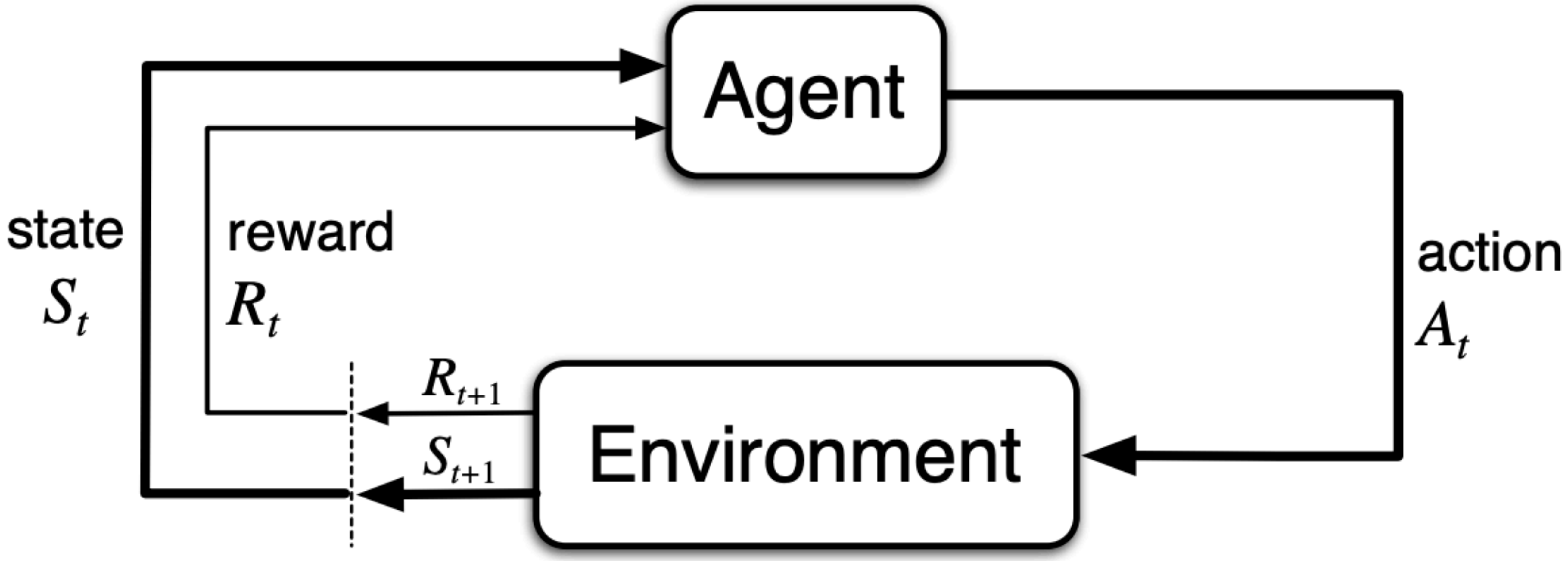
Reinforcement Learning and AlphaGo

COMP3314 — Lecture 10

Lingpeng Kong
Department of Computer Science, The University of Hong Kong

Based on: Probabilistic Machine Learning by Kevin Murphy
Slides from: Saw Shier Nee with special thanks!

Reinforcement Learning

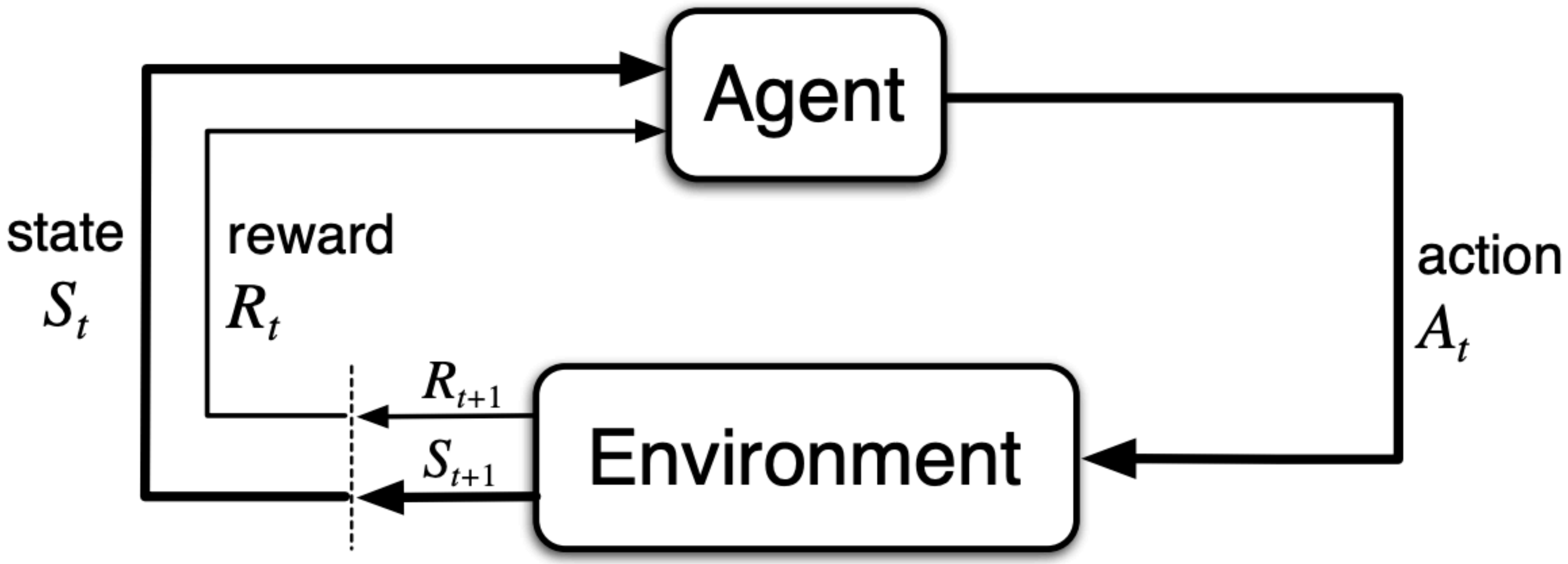


state space

At each time step t, an agent experiences a state s(t) ∈ S.

e.g.
A snapshot of the current game board.
Number of passengers and taxis at different locations in a city.

Reinforcement Learning

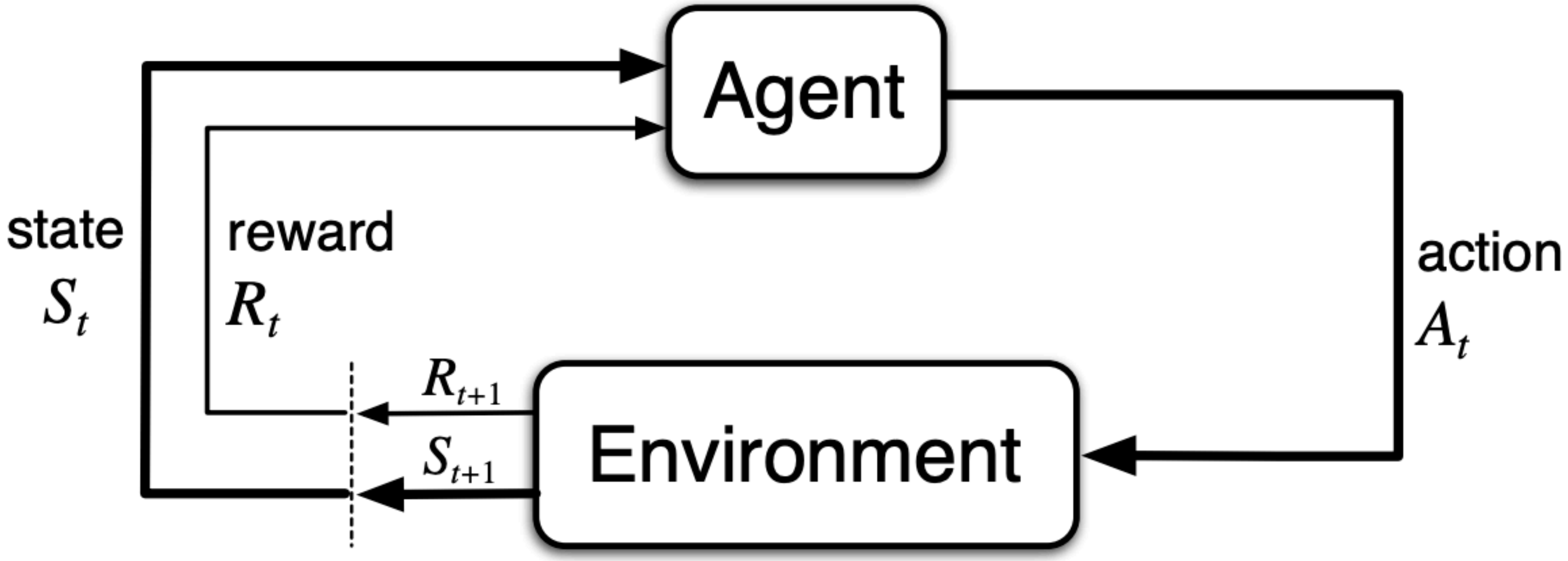


action space

At each time step t, an agent takes an action a(t), chosen from some feasible set A(t).

e.g.
possible moves in a board game.

