ECAI 2020
G.D. Giacomo et al. (Eds.)

doi:10.3233/FAIA200119

SAI: A Sensible Artificial Intelligence That Plays with Handicap and Targets High Scores in 9×9 Go

F. Morandin[1] and **G. Amato**[2] and **M. Fantozzi**[3] and **R. Gini**[4] and **C. Metta**[5] and **M. Parton**[6]

Abstract. We develop a new framework for the game of Go to target a high score, and thus a perfect play. We integrate this framework into the Monte Carlo tree search – policy iteration learning pipeline introduced by Google DeepMind with AlphaGo. Training on 9×9 Go produces a superhuman Go player, thus proving that this framework is stable and robust. We show that this player can be used to effectively play with both positional and score handicap. We develop a family of agents that can target high scores against any opponent, recover from very severe disadvantage against weak opponents, and avoid suboptimal moves.

1 Introduction

The game of Go has been a landmark challenge for AI research since its very beginning. It is no surprise that DeepMind first major effort and achievement was [15, AlphaGo], an AI that plays Go at superhuman level. It is nevertheless quite surprising that the approach for this achievement works even better without human knowledge [17, AlphaGo Zero] and that it is universal enough to be applied successfully to Chess and Shogi [16, AlphaZero].

However, in the game of Go, maximizing the final score difference and the related abilities of playing with positional or score handicap is still an open and important question. AlphaGo is known to play suboptimal moves in the endgame, see for instance [18, moves 210 and 214, page 252], and in general many games in [18] not ending by resignation. This phenomenon is rooted in the win/lose reward implemented in the Deep Reinforcement Learning (DRL) pipeline of AlphaGo. Score is unlikely to be a successful reward, because a single point difference may change the winner, thus inducing instability in the training.

Efforts in the direction of score maximization have been made in [1] and in [11]. However, these attempts do not use any of the modern DRL techniques, and thus their accuracy is quite low. One DRL paper we are aware of is [21], where a Deep Convolutional Neural Network is used to predict the final score and 41 different winrates, corresponding to 41 different scores handicap $\{-20, -19, \dots, 0, \dots, 19, 20\}$. However, their results have not been validated against human professional-level players. Moreover, one single self-play training game is used to train 41 winrates, which is not a robust approach. Finally, in [20, KataGo] the author introduces a heavily modified implementation of AlphaGo Zero, which includes score estimation, among many innovative features. The value to be maximized is then a linear combination of winrate and expectation of a nonlinear function of the score. This could be a promising approach.

In this paper we present a new framework, called Sensible Artificial Intelligence (SAI), which addresses the above-mentioned issues through a novel modification of the AlphaGo framework: the winning probability is modeled as a 2-parameters sigmoid function, making explicit the dependence on the targeted score. Several self-play training games are branched, reinforcing robustness of this model, see Section 2.2. We first introduced this framework in a previous work [14], where we conducted a proof-of-concept in the toy example of 7×7 Go. In this paper we could exploit the 9×9 setting to prove that the results promised in the 7×7 proof-of-concept have been indeed achieved.

The implementation of the SAI framework [7] has been realized as a fork of Leela Zero [6], an open source clean room implementation of AlphaGo Zero. We performed two runs of the learning pipeline, from a random network to very strong play on the 9×9 board, using two PCs with entry-level GPUs, see Section 3.

In Section 4 we apply massive simulations to show that SAI is able to play Go with both positional and score handicap and can maximize the final score. Moreover, SAI can leverage on its strength to recover from very severe disadvantage when playing against weaker opponents. All these features have been validated against human players: SAI won against a professional player and against a professional-level player with substantial handicap, therefore showing superhuman strength and additional ability to win with handicap in games with humans. Finally, human validation shows that SAI minimizes suboptimal moves when compared with Leela Zero.

2 The SAI framework

2.1 Modeling winrate as a sigmoid function of bonus points

In the AlphaGo family the winning probability (or expected winrate) τ of the current player depends on the game state s. In our framework, we include an additional dependence on the number x of possible bonus points for the current player: in this way, trying to win by n points is equivalent to play trying to maximize the winrate in $x = -n$. We modeled $\tau_s(x)$ with a two-parameters sigmoid function, as follows:

$$\tau_s(x) := \frac{1}{1 + \exp(-\beta_s(\alpha_s + x))} \qquad (1)$$

The number α_s is a shift parameter: since $\tau_s(-\alpha_s) = 1/2$, it represents the expected difference of points on the board from the perspective of the current player. The number β_s is a scale parameter:

[1] Università di Parma, Italy, email: francesco.morandin@unipr.it
[2] Università di Chieti–Pescara, Italy, email: gianluca.amato@unich.it
[3] email: marco.fantozzi@gmail.com
[4] Agenzia Regionale di Sanità della Toscana, Italy, email: rosa.gini@ars.toscana.it
[5] Università di Firenze, Italy, email: carlo.metta@gmail.com
[6] Università di Chieti–Pescara, Italy, email: maurizio.parton@unich.it

the higher it is, the steeper is the sigmoid, the higher the confidence that α_s is a good estimate of the difference in points, irrespective of the future moves.

AlphaGo and derivatives all share the same core structure, with neural networks that for every state s provide a probability distribution p_s over the possible moves (the *policy*), trained as to choose the most promising moves for searching the tree of subsequent positions, and a real number $v_s \in [0,1]$ (the *value*), trained to estimate the winning probability for the current player.

In our framework, the neural network was modified to estimate, beyond the usual policy p_s, the two parameters α_s and β_s of the sigmoid, instead of v_s. The winning probability may be computed as $\tau_s(k_s)$ where $k_s = \pm k$ is the *signed komi*, i.e., the bonus points of the current player (if it is negative we often use the term *malus*). Rules generally assign a komi of $k = 7.5$ to white player, to compensate for the disadvantage of playing for second. So the sign of k_s depends on the color of the current player at s.

2.2 Branching from intermediate positions

In order to train the two sigmoid parameters for each position, we relaxed the habit of starting all training games from the initial empty board position, and sometimes branched games at a certain state s, changing the komi of the branch according to the value of α_s. In this way, we generated fragments of games with natural balanced situations but a wide range of komi values. This reinforces robustness of the model. Only a sample of all the possible positions were branched, nevertheless the network was able to generalize from this sample and obtain sharp estimates of α_s, with high values of β_s, for positions near the end of the game.

