Proceedings of the Twenty-Ninth International
Florida Artificial Intelligence Research Society Conference

Neural Networks Learning the Concept of Influence in Go

Gabriel Machado Santos, Rita Maria Silva Julia, and **Marcos Saito**
Computer Science Department
Universidade Federal de Uberlandia

Matheus Araujo Aguiar
Informatics Department
Instituto Federal do Triangulo Mineiro

Abstract

This paper describes an intelligent agent that uses a MLP (Multi-Layer Perceptron) Neural Network (NN) in order to evaluate a game state in the game of Go based, exclusively, in an influence analysis. The NN learns the concept of Influence, which is domain specific to the game of Go. The learned function is used to evaluate board states in order to predict which player will win the match. The results show that, in later stages, the NN can achieve an accuracy of up to 89.3% when predicting the winner of the game. As future work the authors propose the incorporation of several improvements to the NN and also its integration intelligent player agents for the game of go, such as Fuego and GnuGo.

Introduction

Since the early beginning of artificial intelligence science field there has been the interest to implement player agents for board games like chess and checkers(Shannon 1950)(Schaeffer 1997). The task of implementing automatic players for such games involves both theoretical and technical difficulties which are very similar to those that arise in the treatment of a range of problems of everyday life such as urban traffic control(Walker 2000), human-computer interaction(Thorpe and Anderson 1996) and robot autonomous navigation(Monteiro and Ribeiro 2003)(Alers et al. 2014). These similarities are due to the fact that all these problems demand strategic combinations for making decisions in situations where an opponent aims to minimize the positive effects of the actions of the agent. Furthermore, the involved agents acting within these domains need to learn through their interaction with the environment and each other as well as through the state changes which occur after each single action execution. After the chess match between DEEP BLUE and Kasparov, in 1997, many researchers turned their attention to other games, especially Go(van den Herik 2010). This ancient game has been considered a great challenge for researchers in artificial intelligence and it is interesting to note that the classical methods in game tree search, which have worked so successfully for board games like chess and checkers, haven't been able to achieve the same success in the game of Go(Gelly et al. 2012). Because of that, the domain of Go demands from A.I researchers new techniques and approaches to it. This has led to the development, in recent times, of the Monte-Carlo Tree Search (MCTS)(Coulom 2006), which has allowed a substantial change in the computer Go field (Coulom 2010)(Gelly and Silver 2008). Since that, the performance of player agents has been increasingly improving and nowadays programs based on Monte-Carlo Tree Search play at human-master level and are able to challenge top professional players(Gelly et al. 2012). Even with the success of MCTS, top level of playing has not yet been achieved by artificial go players and other well-known techniques, such as Neural Networks (NN), continue to be topic of study and experimentation. For example, the recent works of (Maddison et al. 2014) and (Clark and Storkey 2014) use NN to predict moves after being trained over a set of professional games. It could be inferred that future improvements of the go players agents will be based not only on the promising MCTS results, but also on other known and yet to be discovered tools of artificial intelligence. Considering this context, this work describes the system NN-Influence, an intelligent program that learns to predict the winner of a Go match. The main concept that the NN learns is called influence and it is domain-specific of the game of Go. The influence represents the advantage or potential that a group of stones yields around its neighbouring points. Influence may become points at the end of a match of Go. Thus, the system is able to learn this basic concept of the game and use it to predict the winner of the match given a board state. Also, the system implements a graphical interface that can show the influence on the points of the board. This tool also gives some statistics about the game, such as territory and final score points estimates, groups liberties and other useful information which is not the topic of this work. The remainder of this document is arranged as follows: the concepts and rules of the game of Go are described in section "The Game of Go", followed by section "Related Work", where related and prior work are presented. Section "NN-Influence" describes the architecture of the system NN-Influence implemented in this work, while in section "Experiments and Results" the experiments and results obtained are presented and discussed. Finally, in section "Conclusions", the achievements and problems encountered in this work are summarized and future works are proposed.

Copyright © 2016, Association for the Advancement of Artificial Intelligence (www.aaai.org). All rights reserved.

