-player and accept move commands from player (Human-Machine or Human-Human). This technique allows Go-game program play the game without searching as traditional program but trained by convolutional neural networks. In our tests, we separate in 3 levels and use totally 598,472 board-states for training data. Our main aim is to help people who are the newbie in playing Go-game. With this technique, we expected that we can apply to develop AI programs and devices with more and more effects and higher performance.

Keywords*: Deep learning; Convolutional Neural Networks; Go game; Suggesting Moving Positions; Suggesting Moves.*

1. Introduction

In human history, when people have reached intelligence to build a social lives, our ancients usually look forward finding out the most intelligent people via some tests by creating poems or playing intelligent games. In several most oldest games, Go game is a rare one that well-known in finding out human's intelligence based on the playing skill. Go-game is a traditional game created by Chinese ancients what kings or generals often use to find out most intelligent people who can help them to reign their kingdom efficiently.

Now a days, Go-game is still very popular in our modern lives with millions players all over the world. Despite of today's high technologies, there are many algorithms and methods applied for machine learning to make a machine (computer) becomes intelligent as it has a brain to think and give out decisions themselves (that call artificial intelligence-AI). Gradually, intelligent algorithms and solutions have been becoming so wise that an AI system can win an expert gamer in playing some intelligent games.

The most difficult part in writing a Go-game program is constructing evaluation function that can suggest an advantaged move while playing at a given situation. In board game kinds like Chess, Shogi, Chinese Chess, etc. there are so many possible moves to

Corresponding Authors:
Hoang Huu Duc, Jung Keechul

ISSN: 1738-9968 IJHIT

take. So, what is the wise choice or best move for each turn to take advantage in order to win the game not only base on the very player's strategy but also depend on the opposite player's moves. In each particular situation, there is not only one best choice but normally there are several or even tens good moves with the approximate weight to take advantage with different strategy to win at last. The combination of an enormous state space of 10170 positions combined with sharp tactics that lead to steep non-linearity in the optimal value function has managed many researchers to conclude that representing and learning such a function is impossible. In several recent years, the most success methods have sidestepped this problem using Monte-Carlo tree search (MCTS). MCTS evaluates a position dynamically through random sequences of self-play.

To green players, play on smaller 9×9 and 13×13 boards is the best ways to approach Go-game. We do not care these small sizes because their difficulty levels are low. We just focus on a supervised learning setup, in which the network is trained to suggest expert human moves using a large database of professional 19x19 size Go-game board. The suggestive accuracy of the CNNs on a held-out set of positions can reach 55%; Some strongest Go-games reported suggestive accuracy is about 35% to 39% [10].

We resolve this problem using a deep convolutional neural network (CNN). In spite of reality that some researchers have applied CNN to resolve Go-game and got success as: Schraudolph et al. 1994 [11]; Enzenberger, 1996 [5]; Sutskever & Nair 2008 [12], but there are limitations: computing power and number of hidden layer- they normally use only one hidden layer and the computational power had not been as strong as of today. Our CNN use 5 hidden layers and billions connections to represent and learn Go-game knowledge. Ours increase in depth and size leads to a qualitative jump in performance, a strong function in move evaluation for Go-Game can be represented and learnt by such architectures.

In our work, we figure out the applying of CNNs into making a Go game program which can help green players to learn how to play Go-game by suggesting possible moves, we suggest 10 choices of good moves and the best moves is numbered from 1st to 10th – the higher number is the worse choice.

2. Training Procedure and Architecture of CNN

We describe the details of the training procedure and the precise network architecture. The input to a convolutional layer is an m x m x r image where m is the height and width of the image and r is the number of channels (e.g. an RGB image has r = 3). The convolutional layer have k filters (or kernels) of size n x n x q where n is smaller than m and q can either be the same as the number of channels r or smaller.

