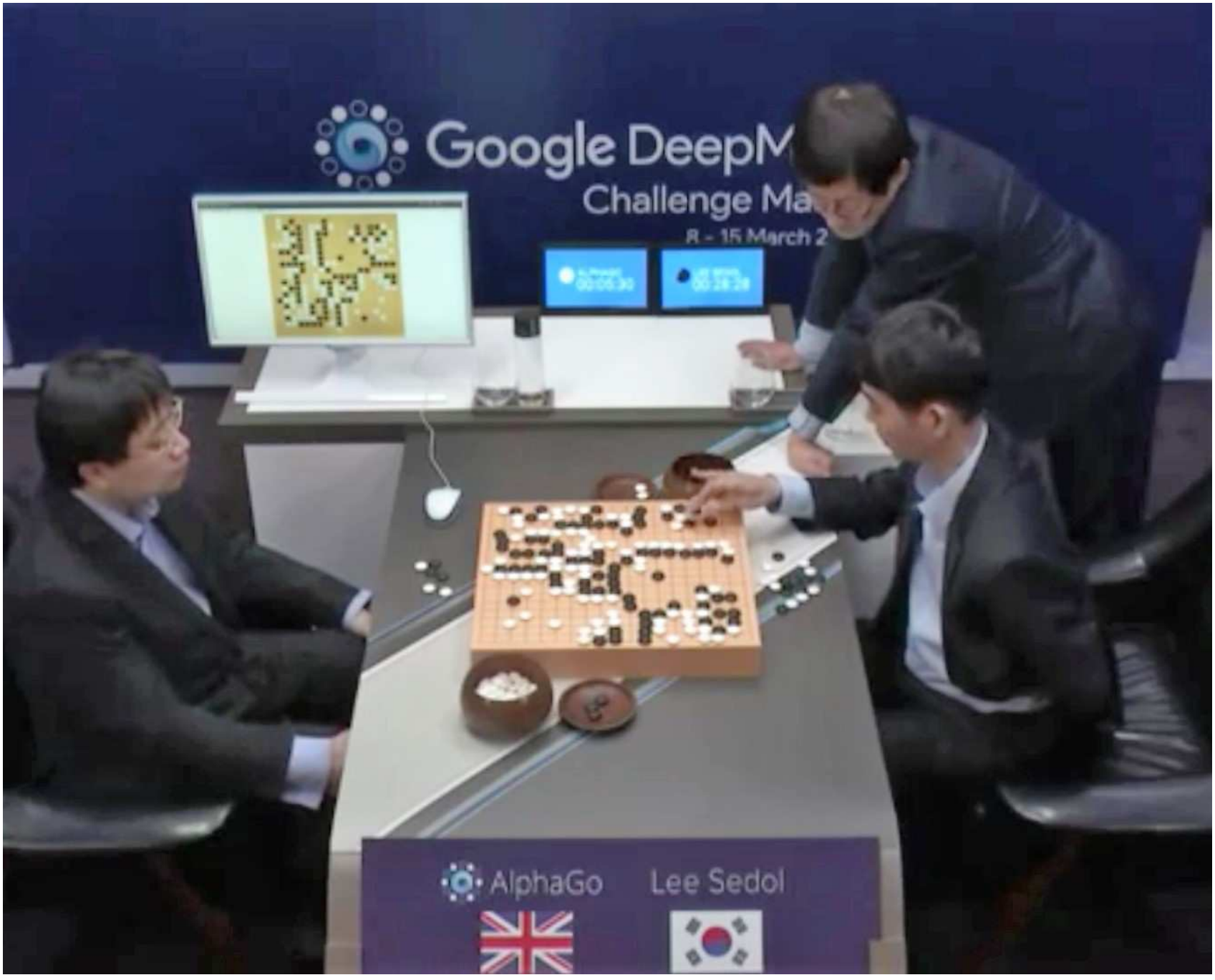