2.3 Parametric family of value functions

In Leela Zero's Monte Carlo tree search, every playout that reaches a state s then chooses among the possible actions a (identified with children nodes) according to the policy $p_s(a)$ and to the evaluation $Q(s,a)$ of the winrate. The latter is the *average* of the value $v_r(s)$ over the visited states r inside the subtree rooted at a; the quantity $v_r(s)$ is the value at r from the point of view of the current player at s, so $v_r(s) = v_r$ or $1 - v_r$ depending on whether the current player at s is the same as at r or the other one. The choice between the actions is done according to AlphaGo Zero UCT formula, see [17].

In our framework, the value function formally equivalent to $v_r(s)$ is the value $\tau_r(k_s)$ of the sigmoid (1). We designed an additional parametric family of value functions $\nu_r(s) = \nu_r^{\lambda,\mu}(s)$, $\lambda \geq \mu \in [0,1]$ computed as

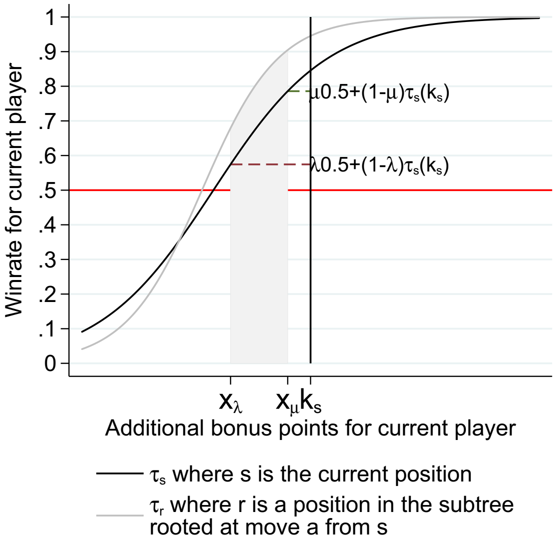
$$\nu_r^{\lambda,\mu}(s) := \begin{cases} \frac{1}{x_\lambda - x_\mu} \int_{x_\mu}^{x_\lambda} \tau_r(u)du & \lambda > \mu \\ \tau_r(x_\lambda) & \lambda = \mu \end{cases}$$

with x_λ and x_μ pre-images via τ_s of convex combinations between $\tau_s(k_s)$ and 0.5, i.e.:

$$x_\theta := \tau_s^{-1}\big(\theta \cdot 0.5 + (1-\theta)\tau_s(k_s)\big), \qquad \theta = \lambda, \mu \tag{2}$$

so that for example $x_0 = k_s$ is the authentic bonus for the current player, and $x_1 = -\alpha_s$ is the virtual bonus that would make the game position balanced. Here, x_λ and x_μ (and hence $\nu_r(s)$) are computed according to the evaluation τ_s at the root node s, so that the integral averages $\nu_r(s)$ entering in the averages $Q(s,a)$ in order to be compared, are all done on the same interval. See Figure 1.

Figure 1. If s is a state, a a possible move from s, and r a position in the subtree rooted at a, the value $\nu_r^{\lambda,\mu}(s)$ is the integral average of τ_r between x_λ and x_μ, where x_λ and x_μ are determined according to (2).



We remark that for $\lambda > 0$ and $\mu \in [0, \lambda]$, $\nu_r^{\lambda,\mu}(s)$ under-estimates or over-estimates the winning probability, according to whether the player's winrate is above or below 0.5. In the extreme scenario $\lambda = \mu = 1$, the agent $\nu^{1,1}$ would always believe to be in a perfectly balanced situation. Thus, it would try to grab every single point, resulting in a greedy score-maximizing agent.

As we will show, when adopted, the parametric family $\nu^{\lambda,\mu}$ is instrumental in pushing SAI towards higher scores.

2.4 Results from the 7×7 proof-of-concept

In our previous work [14] we showed that our framework can successfully be applied to 7×7 Go. In particular the double value head estimated correctly both parameters in final and nearly final positions, with $\alpha_s + k_s$ typically within 0.5 points from the real score difference and $\beta > 5$. This is not trivial, since the target to which the value head is trained is not a pair (α, β), but the boolean game outcome at one single komi value, and in fact we couldn't get this result without the branching mechanism, which seems quite necessary to this end. The present work shows that this was not due to the relative small number of positions on 7×7 Go.

One peculiar result in [14] was that the sequences of moves chosen by the policy at the very beginning of each game correspond to what is believed to be the *perfect play* (there are tentative solutions of 7×7 Go by strong professional players, see [3, 22]). Moreover the estimated parameters for the initial position were $\alpha_\varnothing \approx 9$ and $\beta_\varnothing > 2$, suggesting that the nets properly identified the fair komi of 9 points, and could win with 95% confidence if given just 0.5 bonus points from it. We believe that the play at 250 visits was perfect or very close to perfect, and the fact that almost perfect play can actually be learnt on 7×7 Go, made the possibility of generalization to larger sizes at least dubious.

In this work we exploit the setting of 9×9 Go. This allows us to challenge our assumptions and prove that our framework is effective.

3 Methods of this paper

3.1 Training SAI

We performed two runs of 9×9 SAI. The process we implemented is similar to what was done in Leela Zero [6], with a sequence of *generations*, each one with a single network doing self-play games, beginning with a random net. Differently from Leela Zero, and following AlphaZero [16], in our setting there is no *gating*, meaning that after a fixed number of games the generation ends and a newly trained net is automatically promoted, without testing that it wins against the previous one. Around 2,000 self-plays for generations were enough to get a fast and stable learning (with the possible exception of the first 3-4 generations).

Each training was performed on the self-play games data of a variable number of generations, ranging from 4 to 20, inversely proportional to the speed of the changes in the nets from generation to generation, so that in the training buffer there would not be contradictory information.