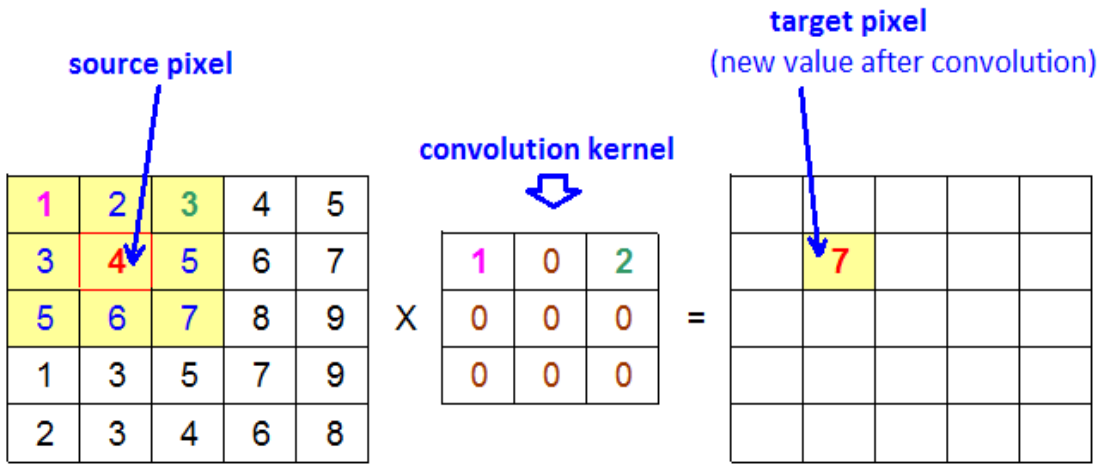


Figure 1. A Kernel Operates on One Pixel

A CNN contains a number of convolutional and subsampling layers followed by fully connected layers shown in figure 2 [13]. The numbers in the kernel of CNN represent the amount by which to multiply the number underneath it. The number underneath represents the intensity of the pixel over which the kernel element is hovering. During convolution, the center of the kernel passes over each pixel in the image. The process multiplies each number in the kernel by the pixel intensity value directly underneath it. The final step of the process sums all of the products together, divides them by the amount of numbers in the kernel, and this value becomes the new intensity of the pixel that was directly under the center of the kernel.

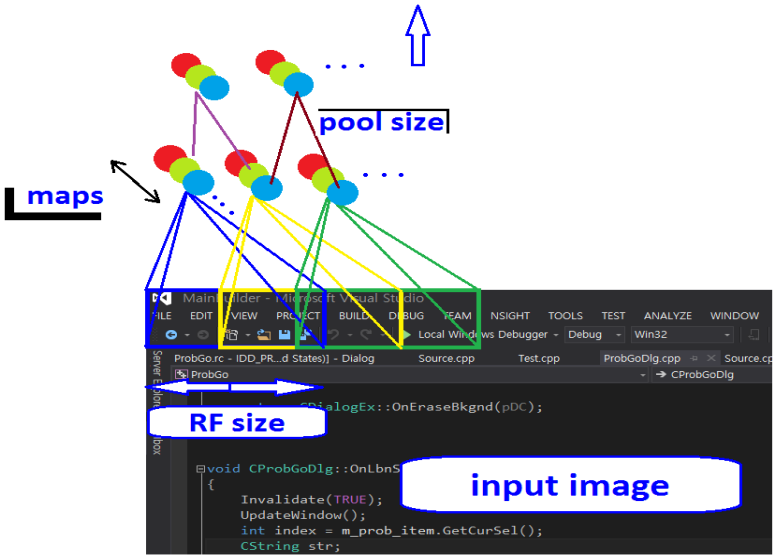


Figure 2. One Layer of the Convolutional Neural Network the Pooling Stage. Units of the Same Color have Tied Weights and Units of Different Color Represent Different Filter Maps

