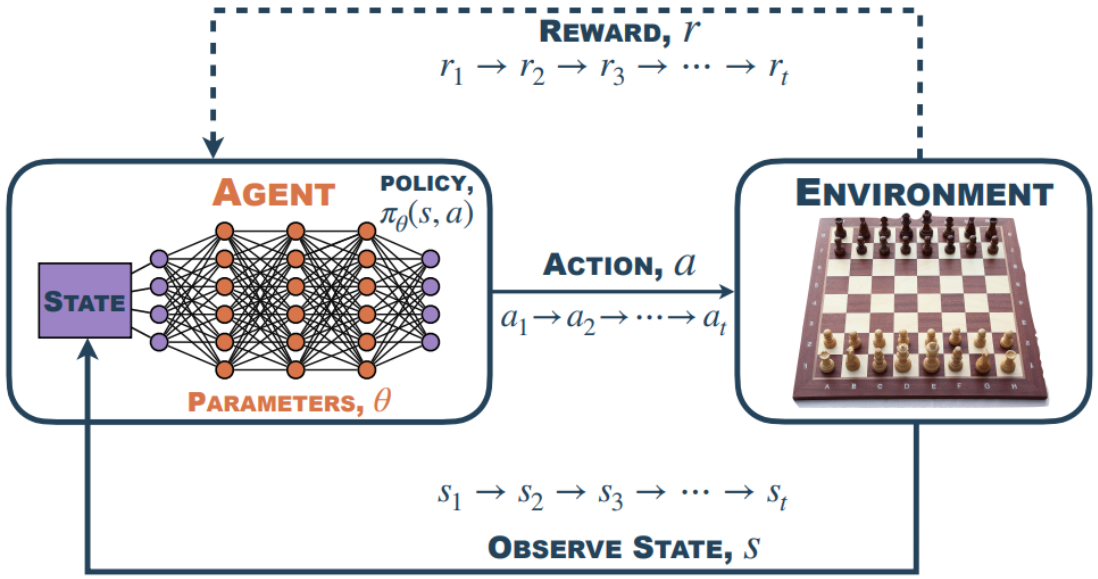
Mastering the game of Go with deep neural networks and tree search

https://storage.googleapis.com/deepmind-media/alphago/AlphaGoNaturePaper.pdf

Hieu Nguyen – Godaddy Inc.

