Game-playing AI masters the game of Go

Duc Vu

I. Introduction

Go is an ancient board game which takes simple elements: line and circle, black and white. The rules of the game are simple: Players take turns to place black or white stones on a board, trying to capture the opponent's stones or surround empty space to make points of territory. As simple as the rules are, Go is a game of profound complexity. There are more possible positions in Go than there are atoms in the universe.

Google Deep-Mind developed AlphaGo, the first Computer Go program to beat a human professional Go player. AlphaGo's algorithm uses a Monte Carlo tree search to find its moves based on knowledge previously "learned" by deep neural networks that have been trained by supervised learning, from human expert games, and by reinforcement learning from games of self-play.

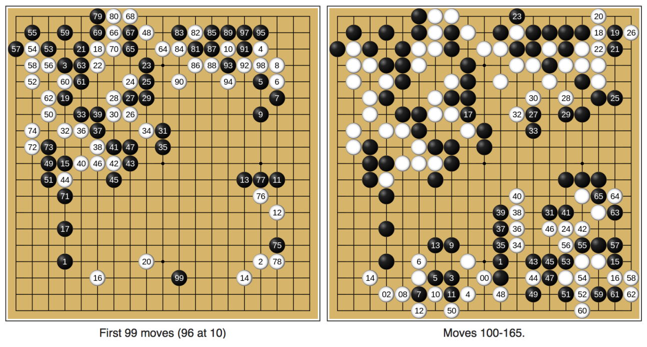


Fig. 1. AlphaGo (black) beat Fan Hui in Game 4 by Wikipedia

II. Techniques

Problem setting: To win the game, computer Go recursively computes the optimal value function $v^*(s)$ (from each board position s) in a search tree containing approximately B^D possible sequences of moves, where B is the games branch (number of legal moves per position) and D is game's depth.

Optimized search space can be obtained by two approaches. reducing the depth of the search D by position evaluation and reducing the branch of the search B by sampling actions from a policy $p(a|s)$ - a probability distribution over possible moves a in position s.

Monte Carlo tree search (MCTS) is used to estimate the value of each state in a search tree and the policy $p(a|s)$ are used to select actions during search which eventually converge to the optimal value function $v^*(s)$. To reduce the depth and branch of the search tree, the Deep convolutional neural networks (DNN) is applied to evaluate positions using a value network, and sample actions using a policy network. DNN training pipeline consists of several stages of machine learning :

1. Supervised learning of policy networks $p_\sigma(a|s)$: is trained directly from expert human moves and outputs a probability distribution over all legal moves a

2. Supervised learning rollout policy p_π : rapidly samples actions during rollouts.

3. Reinforcement learning policy networks p_ρ : adjusts the policy towards the correct goal of winning games, rather than maximizing predictive accuracy by policy gradient reinforcement learning and by optimizing the final outcome of games of self-play.

4. Reinforcement learning value networks $v_\theta(s)$: estimates a value function $s^p(s)$ that predicts the outcome from positions of games played by using policy p for both players. In order to predict the winner of games played by the RL policy network against itself, optimal value function $v^*(s)$ is approximated by a value network $v_\theta(s)$ with weights $\theta(s)$

$$v_\theta \approx v^p(s) \approx v^*(s) \quad (1)$$

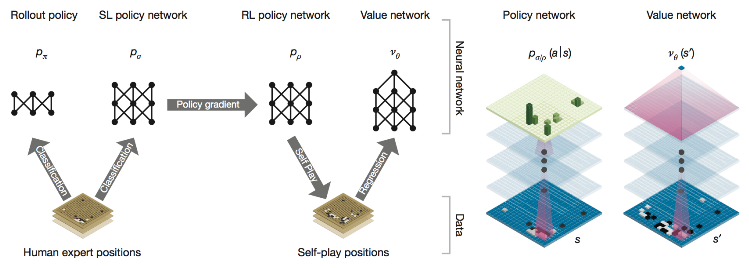


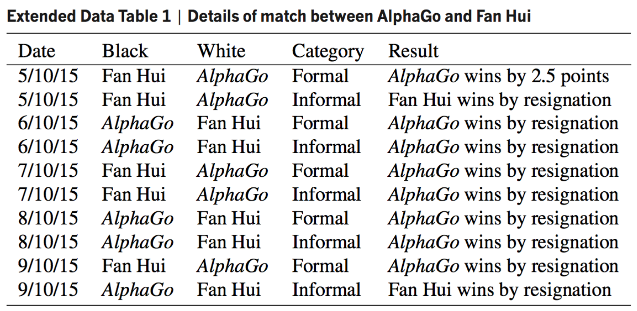
Fig. 2. Neural network training pipeline and architecture

Searching with policy and value networks : computer Go combines the policy and value networks in an MCTS algorithm to select actions that maximize action value $Q(s,a)$ and bonus action $u(s,a)$. At each time step t of each simulation, an action a_t is selected from state s_t

$$a_t = argmax(Q(s_t, s) + u(s_t, a)) \quad (2)$$

Evaluating the playing strength of AlphaGo : computer Go uses value networks to evaluate board positions and policy networks to select moves. New search algorithm that combines Monte Carlo tree search (MCTS) with value and policy networks to achieve a 99.8% winning rate.