Reinforcement Learning

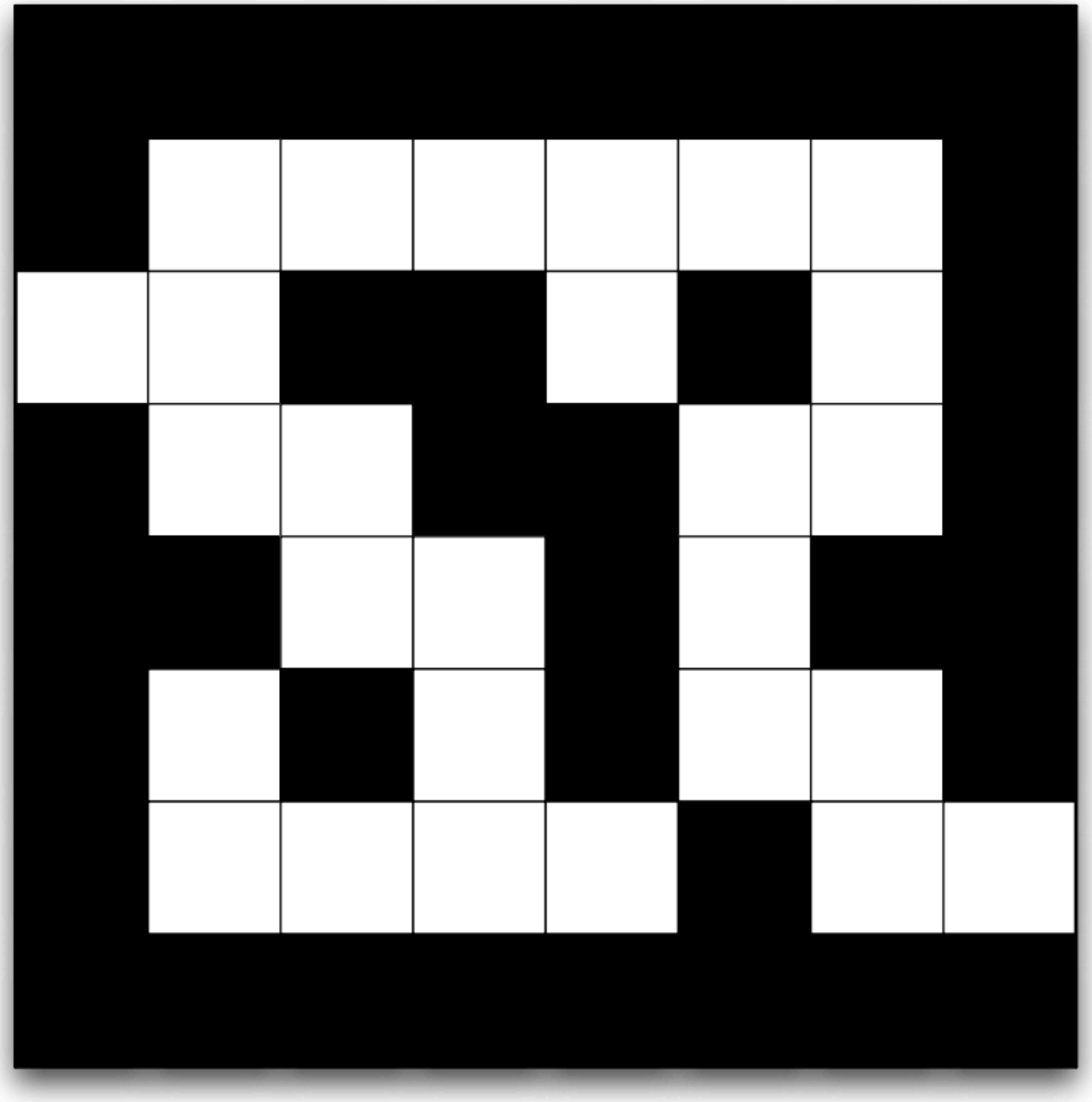


reward

One time step later, in part as a consequence of its action, the agent receives a reward R_{t+1} and find it self in a new state.

The <u>next state</u> (at time t+1) is a (probabilistic) function of the current state and action taken: s(t+1) ~ σ(a(t), s(t)).

Reinforcement Learning



Rewards: -1 per time-step

Actions: N, E, S, W

States: Agent's location

https://www.deepmind.com/learning-resources/introduction-to-reinforcement-learning-with-david-silver

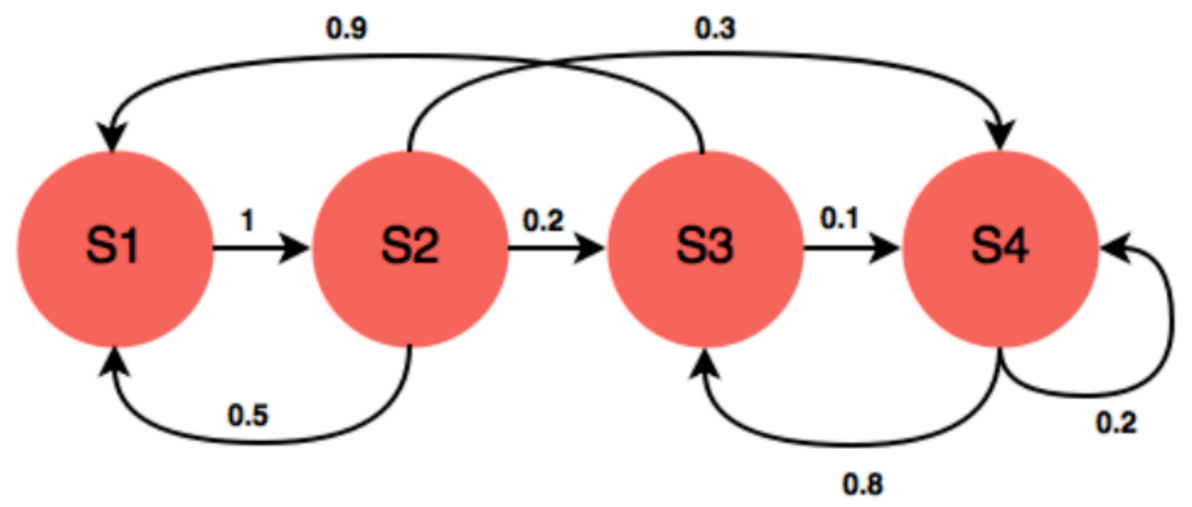
Markov Reward Processes

A Markov chain can be represented with a transition matrix. For a Markov state s and successor state s', the state transition probability is defined by

$$\mathcal{P}_{ss'} = \mathbb{P}\left[S_{t+1} = s' \mid S_t = s\right]$$

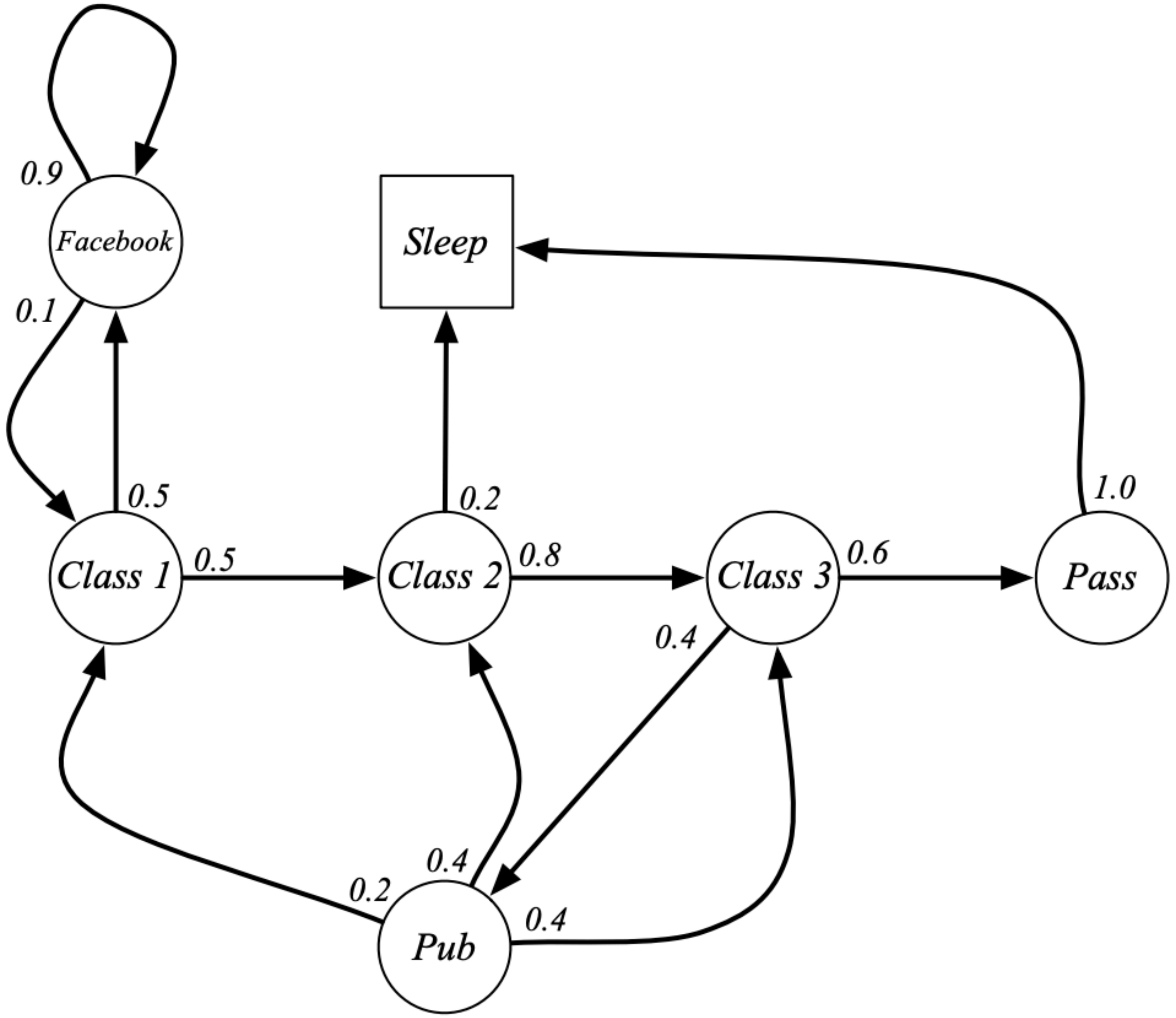
State transition matrix defines transition probabilities from all states s to all successor states s'