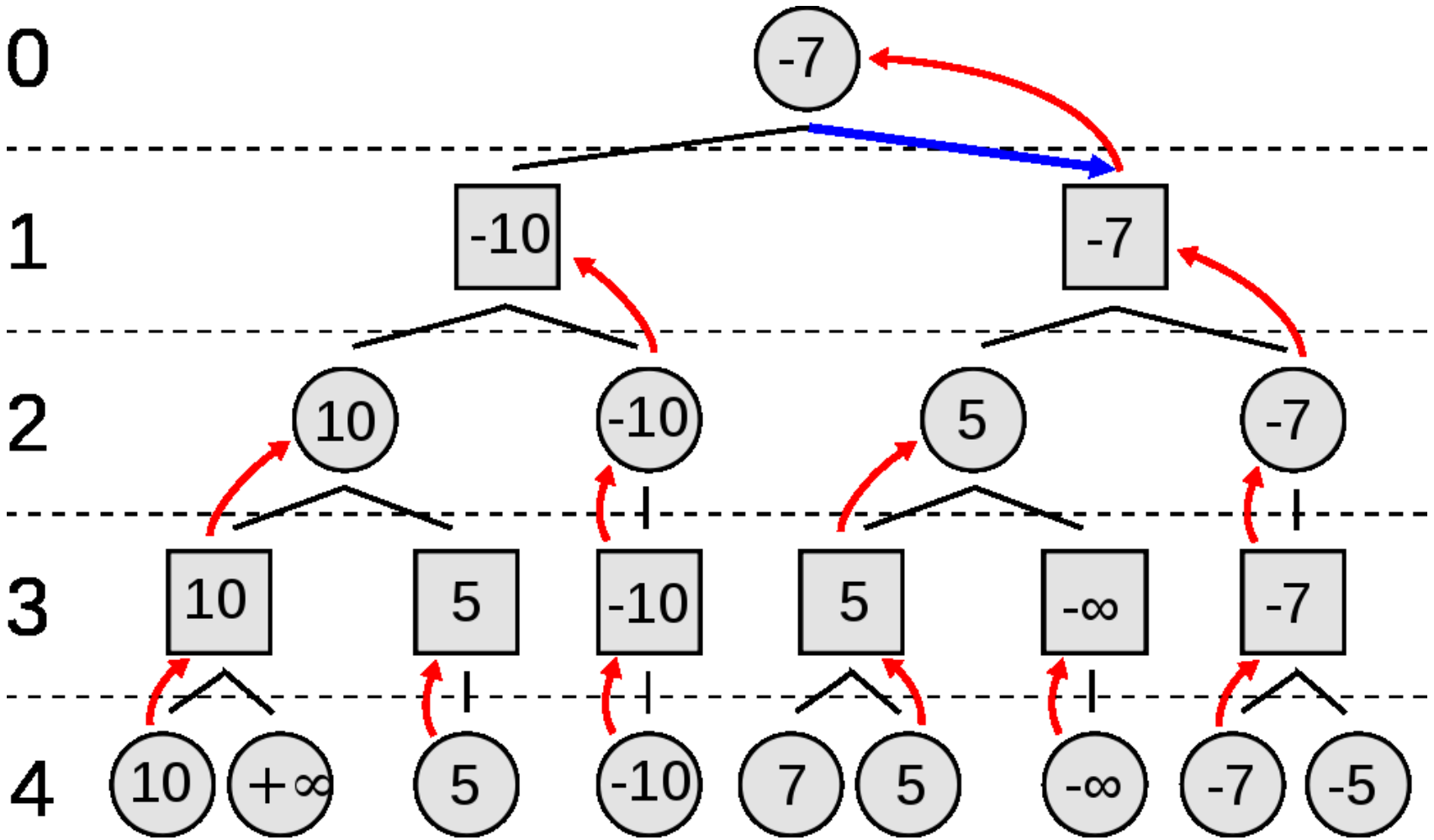

