AlphaGo: Mastering the game of Go with neural networks and tree search

Silver et al. Nature 2016

Presented by Allan Zelener

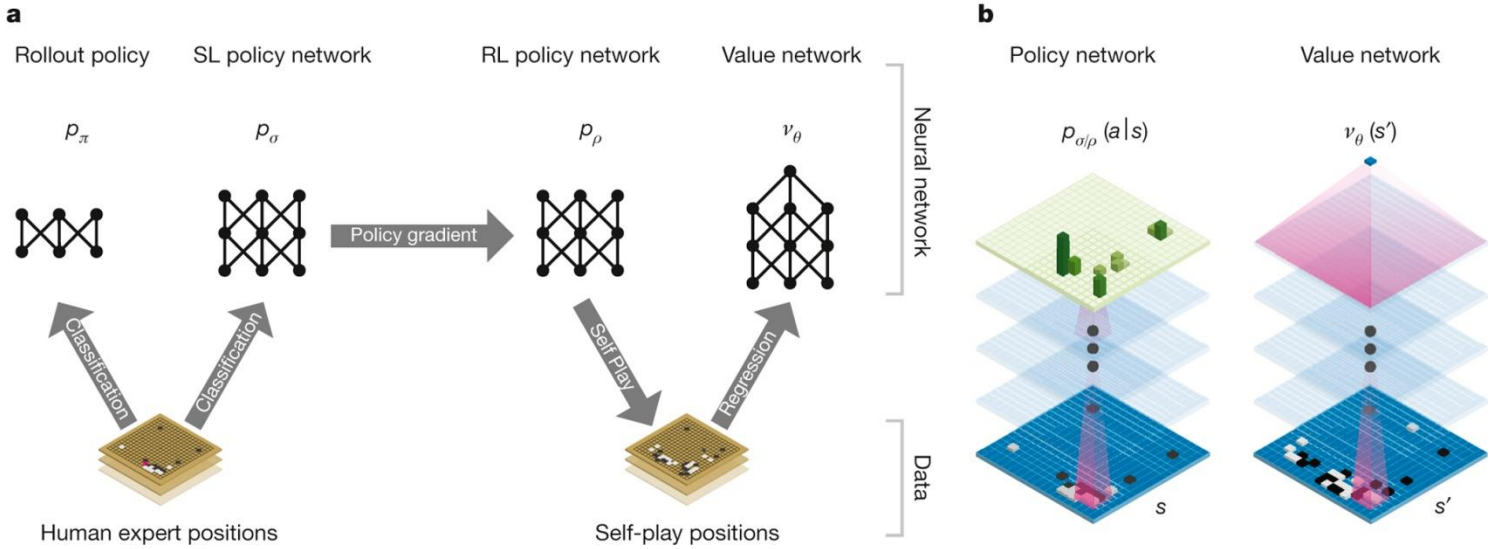
Machine Learning Reading Group at The Graduate Center, CUNY

March 11th 2016

The Key Ingredients

- Convolutional Neural Networks
  - What is the state of the game board?
  - Reduce complexity of the game state.
- Reinforcement Learning
  - What moves to make? How good is a move?
  - Policy and value.
- Monte Carlo Tree Search
  - What moves lead to victory?
  - Approximate optimal minimax game tree.
- Distributed Computation
  - Do it all big and fast.

Neural network training pipeline and architecture



D Silver *et al. Nature* **529,** 484–489 (2016) doi:10.1038/nature16961

- p_π and p_σ are trained to predict moves in human expert games, giving $p(a|s)$.
- The rollout policy is a small network used to quickly simulate entire games.
- p_ρ refines p_σ to predict winning moves in self-play simulation through reinforcement learning.
- v_θ predicts the probability of each player winning given a board state.

nature

Input Features

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).