$$\mathcal{P} = \text{from} \overset{\text{to}}{\begin{bmatrix} \mathcal{P}_{11} & \ldots & \mathcal{P}_{1n} \\ \vdots & & \\ \mathcal{P}_{n1} & \ldots & \mathcal{P}_{nn} \end{bmatrix}}$$



$$P = \begin{bmatrix} 0 & 1 & 0 & 0 \\ 0.5 & 0 & 0.2 & 0.3 \\ 0.9 & 0 & 0 & 0.1 \\ 0 & 0 & 0.8 & 0.2 \end{bmatrix}$$

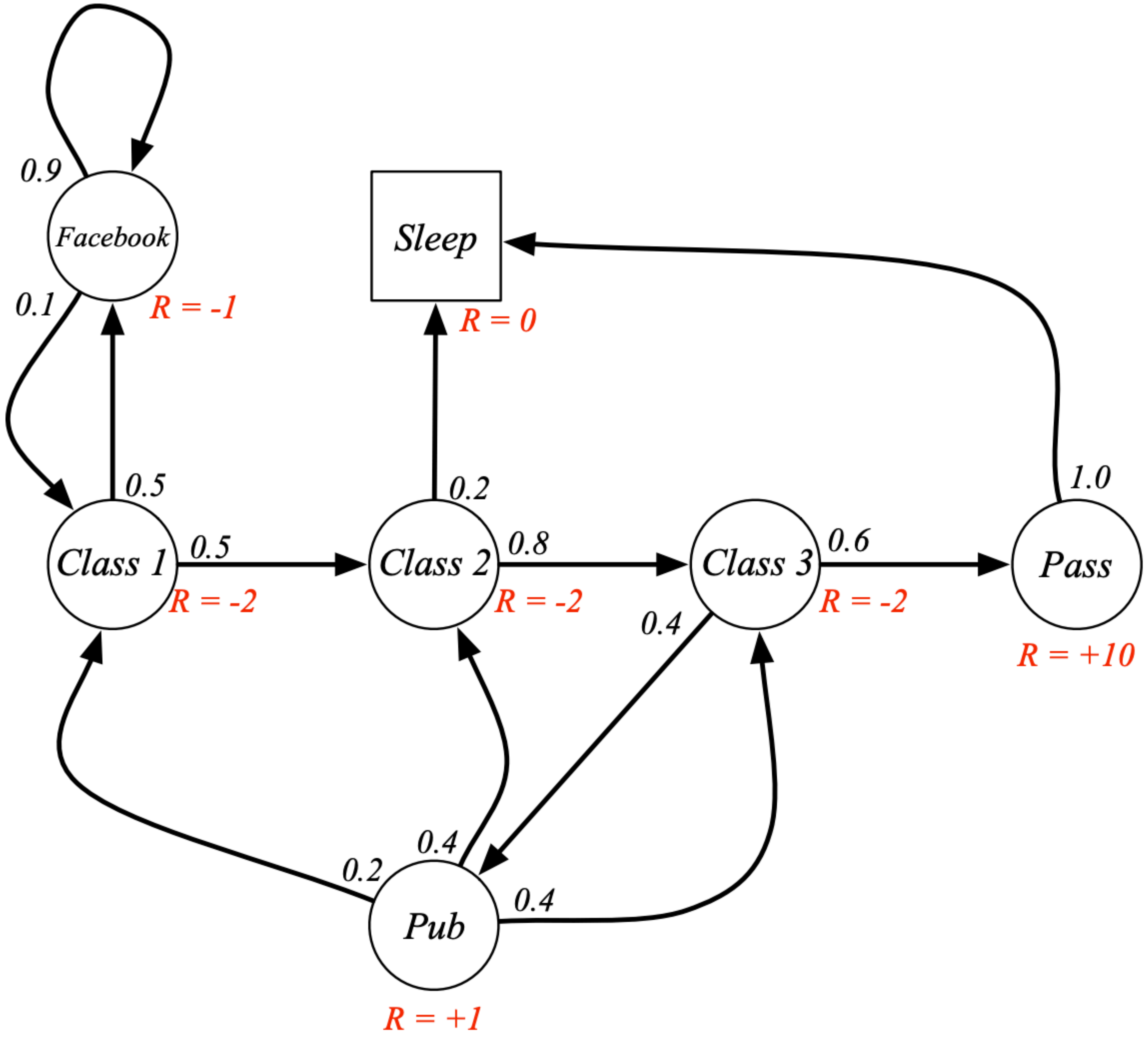
Student Markov Chain Transition Matrix



$$
\mathcal{P} = \begin{array}{c} \\ C1 \\ C2 \\ C3 \\ Pass \\ Pub \\ FB \\ Sleep \end{array}
\begin{array}{c}
\begin{array}{ccccccc} C1 & C2 & C3 & Pass & Pub & FB & Sleep \end{array} \\
\left[\begin{array}{ccccccc}
 & 0.5 & & & & 0.5 & \\
 & & 0.8 & & & & 0.2 \\
 & & & 0.6 & 0.4 & & \\
 & & & & & & 1.0 \\
0.2 & 0.4 & 0.4 & & & & \\
0.1 & & & & & 0.9 & \\
 & & & & & & 1
\end{array}\right]
\end{array}
$$

https://www.deepmind.com/learning-resources/introduction-to-reinforcement-learning-with-david-silver

Markov Reward Processes



$\mathcal{R}$ is a reward function

$$\mathcal{R}_s = \mathbb{E}[R_{t+1} \mid S_t = s]$$

γ is a discount factor $\gamma \in [0, 1]$

https://www.deepmind.com/learning-resources/introduction-to-reinforcement-learning-with-david-silver

Markov Reward Processes

Return: total Discounted reward from time-step t

$$G_t = R_{t+1} + \gamma R_{t+2} + \ldots = \sum_{k=0}^{\infty} \gamma^k R_{t+k+1}$$

γ close to 0 leads to “myopic” evaluation

γ close to 1 leads to “far-sighted” evaluation

The state value function v(s) of an MRP is the expected return starting from state s

$$v(s) = \mathbb{E}[G_t \mid S_t = s]$$

Markov Reward Processes

Sample returns for Student MRP

Starting from S_1 = C1 with γ = 0.5

C1 C2 C3 Pass Sleep	$v_1 = -2 - 2 * \frac{1}{2} - 2 * \frac{1}{4} + 10 * \frac{1}{8}$	$=$	-2.25
C1 FB FB C1 C2 Sleep	$v_1 = -2 - 1 * \frac{1}{2} - 1 * \frac{1}{4} - 2 * \frac{1}{8} - 2 * \frac{1}{16}$	$=$	-3.125
C1 C2 C3 Pub C2 C3 Pass Sleep	$v_1 = -2 - 2 * \frac{1}{2} - 2 * \frac{1}{4} + 1 * \frac{1}{8} - 2 * \frac{1}{16} \dots$	$=$	-3.41
C1 FB FB C1 C2 C3 Pub C1 ...	$v_1 = -2 - 1 * \frac{1}{2} - 1 * \frac{1}{4} - 2 * \frac{1}{8} - 2 * \frac{1}{16} \dots$	$=$	-3.20
FB FB FB C1 C2 C3 Pub C2 Sleep			

Markov Reward Processes

Markov Decision Processes

Markov reward process with decisions.