Each map is then subsampled typically with mean or max pooling over *p* x *p* contiguous regions where *p* ranges between 2 for small images and is usually not more than 5 for larger inputs. Either before or after the subsampling layer an additive bias and sigmoidal nonlinearity is applied to each feature map. Each pixel in the board has a value and it received new value after convolute using a kernel as shown in figure 1 [13].We used a deep convolutional neural network with 5 convolution layers and 1 fully connected layer. Convolution layer's filters were of size 5x5 with the stride is 1. Every convolution layer operated on a 19x19 input space, with no pooling; outputs were zero-padded backup up to 19x19. The number of filters in convolution layer is 64. And number of hidden layer's neurons is 361.

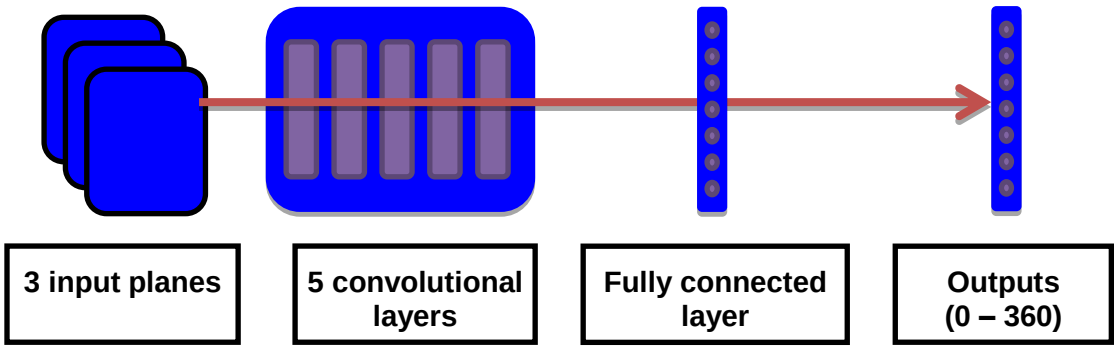


Figure 3. Our networks: Trained Batch Size: 64; Learning Late: 0.001; Number of Iterations: 2,000,000 Times.