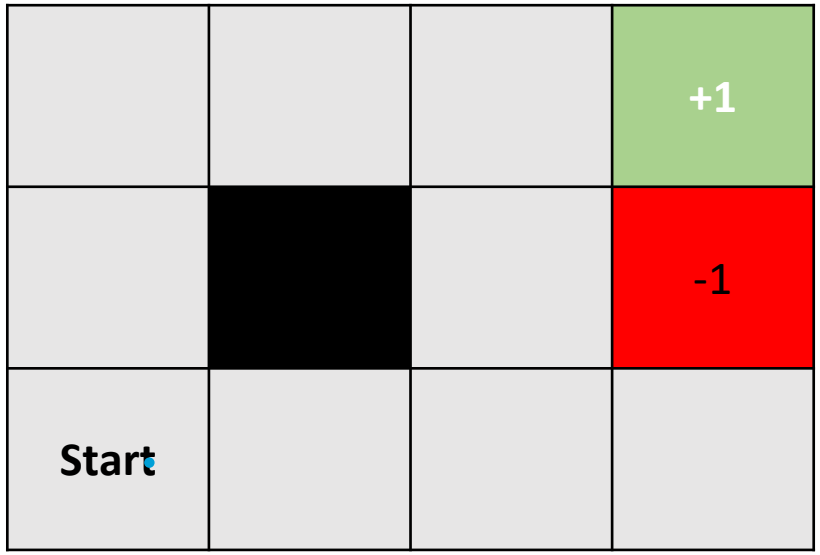
2022-02-18

Reinforcement Learning Overview

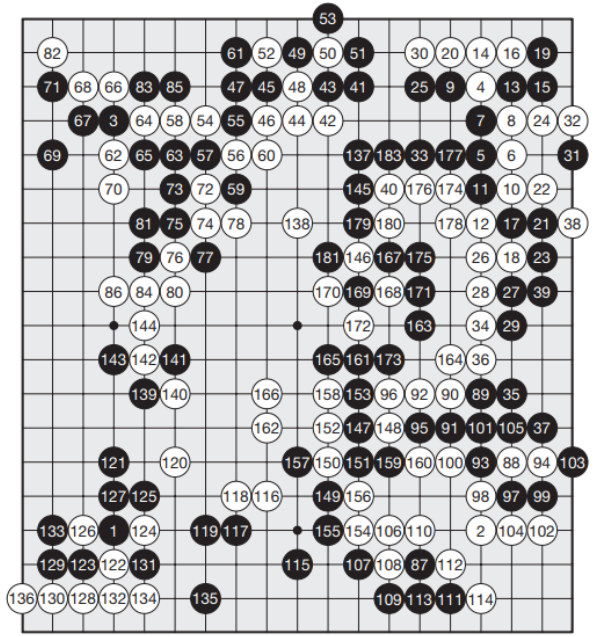


- Data: from interacting with the environment $\{s,a\}_t$
- Model: $\pi_\theta(s,a) = \underset{\mathbf{a}}{\operatorname{argmax}} \sum_{\mathbf{s}'} P(\mathbf{s}' \mid \mathbf{s},\mathbf{a})\,(R(\mathbf{s}',\mathbf{s},\mathbf{a}) + \gamma V(\mathbf{s}'))$
- Goal: maximize expected future rewards

Value and Policy – Main Idea



The Game of Go



Game of Go:

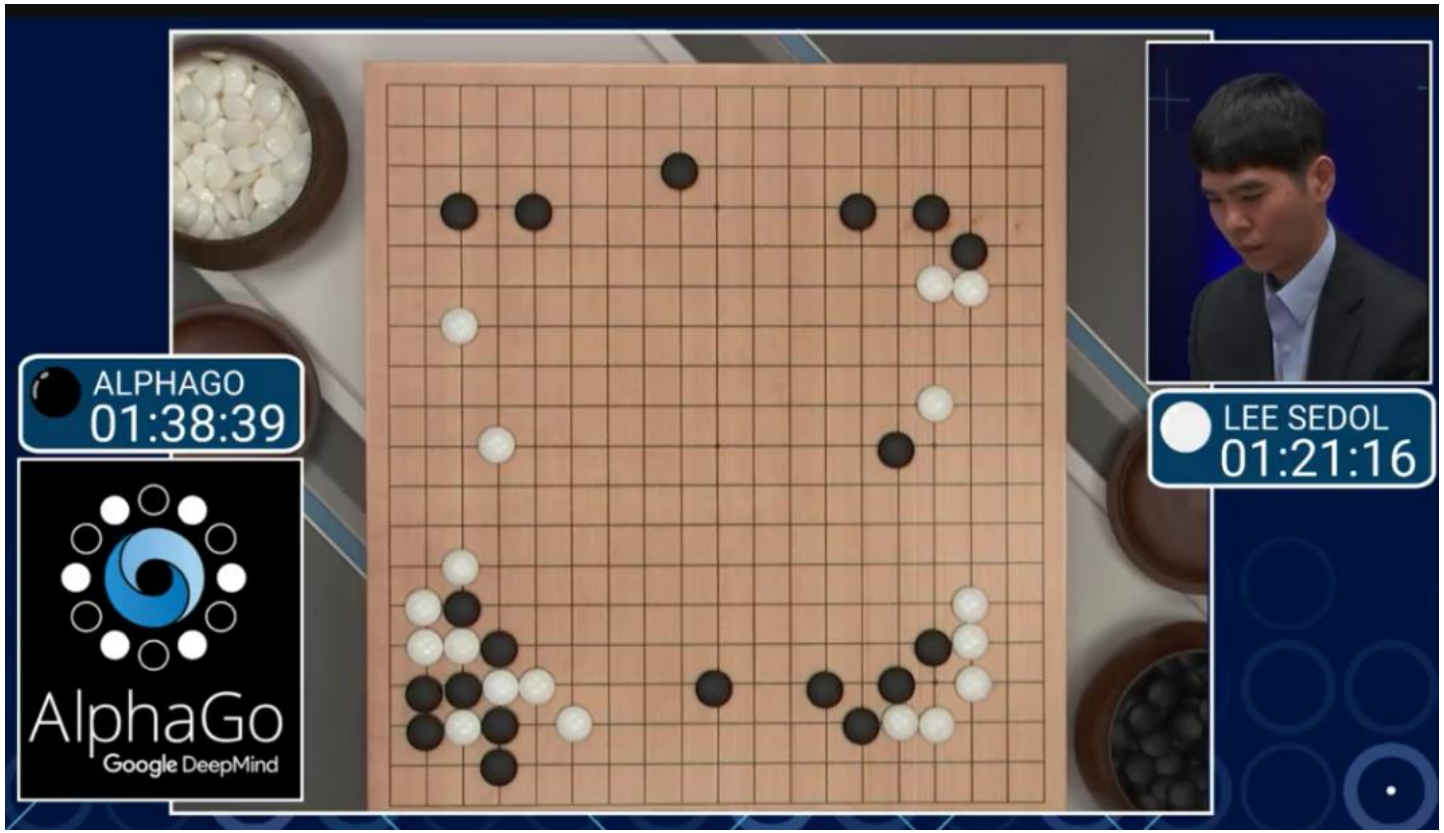
- 19x19 grid ancient board game
- Search space complexity: ~10^170