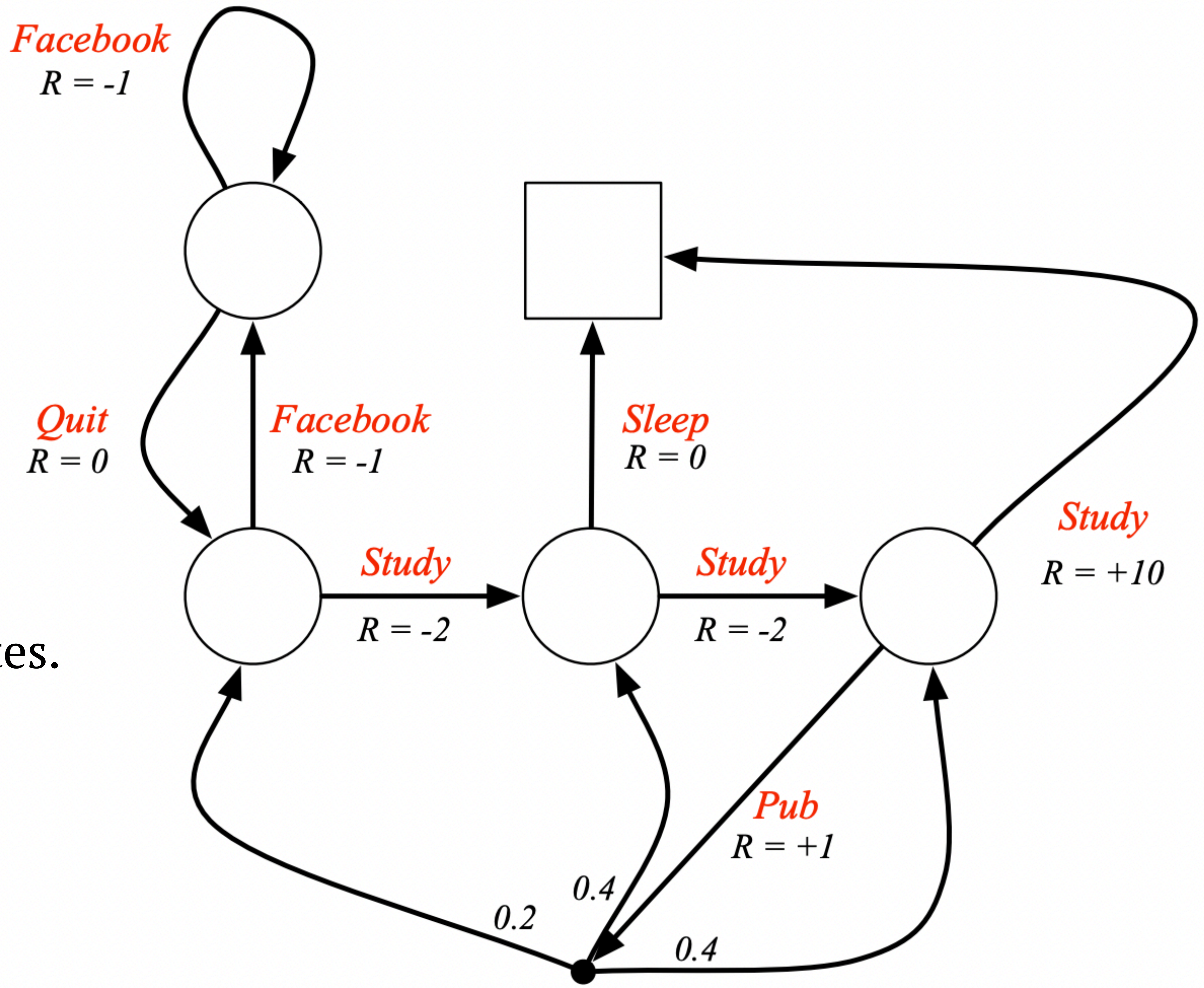
$\mathcal{A}$ is a finite set of actions

$$\mathcal{P}_{ss'}^{a} = \mathbb{P}[S_{t+1} = s' \mid S_t = s, A_t = a]$$

$$\mathcal{R}_{s}^{a} = \mathbb{E}[R_{t+1} \mid S_t = s, A_t = a]$$

A policy π is a distribution over actions given states.

$$\pi(a \mid s) = \mathbb{P}[A_t = a \mid S_t = s]$$



Markov Decision Processes

The state-value function $v_\pi(s)$ of an MDP is the expected return starting from state s, and then following policy π.

$$v_\pi(s) = \mathbb{E}_\pi[G_t \mid S_t = s]$$

The action-value function $q_\pi(s, a)$ is the expected return starting from state s, taking action a, and then following policy π.

$$q_\pi(s, a) = \mathbb{E}_\pi[G_t \mid S_t = s, A_t = a]$$

Bellman Optimality Equation

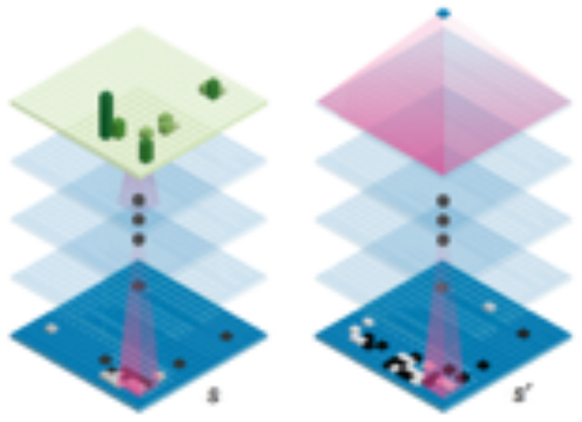
The optimal value functions are recursively related by the Bellman optimality equations:

$$v_*(s) = \max_a q_*(s, a)$$

$$q_*(s, a) = \mathcal{R}_s^a + \gamma \sum_{s' \in \mathcal{S}} \mathcal{P}_{ss'}^a v_*(s')$$

Outline

- Representation of Board (deep convolutional neural networks, CNN)



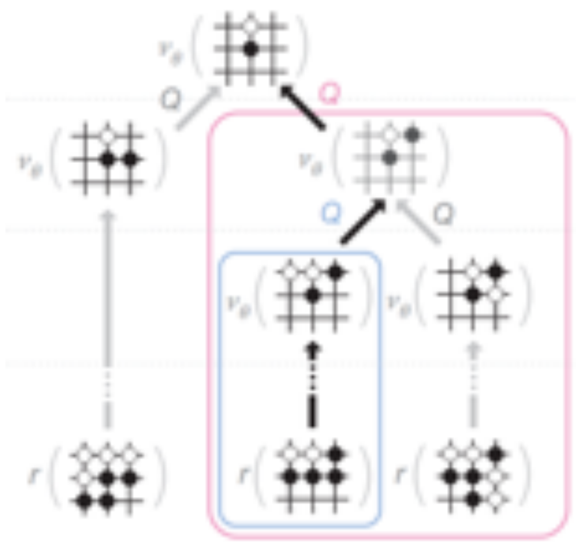
- Imitating expert moves (supervised learning)

Policy network (SL)

- Predicting the wining possibility given a board configuration (reinforcement learning)

