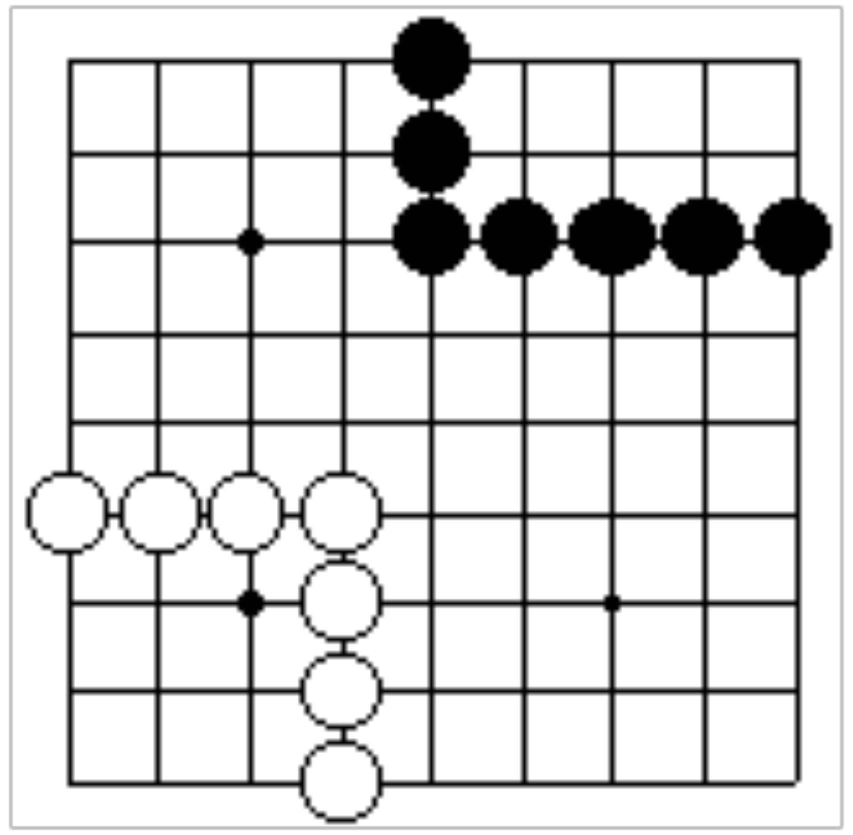
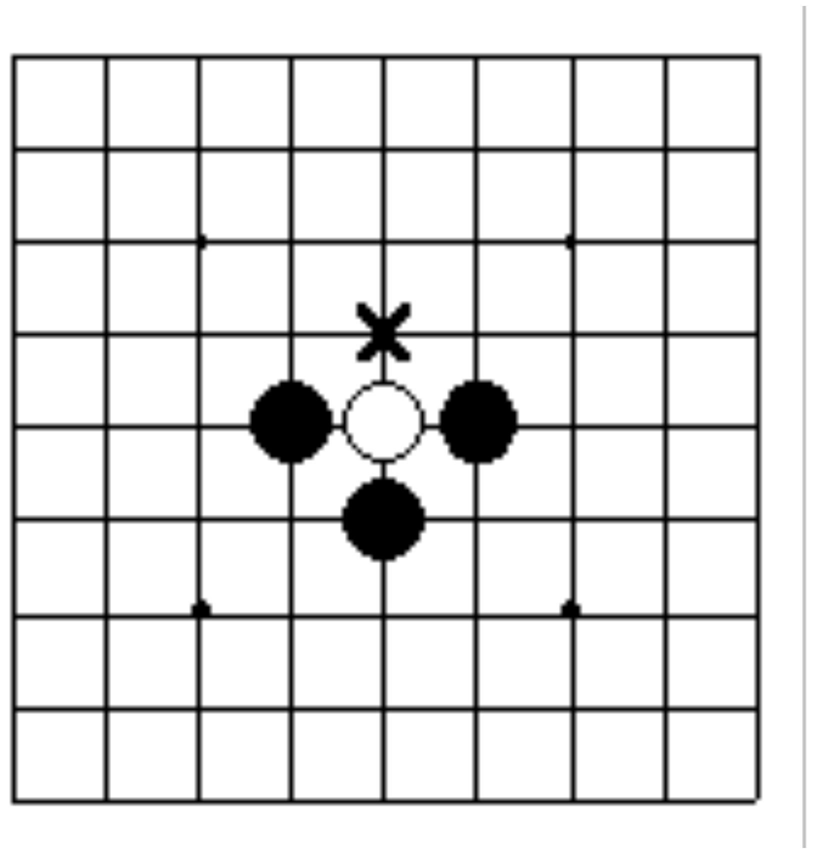
Mastering the game of Go with deep neural networks and tree search