AI Challenge:

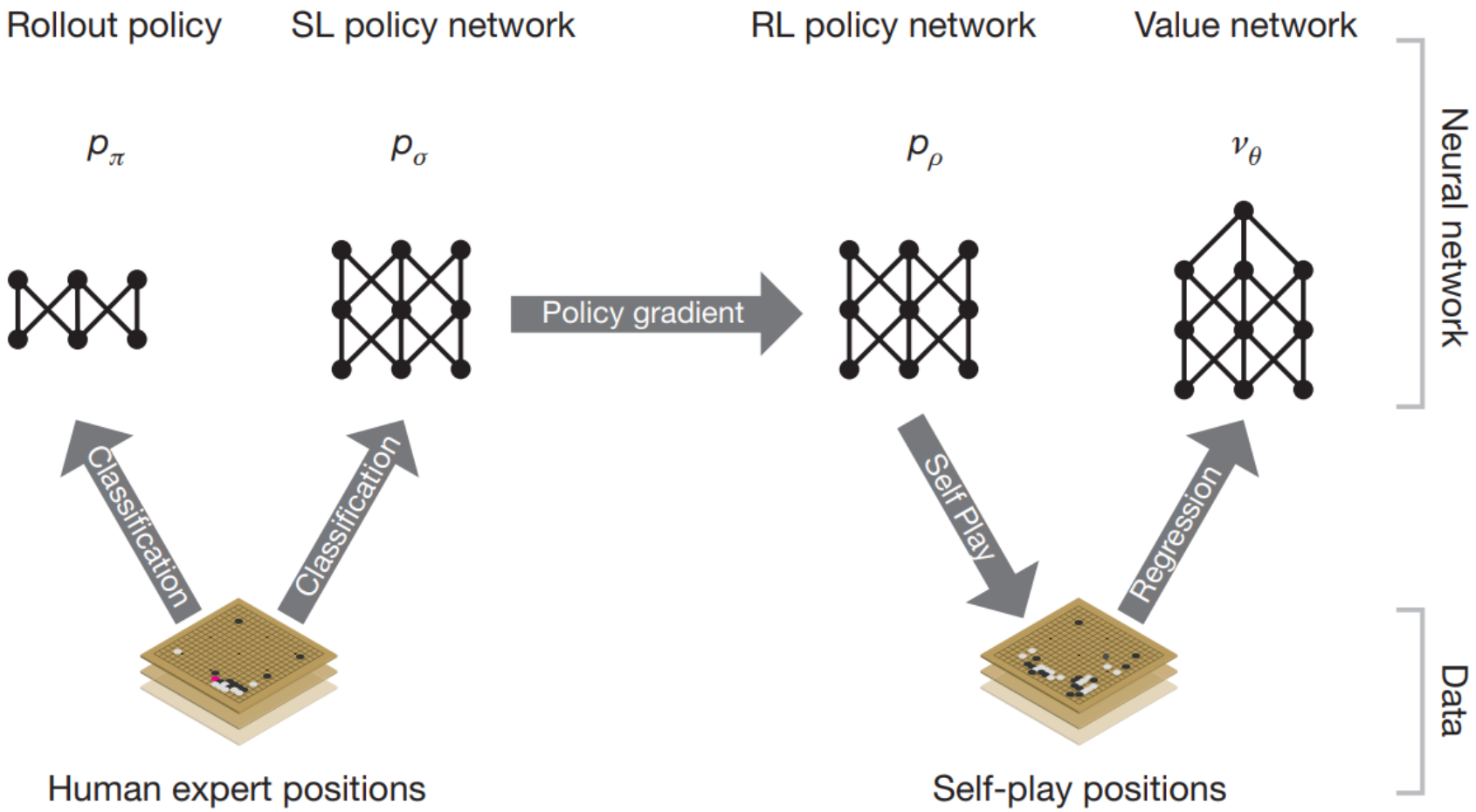
- How to effectively search through an intractable space?