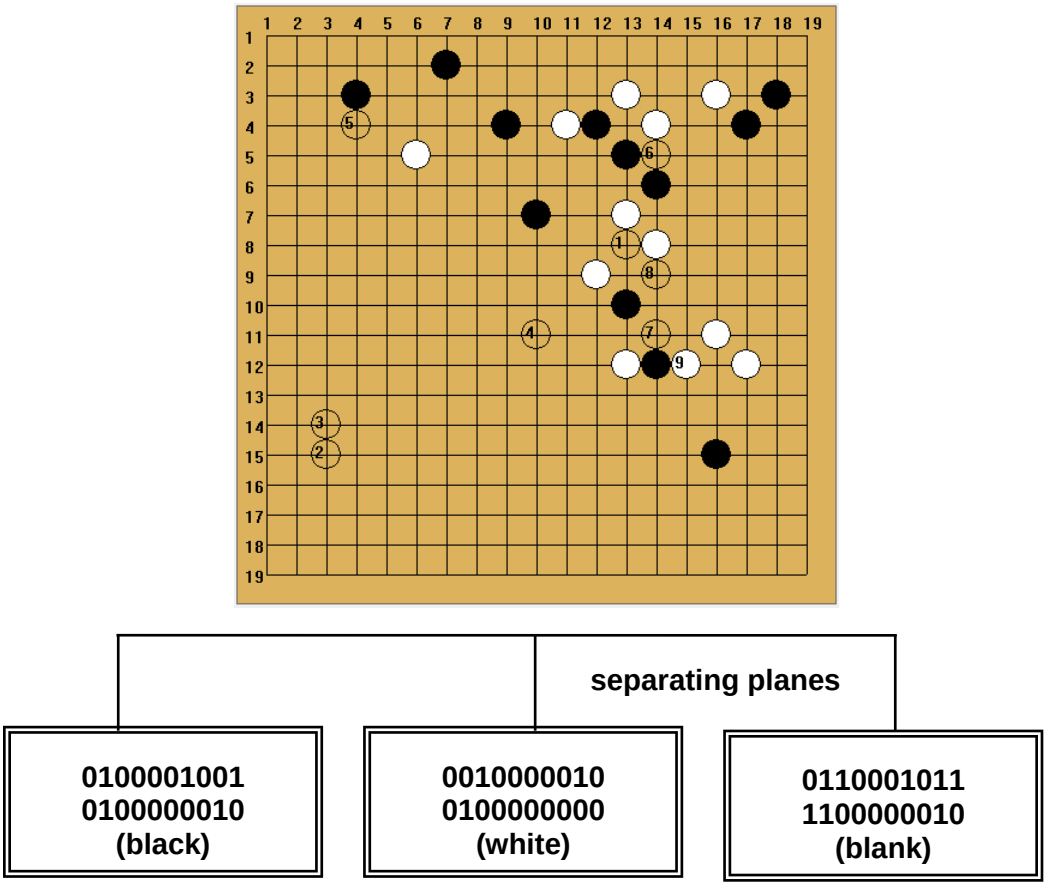


Figure 4. Calculating best Moves using CNNs to Suggest for Player

The 3 input planes are: Black stone; white stone; blank position as shown in the figure 5. After training using hundred thousand board-states, the program calculates the score for each position and making suggestion.

3. Data

To calculate and make comparison with high precision, we using data from cyber-oro database [14]. In this data source, there are sequences of board positions s_t for complete games played between humans of varying rank. A Board-state contains positions of all stones on the 19x19 board and the sequences allow one to determine the sequence of each move. A move at a_t is encoded to a valus between 1 and 361 indicators (numbered from 0 to 360) for each position on the 19x19 board. We use more than 500.000 board-states to train for the inputs using CNNs. We separate board-states in to 3 levels (ranks): basic (newbie); advance; expert.

Every position s_t is preprocessed into 19x19 feature planes B(s_t) also work as input data to the neural network. Each suggestive move must follow Go-game rules about stone, captures, turns. Each turn, we provide 10 best possible positions and first (number 1) is the best choice with highest calculating score as be shown in figure 5.

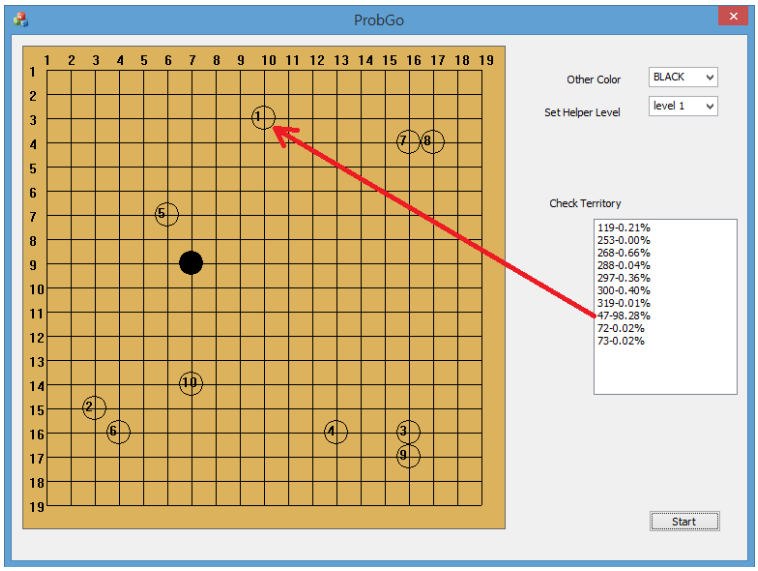


Figure 5. The Interface of Go-Game: Other Player is Black, so it suggests for White Player Only. The Selected Level is 1 (Basic)

In this work, we show the suggestion for only one person optionally and one of three provided levels what people expected to play with. Using the same trained results, the suggestions for both opponent players are possible. In the second way, we can improve our program to play as machine-machine interactions. The table 1 show the particular numbers of board states we used in training progress.