In each generation, the proportion of complete games to branches was 2:1. Complete games always spanned several komi values, chosen in $\frac{1}{2}\mathbb{Z}$ with a distribution obtained by interpreting the sigmoid $\tau_\varnothing$ (of the current net, for the empty board) as a cumulative distribution function. Branches were originated from random positions s (each position in a game or branch had the same probability $p = 0.02$ of originating a new branch) and new komi set equal to $\pm\alpha_s$ (rounded to half an integer) with the sign appropriate for the color of the current player, so that the starting position s with the new komi would be estimated as fair for the two players.

The training hyperparameters changed several times during the two runs, with typical training rate 0.0001, batch size 512 and 4,000–10,000 training steps per generation. In the second run we experimented with a kind of "weak gating", in the sense that for every generation during the training we exported 10 networks, at regular intervals of steps, and then match every one against the previous network, finally promoting the one with the best performance. It is unclear if this choice improves learning, but it seems to reduce strength oscillations.

3.2 Network structure

The structure of the neural networks was always the same during the runs, though with different sizes. The input is formed by 17 bitplanes of size 9×9: one is constant, with 1 in all intersections (useful for the network to be aware of borders, thanks to the zero-padding of subsequent convolutions). The remaining planes hold 4 different features for the last 4 positions in the game: current player stones, opponent stones, illegal moves, last liberties of groups. The first layer is a 3×3 convolutional layer with k filters, followed by batch normalization and ReLU. Then there is a tower of n identical blocks, each one a 3×3 residual convolutional layer with k filters followed by batch normalization and ReLU. On top of that there are the two heads. The policy head is composed by a 1×1 convolutional layer with 2 filters, batch normalization and ReLU, followed by a dense layer with 82 outputs, one per move, and then softmax. The value head starts with a 1×1 convolutional layer with 3 or 2 filters (first and second run respectively), batch normalization and ReLU, on top of which there are two almost identical sub-heads, for α and β. Both sub-heads are composed by two dense layers. The first layer has 384 or 256 outputs (for α or β sub-heads), and is followed by ReLU. The second layer has just 1 output. The β sub-head is concluded by computing the exponential of the last output.

The loss function is the sum of three terms: an l^2 regularization term, the cross-entropy loss between the visits proportion and the network estimate of the policy, and the mean-squared error loss between the game result and the winrate estimate $\hat{\tau}_s(k_s)$, where k_s is the signed komi and $\hat{\tau}_s$ is the sigmoid with parameters $\hat{\alpha}_s$ and $\hat{\beta}_s$ as estimated by the network.

3.3 Scaling up complexity

Since with limited computational resources the training of 9×9 SAI is very long, we decided to start the process with simplified settings, scaling up afterwards as the performance stalled. This approach was introduced with success in Leela Zero, by increasing progressively the network size. We observed that one could also progressively increase the number v of visits, as very small values are more efficient at the beginning, while very large values may be needed for getting to optimal play in the end [14].

In the first run we started with $n = 4$, $k = 128$ and $v = 100$ and progressively increased visits to a maximum value of $v = 850$. Then we started increasing the network size to a maximum of $n = 8$, $k = 160$. In total there were 690 generations, equivalent to about 1.5 million games, 70 million moves and 25 billion nodes. In the second run we tried to keep the network structure large and fixed at $k = 256$, $n = 12$ and scaled only the visits, starting from a lower value of $v = 25$ and going up to $v = 400$ in 300 generations.

3.4 Elo evaluation and training outcome

Every run of a project like Leela Zero or SAI yields hundreds of neural networks, of substantially growing strength, but also with oscillations and "rock-paper-scissors" triples. It is then quite challenging to give them absolute numerical scores to measure their performance.

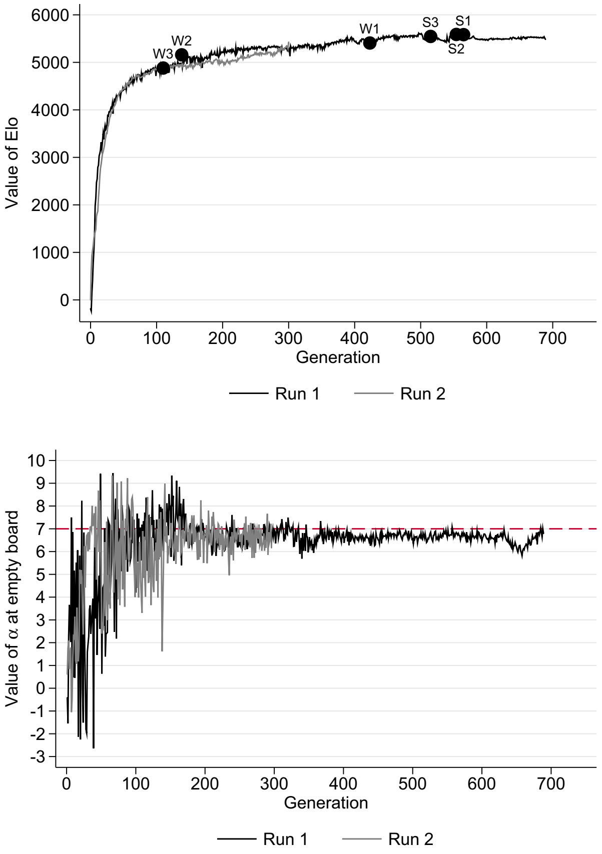
The standard accepted metric for human players is the Elo rating [5, Section 8.4]. In order to get global estimates of the networks strengths, following [19], we confronted every network against several others, of comparable ability, obtaining a graph of pairings with about 1,000 nodes and 13,000 edges. We implemented the maximum likelihood estimator, in a way similar to the classic Bayesian Elo Rating [2], but with a specialization for dealing with draws in the game of Go, which are possible in our framework.

Figure 2 shows the Elo rating of the networks of both runs, anchored to 0 for the random network. It is apparent that the growth is very fast in the beginning, ranging from random play to a good amateur-level playing strength. Elo rating seems less able to express the subtle differences in the playing style and game awareness of more mature networks. In fact our experiments show for example that the very similar ratings of nets S1, S2 and S3 are the overall result of different patterns of winrates against other networks.

3.5 Fair komi for 9×9 Go

A valuable byproduct of the two runs is that we got an estimate of the bonus points for the second player that makes the perfect game a tie, that is, *fair komi*. There are speculations on this subject in the Go community and a general agreement that, with Chinese scoring, a komi of 7.5 should be quite balanced. Figure 2 shows that SAI believes that the fair komi should be 7 points. This is confirmed by both runs, despite the fact that the final networks of the two runs have different preferences for the first moves.