AlphaGo

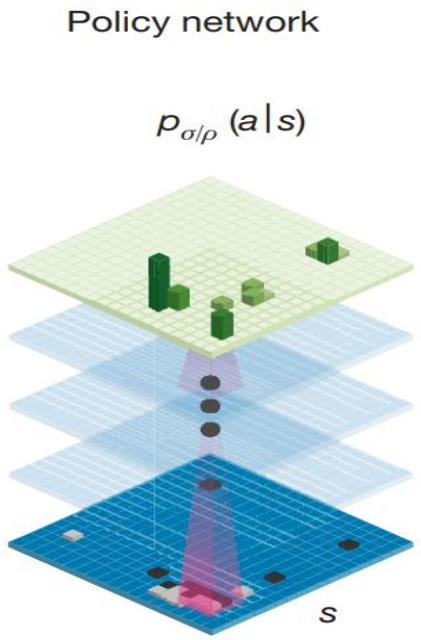
Defeated 18x Go Champion Lee Sedol in 2016



AlphaGo – High Level Training Pipeline



AlphaGo – SL Policy Network



Extended Data Table 2 | Input features for neural networks

Feature	# of planes	Description
Stone colour	3	Player stone / opponent stone / empty
Ones	1	A constant plane filled with 1
Turns since	8	How many turns since a move was played
Liberties	8	Number of liberties (empty adjacent points)
Capture size	8	How many opponent stones would be captured
Self-atari size	8	How many of own stones would be captured
Liberties after move	8	Number of liberties after this move is played
Ladder capture	1	Whether a move at this point is a successful ladder capture
Ladder escape	1	Whether a move at this point is a successful ladder escape
Sensibleness	1	Whether a move is legal and does not fill its own eyes
Zeros	1	A constant plane filled with 0
Player color	1	Whether current player is black

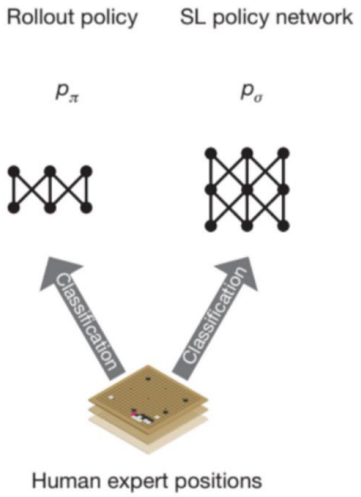
Feature planes used by the policy network (all but last feature) and value network (all features).

- Arch: 12-layer CNN
- Training data: 30M positions from expert games
  - Input: 19x19x48
- Objective function: max. likelihood by SGD

$$\Delta\sigma \propto \frac{\partial \log p_\sigma(a|s)}{\partial \sigma}$$

- Training time: 4 weeks on 50 GPUs
- Results: 57% accuracy on test set (44% state of the art)

AlphaGo – Rollout Policy



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

Features used by the rollout policy (first set) and tree policy (first and second set). Patterns are based on stone colour (black/white/empty) and liberties (1, 2, ≥3) at each intersection of the pattern.