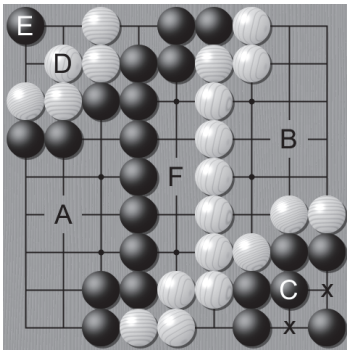

Figure 1: Illustration of a 9x9 Go board configuration after the game has ended.

The Game of Go

The history of Go stretches back some 4,000 years and the game still enjoys a great popularity all over the world(Gelly and Wang 2006). Although its rules are simple, its complexity has frustrated the many attempts to build a good Computer-Go player since the late 70s(Bouzy and Cazenave 2001). The game is played by two players on an empty square board typically in the sizes of 9x9 and 19x19, although other sizes are also possible. Starting with black, each player alternately places a stone of her color (black or white) on an empty point of the board grid. At any moment, a player may opt to skip his turn and not place any stone on the board, in which case it becomes now the turn of the other player. Once the stones are placed on the board, they do not move. Nevertheless, the stones may be removed from the board if they are completely surrounded by enemy stones. In short, the main objective of the game is to surround with stones the most points on the board. The game develops in this fashion until both players pass consecutively. In this case it is assumed that they both agree on the points which they have control of and thus the game is scored and finished. The example of Figure 1 illustrates a board configuration after the game is finished. The black stones have control over the points in the areas **A**, **C** and **E**, while the white stones have control over only the area **B**. Although the white stones marked with **D** seem to be controlling that area, they can be captured because of the presence of the black stone marked with **E**. On the other hand, the black stones marked with **C** may not be captured, because it is impossible for white to play both on the points wich are marked with an **X**. For more information about the rules and dynamics of the game of Go we refer to (Müller 2002) and to the following pages on the Internet: http://gobase.org and http://gogameguru.com.

Related Work

This section briefly presents and describes prior work which are related to the present one. In the early work of (Zobrist 1969), a visual model of the game of go is proposed. The author recognizes the great complexity of the game tree and the difficulty of exploring such a tree with any form of heuristic search. Zobrist also raises the importance of discriminating the different groups of stones and the spheres of influence that are formed by these. The board is represented by a few 19x19 layers of integers, which store information such as the size of the group each stone belongs to, its number of liberties and also under which influence a given point is. The process used by Zobrist computes the influence of a board state by attributing positive or negative values to each position, according to the color of the stone. Then each point receives a value of 1 or -1 from its neighbours. This later process is repeated four times and the values generated are accumulated. The result is considered by Zobrist a "visual organization heuristic". A similar process was more recently proposed in the work of (Bouzy 2003), obtaining successful results. The method applies mathematical morphology to compute influence and territory of a board state. It extends and refines the previous work of Zobrist, allowing that not only influence is calculated but also territories (score points). In order to accomplish this, two operators which are well-known in digital image processing are used, namely, a dilatation operator D_z and a erosion operator E_z. These two operators are combined in the form of an operator $X_z = {E_z}^e$ o ${D_z}^d$, where E_z is iterated *e* times and D_z is iterated d times. In this way, the operator X_z can be used to recognize territories in the game of Go. In regard to approaches that involve neural networks, it is interesting to cite the works of (Schraudolph, Dayan, and Sejnowski 1994) and (Enzenberger 2003). Both works use convolution neural networks (CNN) trained by reinforcement learning from self-play to predict the final territory and they exploit the advantages of the rotational, reflectional, and colour inversion symmetries of the board. In addition to that, *Neurogo* (Enzenberger 2003) use a more sophisticated architecture and could also predict eyes and connectivity of the groups. Also, *Neurogo* was combined with an alpha-beta search, achieving the performance of *GnuGo* (an open-source and benchmark program) on 9x9 board. Later, in the work of (Werf et al. 2003) a two layer NN using a set of hand constructed features was trained by supervised learning of expert moves. The results showed that the NN was able to predict the move an expert would have made with 25% accuracy. Moreover and related to this, the work of (Sutskever and Nair 2008) also uses two-layer CNN trained by supervised learning that outputs the probability an expert player would play in each point of the board. The accuracy achieved was initially 34% and after refinements increased to 37%. More recently, the works of (Maddison et al. 2014) and (Clark and Storkey 2014) both use Deep Convolutional Neural Networks (DCNN) trained by supervised learning on a database of expert games to predict moves. In the former work, it used a technique of tying the weights in the network to take advantage of the symmetries expected to occur on the board. The best NN had 8 layers and was able to 41% of accuracy after training and testing on a dataset of professional games. In the later work, a DCNN with 12 layers also exploited the advantages of the symmetries of the board was used. The results from the work reveal that the NN was able to correctly predict the move of the expert in 55% of the board states. The playing strength of this NN was able to surpass *GNUGo* in 97% of the time and matched the performance of state-of-