Google DeepM
Challenge Ma
R - 15 March 2
AlphaGo
Lee Sedol

Go

▸ The Way To Go


0
1
2
3
4
-7
-10
-7
10
-10
5
-7
10
5
-10
5
-∞
-7
10
+∞
5
-10
7
5
-∞
-7
-5

How many games?

Branching factor: At each position, n possible moves

Length: game lasts m moves

n^m possible games

Size of games

Game	Branching factor	Length	$\log_{10}$ (# Games)
Tic-Tac-Toe	4	9	5
Connect Four	4	36	21
Checkers (8×8)	2.8	70	31
Chess	35	70	108
Backgammon	250	55	132
Carcassonne	55	71	195
Go	250	150	360

Reducing the computational complexity

Length:

- Estimate the value of non-terminal states

Branching factor :

- Pruning
  - α-β: $O\left(\sqrt{n^m}\right)$ if moves are sorted
  - Ad-hoc methods...
- Monte Carlo Tree Search of promising moves

Intuition!

Static Evaluation

Checkers:

- Material
- Position of non-kings

Chess:

- Material
- Control centre of board

Go:

- "Strength" / "Influence"
  - Um...

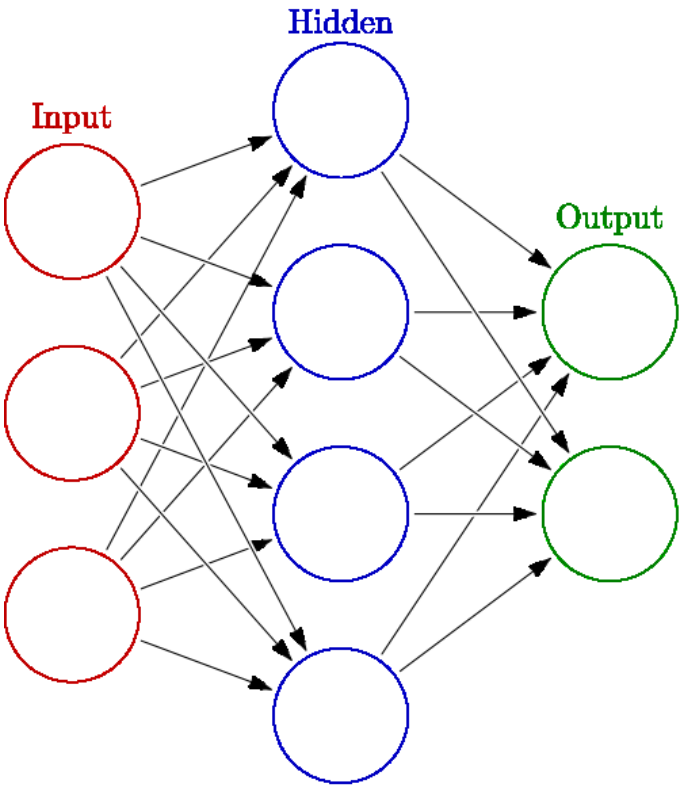
Move selection

Only examine promising moves.

> In the beginner's mind there are many possibilities. In the expert's mind there are few.

How to build an expert?

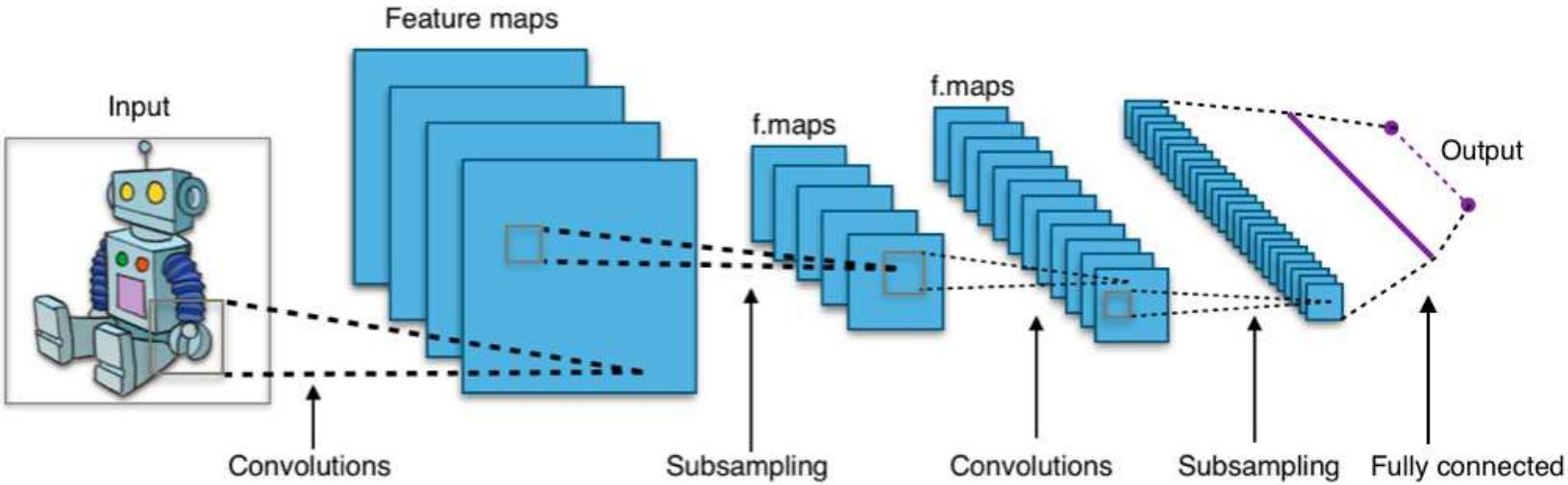
Neural Networks



- Input pattern ξ
- Weights W
- $y = W_1 \tanh(W_0 \xi)$
- W_i are trained with backpropagation of errors

Convolutional Neural Networks

Inspired by Hubel & Wiesel 1968



▸ LeNet

Supervised learning of policy networks: $p_\sigma(a|s)$

Predict the expert's move:

- Given board state s, predict probability of move a
- 13 convolutional layers
  - Input: 48 features
  - Layer 1: 192 of 5×5
  - Layers 2–12: 192 of 3×3
  - Rectifier nonlinearities
- Training: 30 million positions from KGS
- 57% accurate
- Evaluation time: 3 ms

$$\Delta\sigma \propto \frac{\partial \log p_\sigma(a_t|s_t)}{\partial\sigma}$$

Supervised learning of policy networks II: $p_\pi(a|s)$

Predict the expert's move (faster):

- Given board state s, predict probability of move a
- Small pattern features
- Training: 8 million positions from Tygem (?)
- 24% accurate
- Evaluation time: 2 μs

Why? To be continued...