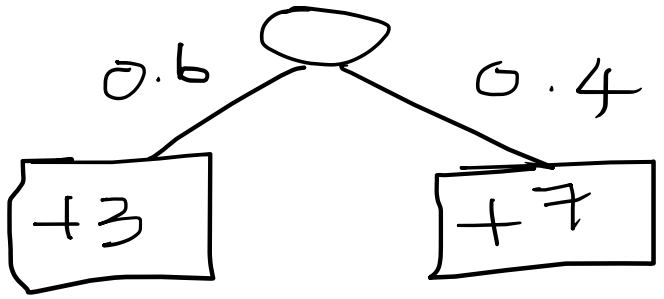
- Usage: fast rollout enables narrower search for moves during simulation
- Arch: more simple linear softmax classifier
- Training data: 8M positions from expert games
- Objective function: max. likelihood by SGD

$$\Delta\sigma \propto \frac{\partial \log p_\sigma(a|s)}{\partial \sigma}$$

- Training time: N/A
- Results: 24.2% accuracy on test set
- Speed: 2us vs 3ms in SL policy network (1500x faster)

Policy Gradient Theorems

$$\nabla_\theta J(\pi_\theta) = \mathop{\mathrm{E}}_{\tau \sim \pi_\theta}\left[\sum_{t=0}^{T} \nabla_\theta \log \pi_\theta(a_t|s_t) A^{\pi_\theta}(s_t, a_t)\right]$$



$$E[R] = P_L(3) + P_R(7)$$
$$= (0.6)(3) + (0.4)(7)$$
$$= 4.6$$

$$\frac{\partial R}{\partial P_L} = 3(1) + 0 = 3$$

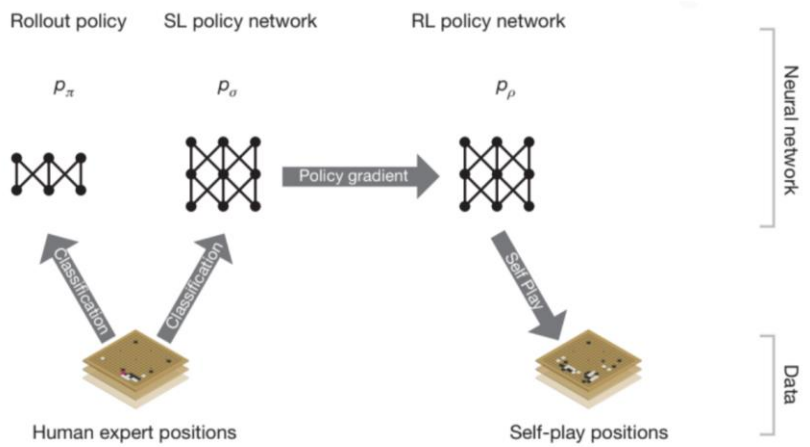
$$\frac{\partial R}{\partial P_R} = 0 + 7 = 7$$

Main idea:

- Compute how much the expected rewards changes wrt to how much each P changes
- Backprob these partial derivatives to the NN, the weights will update to produce new Ps such that the E[R] is maximized

https://spinningup.openai.com/en/latest/algorithms/vpg.html

AlphaGo – RL Policy Network

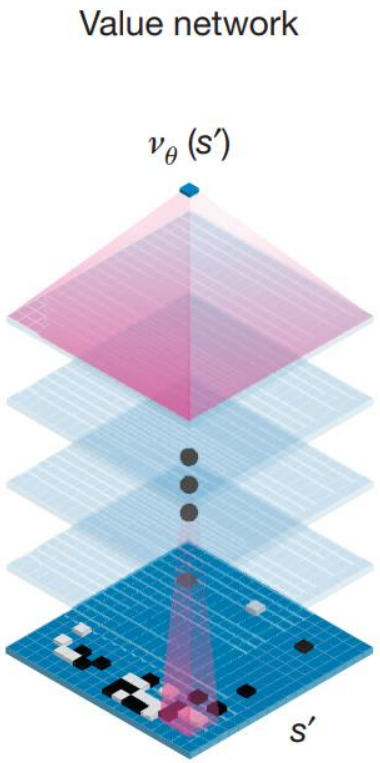


- Usage: reinforce current SL policy and will be used for self-play to generate training data for value network
- Arch: 12-layer CNN
- Training data: 10,000 mini-batches of 128 self-play games between policy networks
- Objective function: max. rewards z_t by policy gradient reinforcement learning

$$\Delta\rho \propto \frac{\partial \log p_\rho(a_t \mid s_t)}{\partial \rho} z_t$$

- Training time: 1 week on 50 GPUs
- Results: 80% win rate over SL policy network
- Notes:
  - They don't play each other on the same policy