the-art Monte-Carlo Tree Search programs as *Pachi*(Baudis and Gailly 2011) and *MoGo*(Gelly and Silver 2007). It is important to note that while the related work presented in this section seek to predict the move of an expert on a given board state, the system NN-Influence, presented in next section, is intended to predict the winner of the match.

NN-Influence

This section presents the system NN-Influence: an intelligent program that uses the strong concept of stone influence, specifically for the game of Go, in order to evaluate a board configuration trying to predict the winner at the end of that match. In order to do that, the system takes into account all influence power computed from a specific board state. Albeit it is, in a broad sense, a simple approach, the concept of Influence is one of the basic aspects of the dynamics of the game and even defines a playing style. Such an analysis, based on the influence, allows for a more concise evaluation which takes into account the global state of the board state, which is a quite difficult task for intelligent agents for the game of Go(Wang and Gelly 2007). The process used by the NN-Influence system to predict an winner for an ongoing match is illustrated in Figure 2.

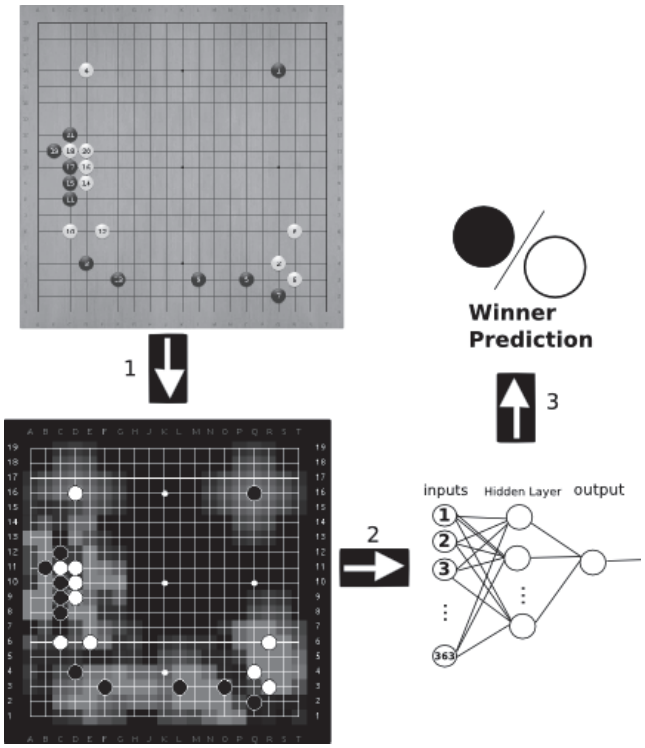

Figure 2: The steps taken in predicting a winner for an ongoing match of the game of Go.

NN-Influence Agent

As depicted in Figure 2, the prediction of a winner for a match is composed of three main tasks:

1. **Influence Transformation**: at a specific moment of the game (after 21 ply moves in Figure 2, for example), the agent will evaluate the board configuration and apply the model of influence used in order to obtain information about the influence projected by each stone on the board.
2. **Feed the Neural Network**: Once the influence transformation is performed, the next task is to feed the neural network, already trained, according to its proper inputs in a specific format.
3. **Interpret the Output**: Finally, after receiving the input, the neural network resumes its task, performing the prediction based exclusively on the features learned during the training process.

Influence Transformation

The process of Influence Transformation consists of taking a board state at a given moment of the game and interpret each stone individually according to an Influence Model. Once the power of influence for each stone is calculated, it has computed the influence value for each intersection point individually, subsequently, generating the influence map, altogether.