Table 1. Training Data Set in 3 Ranks

Level (Rank)	1 (basic)	2 (Advanced)	3 (Expert)
Number of used board States	155,305	219,911	223,256

4. Results

Using our structure and data from cyber-oro server, after installing and training stages, we have tested our Go-game program and it has worked fluently. All posible options have been made. For instance: changing between BLACK and WHITE player; selecting randomly among 3 levels. The accuracy table of testing progress shows large difference between level 1 and the two others.

Table 2. Accuracy of 3 Given Levels

Level (Rank)	1 (basic)		2 (Advanced)		3 (Expert)	
Color	black	white	black	white	black	white
Accuracy (%)	97.92	95.15	44.18	51.79	47.58	48.78

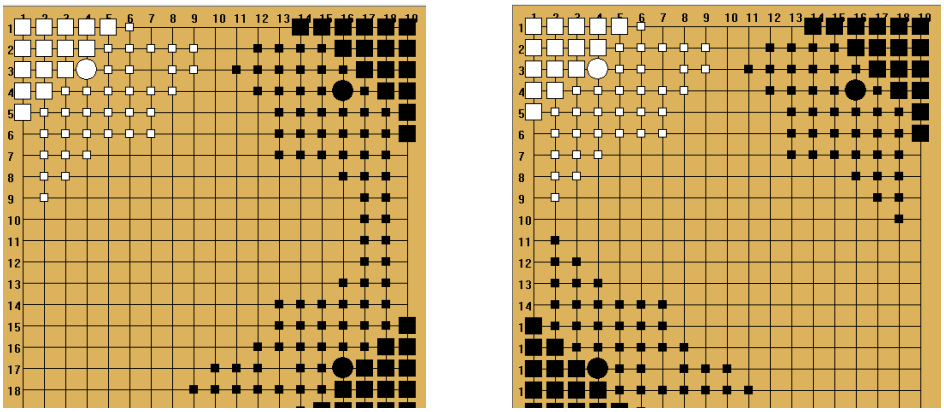
- 1st rank: This level has fewest board-states using for input training. So, it takes shortest time in learning and suggesting next moves for a turn. The accuracy of this level is almost perfect because at this level, people normally make their moves not so intelligent that the machine can win easily.

- 2nd rank: Using more than 200.000 board-states in training for input data and take more time in processing in learning and suggesting next moves but the efficiency is low, only white player (play at second turn) over 50%. At this rank, player can make more intelligent moves. That means machine harder to suggest next moves better than human.

- 3rd rank: Using the largest number of board-states in training for input data and take more time in processing in learning and suggesting next moves but the efficiency is low because at this highest rank, the most Go-game players can choose the optimal moves.

5. Calculating Score Table

Actually, there are many other output results for a turn in Go-game in CNN's suggesting process and many of them are not the garbage with unusable value. Our program try to calculate the 10 best score positions to show in newbie-helping task.



a) Level 1st Expected Value b) Level 3rd Expected Value

Figure 6. Apply the Go Scoring on the Current State in Different Ranks

Figure 6 shows the situation at the third turn (black player). When confirming the score at level 1 and level 3 is not a big difference.

6. Discussion

We have showed that large deep convolutional neural networks can caculate and suggest the next move made by Go-game players. In our work, we seperate into three ranks and our program works very good to help newbie-players. With the results that using CNN can suggest good moves, and it is very well in helping green players to learn playing the Go-game, our work has distributed a new approach in machine learning using CNN. Besides that, the suggestive accuracy translates into much stronger move evaluation and playing strength than has previously been shown. So, the CNN is able to outperform traditional search based programs such as GnuGo, and compete with state-of-the-art MCTS programs such as Pachi and Fuego. This work also proves that we can use CNNs into supervised deep learning field as well as many traditional methods with high effect and performance.