AlphaGo – Value Network



- Usage: quantify how good a board position is
- Arch: 12-layer CNN
- Training data: 30 million games of self-play generated from RL policy network
  - Input: 19x19x(48 + 1) (colour to play)
- Objective function: min. MSE by SGD

$$\Delta\theta \propto \frac{\partial v_\theta(s)}{\partial\theta}(z - v_\theta(s))$$

- Training time: 1 week on 50 GPUs

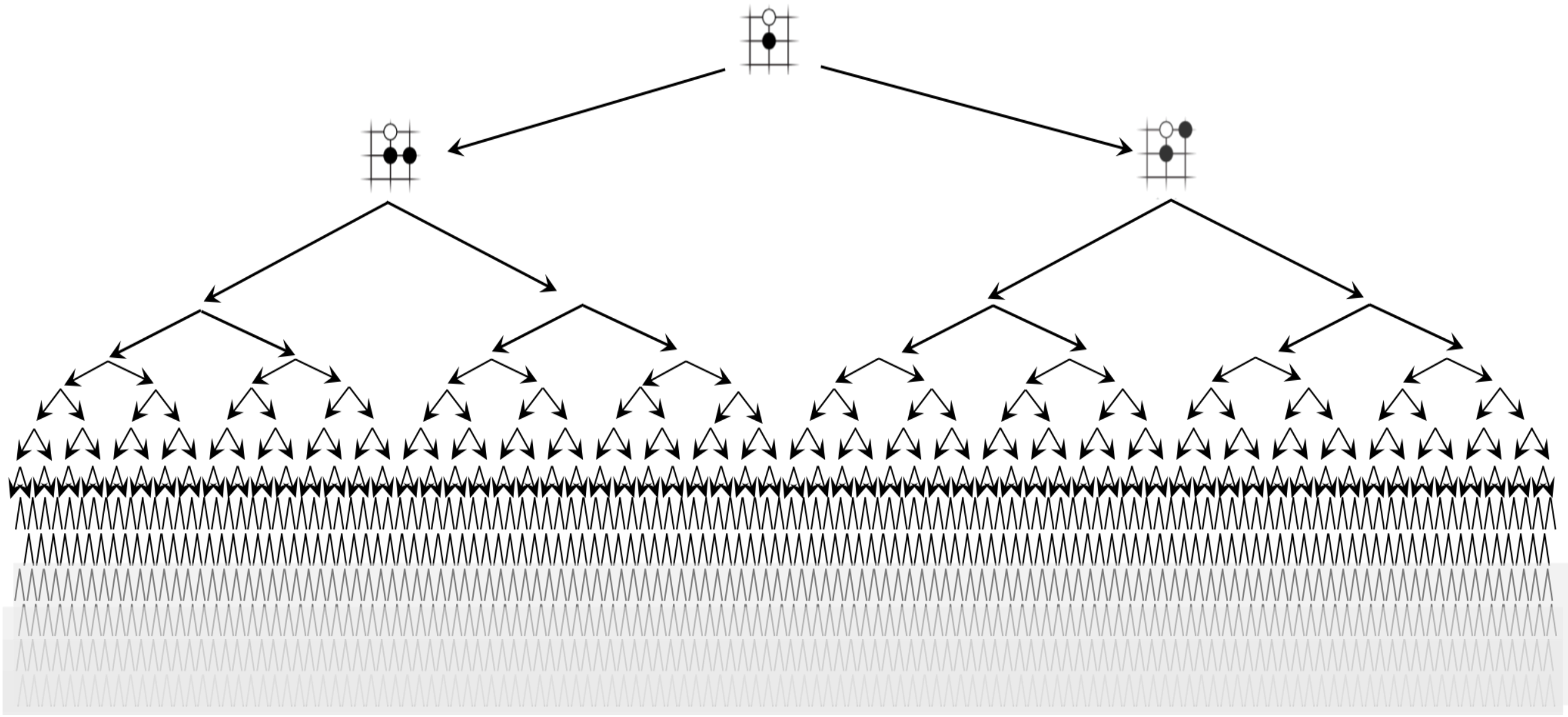
AlphaGo – Quick Recap

- Policy network: guides us to the next best moves
- Value network: quantify the quality of a board position
- Next, they complement these networks to help with search priorities during game simulations.