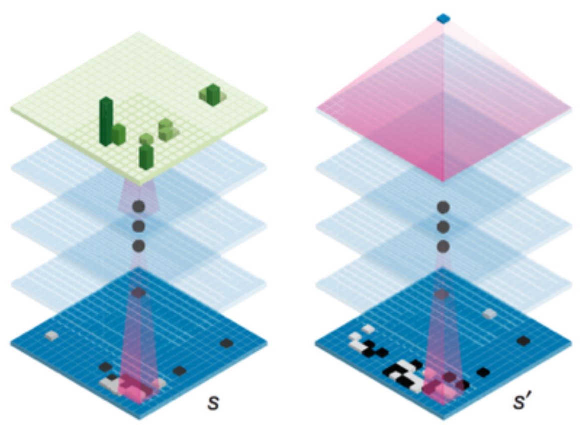
Value network

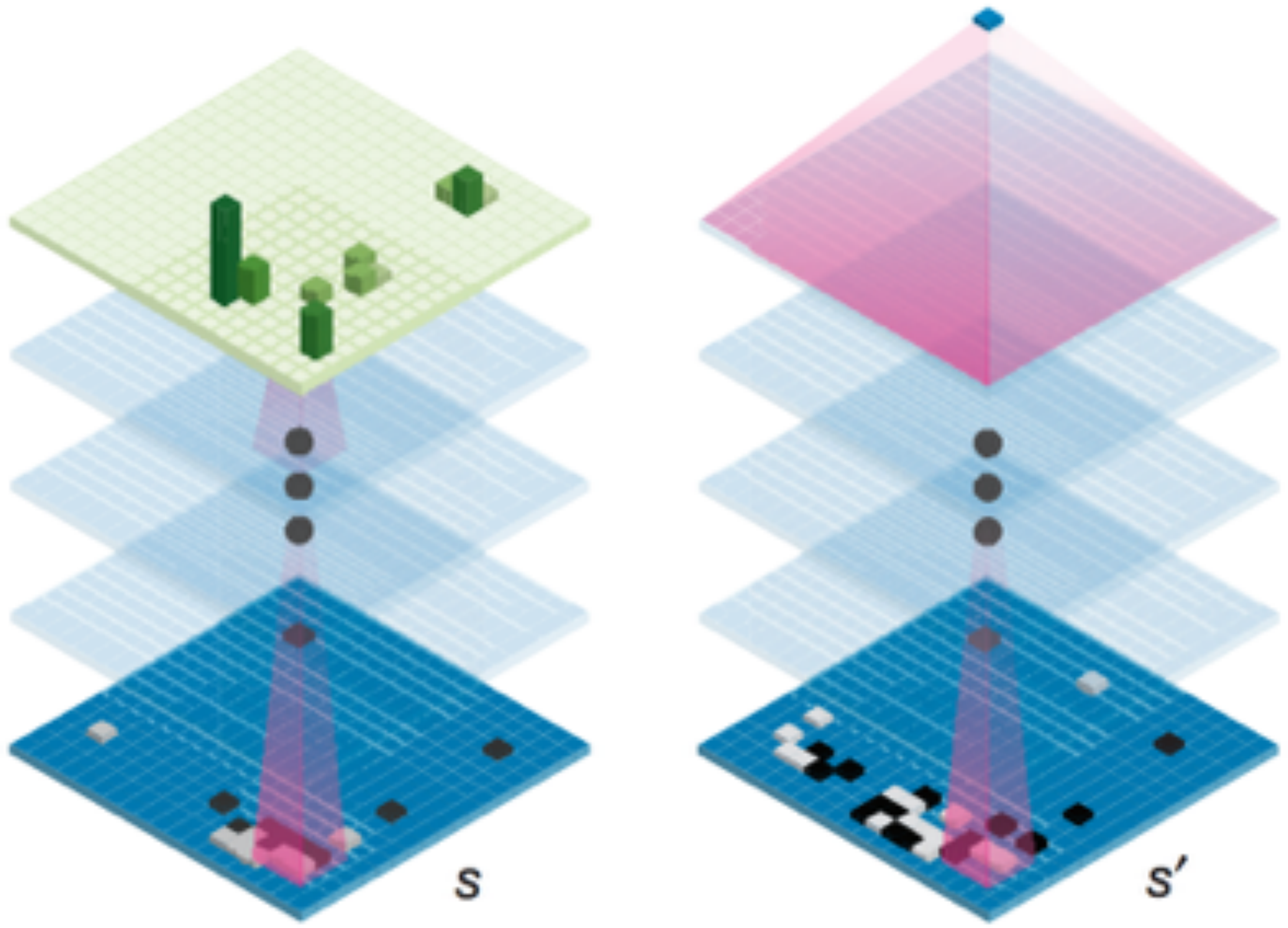
- Select a move, more wisely (Monte Carlo tree search)

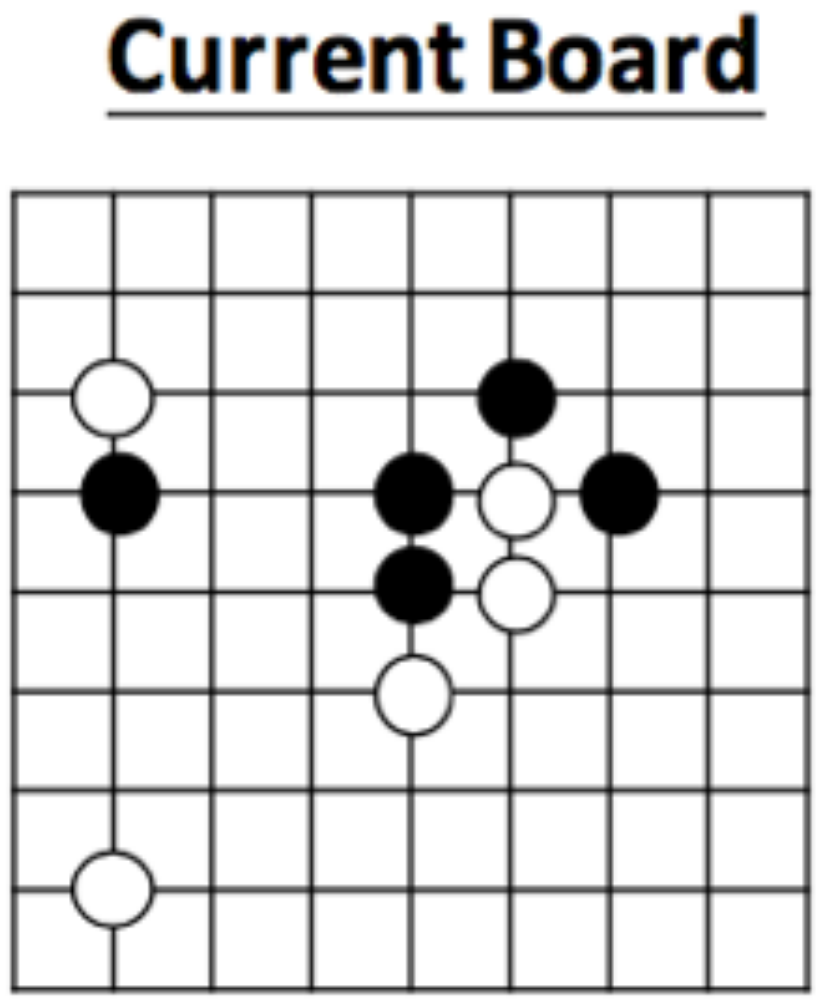


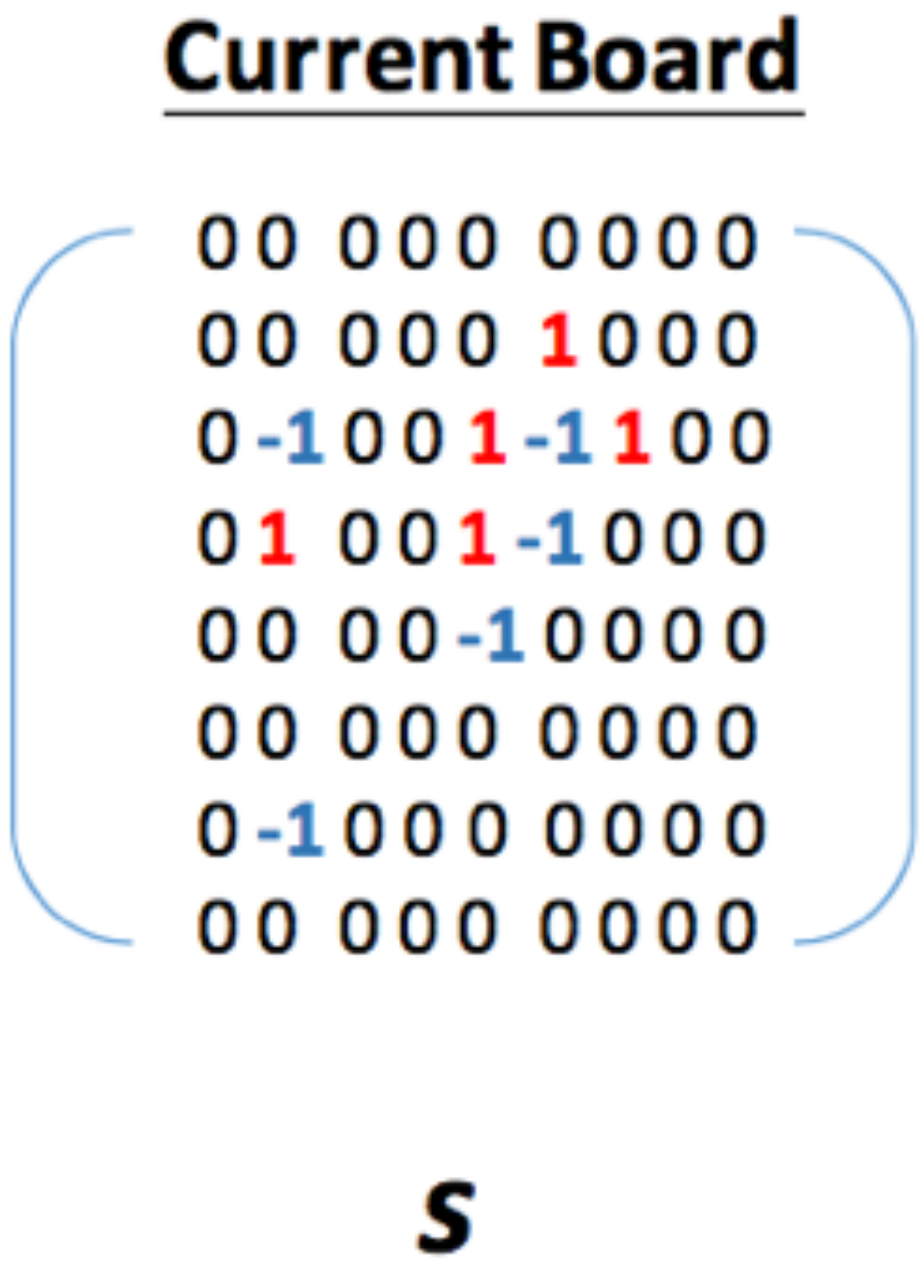
Deterministic MDP

- Representation of Board (deep convolutional neural networks, CNN)