Figure 2. On the upper graph, estimate of the Elo rating as function of the generation for the two runs of 9×9 SAI. The nets S1, S2, S3, W1, W2, W3 are described in Section 3.7. On the lower graph, evolution of the estimates $\hat{\alpha}_{\varnothing}$ of the initial empty board position for the two runs of SAI.



3.6 Score in a Go game

For our experiments we needed a precise evaluation of the score of a finished game, even if it ended by resignation at a fairly early position. But it is difficult to implement an algorithm to compute exactly the score of such a game, as it would in practice require to conduct the game up to the point where scoring can be applied, and this leads to a painstakingly long final part, after the winner is already decided.

In this work we exploited the features of SAI itself to estimate score. The first choice could have been to have a standard strong SAI network compute α_s on the position s when the loser resigns. However we realized after some experiments that this estimate is fairly unstable, in particular when β_s is low, and decided to make it more precise, by aggregating information from a large sample of positions in the subtree of visited nodes rooted at s. This is the same principle introduced by AlphaGo Zero to assess the winrate, but instead of the average, we chose the median, which proved to be stable when based on 1,000 visits. The algorithm was validated by an expert player on a set of 20 games.

3.7 The experimental setting

In order to conduct our experiments, we selected a set of strong and a set of progressively weaker nets (see Figure 2). According to a qualitative assessment, W3 is stronger than a strong amateur, but is not at professional level. To calibrate the nets, we had each of the strong nets play against itself and against all the weak nets 100 times, half times with black and half times with white, with komi 7.5 and with 1,000 visits. As expected, each net won against itself around half of the times, although it would win more often with white, consistently with the assessment that komi 7.5 is unbalanced in favor of White (see Subsection 3.5). Strong nets won 71-73% of the times against W1, 73-90% against W2 and 96-98% against W3. Since the results on 100 games showed a little too much variability, in the next experiments we played 400 games for each setting, in order to reduce the uncertainty by half. The code of the experiment, as well as the files of the games in Smart Game Format, is available at the link [8], and the datasets containing the results underlying the figures of the next section is available at the link [9].

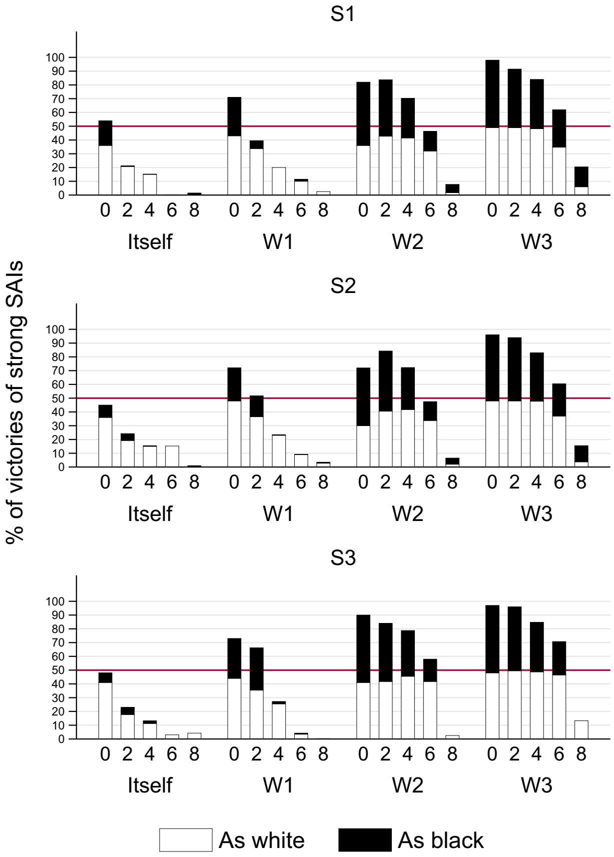
4 Results

4.1 SAI can play with handicap

We conducted two experiments to prove that SAI is able to play with handicap. In the first experiment, we had each of the strong nets play against itself and against all the weak nets, half times with black and half times with white, with an increasing level of komi. The result is shown in Figure 3, where komi is represented in relative terms with respect to an even game of 7.5, i.e. "2" means 5.5 if the strong net is white, and 9.5 if the strong net is black.

When playing against themselves and against W1 as white, the strong nets were able to win a sensible share of games with up to 6

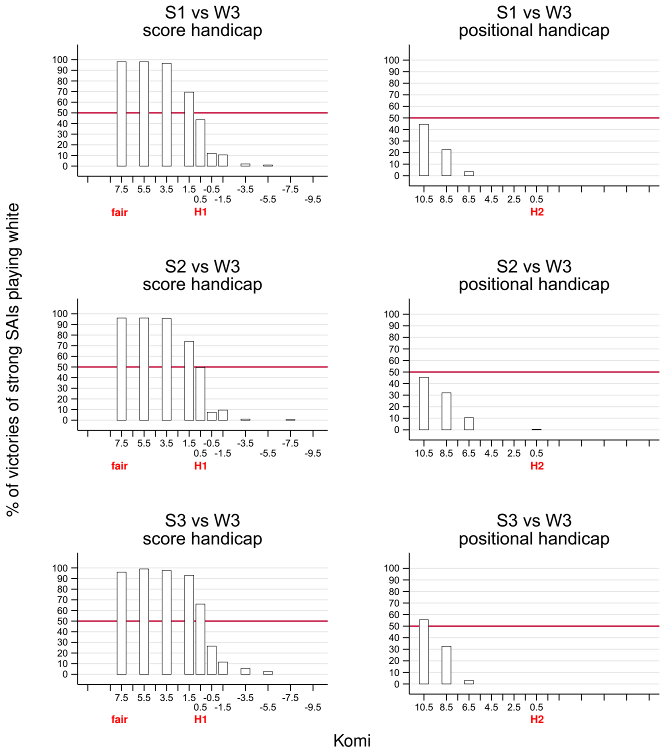
Figure 3. Games of the three strong nets versus themselves and each of the three weak nets, with komi increasingly disadvantageous for the strong net (in relative terms with respect to 7.5).



points of unfavorable komi, while as black their share of victories decreased to less than 2% on average at 4 points. Against the two weaker nets, all the strong nets kept an average share of victories higher than 70% with white, and a reasonably high share of victories with black (higher than 29% on average), when playing with up to 6 points of malus. With 8 malus points, the strong nets could still win an appreciable share of times (5.6% and 16.4%), with both colors.