AlphaGo – Why Simulation?

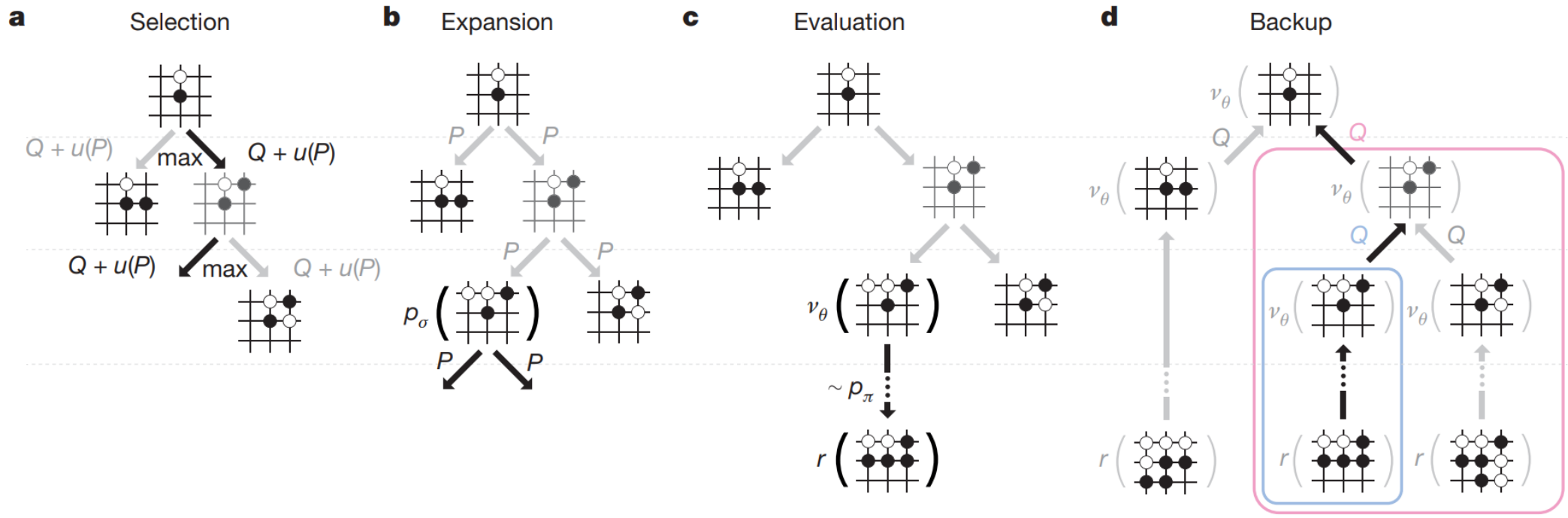
With sufficient games are simulated till the end from current position, we will get an idea which moves likely lead to the most wins. In the process, they build a search tree recording sequences of moves and their corresponding winning rates.

Exhaustive Search



Monte Carlo Tree Search (MCTS)

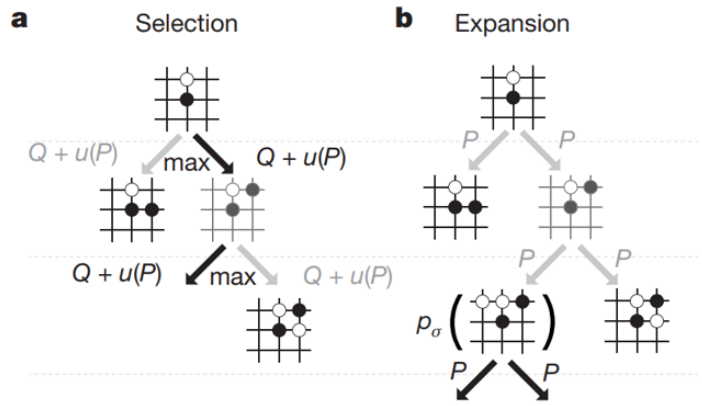
Monte Carlo Tree Search (MCTS) consists of 4 main steps:



MCTS is run on 48 CPUs in 40 threads; the policy and value evaluation run on 8 GPUs.

