AlphaGo & AlphaGo Zero

Google DeepMind

Contents

- Background
  - David Silver
  - Elo Rating
  - Game Tree
  - Examples
- Frameworks
  - AlphaGo
  - AlphaGo zero
- Experiments

Background

David Silver
Google DeepMind
Professor of Computer Science
University College London

- Playing Atari with Deep Reinforcement learning_DQN (Nature-15)
- Mastering the game of Go with deep neural networks and tree search (AlphaGo, Nature-16)
- Mastering the game of Go without human knowledge (AlphaGo Zero, Nature-17, Science-19)
- Starcraft, Civilization, Poker, Chess, ……

http://www0.cs.ucl.ac.uk/staff/D.Silver/web/Applications.html

Background - Elo Rating

- An evaluation method for measuring the level of various game activities. Games, Go, Chess, Basketball, Football...
- Processes
  - Initialization point
  - Calculating winning probability of two players according to the difference of points
  - Each player updates his point based on opponent's points and game results

Background - Elo Rating

- Expectations of two players' Winning Probability

$$E_A = \frac{1}{1+10^{(R_A - R_B)/400}} \qquad E_B = \frac{1}{1+10^{(R_B - R_A)/400}}$$

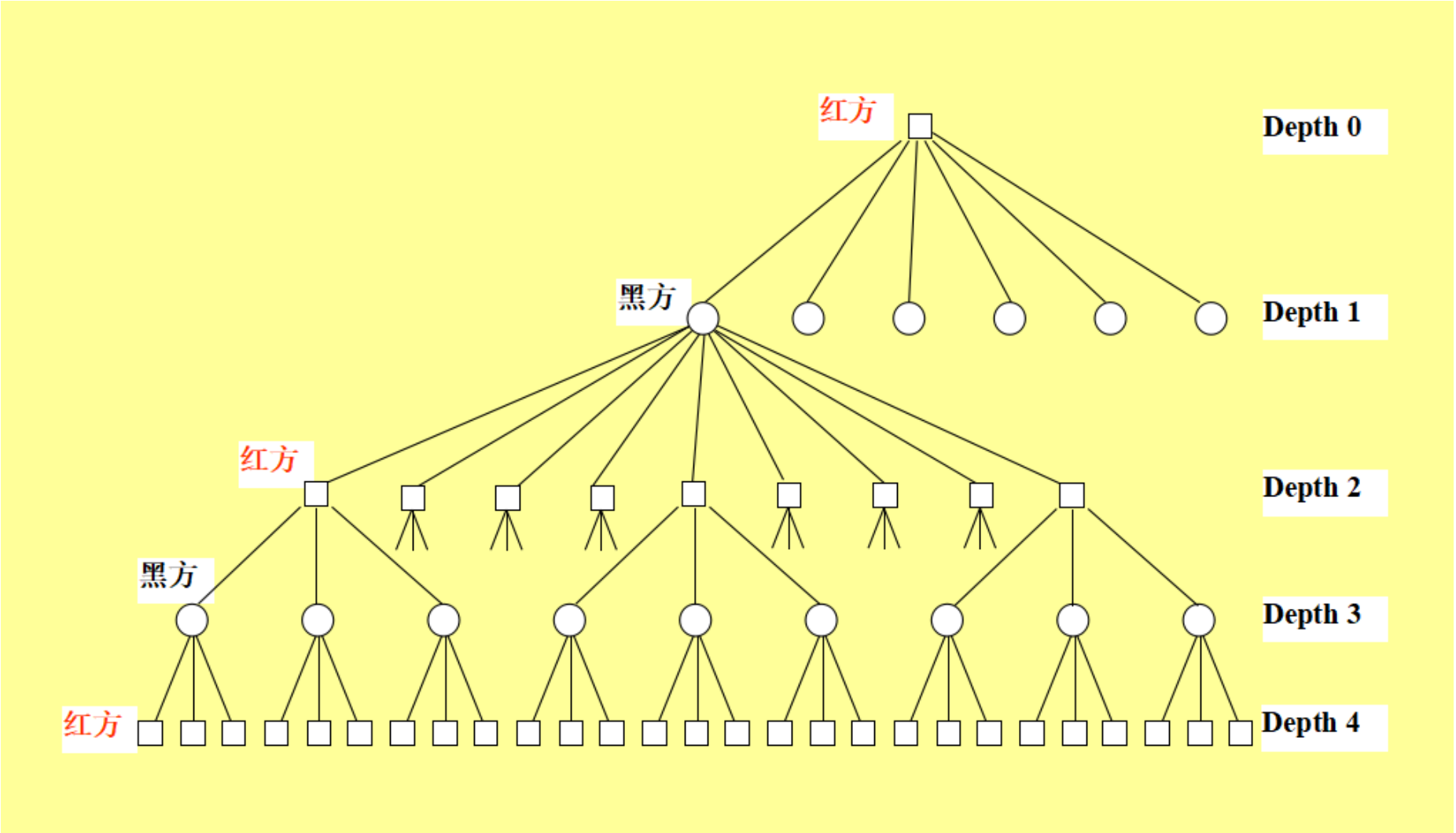
- Points updating

$$R'_A = R_A + K(S_A - E_A) \qquad R'_B = R_B + K(S_B - E_B)$$

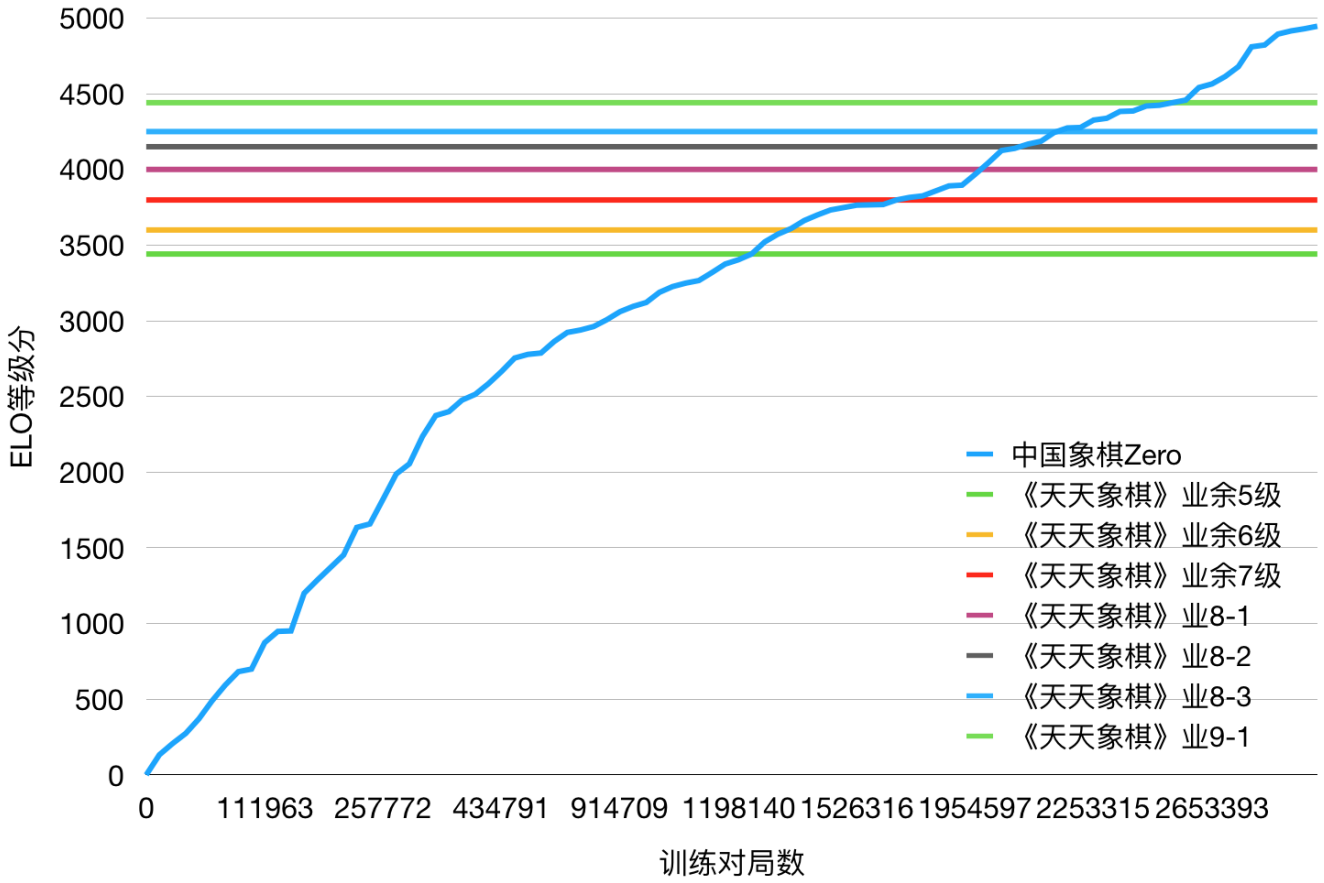
(**A Good Evaluation Method in Game**)

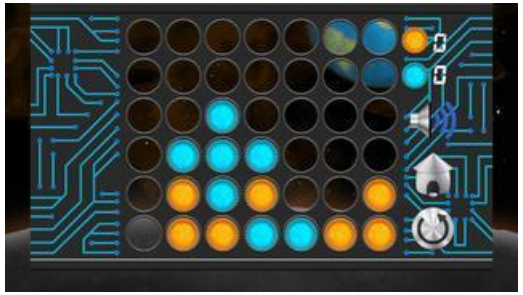
Background - Game Tree

alpha-beta



Background - Examples





```
AI move: 4,2

Player 1 with X
Player 2 with O

      0     1     2     3     4     5     6     7
  7   _     _     _     _     _     _     _     _
  6   _     _     _     _     _     _     _     _
  5   _     _     _     _     _     _     _     _
  4   _     _     X     _     _     _     _     _
  3   _     _     _     _     _     _     _     _
  2   _     _     _     _     _     _     _     _
  1   _     _     _     _     _     _     _     _
  0   _     _     _     _     _     _     _     _
```