Котельников Аким, БПМИ182

Go Game

- 19x19 game board
- A player's area consists of all the points the player has either occupied or surrounded. The player with more area wins.
- $b^d \approx 250^{150}$ possible sequences of moves, where b is the games' breadth and d is the game's depth. (For chess $b \approx 35,\ d \approx 80$)





Stages of AlphaGo

1. Supervised Learning (SL) policy network p_σ
2. Rollout policy network p_π
3. Reinforcement Learning (RL) policy network p_ρ
4. Value network v_θ
5. Monte-Carlo tree search (MCTS)

SL Policy Network

- *Architecture:* 13 convolutional layers, non-linear activation functions, softmax
- *Input:* board representation using simple features
- *Output:* probability distribution over all legal moves
- *Training:* gradient ascent to maximize the likelihood of the human move
- *Result:* 56% test accuracy

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).

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

Rollout Policy Network

- *Architecture:* linear, softmax
- *Input:* board representation using other features
- *Output:* probability distribution over all legal moves
- *Result:* 24% test accuracy, but using $2\mu s$ to select an action instead of 3ms

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, $\geq$3) at each intersection of the pattern.

RL Policy Network

- Is identical in structure to the SL policy network
- Weights ρ are initialized to the same values σ
- Trained by playing games between the current policy network p_ρ and a randomly selected previous iteration of the policy network
- Updating weights ($z_t = \pm 1$):

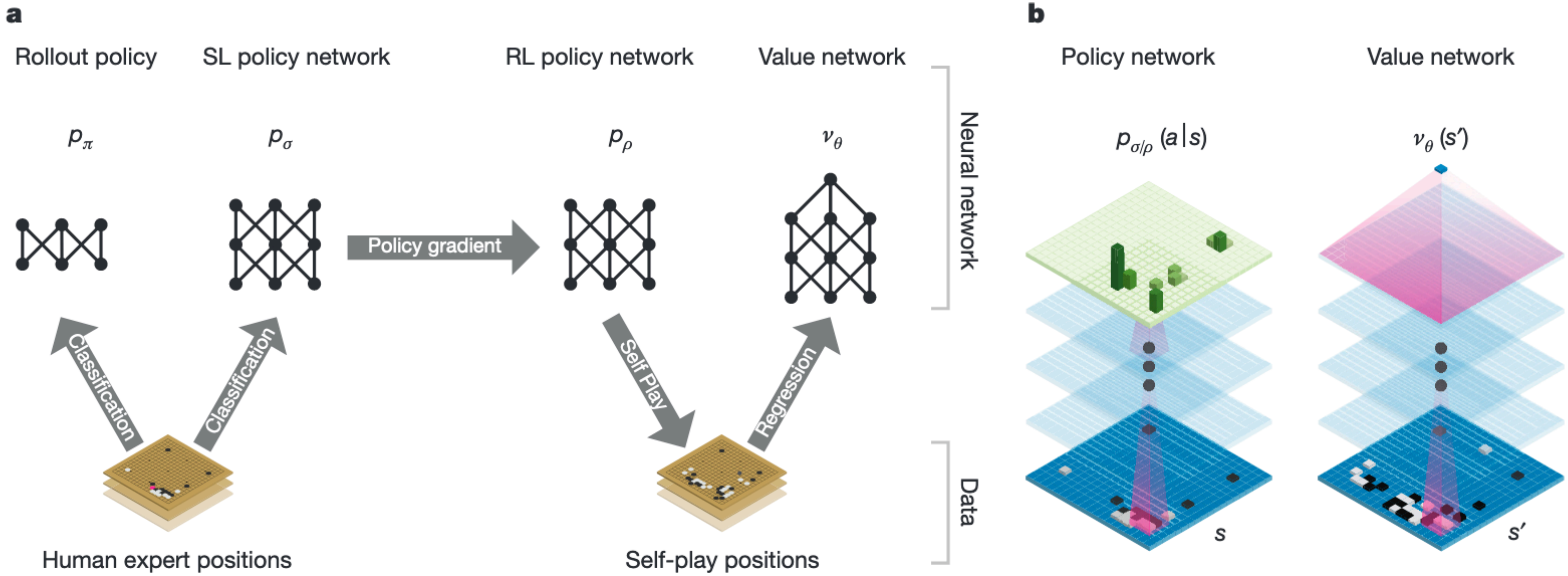
$$\Delta\rho \propto \frac{\partial \log p_\rho(a_t \mid s_t)}{\partial \rho} z_t$$