Influence Model The influence model is a quantitative representation of the intersection points affected by a single stone. This model corresponds roughly with the manhattan distance(Craw 2010) from the stone projecting influence. Figure 3 shows a representation of a white stone and its influence power according to the cross-influence model. This model is named cross-influence by the authors and it was conceived to reflect common connection relations between stones. For example the *one-space jump* has an influence value of 6, which is greater than the influence value of the *two-space jump*, which is 4.

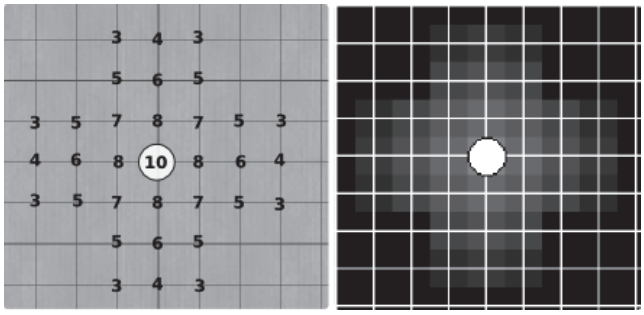

Figure 3: The values of the influence model (left) and its graphical representation in the Influence Goban (right).

As it is show in the Figure 3, the nearer the point is of the stone, the stronger is the influence value attributed. In the same way, the farthest the point is the weaker the influence, until at a certain distance it becomes null. The black stones projects influence values that are positive integers, while white stones projects influence values that are negative integers. Thus, stones from different values project different signed integers of the same magnitude. In this way, if an intersection is at the same distance from a white stone and

a black stone simultaneously, then, for example, the white stone projects a value of -6 and the black stone projects a value of $+6$. The resulting influence value is the sum of all influence taken, which becomes 0 in the previous example. After the influence of every stone in the board is computed, an influence map is generated representing the given board state. Figure 4 shows an example of the generated influence map of a board state.

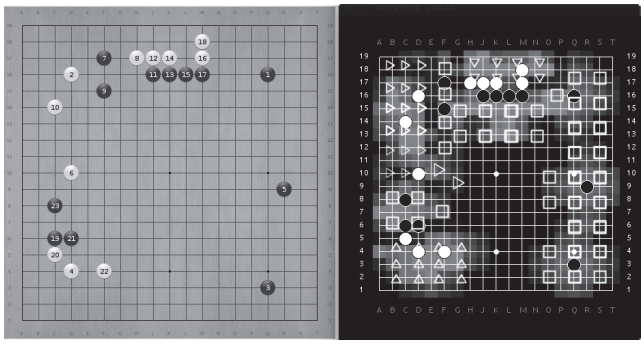

Figure 4: Illustration of a Go board configuration and its influence map.

Feed the Neural Network

Once the board state is interpreted as an influence map, the information encoded in it must be inputed, in a specific format, to the already trained Neural Network module which will, subsequently, predict the player who has the greatest chance of winning.

The Neural Network Architecture The Neural Network implemented in this work is a well known Multilayer Perceptron (MLP)(Bishop 1995) with 363 inputs, one hidden layer composed of 200 neurons and only one neuron in the output layer. The number of neurons of the hidden layer was choosen to be $2/3$ (or 70% to 90%) of the number of neurons in the input layer(Boger and Guterman 1997). The input layer comprises 363 neurons, in which 361 ones correspond to each intersection present in a 19x19 layer, which is the matrix of positions of the Go board. Each one of the 361 inputs is represented as a scalar which is equal to the influence value for each corresponding intersection in the board, as previously explained. There are two more inputs which correspond to the number of black and white prisoners, respectively. All the neurons present in the hidden layer as well as in the output layer use the sigmoid function as activation function, equation 1, where x and y are the input and output value for each neuron, respectively.

$$y \leftarrow 1/(1 + exp(-x)) \tag{1}$$

The Training Process During the training process, two different databases of Go games were used, resulting in almost $160,000$ games:

- The Games of Go on Download (GoGoD): The GoGoD library(GoG) comprises over $82,300$ pro games (as of Winter 2014), all in the SGF format.
- The KGS amateur games: The KGS database(KGS) contains about $77,000$ games from strong amateur players.