In the second experiment, we pushed the handicap further, having the three strong nets play white against W3 with additional levels of komi and with the "traditional" concept of handicap: H1 is having no komi (actually, 0.5, to avoid ties), H2 is having white start the game but with two black stones placed on the board. To make H2 less prohibitive, we countered it with komi points in favor of white. The result is shown in Figure 4, where komi is expressed in absolute points. As expected, H2 proved prohibitive, even though in one single game S2 was able to win. When H2 was played with 6.5 komi, in few cases strong nets were able to recover, and with 8.5 the setting was challenging again for W3. With 10.5 the game was again approximately even.

Figure 4. Games of the three strong nets as white versus W3, with various levels of handicap. Komi is expressed in absolute terms. Score handicap (left): the disadvantage is malus points, and H1 means komi 0.5. Positional handicap (right): the starting board contains 2 black stones and white plays first, and H2 means komi 0.5. Since the strong nets play white, lower komi points (from left to right on the x-axis) mean higher disadvantage.

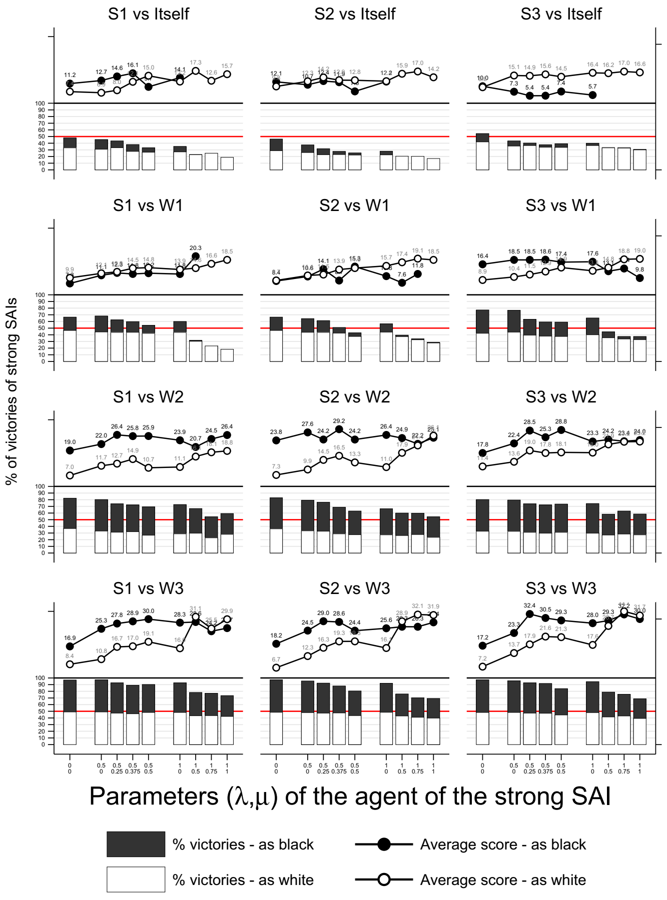


4.2 SAI can target high scores

SAI nets can play with different agents, according to which of the value functions introduced in Section 2.3 is used. We expected that SAI, when playing with agents with high parameters, due to the systematic underestimation of the number of advantage points, would have targeted high margins of victories. We also expected that this would come at the price of reducing its winning probability, due to two composite effects: on the one hand, when playing against a strong opponent, SAI would overlook more solid moves, towards moves riskier but with a higher reward; on the other, when being in disadvantage, the agent would provide SAI with a delusional overestimate of its situation, therefore jeopardizing its choices.

To explore those effects, we had the three strong nets play against themselves and against the three weak nets with a family of agents: λ equal to 0.5 or 1, and μ being in turn 0, 0.5λ, 0.75λ, and λ. The result is shown in Figure 5.

Figure 5. Games of the three strong nets versus themselves and each of the three weak nets, with agents adopting different value functions parameterized by λ and μ. In each subfigure, the lower part shows the winning probability of the strong net, the upper part shows the average final score of the games won by the strong net.



As expected, the winning probability of the strong nets when playing against themselves and against W1 decreased rapidly with increasing λ, especially for high values of μ and when playing as black. However, the average score when winning as white, which was around 10 points at baseline with $\lambda = \mu = 0$, showed a remarkable improvement, up to +7.6 points on average when playing against itself with $\lambda = 1$ and $\mu = 0.5$, and up to +9.5 points on average when playing against W1 with $\lambda = \mu = 1$. The loss of strength against W2 and W3 was not so apparent, even for high values of λ and μ. On the other hand, the improvement in score was substantial.

We then reasoned that the potential of the family of value functions would be better exploited if the choice of the value function was targeted to the situation in the game, for instance changing with the winning probability or with the strength of the opponent. Therefore we tested a *variable agent* by having the strong nets play against themselves and against W3 using three agents ($(0.5, 0)$, $(1, 0)$ and $(1, 0.5)$) which were only activated if the winrate was higher than a threshold of, in turn, 50% and 70%.

The result of the games between strong nets and themselves (and between strong nets and W3) is shown in Figure 6. The threshold effectively eliminated or contained the loss in strength. The gains in score were maintained or increased with thresholds, especially for the 50% case. We tested whether loss in strength and increase in score were significant on average across the three strong nets. The results are reported in in Table 1. Loss in strength as white was more often significant when the strong net was playing against itself than when it was playing against W3; loss in strength as black was not significant, except when $(\lambda, \mu) = (1, 0.5)$. The increase in score was always statistically significant, but was more substantial with threshold 50%.

Figure 6. Games of the 3 strong nets versus themselves and W3, with agents adopting different value functions parameterized by λ and μ, when the pointwise estimate of victory is above a pre-defined threshold of 0%, 50% and 70%. In each subfigure, the lower part shows the winning probability of the strong net with variable agent, the upper part shows the average final score of the games won by the strong net.