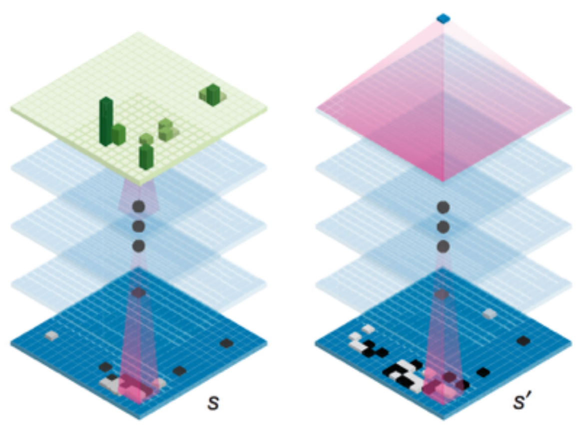




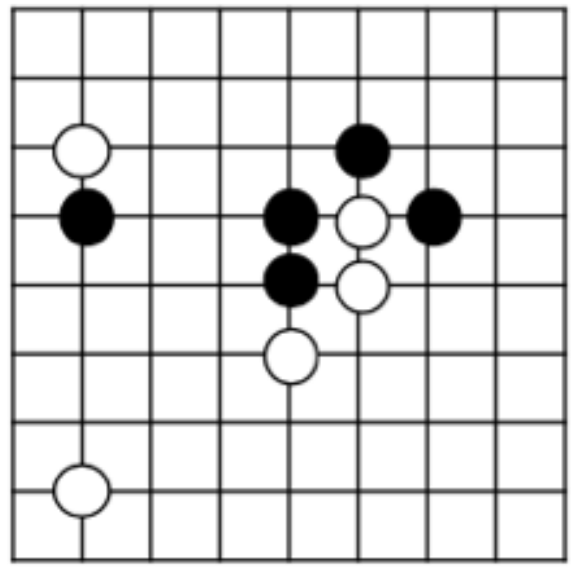




- Representation of Board (deep convolutional neural networks, CNN)



Current Board



Current Board

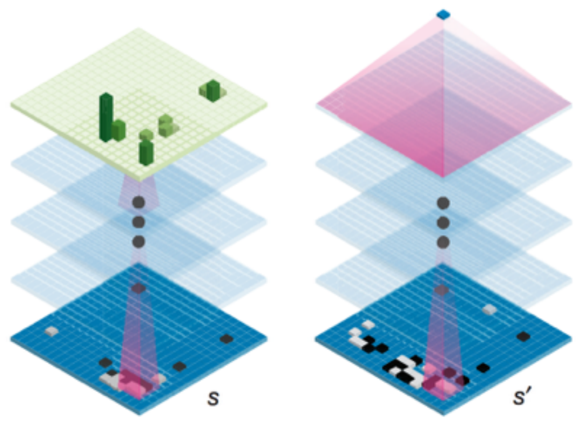
```
00 000 0000
00 000 1000
0-1001-1100
01 001-1000
00 00-10000
00 000 0000
0-1000 0000
00 000 0000
```