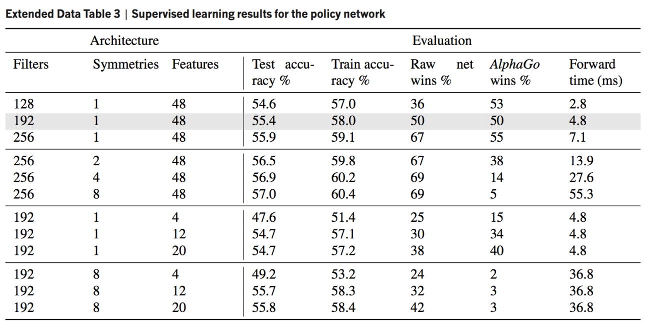
III. Resuts



Extended Data Table 1 | Details of match between AlphaGo and Fan Hui

Date	Black	White	Category	Result
5/10/15	Fan Hui	*AlphaGo*	Formal	*AlphaGo* wins by 2.5 points
5/10/15	Fan Hui	*AlphaGo*	Informal	Fan Hui wins by resignation
6/10/15	*AlphaGo*	Fan Hui	Formal	*AlphaGo* wins by resignation
6/10/15	*AlphaGo*	Fan Hui	Informal	*AlphaGo* wins by resignation
7/10/15	Fan Hui	*AlphaGo*	Formal	*AlphaGo* wins by resignation
7/10/15	Fan Hui	*AlphaGo*	Informal	*AlphaGo* wins by resignation
8/10/15	*AlphaGo*	Fan Hui	Formal	*AlphaGo* wins by resignation
8/10/15	*AlphaGo*	Fan Hui	Informal	*AlphaGo* wins by resignation
9/10/15	Fan Hui	*AlphaGo*	Formal	*AlphaGo* wins by resignation
9/10/15	*AlphaGo*	Fan Hui	Informal	Fan Hui wins by resignation

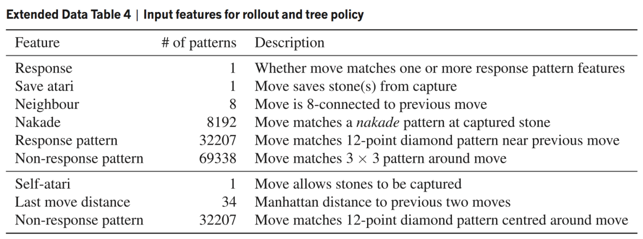
Fig. 3. AlphaGo (black) vs Fan Hui



Extended Data Table 3 | Supervised learning results for the policy network

Architecture			Evaluation				
Filters	Symmetries	Features	Test accuracy %	Train accuracy %	Raw net wins %	*AlphaGo* wins %	Forward time (ms)
128	1	48	54.6	57.0	36	53	2.8
192	1	48	55.4	58.0	50	50	4.8
256	1	48	55.9	59.1	67	55	7.1
256	2	48	56.5	59.8	67	38	13.9
256	4	48	56.9	60.2	69	14	27.6
256	8	48	57.0	60.4	69	5	55.3
192	1	4	47.6	51.4	25	15	4.8
192	1	12	54.7	57.1	30	34	4.8
192	1	20	54.7	57.2	38	40	4.8
192	8	4	49.2	53.2	24	2	36.8
192	8	12	55.7	58.3	32	3	36.8
192	8	20	55.8	58.4	42	3	36.8

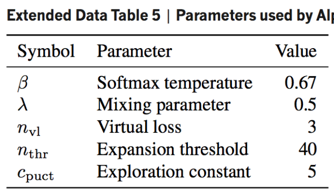
Fig. 4. Supervised learning results for policy networks



Extended Data Table 4 | Input features for rollout and tree policy

Feature	# of patterns	Description
Response	1	Whether move matches one or more response pattern features
Save atari	1	Move saves stone(s) from capture
Neighbour	8	Move is 8-connected to previous move
Nakade	8192	Move matches a *nakade* pattern at captured stone
Response pattern	32207	Move matches 12-point diamond pattern near previous move
Non-response pattern	69338	Move matches 3×3 pattern around move
Self-atari	1	Move allows stones to be captured
Last move distance	34	Manhattan distance to previous two moves
Non-response pattern	32207	Move matches 12-point diamond pattern centred around move

Fig. 5. Input features for rollout and tree policy



Extended Data Table 5 | Parameters used by AlphaGo

Symbol	Parameter	Value
β	Softmax temperature	0.67
λ	Mixing parameter	0.5
n_{vl}	Virtual loss	3
n_{thr}	Expansion threshold	40
c_{puct}	Exploration constant	5

Fig. 6. AlphaGo parameters

References

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