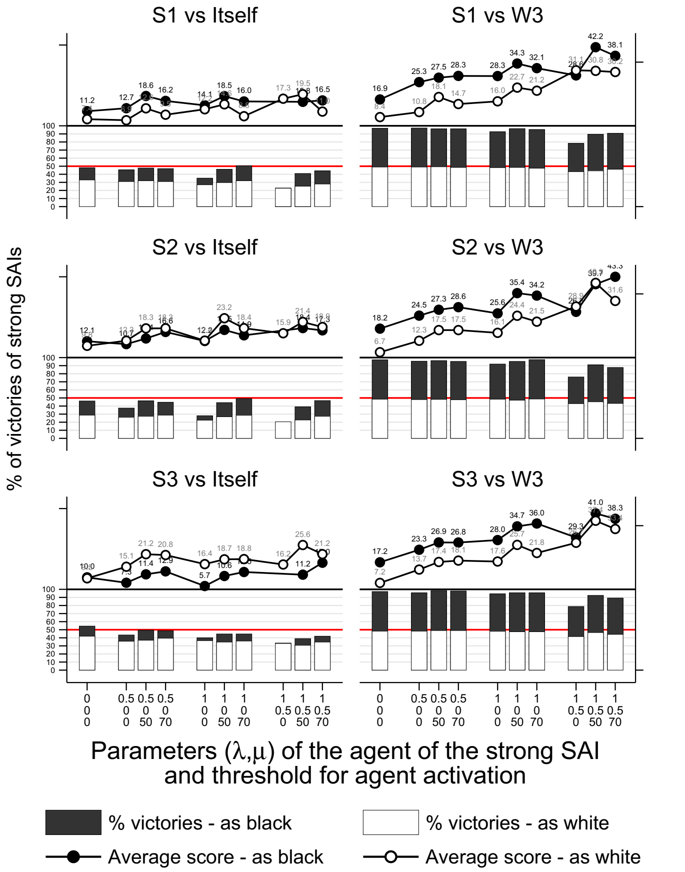


Table 1. Analysis of the data in Figure 6. The winrates and the scores are averages across the three strong nets. The differences (indicated by Δ) both in winrates and in scores are between each row and the corresponding row with $\lambda = \mu = 0$. A binomial test for difference smaller than 0 was conducted for winrates, a t-test for difference larger than 0 was conducted for score. Differences are marked with * if the p-value is < 0.05 and with ** if the p-value is < 0.001.

Threshold=50%, opponent: itself

Color	λ	μ	Winrate (%)	Δ in winrate with (0,0)	Score	Δ in score with (0,0)
White	0	0	69.2	-	8.9	-
	0.5	0	64.2	-5.0^{**}	17.5	$+8.6^{**}$
	1	0	60.8	-8.3^{**}	18.7	$+9.8^{**}$
		0.5	52.5	-16.7^{**}	22.4	$+13.5^{**}$
Black	0	0	30.0	-	11.2	-
	0.5	0	31.7	$+1.7$	14.6	$+3.4^{**}$
	1	0	29.3	-0.7	16.3	$+5.1^{**}$
		0.5	26.8	-3.2^{*}	15.9	$+4.7^{**}$

Threshold=70%, opponent: itself

Color	λ	μ	Winrate (%)	Δ in winrate with (0,0)	Score	Δ in score with (0,0)
White	0	0	69.2	-	8.9	-
	0.5	0	66.2	-3.0	16.5	$+7.6^{**}$
	1	0	64.3	-4.8^{**}	15.4	$+6.5^{**}$
		0.5	60.0	-9.2^{**}	17.4	$+8.4^{**}$
Black	0	0	30.0	-	11.2	-
	0.5	0	27.5	-2.5	15.6	$+4.4^{**}$
	1	0	32.0	$+2.0$	14.9	$+3.7^{**}$
		0.5	28.7	-1.3	17.0	$+5.8^{**}$

Threshold=50%, opponent: W3

Color	λ	μ	Winrate (%)	Δ in winrate with (0,0)	Score	Δ in score with (0,0)
White	0	0	97.2	-	7.5	-
	0.5	0	97.7	$+0.5$	17.7	$+10.2^{**}$
	1	0	95.5	-1.7^{*}	24.2	$+16.8^{**}$
		0.5	90.7	-6.5^{**}	36.2	$+28.8^{**}$
Black	0	0	97.2	-	17.4	-
	0.5	0	96.3	-0.8	27.2	$+9.8^{**}$
	1	0	96.2	-1.0	34.8	$+17.3^{**}$
		0.5	91.3	-5.8^{**}	41.0	$+23.5^{**}$

Threshold=70%, opponent: W3

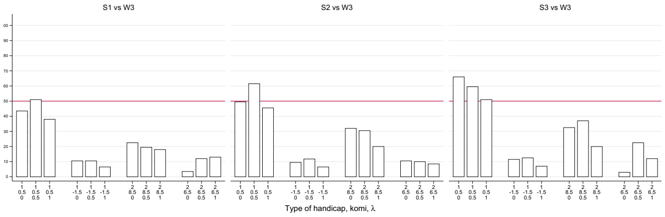
Color	λ	μ	Winrate (%)	Δ in winrate with (0,0)	Score	Δ in score with (0,0)
White	0	0	97.2	-	7.5	-
	0.5	0	96.8	-0.3	16.7	$+9.3^{**}$
	1	0	95.8	-1.3^{*}	21.5	$+14.0^{**}$
		0.5	89.2	-8.0^{**}	31.7	$+24.3^{**}$
Black	0	0	97.2	-	17.4	-
	0.5	0	96.2	-1.0	27.9	$+10.5^{**}$
	1	0	96.7	-0.5	34.1	$+16.7^{**}$
		0.5	89.3	-7.8^{**}	39.9	$+22.4^{**}$

4.3 SAI can recover from very severe disadvantage

As demonstrated in Section 4.1, SAI can play with handicap. However, victory becomes rare in H1 and H1 with an additional negative malus point; or H2 with 6 or even 8 bonus points. In such severe situations we investigated whether having the strong net overestimate its advantage would help it keeping a solid game until the weak net made some error, allowing the strong net to leverage on its own superiority and win. To this aim we had the 3 strong nets play with W3 in the four severe disadvantageous situations, with λ set to 0.5 and to 1.