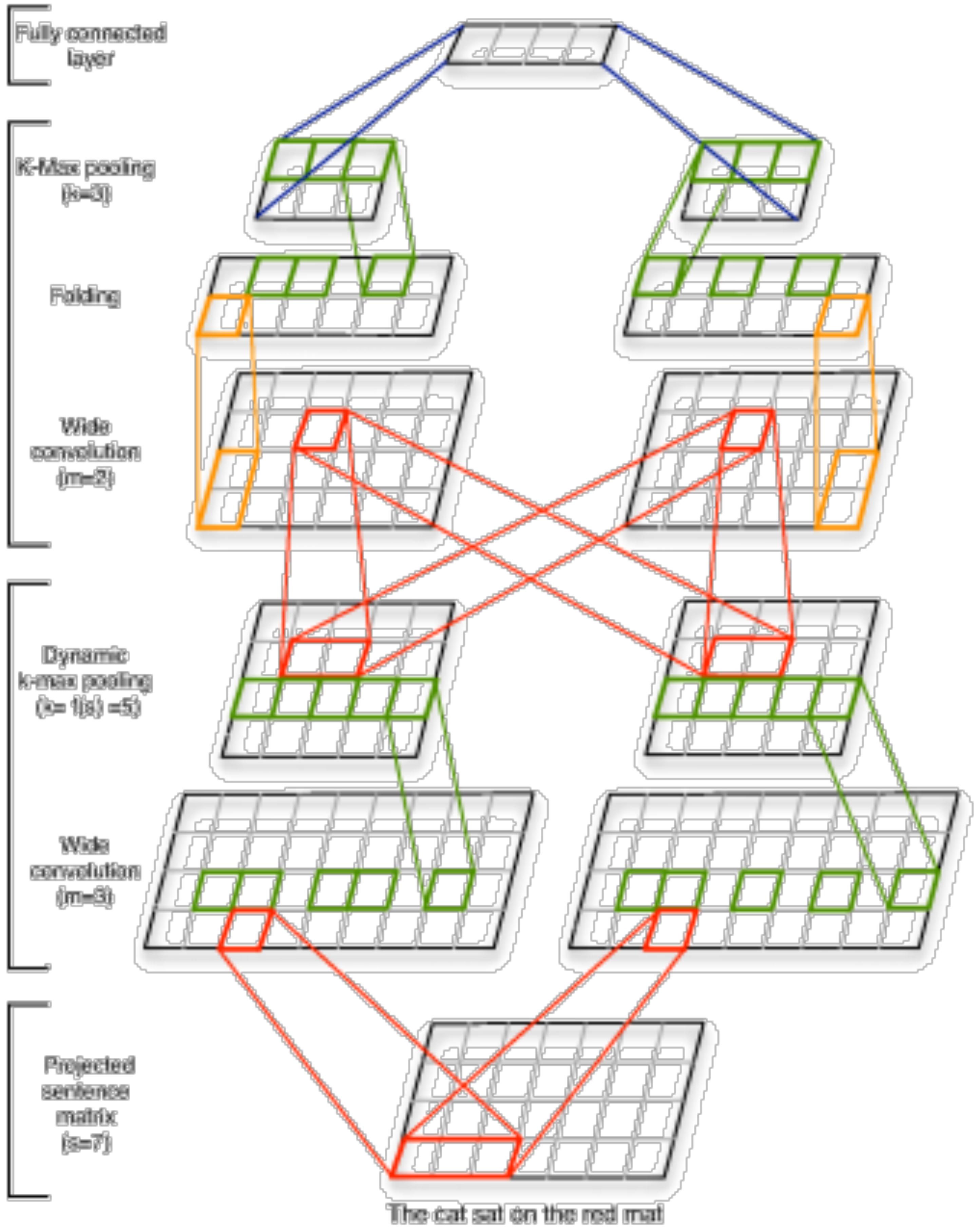
$$\theta^T \phi(s)$$

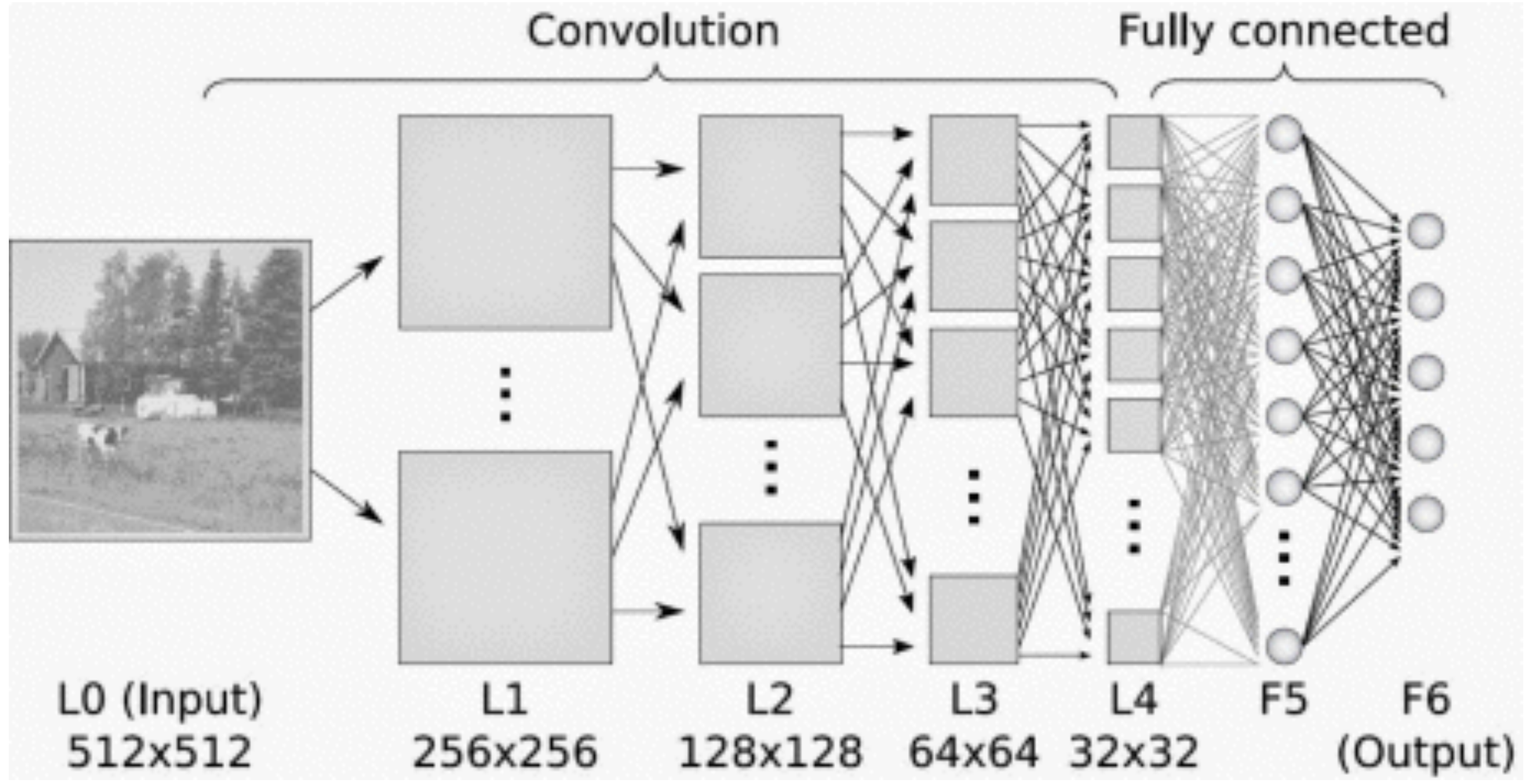
Feature	# of planes	Description
Stone colour	3	Player stone / opponent stone / empty
Ones	1	A constant plane filled with 1
Turns since	8	How many turns since a move was played
Liberties	8	Number of liberties (empty adjacent points)
Capture size	8	How many opponent stones would be captured
Self-atari 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

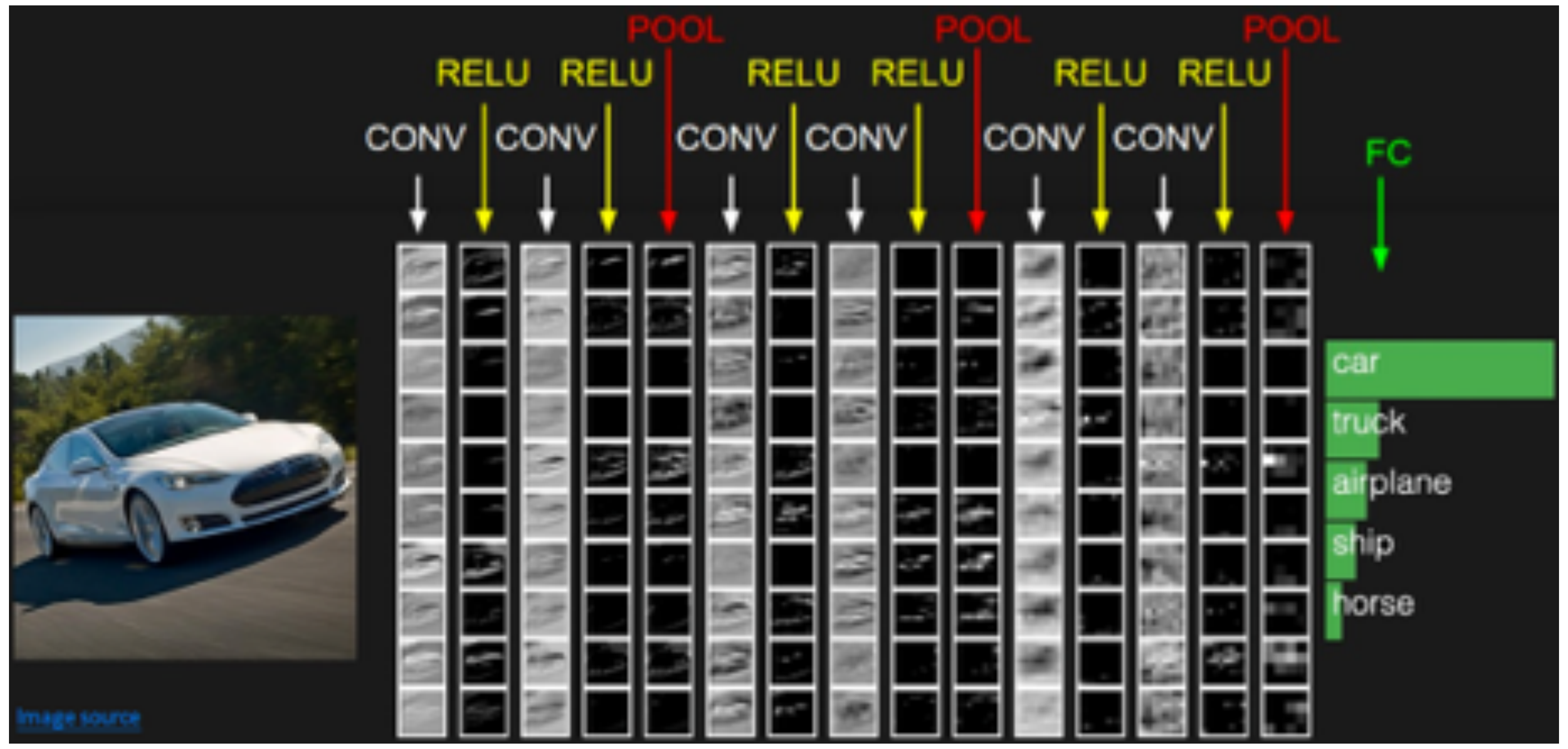
Feature planes used by the policy network (all but last feature) and value network (all features).

- Representation of Board (deep convolutional neural networks, CNN)