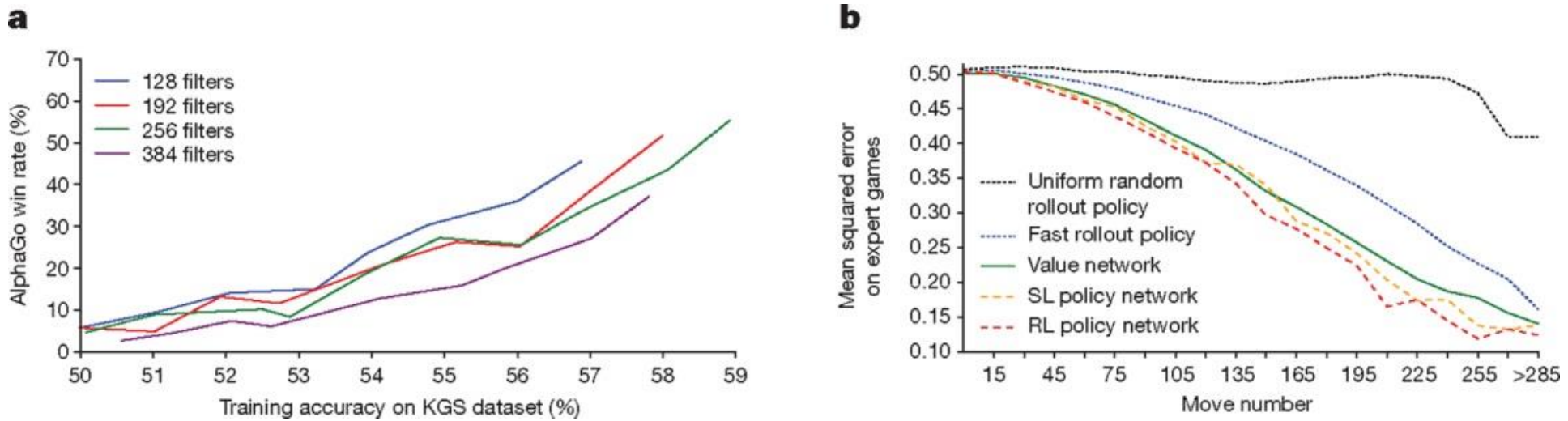
Input Features

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.

Strength and accuracy of policy and value networks



- a) Comparison of policy networks with varying number of convolutional filters versus the match ready version of AlphaGo.
- b) Comparison of value network and mean of 100 rollouts by various policies on predictions of outcomes on human expert games.