The result is shown in Figure 7. With $\lambda = 0$, both S1 and S3 had won less than 5% of the times with 2 handicaps and a bonus of 6 points (komi 8.5). In both cases, setting λ to 0.5 increased substantially the winning probability, to 12.0% and 22.5% respectively. In the case of S2, that had had a better performance than S1 and S3 in this extreme situation, increasing λ to 0.5 didn't have any noticeable effect. Setting λ to 1 did not further improve the winning probability for any of the strong nets. In the other, less extreme situations of disadvantage, the effect of setting λ to 0.5 was inconsistent, while further increasing λ to 1 never improved the winning probability.

Figure 7. Games of the three strong nets versus W3, playing white with handicap incremented or decreased by komi points. Since the strong nets play white, lower komi points mean higher disadvantage. Type of handicap 1: the disadvantage is malus points. Type of handicap 2: the starting board contains 2 black stones and white plays first.



4.4 SAI can minimize suboptimal moves

As mentioned in the Introduction, in the view of the content experts, AlphaGo wins often by a small margin and plays suboptimal moves, since no direct notion of score is incorporated in its DRL pipeline. This is common knowledge among Go players, based on professional analysis of public AlphaGo games against humans [18]. In order to quantify this claim and, at the same time, prove that SAI acts less suboptimally, in July 2019 the following experiment was organized. Since AlphaGo and AlphaGo Zero are not publicly available, we selected a recent, unofficial but very strong Leela Zero (LZ) net for 9×9 Go.[7] We ran 200 games between LZ and W3, the weakest net in our experiments. LZ was assigned white and won all the games. We drew a 10% random sample from the 200 games. Our strongest net, S1, had won 194 times out of 200 games played as white against W3, with $(\lambda, \mu) = (1, 0)$ (see Subsection 4.2). We drew a random sample of 20 games from the 194. In total, we obtained a sample of 40 games, 20 played by LZ and 20 played by S1. We shuffled the 40 games and labeled them with an anonymous identifier.

We asked two strong amateur players (4D and 3D) to score the 40 games and to identify whether the winner had played suboptimal moves. We also asked them to rate their own estimate of score as 'reliable' or 'non reliable'. The two assessors did not communicate with each other during assessment. As a result, the scores of 32 games were rated 'reliable' by both assessors. The average difference in score between the two assessors was 1.53, in detail 0.8 among 'reliable' and 4.3 among 'non reliable' games. We computed the mean of the two scores and we linked the data to the identity of the winners: LZ or SAI. We found that the average scores of LZ and SAI were, respectively, 6.3 and 16.0 (single-tail t-test: $p < 0.001$), or 6.0 and 15.0 when restricting to 'reliable' games (single-tail t-test: $p = 0.006$). The games with no suboptimal moves were 18 (90%) for SAI and 11 (55%) for LZ (χ^2 test: $p = 0.013$). The files of the games in Smart Game Format, the manual assessment and analysis are available at this link [10]. Finally, to understand the comparative strength of LZ with respect to S1 playing with this agent, we had them play 400 games: LZ won 359 times (89.8%). In summary, even though LZ was stronger than S1, it got significantly lower scores, and made significantly more suboptimal moves, with respect to SAI playing with an agent targeting high scores.

4.5 SAI is superhuman

A match was scheduled on May 2019 between SAI and Hayashi Kozo 6P, a professional Japanese player. The match was composed by 3 games, at alternate colors, and komi 7.5. In the first game white was assigned to Hayashi Kozo 6P, out of respect, because it is traditional in Go that the more expert player plays white first.[8] SAI was set to play with net S1, 50,000 visits, $\lambda = \mu = 0$ and resign threshold set at 5%. SAI won all games, two playing black and one playing white.

Another match was also scheduled on May 2019 between SAI and Oh Chimin 7D, a 7-Dan Korean amateur player whose strength is estimated to be that of a low dan professional player. The match was composed by 5 games, with different values of komi. SAI played with the same settings of the previous match and won 4 games, three of them while playing white with komi 7.5, 5.5 and 3.5, one playing black with komi 13.5 (which amounts to 6 malus points for SAI). Oh Chimin 7D won a game playing black with komi 1.5. According to expert Go knowledge, winning against a professional-level player with 6 points of handicap on a 9×9 board is an achievement that classifies SAI as superhuman.

5 Developments

5.1 SAI and the game of Go

Many opportunities associated with SAI's framework are still unexplored. For instance, the variable agents introduced at the end of Section 4.2 may be used in dependence of other parameters with respect to the pointwise estimate of the winrate, or used in conjunction with an estimate of the strength of the opponent. Symmetrically, the ability of SAI to recover from very severe disadvantage against a weaker opponent that was highlighted in Section 4.3, may probably be better exploited if the agent were variable. Indeed, it may be suggested that, as soon as a mistake from the weaker opponent re-opens the game, a variable agent could restore its proper assessment of the winrate, thus switching its aim from recovering points to targeting victory and improving effectively its chances.

Moreover, note that SAI is able to win with substantial handicap, even in situations when the information stored in the sigmoid is of

[7] See https://github.com/leela-zero/leela-zero/issues/863#issuecomment-497599672.

[8] This is rooted in the fact that, in the past, komi was non assigned, and later it was smaller than 7.5: as a consequence, white was at a disadvantage. It must be noted that nowadays, however, the conventional 7.5 points komi gives a small advantage to the expert player, since, to avoid ties, the number of games scheduled for a match is generally odd.

little help because the winrate is very low. Similarly, a SAI net is able to win when playing with $\lambda = \mu = 1$, when the winrate is constantly close to 0.5, even when playing with an equally strong opponent (itself). We speculate that both those abilities amount to the quality of the *policy*, that stems from experience gained during branched games. It would be interesting to test this hypothesis.

In the games we organized with human players, see Section 4.5, our primary goal was victory. For this reason we used $\lambda = \mu = 0$. As a consequence, we observed that SAI made some suboptimal moves. A promising development is using SAI with variable λ and μ against humans, for competitive and teaching purposes, in order for it to choose optimal moves with humans even when it is winning.