With this new research trend, we can apply this technique into not only Go-Game but many others like GnuGo, Pachi,… and in the common, training data using CNNs can be applied into AI field to develop many other Human Computer Interaction programs and devices with more effects and more performances.

In future, with stronger computing capacity, speed and power, we can use more than 5 hidden layers and get higher accuracy, this method will brings outcome performance. We hope, our new given method can be applied widely in AI field, and many smart devices using this technique will appear in the near future.

References

[1] C. Bowman, M. J-B. Guillaume, Winands, Mark H. M., van den Herik, H. Jaap, Uiterwijk, Jos W.H. M., and Bouzy, Bruno. Progressive strategies for Monte-Carlo tree search. "New Mathematics and Natural Computation", vol. 4, **(2008)**, pp. 343–357.
[2] Clark, Christopher and Storkey, Amos. "Teaching deep convolutional neural networks to play Go". arXiv preprint arXiv:1412.3409, **(2014)**.
[3] Coulom, R´emi. "Efficient selectivity and backup operators in Monte-Carlo tree search". In Computers and games, pp. 72–83. Springer, **(2007)**.
[4] Dean, Jeffrey, Corrado, Greg, Monga, Rajat, Chen, Kai, Devin, Matthieu, Mao, Mark, aurelio Ranzato, Marc', Senior, Andrew, Tucker, Paul, Yang, Ke, Le, Quoc V., and Ng, Andrew Y. Large scale distributed deep networks. In Pereira, F., Burges, C.J.C., Bottou, L., and Weinberger, K.Q. (eds.), "Advances in Neural Information Processing Systems 25", Curran Associates, Inc., **(2012)**, pp. 1223–1231.
[5] Enzenberger, Markus, M¨uller, Martin, Arneson, Broderick, and Segal, "R. Fuego - an open-source framework for board games and Go engine based on monte carlo tree search". IEEE Trans. Comput. Intellig. and AI in Games, vol. 2, no. 4, **(2010)**, pp. 259–270.
[6] Kocsis, Levente and Szepesv´ari, Csaba. "Bandit based Monte-Carlo planning". In Machine Learning: ECML 2006, Springer, **(2006)**, pp. 282–293..
[7] J. Chris, Maddison, Aja Huang, Ilya Sutskever, David Silver conference paper at ICLR 2015, "Move Evaluation in go Using Deep Convolutional Neural Networks". arXiv: 1412.6564v2 [cs.LG], **(2015)**.
[8] Sutskever, Ilya and Nair, Vinod. "Mimicking Go experts with convolutional neural networks". In Artificial Neural Networks-ICANN **2008**, pp. 101–110. Springer, **2008**.
[9] Gelly, S. and Silver, "D. Combining online and offline learning in UCT". In 17th International Conference on Machine Learning, **(2007)**, pp. 273–280.
[10] Krizhevsky, Alex, Sutskever, Ilya, and Hinton, Geoffrey E. "Imagenet classification with deep convolutional neural networks". In Advances in neural information processing systems, **(2012)**, pp. 1097–1105.
[11] N. N Schraudolph, P. Dayan and T. J. Sejnowski, "Temporal difference learning of position evaluation in the game of Go". Advances in Neural Information Processing Systems, **(1994)**, pp. 817–817.
[12] S. Gelly, and Silver, "D. Monte-Carlo tree search and rapid action value estimation in computer Go". Artificial Intelligence, vol. 175, **(2011)**, pp.1856–1875.
[13] Website: "http://ufldl.stanford.edu". Computer Science Department of Stanford University.
[14] Website: "http://cyberoro.com". A Go-game news website of Korea.