D Silver *et al. Nature* **529,** 484–489 (2016) doi:10.1038/nature16961

nature

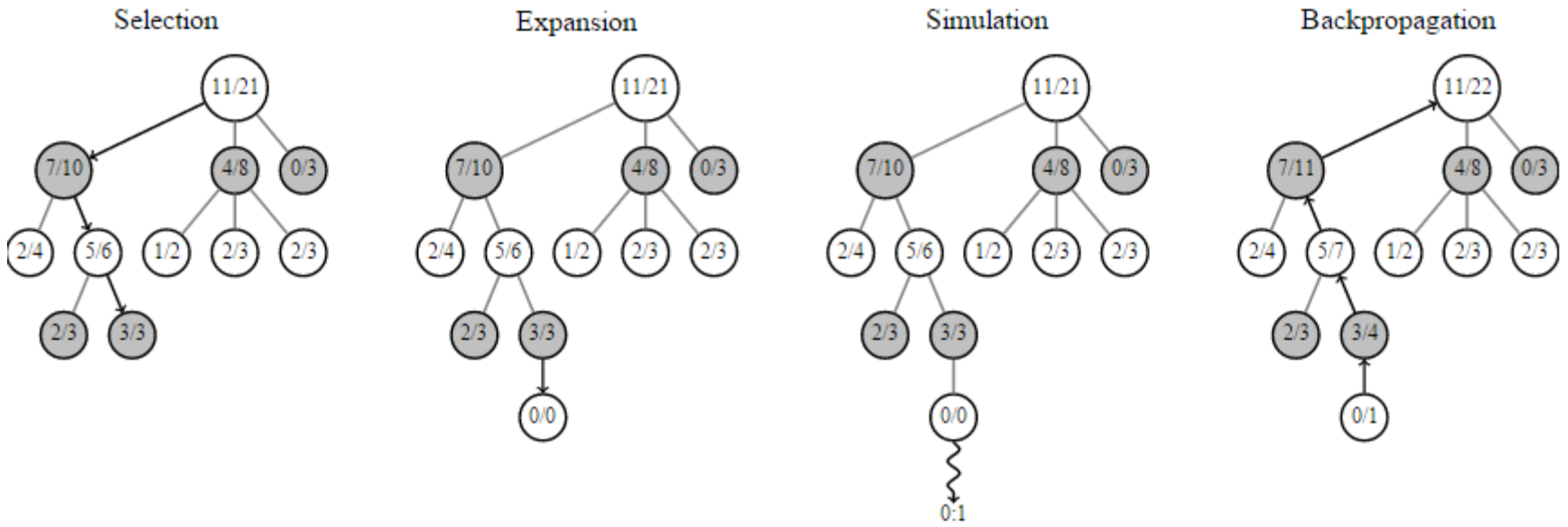
Reinforcement Learning

- Initialize p_ρ with p_σ, policy from human expert moves.
- Simulate game of $p_\rho^{(k)}$ with randomly selected $p_\rho^{k'}, k' < k$.
- Outcome at move t, z_t = +1 for winning and -1 for losing.
- SGD for each game move: $\Delta\rho \propto \frac{\partial \log p_\rho(a_t|s_t)}{\partial \rho} \cdot z_t$.
- 80% win rate versus p_σ.
- 85% win rate versus Pachi (previous best Go AI, search-based).

Reinforcement Learning

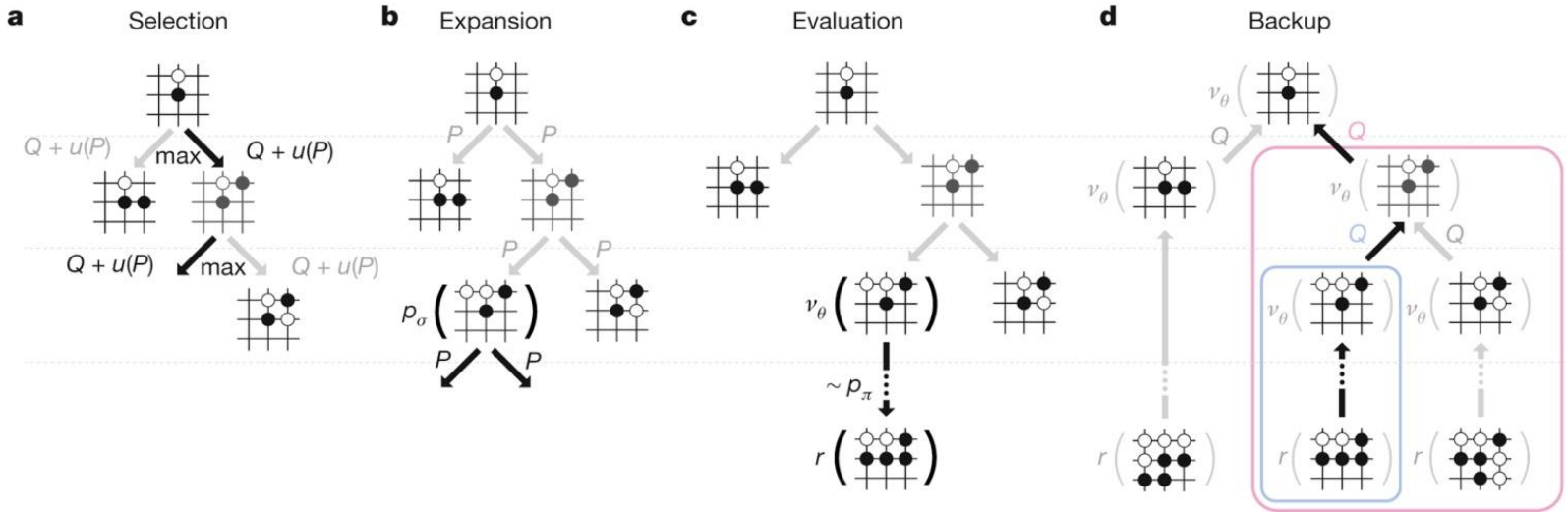
- $v^p(s) = \mathbb{E}[z_t \mid s_t = s, a_{t\ldots T} \sim p]$
- $v_\theta(s) \approx v^{p_\rho}(s) \approx v^*(s)$
- For state-outcome pairs (s, z), $\Delta\theta \propto \frac{\partial v_\theta(s)}{\partial \theta} \cdot \big(z - v_\theta(s)\big)$
- Successive positions in the same game are too strongly correlated, leads to overfitting.
- Sample (s, z) from different games in the self-play simulations.
- Approaches accuracy of rollouts of p_ρ with 15,000x less computation.

Monte Carlo Tree Search



- Explore move and simulate to game conclusion.
- Balance exploration vs exploitation.
- Converges to optimal value function with infinite simulations.