Reinforcement learning of policy networks: $p_\rho(a|s)$

Don't predict expert moves: Win!

- Start with the supervised move predictor: $\rho \leftarrow \sigma$
- Training: self-play vs. a previous iteration of ρ
- Result of game is z

$$\Delta\rho \propto \frac{\partial \log p_\rho(a_t|s_t)}{\partial \rho} z$$

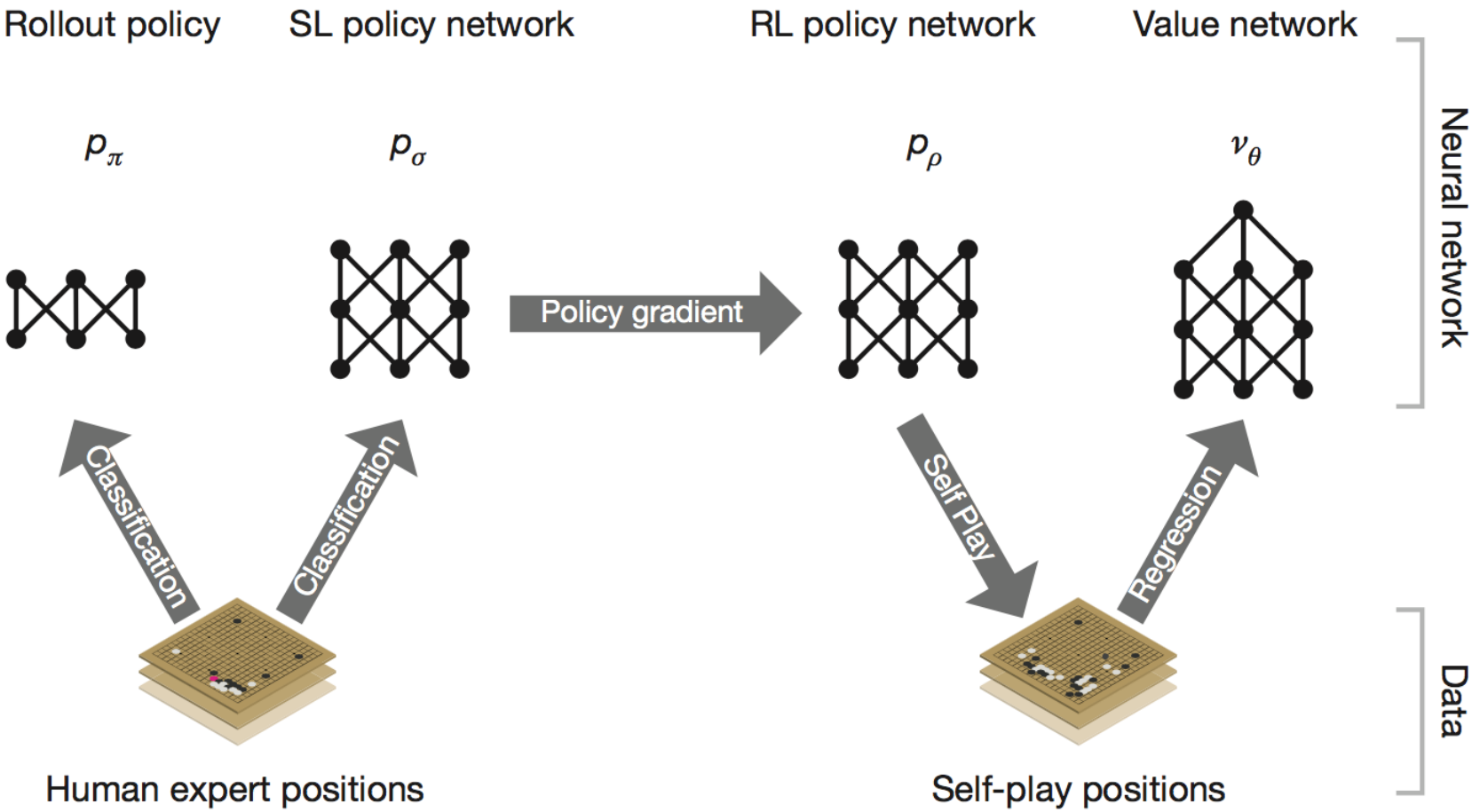
p_ρ won 80% of its games vs. p_σ

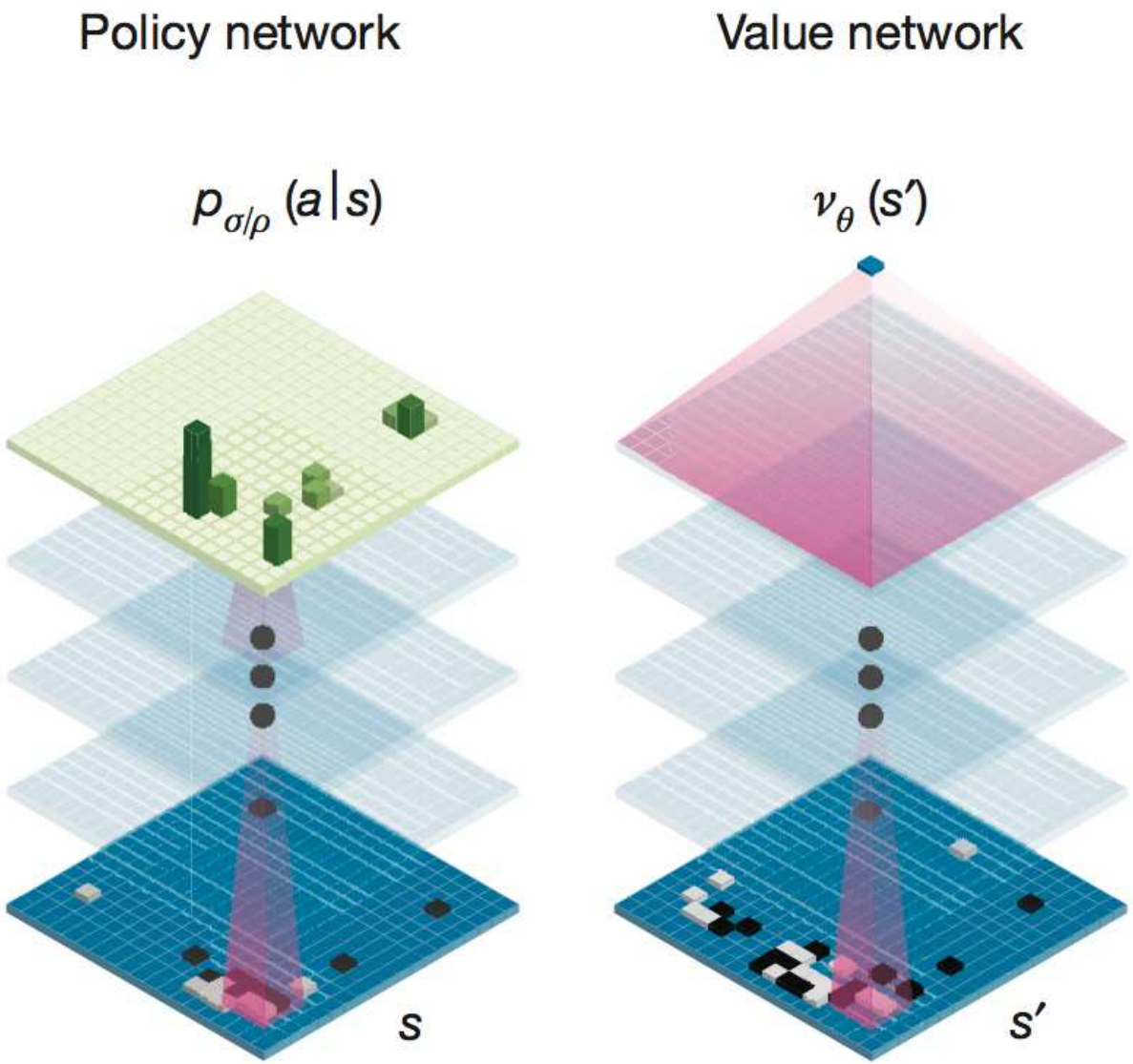
Reinforcement learning of value networks: $v^p(s)$

Predict the likelihood of winning from any board position

- $v^p(s) = \mathbb{E}\left[z|s = s_t, a_{t\ldots T} \sim p\right]$
- Similar architecture to p_ρ, but outputs scalar v
- Trained from 30 million samples from self-play by $p_\rho(s, a)$
  - Decorrelate sequences of moves

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


Rollout policy
SL policy network
RL policy network
Value network
p_π
p_σ
p_ρ
v_θ
Policy gradient
Classification
Classification
Self Play
Regression
Neural network
Data
Human expert positions
Self-play positions