Value Network

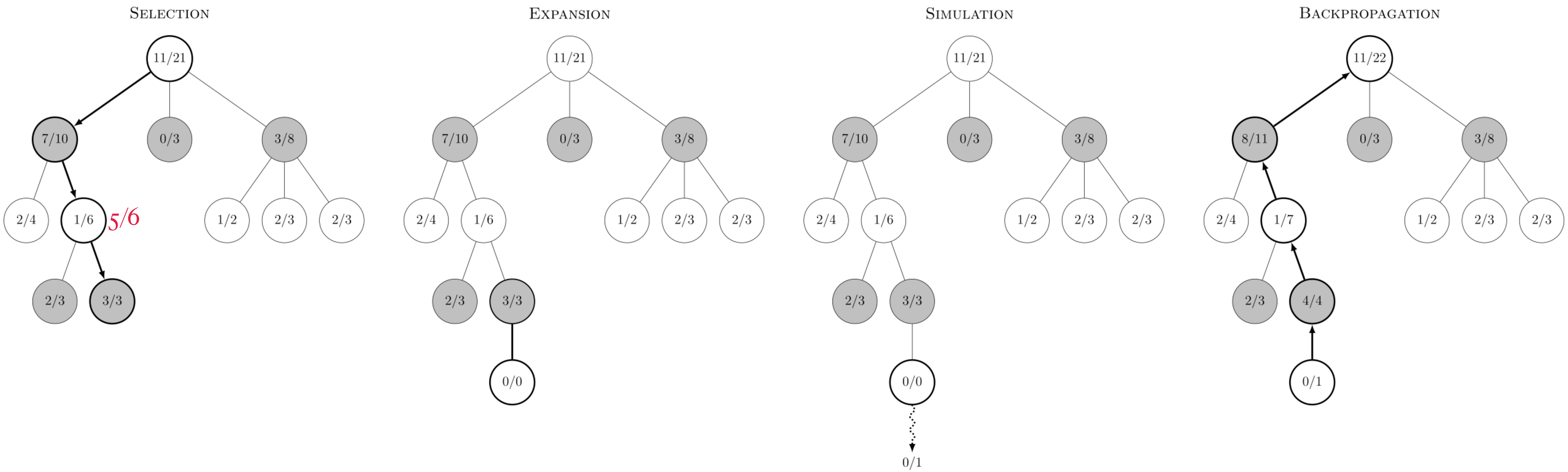
- Is identical in structure to the SL policy network
- Outputs a single prediction
- Trained by regression on state-outcome pairs using stochastic gradient descent to minimize the MSE
- At first, trained on expert games -> overfitting $\Rightarrow$ trained on self-play games.
- Updating weights ($z = \pm 1$):