Monte Carlo tree search in AlphaGo



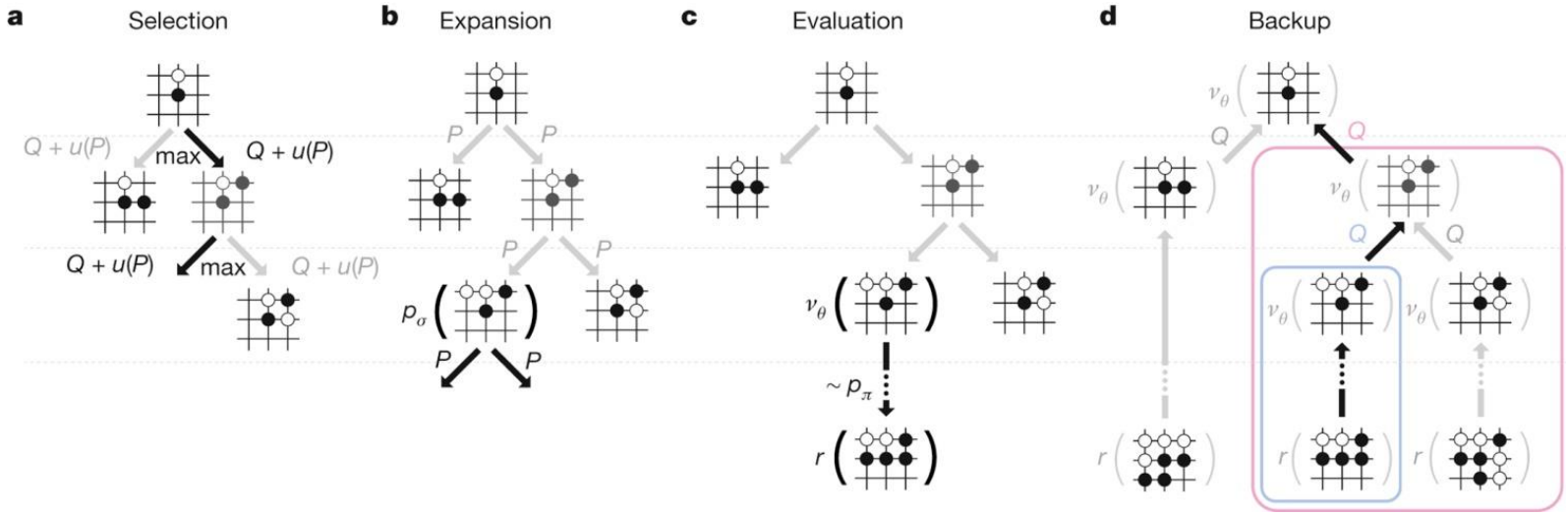
- a) Traverse tree from root to leaf my selecting actions with $\text{argmax}_a Q(s,a) + u(s,a)$, the action value plus a prior-based bonus $u(s,a) \propto \frac{P(s,a)}{1+N(s,a)}$, that decays with repeated visits.

- b) Expand leaf node and set prior $P(s,a) = p_\sigma(a|s)$.

D Silver *et al. Nature* **529,** 484–489 (2016) doi:10.1038/nature16961

nature

Monte Carlo tree search in AlphaGo



- c) Evaluate leaf node, $V(s_L) = (1-\lambda)v_\theta(s_L) + \lambda z_L$ where z_L is the rollout outcome using p_π.

- d) Update Q along tree path to track mean action value.

$$N(s,a) = \sum_{i=1}^{n} \mathbf{1}(s,a,i)\,, \qquad Q(s,a) = \frac{1}{N(s,a)} \sum_{i=1}^{n} \mathbf{1}(s,a,i) V\left(s_L^i\right).$$