For the training of the network, 80% of the whole database ($\sim 128,000$ games) was used, while the remaining 20% ($\sim 32,000$ games) were used as the validation set of the network. The games were randomly choosen to compose both the training and validation set. The process consisted of taking a "*snapshot*" at a specific moment of each training sample board state, interpret it as an influence map and send its respective scalars values to the input layer of the neural network. Also, the input layer receives the number of prisoners taken for each player. For the output layer, as a sigmoid function was used, the values may vary between 0 and 1. The output values equal or below 0.5 are interpreted as victory for white and the output values above 0.5 as victory for the Black. During the training of the neural network it was used a maximum value of $10,000$ epochs and a desired error below of 0.03. For comparison matters, the authors have trained four different networks in order to evaluate the learning power of the agent regarding distinct moments of each matching, as shown in the Experiments section. It is important to remark that the thorough process was repeated for each network trained during the experiments.

Interpret the Output

As soon as the network receives all the inputs regarding a particular board state, it will compute, as a result, a value Y where $\{Y|0 < Y < 1\}$. This is interpreted as follows:

1. If $Y \leq 0.5$, then the White player has more probability of winning that very match according to the current influence power.
2. Otherwise, the power balance is more favorable to the Black player, who, consequently, has better chances of victory in that match.

This estimative allows for a player to scrutinize if a specific move increases or not its chances of winning a match. In other words, a player may simulate a move and check whether its probability of winning increases or decreases. It is important to highlight that this process may be used in the future as a move generator for an automatic playing agent.

Agent Details

The agent was completely implemented in the C++ language, using the Fast Artificial Neural Network library (FANN) for the Neural Network module. As explained previously, the goal of the system is to predict the winner of a match. Nevertheless, in order to accomplish that, the agent generates an influence map, as well as gathers statistics about the groups of stones such as liberties and territory estimative. The information that is encoded in the influence map can be better visualized by humans if shown graphically over the board. Thus, the system NN-influence has a graphic interface where the influence can be visualized in terms of colors over the areas of the board. This is illustrated next in Figure 5.

In the figure, the triangles indicate the areas that are colored as white influence, while the squares indicate the areas that are colored as black influence.

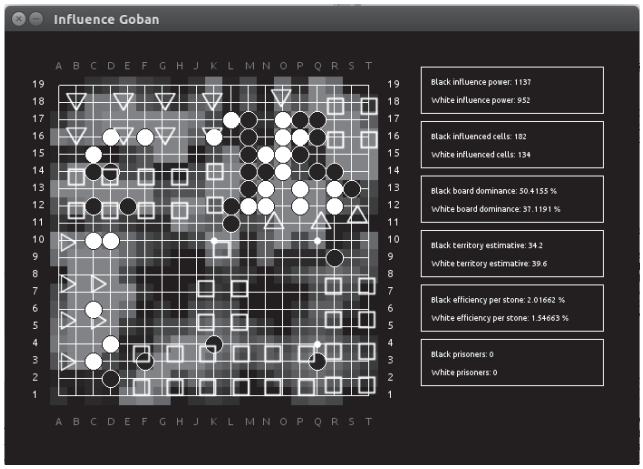

Figure 5: The Influence Goban Program: an additional tool created in order to help visualize the influence map for a game of Go.

Experiments and Results

This section presents all the test scenarios performed in order to evaluate the performance of the NN-Influence agent. First off, it is important to remark that in a Go match, even for strong humans players, is a quite harsh task to predict a winner depending on the game phase. Obviously, as soon as the match advances it becomes clear the player who has more advantage in the game. In this sense, as shown in this section, the more the match advances the more assertive is the agent predictions. The goal of the authors, in this paper, is to find the earlier moment in the game that comprises a good correctness rate of prediction, so an automatic agent based on MCTS simulations, for instance, could benefit from that approach by combining the power of the predictions to stop the simulations in the play-out phase as soon as possible, which could allow for it to perform more simulations during the search process, increasing the agent power. It is important to mention that a 19x19 Go game has an average length of 200 moves. In this sense, the authors have performed tests in four different scenarios:

I Phase 30: in this scenario, a snapshot was taken of the board around the 30th move for each game in the training and the validation set. Through this scenario it is possible to evaluate the learning power of the network for opening phases of the game.