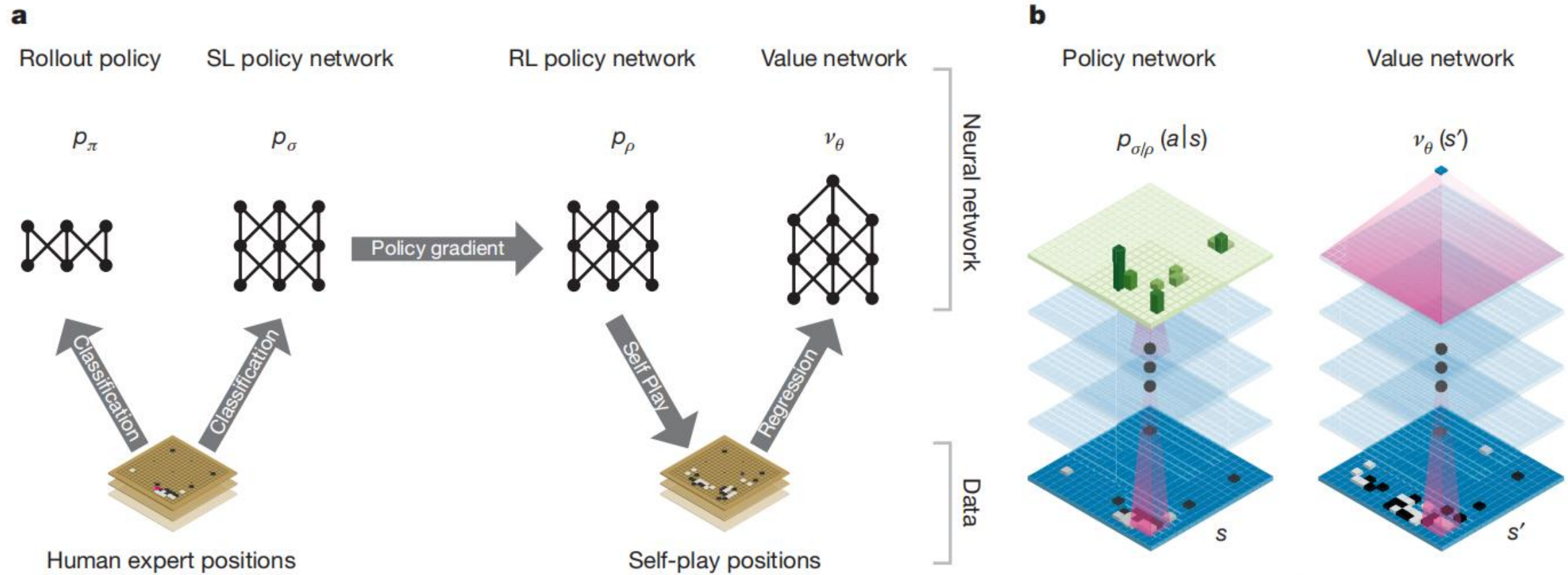


Figure 1 | Neural network training pipeline and architecture. a, A fast rollout policy p_π and supervised learning (SL) policy network p_σ are trained to predict human expert moves in a data set of positions. A reinforcement learning (RL) policy network p_ρ is initialized to the SL policy network, and is then improved by policy gradient learning to maximize the outcome (that is, winning more games) against previous versions of the policy network. A new data set is generated by playing games of self-play with the RL policy network. Finally, a value network v_θ is trained by regression to predict the expected outcome (that is, whether the current player wins) in positions from the self-play data set. **b**, Schematic representation of the neural network architecture used in AlphaGo. The policy network takes a representation of the board position s as its input, passes it through many convolutional layers with parameters σ (SL policy network) or ρ (RL policy network), and outputs a probability distribution $p_\sigma(a|s)$ or $p_\rho(a|s)$ over legal moves a, represented by a probability map over the board. The value network similarly uses many convolutional layers with parameters θ, but outputs a scalar value $v_\theta(s')$ that predicts the expected outcome in position s'.

Frameworks - AlphaGo

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).

KGS dataset

Frameworks - AlphaGo

- Reinforcement learning of policy networks
  - The RL policy network and SL policy network have the same structure and values.

$$\rho = \sigma$$

  - We play games between the **current policy network** and a randomly selected **previous iteration of the policy network**.
    (Prevent overfitting to the current policy)
  - Maximize the expected outcome

$$\Delta\rho \propto \frac{\partial \log P_\rho(\mathrm{a_t} \mid \mathrm{s_t})}{\partial \rho}\mathrm{z_t} \qquad \text{where} \quad \mathrm{a_t} \sim P_\rho(\cdot \mid \mathrm{s}_t),\ \mathrm{z_t} = \begin{cases} 1 & \text{win} \\ -1 & \text{lose} \\ 0 & \text{non-terminal} \end{cases}$$

Frameworks - AlphaGo VS. “绝艺”, AI Lab, Tencent

- Reinforcement learning of value networks
  - Estimating a value function $\mathrm{v}^{p}(s)$ that predicts the outcome from position s of games played by using policy P for both players

$$\mathrm{v}^{p}(s) = \mathrm{E}(z_{\mathrm{t}} \mid s_{\mathrm{t}} = s, \mathrm{a}_{\mathrm{t}\ldots\mathrm{T}} \sim P)$$

  - We estimate the value function $\mathrm{v}_{\theta}(s)$ for our strongest policy, using RL policy network P_{ρ} .
  - Minimize mean squared error (MSE)

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

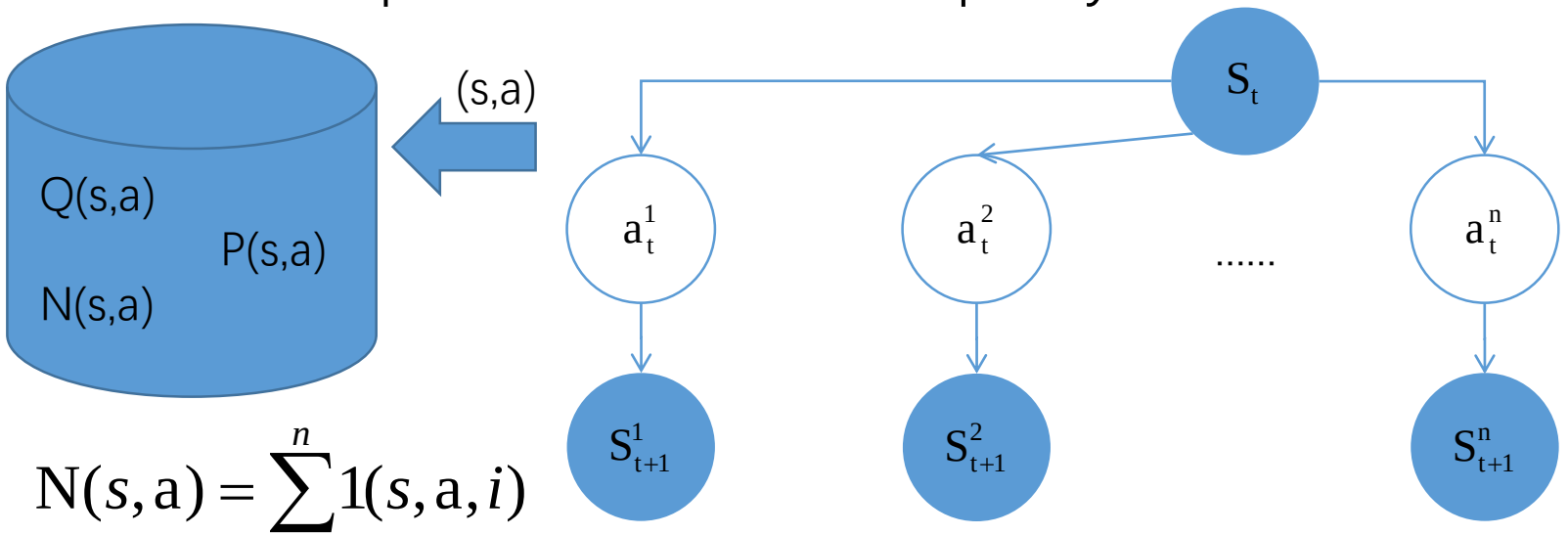
tricks

Method	Training Set	Test Set
1 (KGS dataset)	0.19	0.37
2 (self-play dataset)	0.234	0.226