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

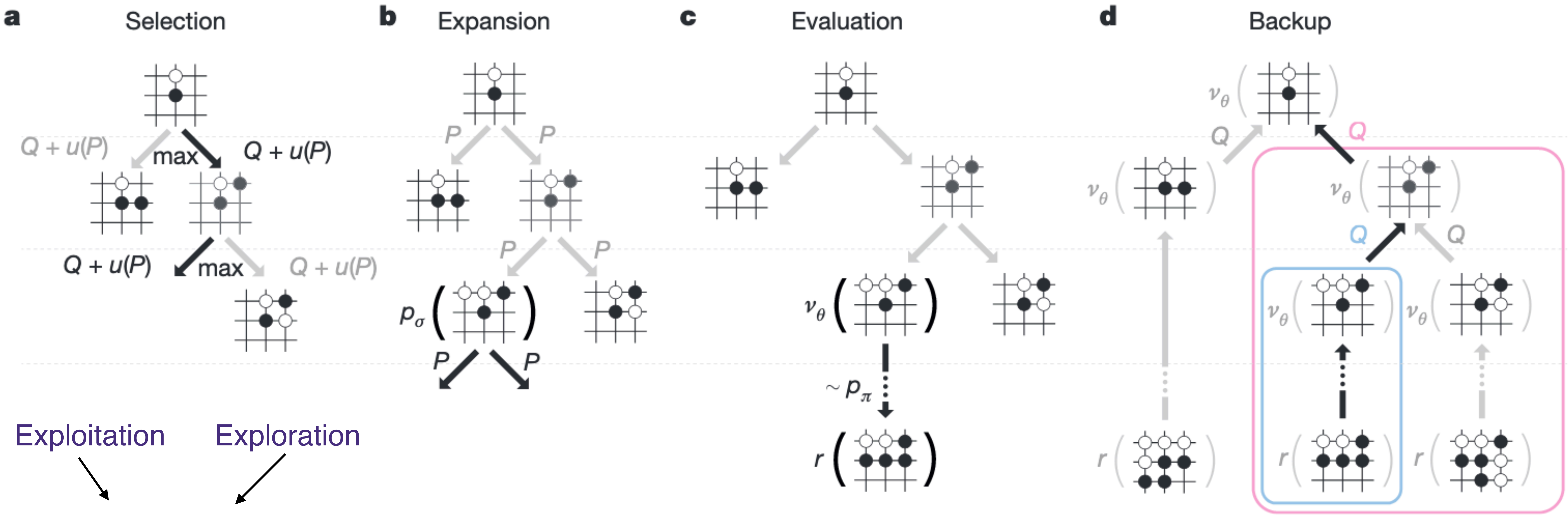
Training pipeline



MCTS


Selection
11/21
7/10
0/3
3/8
2/4
1/6
5/6
1/2
2/3
2/3
2/3
3/3
Expansion
11/21
7/10
0/3
3/8
2/4
1/6
1/2
2/3
2/3
2/3
3/3
0/0
Simulation
11/21
7/10
0/3
3/8
2/4
1/6
1/2
2/3
2/3
2/3
3/3
0/0
0/1
Backpropagation
11/22
8/11
0/3
3/8
2/4
1/7
1/2
2/3
2/3
2/3
4/4
0/1