MCTS – Selection & Expansion

Goal: to prioritize most promising moves for further simulations.
This allows finding good moves with fewer games played



$a_t = \underset{a}{\operatorname{argmax}}(Q(s_t, a) + u(s_t, a))$ - Q: exploitation; u: exploration

$u(s,a) \propto \frac{P(s,a)}{1 + N(s,a)}$

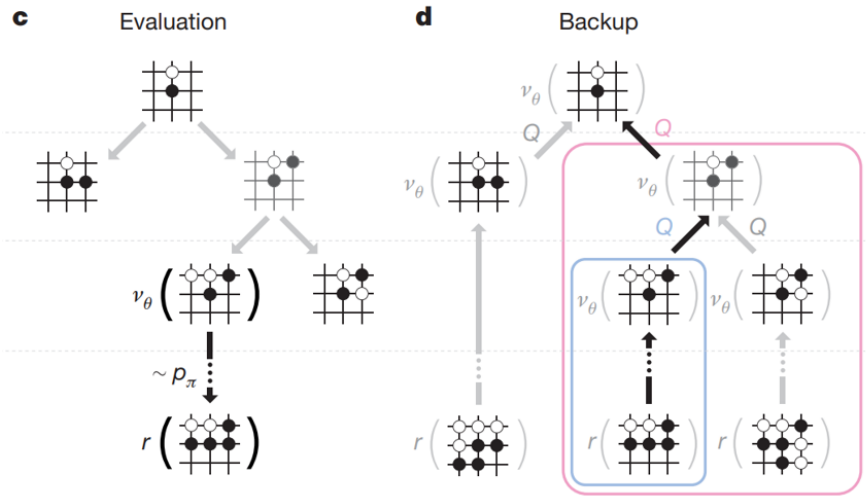
$P(s,a) = p_\sigma(a|s)$ - Policy Network: Probability taking action a in state s

$N(s,a) = \sum_{i=1}^{n} 1(s,a,i)$ - Number of time action a has been selected in state s

$Q(s,a) = \frac{1}{N(s,a)} \sum_{i=1}^{n} 1(s,a,i) V(s_L^i)$ - The quality of being in state s and taking action a

$V(s_L) = (1-\lambda) v_\theta(s_L) + \lambda z_L$ - Value Network: how advantageous is it to be in this position

MCTS – Evaluation & Backup



- In the evaluation, simulate the rest of the game using rollout policy starting from the leaf node until the end of the game to see whether it loses or wins.
- After the evaluation, we know our moves win/lose statistics. In the backup phase, update **Q(s,a)** to remember how well to make a move from the game results and the leaf node's value function.

MCTS – Results

$$V(s_L) = (1 - \lambda)v_\theta(s_L) + \lambda z_L$$

Extended Data Table 7 | Results of a tournament between different variants of AlphaGo

Short name	Policy network	Value network	Rollouts	Mixing constant	Policy GPUs	Value GPUs	Elo rating
α_{rvp}	p_σ	v_θ	p_π	$\lambda = 0.5$	2	6	2890
α_{vp}	p_σ	v_θ	–	$\lambda = 0$	2	6	2177
α_{rp}	p_σ	–	p_π	$\lambda = 1$	8	0	2416
α_{rv}	$[p_\tau]$	v_θ	p_π	$\lambda = 0.5$	0	8	2077
α_v	$[p_\tau]$	v_θ	–	$\lambda = 0$	0	8	1655
α_r	$[p_\tau]$	–	p_π	$\lambda = 1$	0	0	1457
α_p	p_σ	–	–	–	0	0	1517

Evaluating positions using rollouts only (α_{rp}, α_r), value nets only (α_{vp}, α_v), or mixing both (α_{rvp}, α_{rv}); either using the policy network $p_\sigma(\alpha_{rvp}, \alpha_{vp}, \alpha_{rp})$, or no policy network ($\alpha_{rvp}$, α_{vp}, α_{rp}), that is, instead using the placeholder probabilities from the tree policy p_τ throughout. Each program used 5 s per move on a single machine with 48 CPUs and 8 GPUs. Elo ratings were computed by BayesElo.

References/Resources:

- David Silver RL Series, DeepMind
- Steve Brunton, uWashington – Data Driven Science & Engineering
- Pascal Poupart, uWaterloo – CS885: Reinforcement Learning
- Martin Lysy, uWaterloo – Stats946: Advanced Computational Statistics