II Phase 80: a snapshot of the board at the 80th move for each game in the training and validation set. This phase represents the early middle-game.

III Phase 130: a snapshot of the board at the 130th move for each game in the training and validation set. This phase represents a final middle-game phase.

IV Phase 180: it represents an endgame phase, also known as *yose*, and starts around the 170th move.

The processor used for training and executing the agent is an Intel Core 2 Quad 2.4 GHz with 8GB of RAM. The results for all scenarios are shown in Table 1. The success rate represents the percentage of correct predictions made over the validation set (around $32,000$ games), after each network related to its specific scenario has been trained.

Table 1: The success rate in predicting a winner, based on influence power, for the game of Go in four different phases.

Test Scenario	Success Rate
I - Phase 30	52.1%
II - Phase 80	61.8%
III - Phase 130	76.7%
IV - Phase 180	89.3%

Table 1 shows the results for each test scenario. The results show that is clear that the later the moment in the game, the better the success rate of the agent. Professional and strong amateur games are, in a broad sense, quite balanced since the players are experts in the game. This fact hinders the success of predictions when the game is in the opening phase, once it is pretty difficult to say if a player has a much stronger position. As it can be seen, for Test Scenario I (phase 30 in the game) the agent behaves quite random, succeeding in just 52.1% of the games used in the database. The results improve slightly for Test Scenario II (phase 80), where the success rate comes to 61.8%. In Test Scenario III (phase 130), it is observed an improvement of almost 15%, raising the success rate to 76.7%. Although it is not an opening, it is still a good point in the game, since the match is still in the middle phase. For the Test Scenario IV, the endgame phase, the agent has shown a success rate of 89.3%, which is an excellent outcome. However, as previously mentioned, it is a bit easier to predict a winner in the endgame phase, as long as the majority of the areas in the board are already taken and, in most of the cases, one player has already greater influence than the other one. Despite the random behavior and the poor learning power presented by the agent in the opening phase, the results confirm that the usage of an influence model combined with the learning power of a MLP Neural Network is a good approach in the task of predicting a winner for the game of Go.

Conclusions

It was presented in this work the NN-Influence Agent, a program that uses an influence model, specific for the game of Go, and a MLP Neural Network in order to predict the winner of a match based exclusively on its board dominance aspect, also called influence power. The authors had two motivations (and goals) in creating such a tool: (1) Create a tool capable of graphically indicating the dominance limits in a board game of an ongoing match in addition to have a intelligent system capable of evaluating the decision made (by indicating the probability of victory) based on its influence power and (2) create an evaluation function, that could receive as parameter a board state and return who is the player with more chances of winning in that specific situation. Although it is conspicuous that the task of effectively predicting a winner in the early opening phase is still a harsh

challenge for all researchers in the field, the results show that the authors have accomplished some positive effects regarding their two goals. In addition, they have also developed a tool in order to help visualize the influence power of each stone in a board, which is released under the LGPL license and available at http://bitbucket.org/goresearchers/go_tools.git. As further developments, the authors will implement a DCNN and compare the results with the current approach. In a second moment, the authors will also implement the approach here presented into intelligent agents that use the MCTS algorithm as main technique and compare the performance with other well-known agents. Finally, they will scrutinize different models of influence for the game, study different approaches to use the prediction, such as computing the delta between the NN's prediction and the actual move played and also create more indicators of performance for a player during the game.

References

Alers, S.; Claes, D.; Fossel, J.; Hennes, D.; and Tuyls, K. 2014. Applied robotics: Precision placement in robocup@work. In *Proceedings of the 2014 International Conference on Autonomous Agents and Multi-agent Systems*, AAMAS '14, 1681–1682. International Foundation for Autonomous Agents and Multiagent Systems.

Baudis, P., and Gailly, J.-L. 2011. Pachi: State of the art open source go program. In *ACG*, volume 7168 of *Lecture Notes in Computer Science*, 24–38. Springer.

Bishop, C. M. 1995. *Neural networks for pattern recognition*. Oxford university press.