MCTS in AlphaGo



Exploitation

Exploration

$$a_t = \underset{a}{\operatorname{argmax}}(Q(s_t, a) + u(s_t, a))$$

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

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

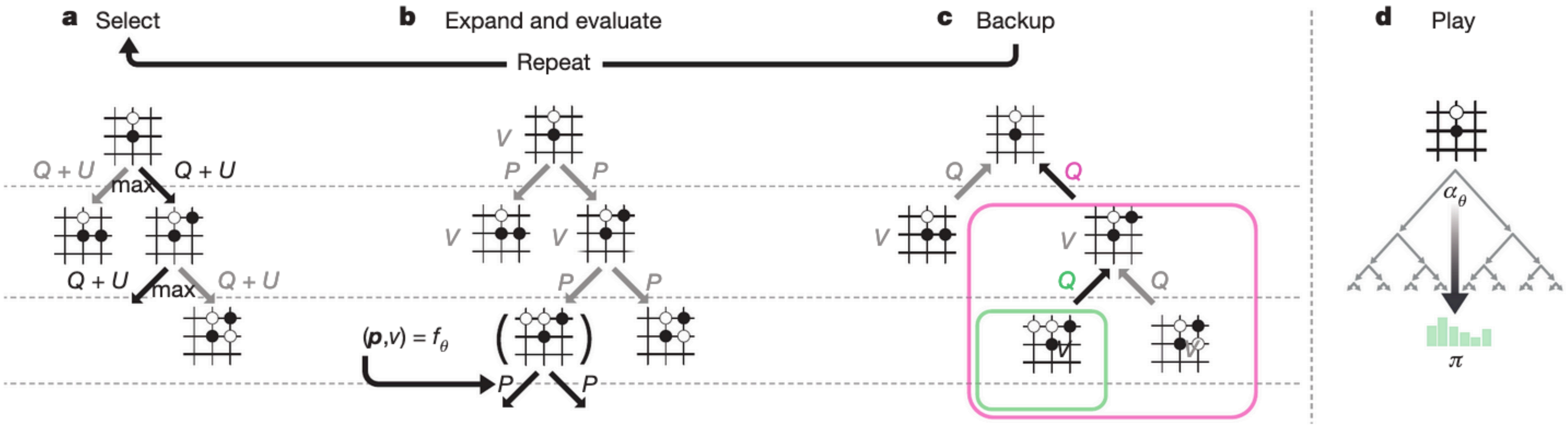
$$N(s,a) = \sum_{i=1}^{n} 1(s,a,i)$$

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

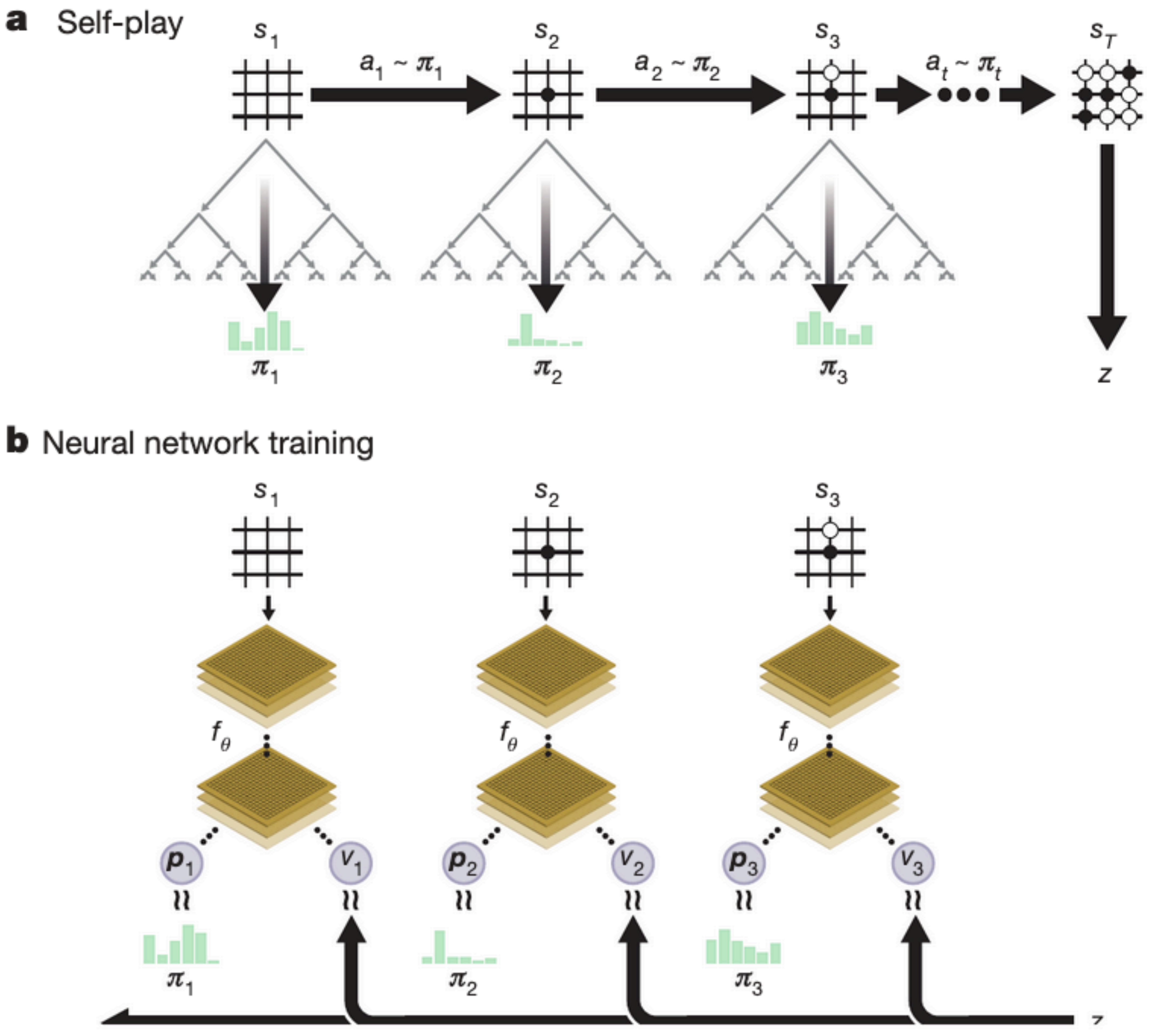
AlphaGo Zero

- No expert games
- Network f_θ (Conv, Res, BatchNorm) that takes as an input the raw board representation s and outputs both move probabilities and a value: $(\mathbf{p}, v) = f_\theta(s)$
- We use the search tree to create a policy π to pick our next move for the board
- π is derived from the visit count N: $\pi_a \propto N(s,a)^{1/\tau}$

AlphaGo Zero MCTS



AlphaGo Zero Self-play



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

References

- https://storage.googleapis.com/deepmind-media/alphago/AlphaGoNaturePaper.pdf
- https://deepmind.com/blog/article/alphago-zero-starting-scratch
- https://jonathan-hui.medium.com/alphago-zero-a-game-changer-14ef6e45eba5