MSEs

Frameworks - AlphaGo

- Searching with policy and value networks
  - AlphaGo combines the policy and value networks in an MCTS algorithm.



select

Upper Confidence Bound (UCB)

$$a_t = \arg\max_a (Q(s_t, a) + \mu(s_t, a))$$

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

$$N(s, a) = \sum_{i=1}^{n} 1(s, a, i)$$

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

$$P(s, a) = P_\rho(s, a)$$

Frameworks - AlphaGo

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

- Searching with policy and value networks

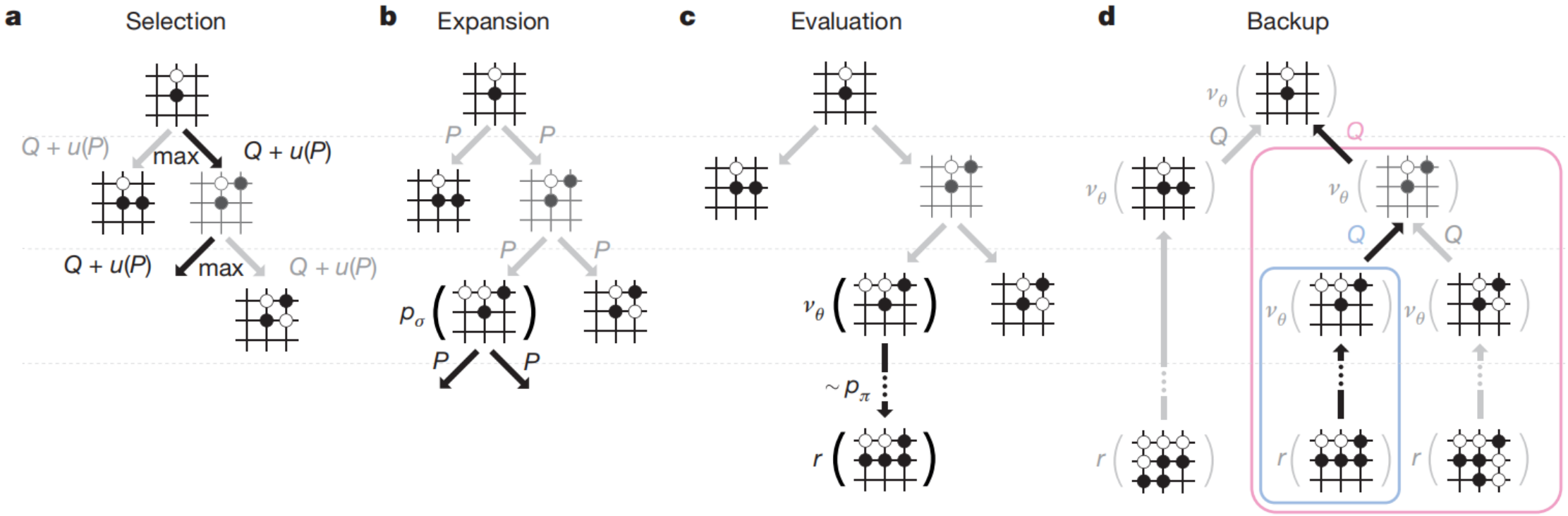
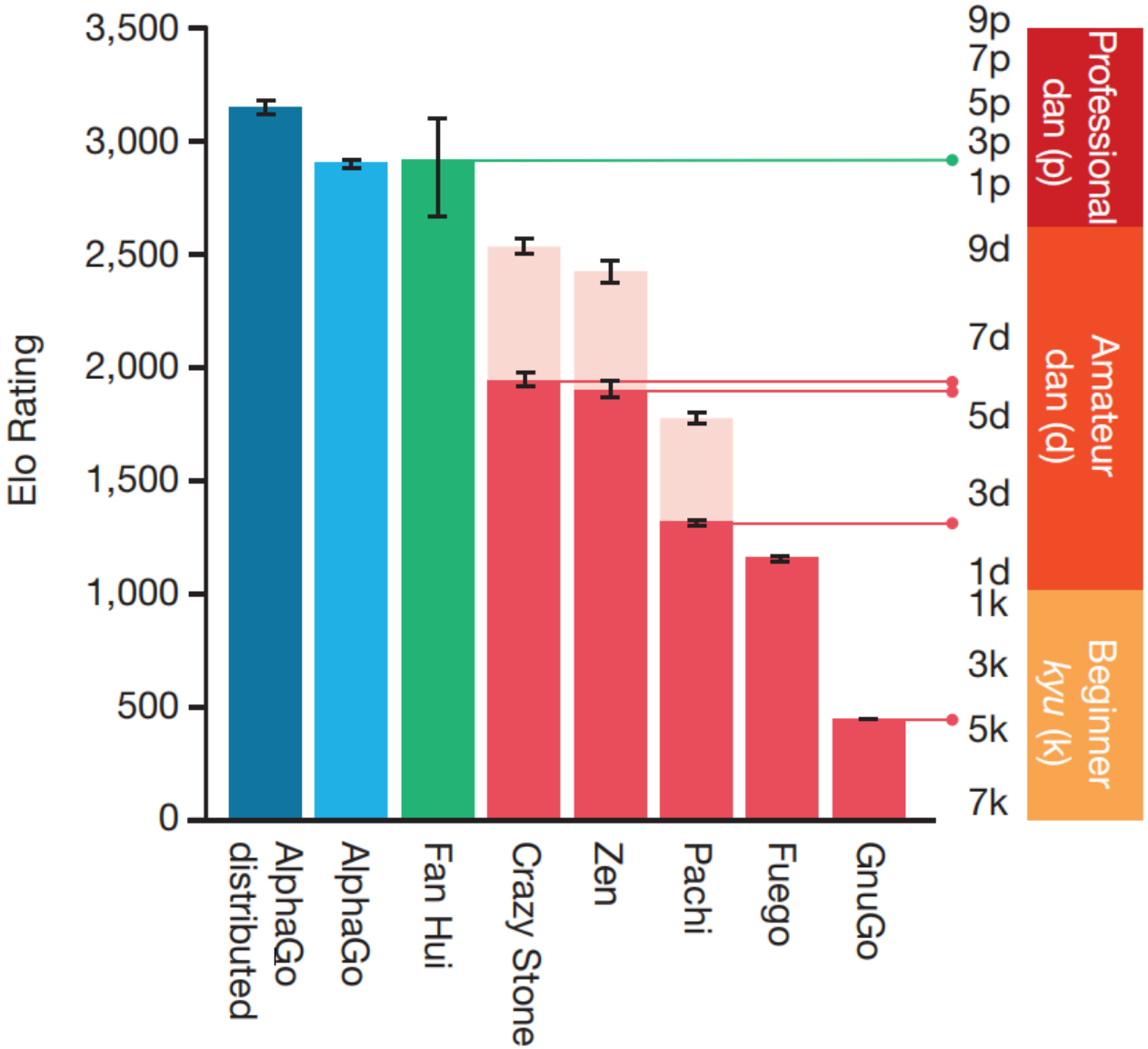