Boger, Z., and Guterman, H. 1997. Knowledge extraction from artificial neural network models. In *1997 IEEE International Conference on Systems, Man, and Cybernetics*, volume 4, 3030–3035.

Bouzy, B., and Cazenave, T. 2001. Computer go: an ai oriented survey. *Artificial Intelligence* 132:39–103.

Bouzy, B. 2003. Mathematical morphology applied to computer go. *IJPRAI* 17(2):257–268.

Clark, C., and Storkey, A. J. 2014. Teaching deep convolutional neural networks to play go. *CoRR* abs/1412.3409.

Coulom, R. 2006. Efficient selectivity and backup operators in monte-carlo tree search. *Proceedings of the 5th international conference on Computers and games* 72–83.

Coulom, R. 2010. Computing "elo ratings" of move patterns in the game of go. *ICGA Journal* 30(4):198–208.

Craw, S. 2010. Manhattan distance. In Sammut, C., and Webb, G., eds., *Encyclopedia of Machine Learning*. Springer US. 639–639.

Enzenberger, M. 2003. Evaluation in go by a neural network using soft segmentation. *10th advances in computer games conference* 97–108.

Gelly, S., and Silver, D. 2007. Combining online and offline knowledge in uct. 273–280.

Gelly, S., and Silver, D. 2008. Achieving master level play in 9 x 9 computer go. In *AAAI*, 1537–1540. AAAI Press.

Gelly, S., and Wang, Y. 2006. Exploration exploitation in go: Uct for monte-carlo go. in: Nips-2006: On-line trading of exploration and exploitation workshop. In *In Twentieth Annual Conference on Neural Information Processing Systems (NIPS)*.

Gelly, S.; Kocsis, L.; Schoenauer, M.; Sebag, M.; Silver, D.; Szepesvári, C.; and Teytaud, O. 2012. The grand challenge of computer go: Monte carlo tree search and extensions. *Commun. ACM* 55(3):106–113.

GoGoD - Games of Go on Download. http://gogodonline.co.uk. [Online].

KGS - KGS Go Server. https://www.gokgs.com/. [Online].

Maddison, C. J.; Huang, A.; Sutskever, I.; and Silver, D. 2014. Move evaluation in go using deep convolutional neural networks. *CoRR* abs/1412.6564.

Monteiro, S. T., and Ribeiro, C. 2003. Aprendizagem da navegacao em robos moveis a partir de mapas obtidos autonomamente. In *Anais do XXIII Congresso da Sociedade Brasileira de Computacao*, volume 1, 152–162.

Müller, M. 2002. Computer go. *Artificial Intelligence* 134(1-2):145–179.

Schaeffer, J. 1997. *One Jump Ahead: Challenging Human Supremacy in Checkers*. New York, NY, USA: Springer-Verlag New York, Inc.

Schraudolph, N. N.; Dayan, P.; and Sejnowski, T. J. 1994. Temporal difference learning of position evaluation in the game of go. volume 6, 817–824. Morgan Kaufmann, San Francisco.

Shannon, C. E. 1950. Programming a computer for playing chess. *Philosophical Magazine* (314):256–275.

Sutskever, I., and Nair, V. 2008. Mimicking go experts with convolutional neural networks. In *Proceedings of the 18th International Conference on Artificial Neural Networks, Part II*, ICANN '08, 101–110. Berlin, Heidelberg: Springer-Verlag.

Thorpe, L., and Anderson, C. W. 1996. Traffic light control using sarsa with three state representations. *Technical Report, IBM Corporation*.

van den Herik, H. J. 2010. The drosophila revisited. *ICGA Journal* 33(2):65–66.

Walker, M. A. 2000. An application of reinforcement learning to dialogue strategy in a spoken dialogue system for email. *Artificial Intelligence Research 12* 387–416.

Wang, Y., and Gelly, S. 2007. Modifications of uct and sequence-like simulations for monte-carlo go. In *CIG*, 175–182. IEEE.

Werf, E. V. D.; Uiterwijk, J.; Postma, E.; and Herik, J. V. D. 2003. Local move prediction in go.

Zobrist, A. L. 1969. A model of visual organization for the game of GO. In *American Federation of Information Processing Societies: AFIPS Conference Proceedings*, 103–112.