All such opportunities will have a chance to be fully exploited when SAI is trained to play on the 19×19 board.

5.2 SAI and Deep Reinforcement Learning

After the seminal papers on Atari [13] and AlphaGo [15], Deep Reinforcement Learning (DRL) has been a major research topic. The SAI framework, at its essence, is a variation into the high-level, domain-independent aspects of DRL, stemming from the assumption that probability of success is a function of the targeted score, belonging to a parametric family of functions whose parameters can be learned by a neural network. The only requirement for SAI to contribute to an application is that success is linked to some score in the first place. This is true in many instances of zero-sum two-player games, for instance Othello, where a Leela Zero approach has been advocated for but not yet pursued [12]. In fact, DRL was recently used to address broader challenges, such as multi-player games, with incomplete information, and/or with non-zero sum. Whenever score is relevant for success, SAI can contribute with parametric modeling of winrate as a function of targeted score, with branching training techniques, with real-time score estimates, with a parametric family of agents which allow real-time tuning the playing style to the opponent and to the game circumstances. For instance Puerto Rico [4] could benefit from the general-purpose aspects of SAI. Finally, DRL is expected to extend to other domains outside of games, spanning from robotics to complex behaviors in 3D environments and character animation: as science progresses by contamination, we would not be surprised if SAI could be part of this conversation, too.

Acknowledgements. We thank Hayashi Kozo 6P and Oh Chimin 7D for accepting the challenge of playing with SAI; Alessandro Martinelli 2D, Davide Minieri 3D and Alessandro Pace 4D for support in the validation of suboptimal moves, and Michele Piccinno and Francesco Potortì for technical support.

REFERENCES

[1] Petr Baudiš, 'Balancing MCTS by Dynamically Adjusting the Komi Value', *ICGA Journal*, **34**(3), 131–139, (2011).

[2] Rémy Coulom. Bayesian Elo rating, 2010. http://www.remi-coulom.fr/Bayesian-Elo/ [Online; accessed 10-Nov-2019].

[3] James Davies, '7× 7 Go', *American Go Journal*, **29**(3), 11, (1995).

[4] Rafal Drezewski and Maciej Kleczar, 'Artificial Intelligence Techniques for the Puerto Rico Strategy Game', volume 74, (06 2017).

[5] Arpad E. Elo, *The Rating of Chessplayers, Past & Present*, ISHI Press International, Bronx NY 10453, 2008.

[6] Gian-Carlo Pascutto and contributors. Leela Zero, 2018. http://zero.sjeng.org/home [Online; accessed 17-August-2018].

[7] Gianluca Amato and contributors. SAI: a fork of Leela Zero with variable komi. https://github.com/sai-dev/sai [Online; accessed 10-Nov-2019].

[8] Rosa Gini and contributors. Experiments with SAI 9 × 9, 2019. https://drive.google.com/file/d/1vGMlc9VcjwOFxV1MxytbrrbgVOBSP4pA [Online; accessed 11-Nov-2019].

[9] Rosa Gini and contributors. Tables of results, 2019. https://drive.google.com/open?id=1d07zfPxITgA6YQiT5smSXuvDrFdOe6pg [Online; accessed 11-Nov-2019].

[10] Rosa Gini and contributors. Validation of suboptimal moves in SAI and LZ games, 2019. https://drive.google.com/open?id=1VARf2fLpJvurdXSGsV9z_mkqdk3kQzmM [Online; accessed 11-Nov-2019].

[11] Jilmer Justin. GoCNN - using CNN to do move prediction and board evaluation for the board game Go. https://github.com/jmgilmer/GoCNN [Online; accessed 10-Nov-2019].

[12] Pawel Liskowski, Wojciech Jaskowski, and Krzysztof Krawiec, 'Learning to Play Othello With Deep Neural Networks', *IEEE Trans. Games*, **10**(4), 354–364, (2018).

[13] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis, 'Human-level control through deep reinforcement learning', *Nature*, **518**(7540), 529–533, (February 2015).

[14] Francesco Morandin, Gianluca Amato, Rosa Gini, Carlo Metta, Maurizio Parton, and Gian-Carlo Pascutto, 'SAI a Sensible Artificial Intelligence that plays Go', in *International Joint Conference on Neural Networks, IJCNN 2019 Budapest, Hungary, July 14-19, 2019*, pp. 1–8, (2019).

[15] David Silver, Aja Huang, Chris J Maddison, Arthur Guez, Laurent Sifre, George Van Den Driessche, Julian Schrittwieser, Ioannis Antonoglou, Veda Panneershelvam, Marc Lanctot, et al., 'Mastering the game of Go with deep neural networks and tree search', *Nature*, **529**(7587), 484, (2016).

[16] David Silver, Thomas Hubert, Julian Schrittwieser, Ioannis Antonoglou, Matthew Lai, Arthur Guez, Marc Lanctot, Laurent Sifre, Dharshan Kumaran, Thore Graepel, et al., 'A general reinforcement learning algorithm that masters chess, shogi, and Go through self-play', *Science*, **362**(6419), 1140–1144, (2018).

[17] David Silver, Julian Schrittwieser, Karen Simonyan, Ioannis Antonoglou, Aja Huang, Arthur Guez, Thomas Hubert, Lucas Baker, Matthew Lai, Adrian Bolton, et al., 'Mastering the game of Go without human knowledge', *Nature*, **550**(7676), 354, (2017).

[18] Antti Törmänen, *Invisible: the games of AlphaGo*, Hebsacker Verlag, 2017.

[19] Seth Troisi. MiniGo Model Evaluation, 2019. https://cloudygo.com/all-eval-graphs [Online; accessed 10-Nov-2019].

[20] David J Wu, 'Accelerating Self-Play Learning in Go', *arXiv:1902.10565*, (2019).

[21] Ti-Rong Wu, I Wu, Guan-Wun Chen, Ting-han Wei, Tung-Yi Lai, Hung-Chun Wu, Li-Cheng Lan, et al., 'Multi-Labelled Value Networks for Computer Go', *arXiv:1705.10701*, (2017).

[22] Li Zhe, 'Qi lu qipan zui you jie [7x7 Weiqi Optimal Solution]', *Weiqi tiandi [The World of Weiqi]*, (20), 11, (2015).