Figure 3 | Monte Carlo tree search in AlphaGo. a, Each simulation traverses the tree by selecting the edge with maximum action value Q, plus a bonus $u(P)$ that depends on a stored prior probability P for that edge. **b**, The leaf node may be expanded; the new node is processed once by the policy network p_σ and the output probabilities are stored as prior probabilities P for each action. **c**, At the end of a simulation, the leaf node is evaluated in two ways: using the value network v_θ; and by running a rollout to the end of the game with the fast rollout policy p_π, then computing the winner with function r. **d**, Action values Q are updated to track the mean value of all evaluations $r(\cdot)$ and $v_\theta(\cdot)$ in the subtree below that action.

Experiments - AlphaGo



Frameworks - AlphaGo Zero

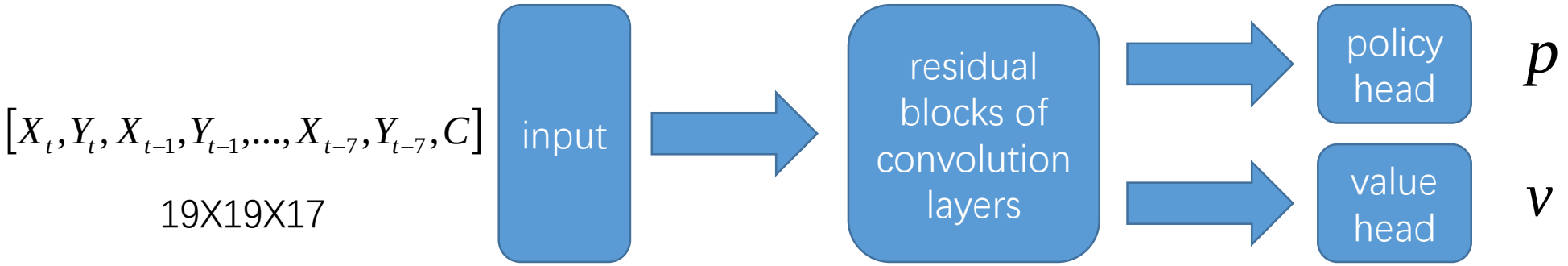
- Achievement
  - Winning 100-0 against the previously published, champion-defeating AlphaGo.
  - Without any human data, guidance or domain knowledge.
  - The form is simpler, the design is more reasonable and the method is more effective.

Frameworks - AlphaGo Zero

- Reinforcement learning in AlphaGo Zero
  - Use a deep neural nework f_θ with parameters θ to estimate both move probabilities and a value.

$$(p, v) = f_\theta(s)$$

  - This neural network combines the roles of both policy network and value network into a single architecture.



256 kernels size 3X3 with stride 1

Frameworks - AlphaGo Zero

- Reinforcement learning in AlphaGo Zero
  - Use a deep neural nework f_θ with parameters θ to estimate both move probabilities and a value.

$$(p, v) = f_\theta(s)$$

  - In each postition s, and MCTS search is executed, guided by the neural network f_θ. **The MCTS outputs probabilities π of playing each move.** (These search probabilities usually select much stronger moves than the raw move probabilites π of the neural network $f_\theta(s)$; MCTS may therefore be viewed as a powerful policy improvement operator.)