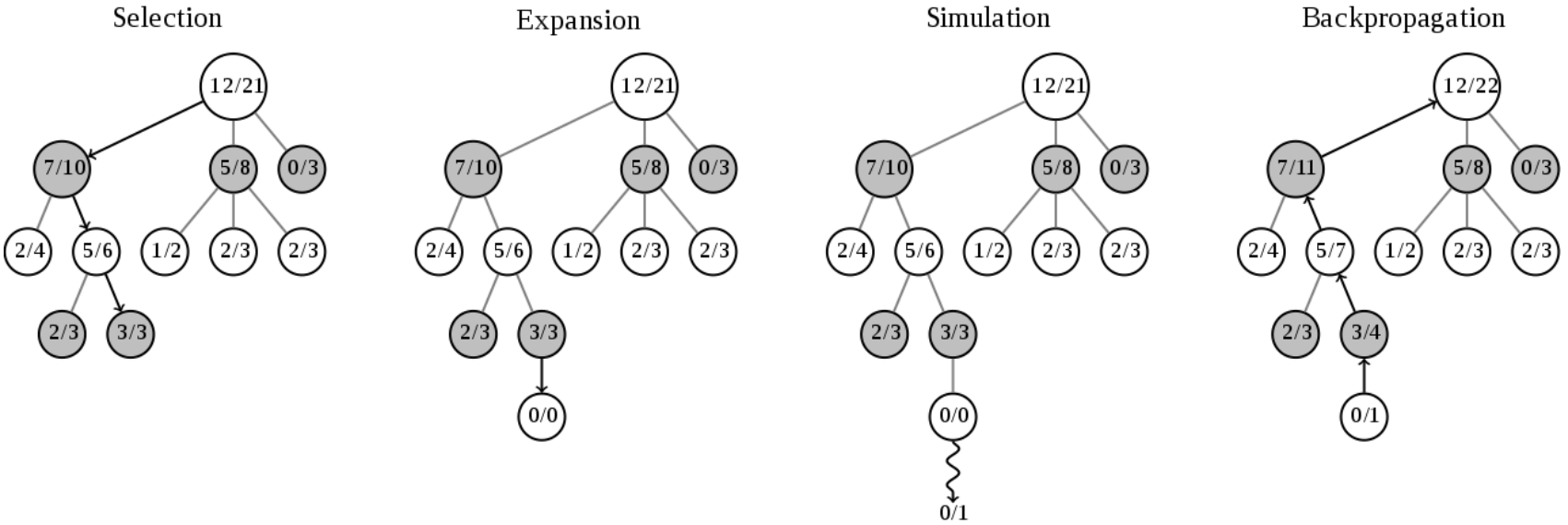

Policy network
$p_{\sigma/\rho}(a|s)$
s
Value network
$v_\theta(s')$
s'

Evaluating a potential move

- Choose action a from p_σ + exploration term
- Evaluate new position 2 ways:
  - $v_\theta(s)$
  - Monte Carlo rollout (using the fast policy p_π)
- Combine the evaluations

Monte Carlo Tree Search

New node: get priors from $p_\sigma(s, a)$ + exploration term



Early:

- High prior probability
- Low visit count

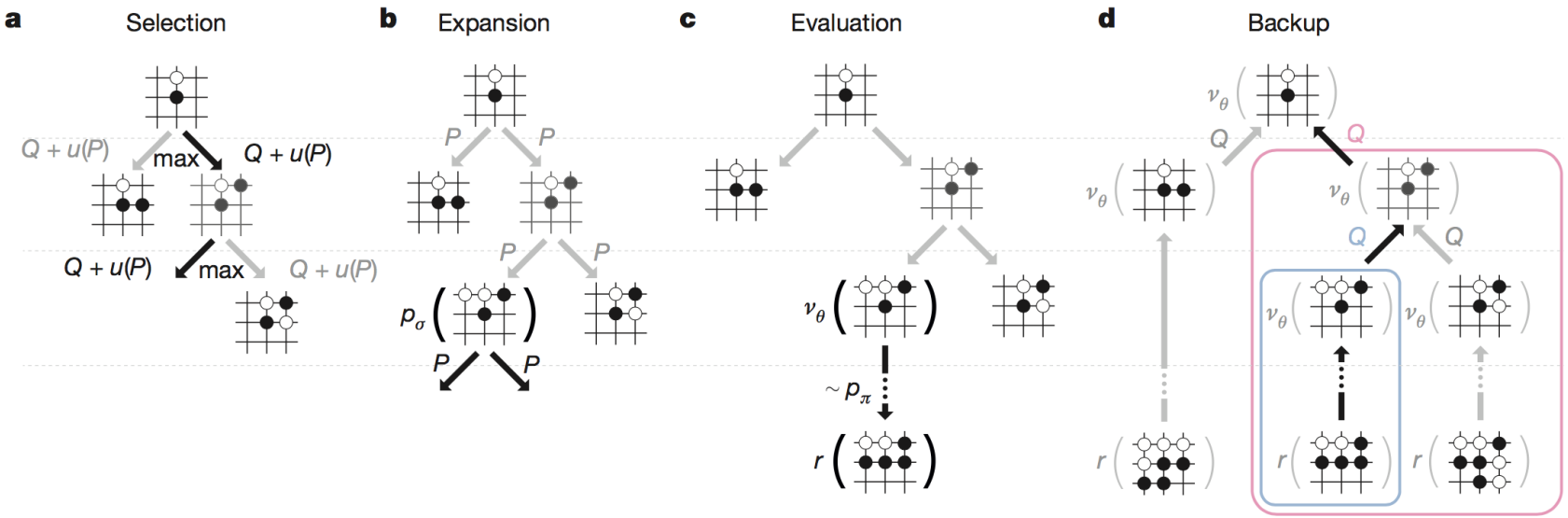
Later:

- High action value

Return the most visited move

Monte Carlo Tree Search

New node: get priors from $p_\sigma(s, a)$ + exploration term



Early:

- High prior probability
- Low visit count

Later:

- High action value

Return the most visited move

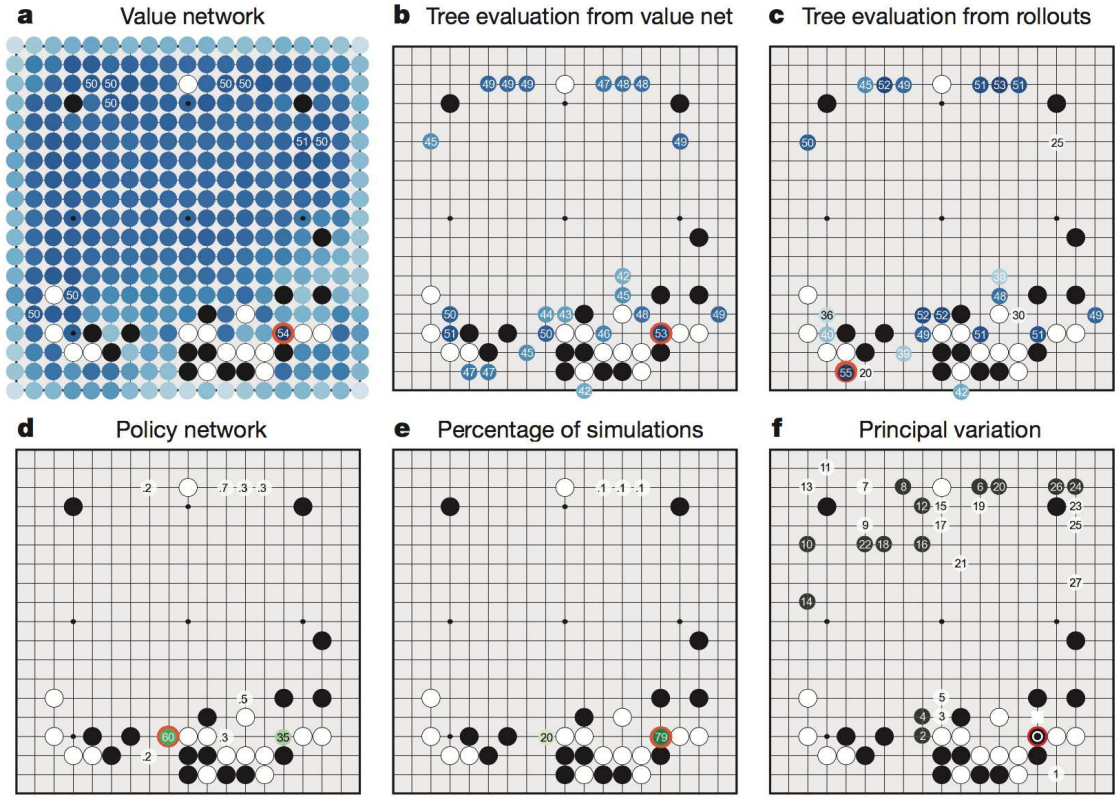
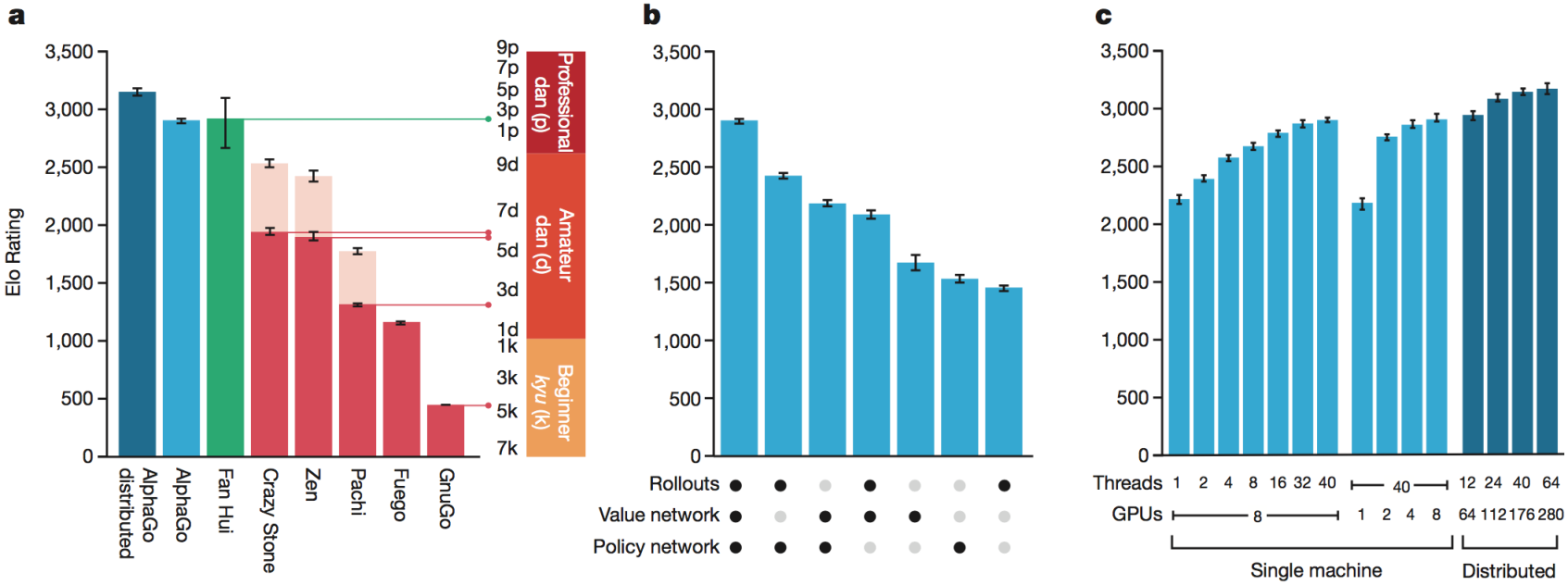