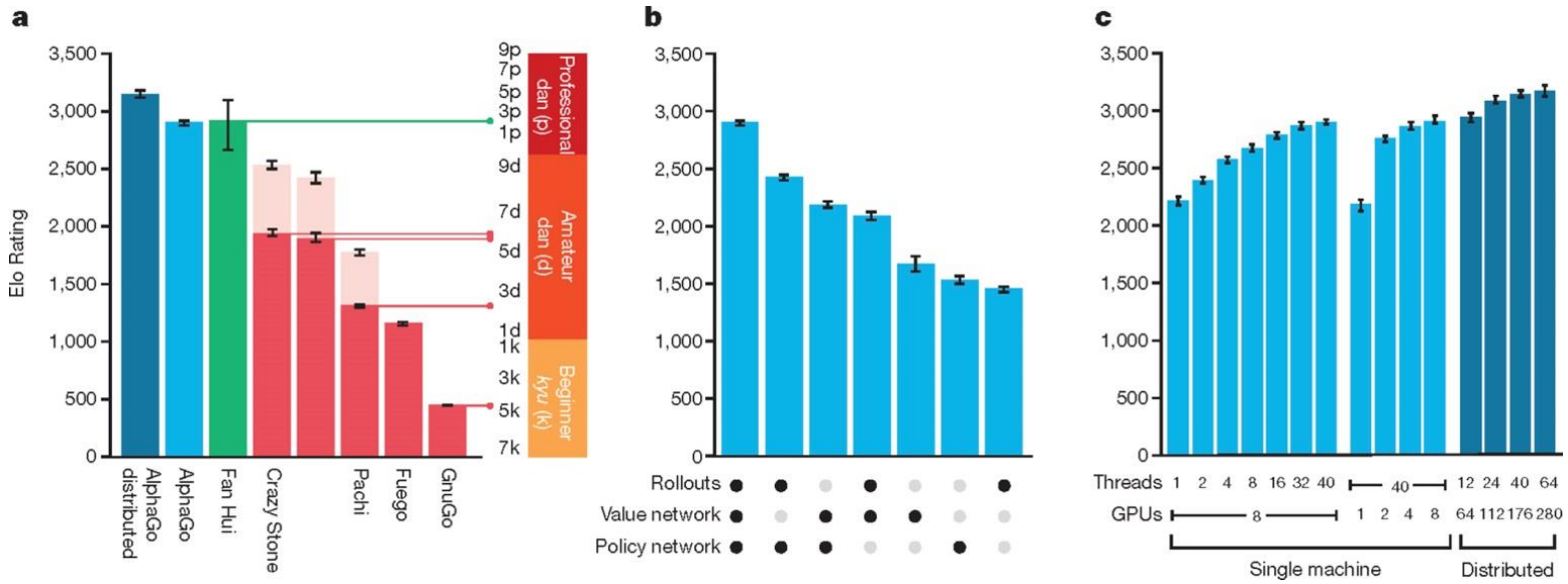
D Silver *et al. Nature* **529,** 484–489 (2016) doi:10.1038/nature16961

nature

The Networks in AlphaGo MCTS

- Note that p_σ served as a better prior than p_ρ, presumably because humans consider a diverse beam of moves whereas RL optimizes for a single best move.
- However the value network derived from the RL policy network was better than the value network derived from the SL policy network.
- NNs are relatively expensive search heuristics for MCTS.
- Asynchronous multithreaded search:
  - 40 search threads, 48 CPUs, 8 GPUs.
- Distributed AlphaGo:
  - 40 search threads, 1202 CPUs, 176 GPUs.
- See the paper for more technical details.