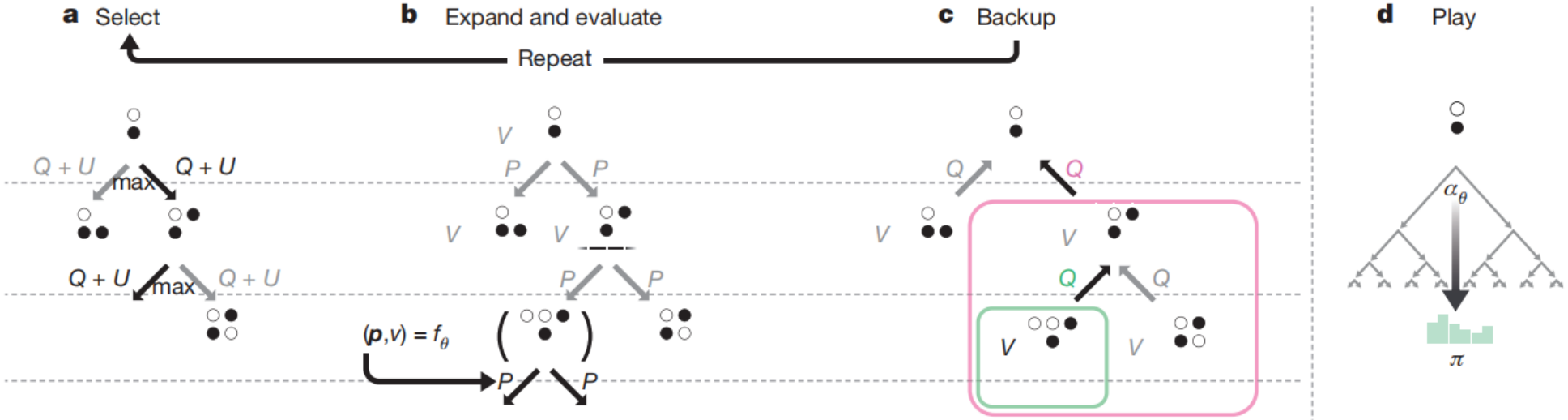
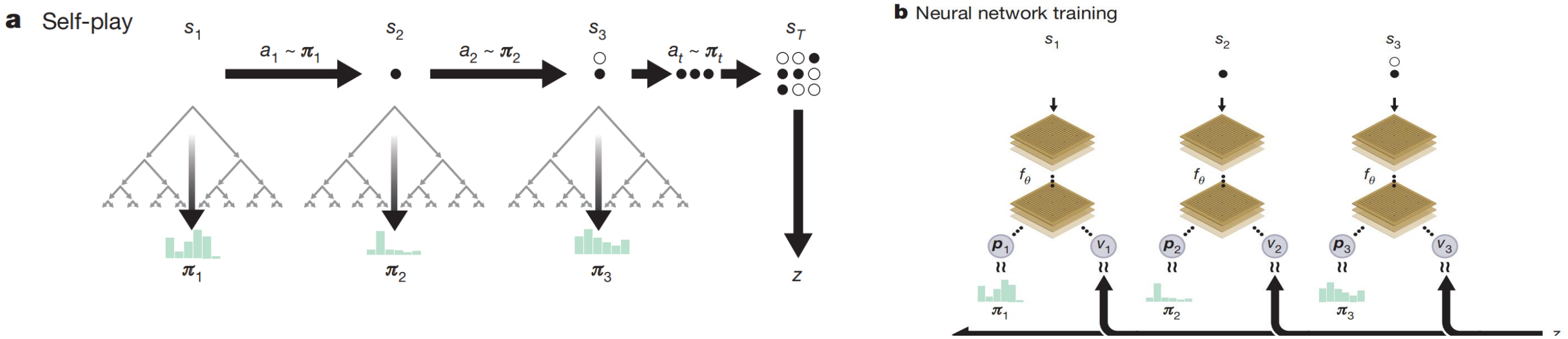


Figure 2 | MCTS in AlphaGo Zero. a, Each simulation traverses the tree by selecting the edge with maximum action value Q, plus an upper confidence bound U that depends on a stored prior probability P and visit count N for that edge (which is incremented once traversed). **b**, The leaf node is expanded and the associated position s is evaluated by the neural network $(P(s,\cdot),V(s)) = f_\theta(s)$; the vector of P values are stored in the outgoing edges from s. **c**, Action value Q is updated to track the mean of all evaluations V in the subtree below that action. **d**, Once the search is complete, search probabilities $\boldsymbol{\pi}$ are returned, proportional to $N^{1/\tau}$, where N is the visit count of each move from the root state and τ is a parameter controlling temperature.



Frameworks - AlphaGo Zero

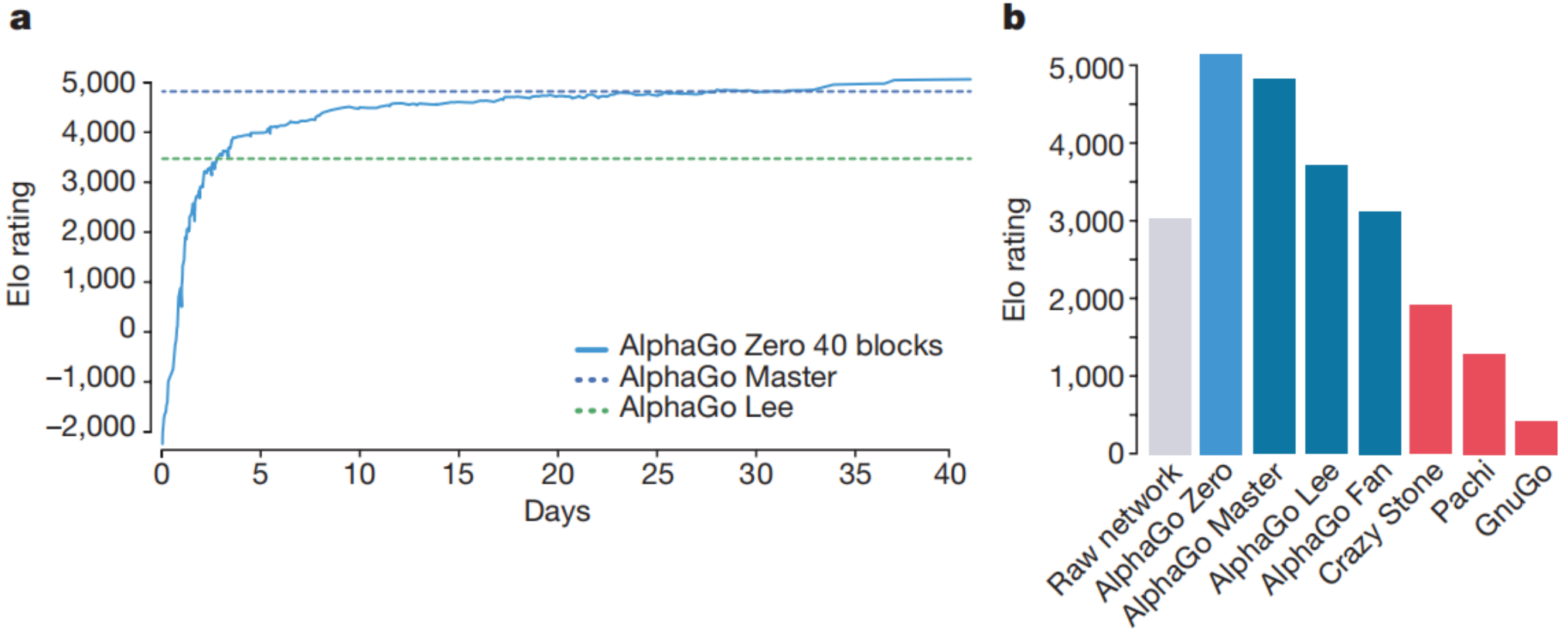
- Reinforcement learning in AlphaGo Zero
  - The parameters θ are adjusted by gradient desent on a loss function l that sums over the mean-squared error and cross-entropy losses.