Figure 5 | How AlphaGo (black, to play) selected its move in an informal game against Fan Hui. For each of the following statistics, the location of the maximum value is indicated by an orange circle. **a**, Evaluation of all successors s' of the root position s, using the value network $v_\theta(s')$; estimated winning percentages are shown for the top evaluations. **b**, Action values $Q(s, a)$ for each edge (s, a) in the tree from root position s; averaged over value network evaluations only ($\lambda = 0$). **c**, Action values $Q(s, a)$, averaged over rollout evaluations only ($\lambda = 1$). **d**, Move probabilities directly from the SL policy network, $p_\sigma(a|s)$; reported as a percentage (if above 0.1%). **e**, Percentage frequency with which actions were selected from the root during simulations. **f**, The principal variation (path with maximum visit count) from AlphaGo's search tree. The moves are presented in a numbered sequence. AlphaGo selected the move indicated by the red circle; Fan Hui responded with the move indicated by the white square; in his post-game commentary he preferred the move (labelled 1) predicted by AlphaGo.

Results


a
Elo Rating
3,500
3,000
2,500
2,000
1,500
1,000
500
0
9p
7p
5p
3p
1p
9d
7d
5d
3d
1d
1k
3k
5k
7k
Professional dan (p)
Amateur dan (d)
Beginner kyu (k)
AlphaGo distributed
AlphaGo
Fan Hui
Crazy Stone
Zen
Pachi
Fuego
GnuGo
b
Rollouts
Value network
Policy network
c
Threads 1 2 4 8 16 32 40 40 12 24 40 64
GPUs 8 1 2 4 8 64 112 176 280
Single machine
Distributed

Victory!

- Beat Lee Sedol 9p 4-1
- 10^3 times fewer position evaluations than Deep Blue!
  - Learns to aggressively examine only a few promising moves
  - Learns intuition for values of intermediate positions
  - Plays like a human?