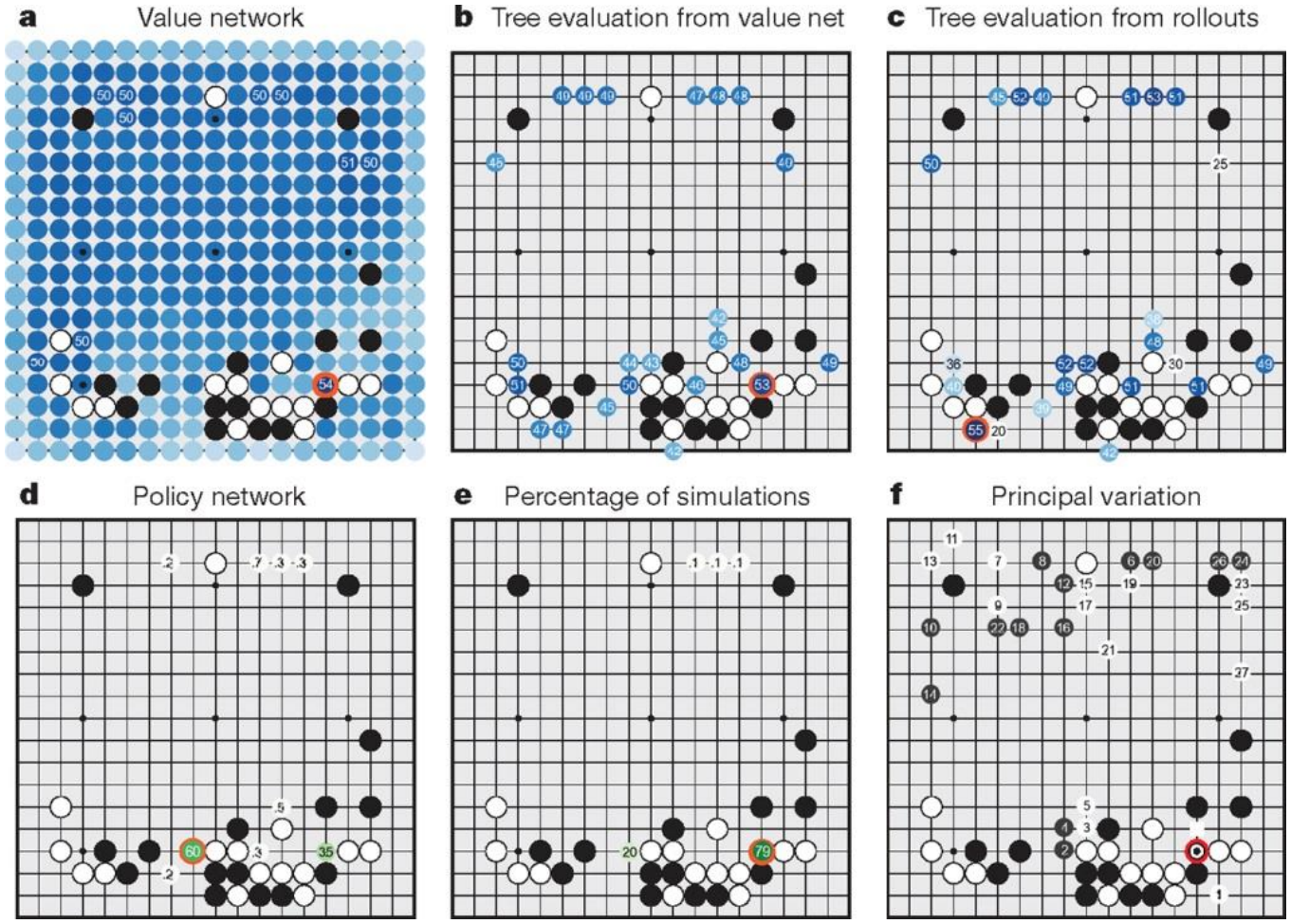
Tournament evaluation of AlphaGo



- Each program used about 5s per move.
- Pale bar indicates games where a four stone handicap was given.
- Games against human European champion Fan Hui used longer time control.
- Elo rating: 100 point gap $\sim 64\%$ win rate, 200 point gap $\sim 76\%$ win rate.

D Silver *et al. Nature* **529,** 484–489 (2016) doi:10.1038/nature16961 nature

How AlphaGo (black, to play) selected its move in an informal game against Fan Hui



a) $v_\theta(a|s)$
b) MCTS using only $v_\theta(s)$ $(\lambda = 0)$
c) MCTS using only $p_\pi(s)$ $(\lambda = 1)$
d) $p_\sigma(a|s) > 0.1\%$
e) % selections during MCTS
f) Most visited path is selected. White square indicates Fan Hui's move. In post-game commentary he preferred the move labeled 1 predicted by AlphaGo.

D Silver *et al. Nature* **529,** 484–489 (2016) doi:10.1038/nature16961

nature

Comparison with Deep Blue

- AlphaGo evaluated thousands of times fewer positions in its matches against Fan Hui than Deep Blue against Garry Kasparov.
- Deep Blue used a handcrafted evaluation function while AlphaGo was trained directly from gameplay data.
  - Some AlphaGo features are still handcrafted, particularly for rollout.
- AlphaGo demonstrates a better capability of approximately solving a seemingly intractable problem using less brute-force effort.

AlphaGo vs Lee Se-dol

- AlphaGo is the first AI Go system to beat a human pro player.
- Fan Hui is the European champion and a 2 dan pro.
- Lee Se-dol has won 18 international titles and is a 9 dan pro.
- Based on Elo ratings, Lee would win 75% of games against Fan, and this may be an underestimate since 9 dan is the maximum rank.

- AlphaGo plays Lee Se-dol on March 9^{th},10^{th} ,12^{th} ,13^{th} ,and 15^{th} at 11:00pm ET. Matches are expected to take 4 - 5 hours.
- The winner will get a $1,000,000 prize.