$$(p, v) = f_\theta(s) \quad \text{and} \quad l = (z - v)^2 - \pi^t \log p + c \| \theta \|^2$$

    - where C is a parameter controlling the level of L2 weight regularization. (to prevent overfitting)

Update θ

Experiments - AlphaGo Zero



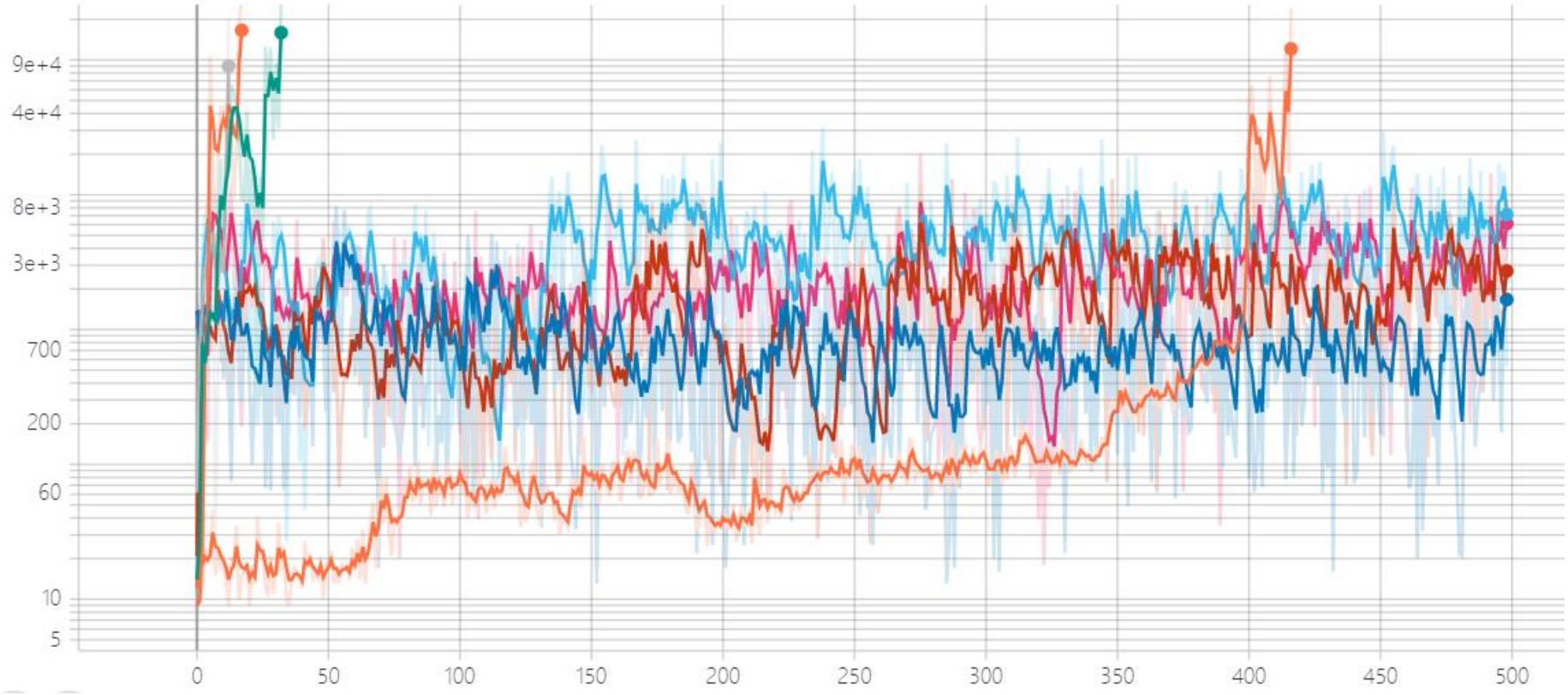
Experiments - AlphaGo Zero

自己	对手	结果
AlphaGo Fan	Fan Hui (欧洲围棋冠军)	5:0
AlphaGo Lee	李世石 (围棋世界冠军)	4:1
AlphaGo Master	10+围棋高手	60:0
AlphaGo Master (之后宣布退出围棋比赛)	柯洁 (人类第一)	3:0
AlphaGo Zero (不选择和人类进行比赛, 只和机器内战)	AlphaGo Lee	100:0

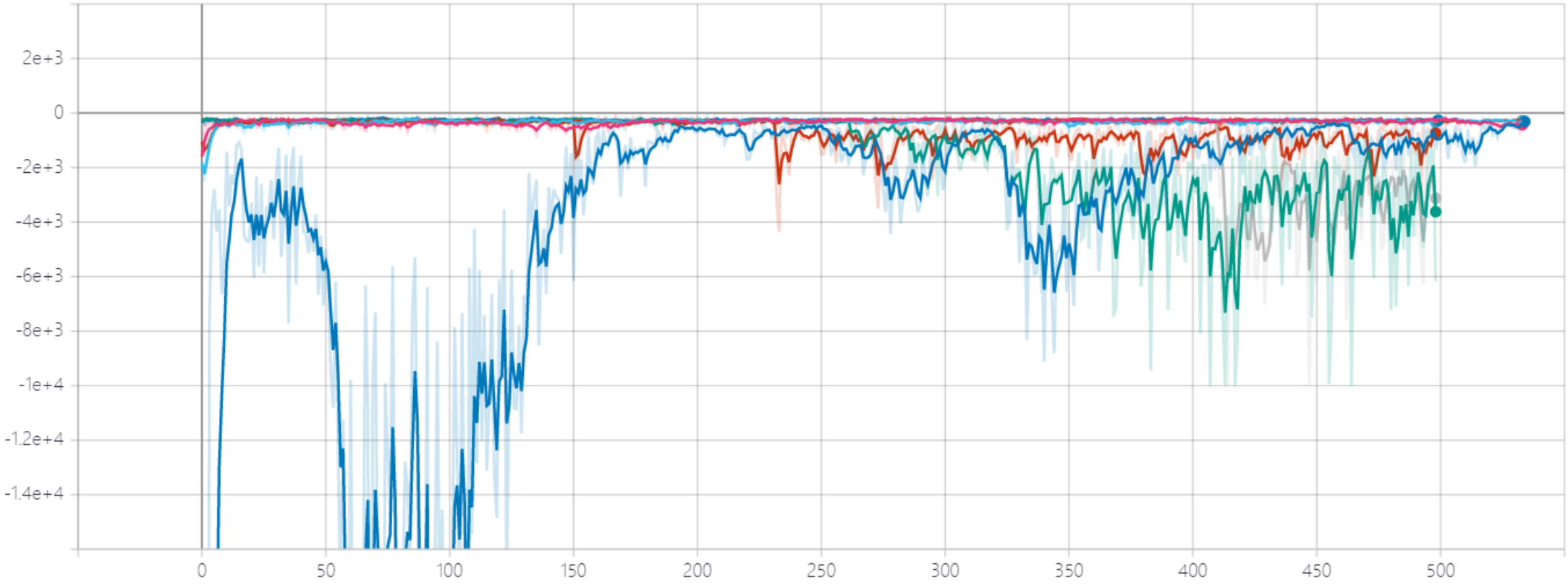
Experiments

- Imitation learning
- Mujoco Environment

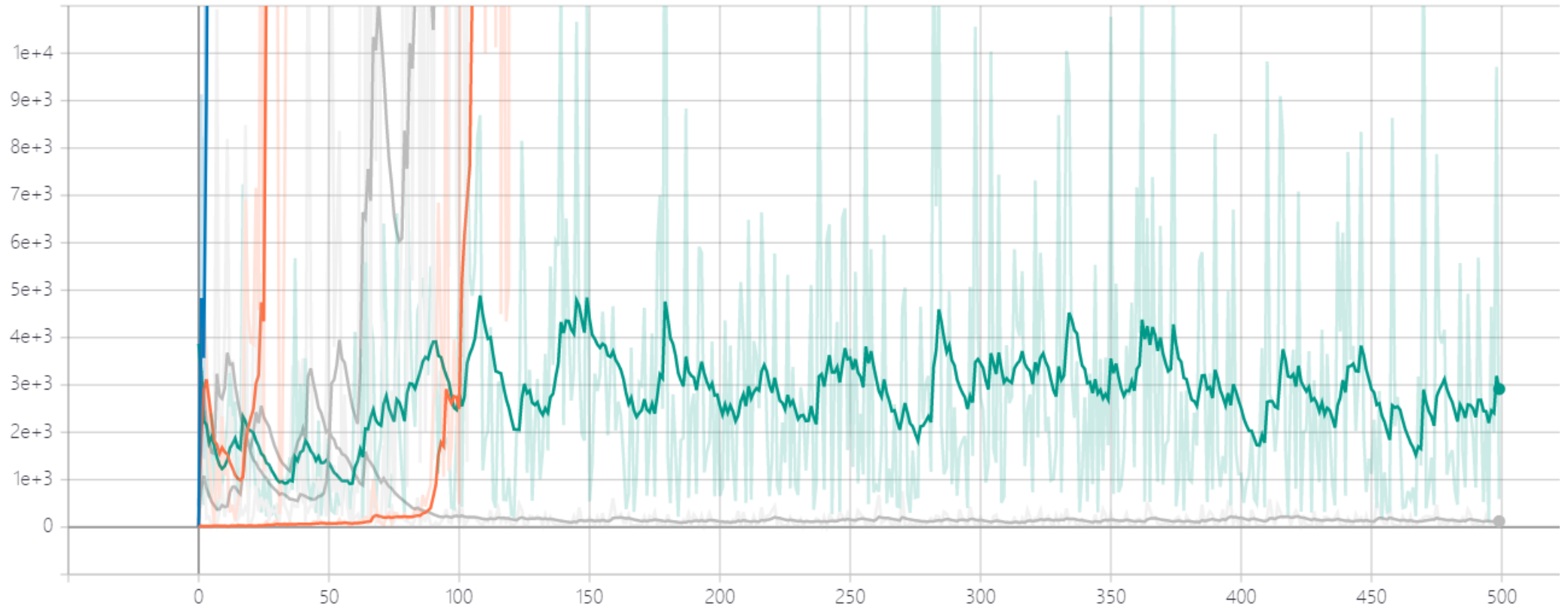
Experiments - PG_CartPole



Experiments - PG_MountainCar



Experiments - ATRPO_CartPole



Experiments - ATRPO_MountainCar