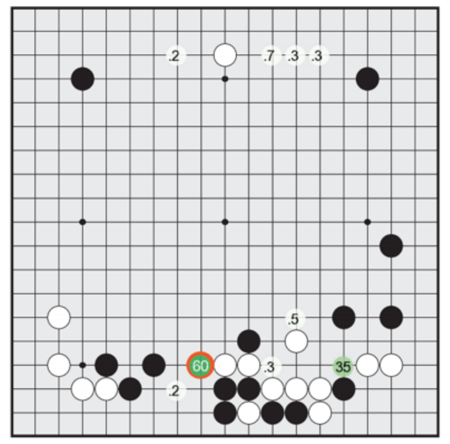


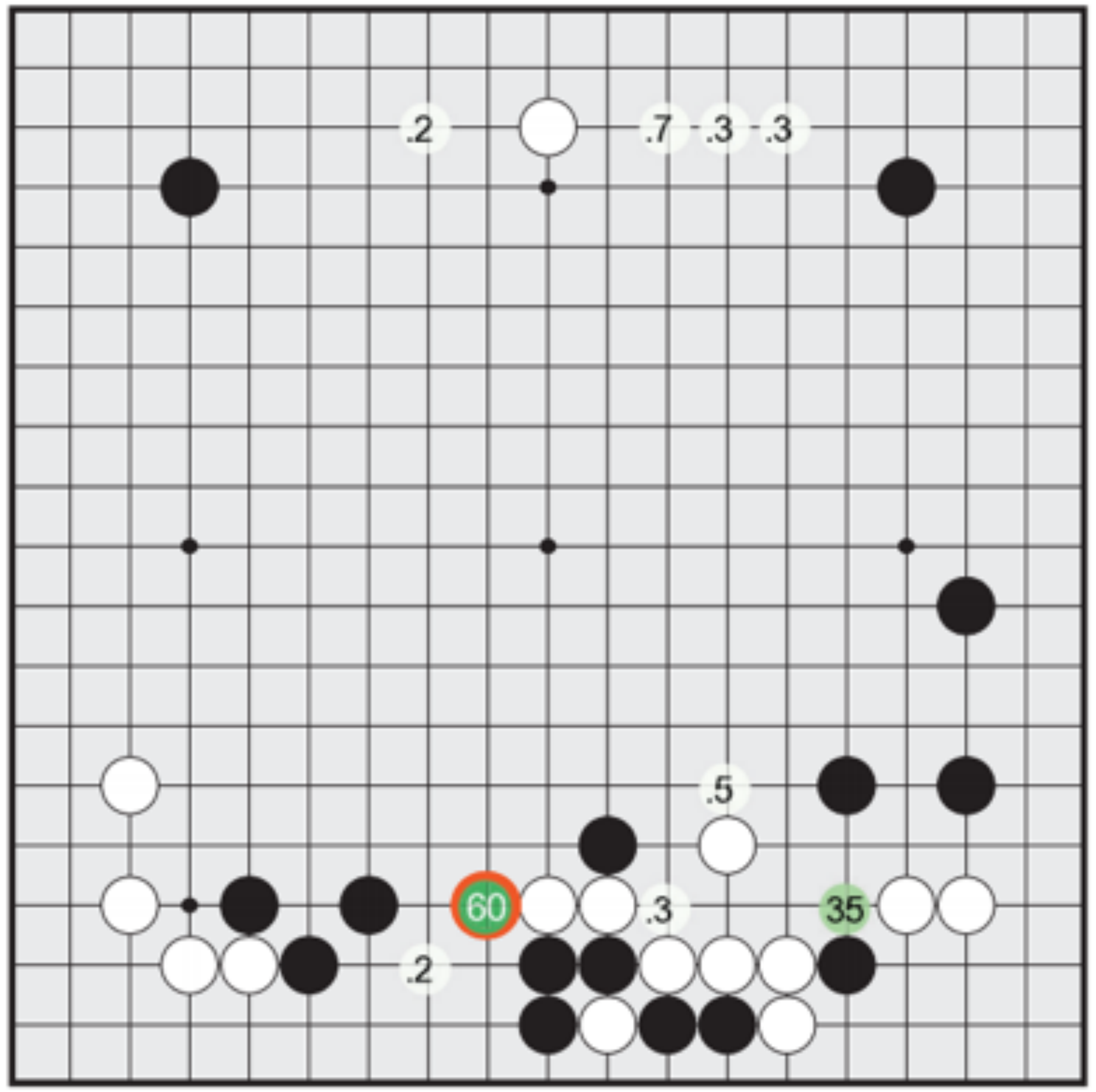


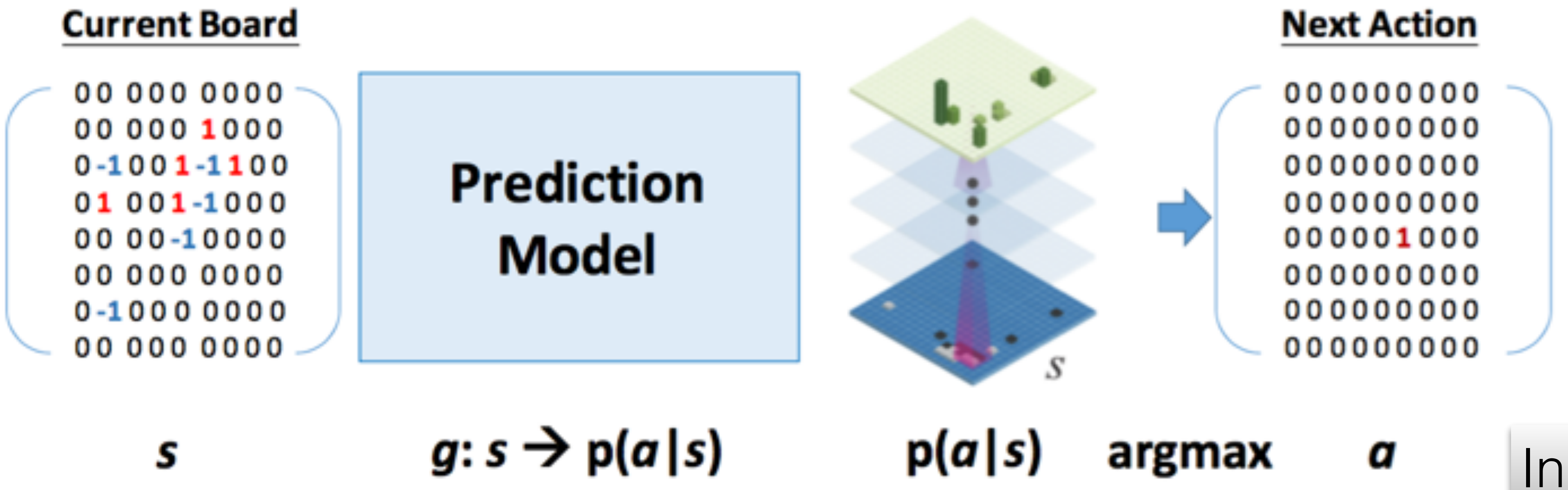




- Imitating expert moves (supervised learning)

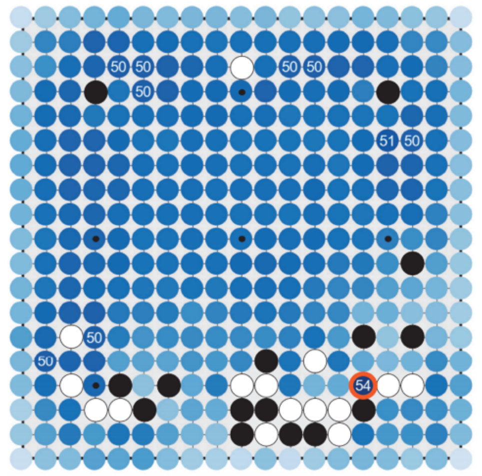


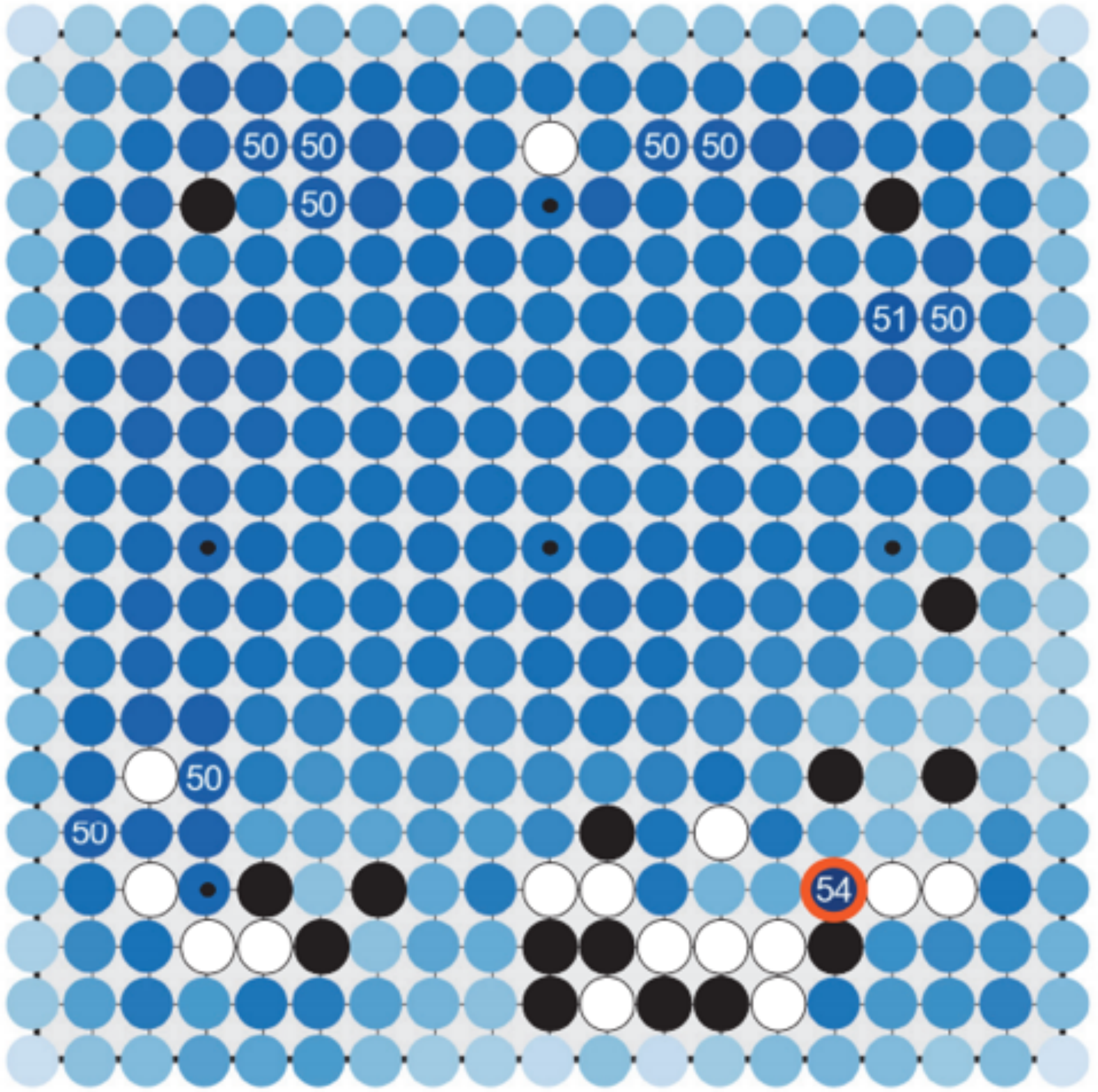


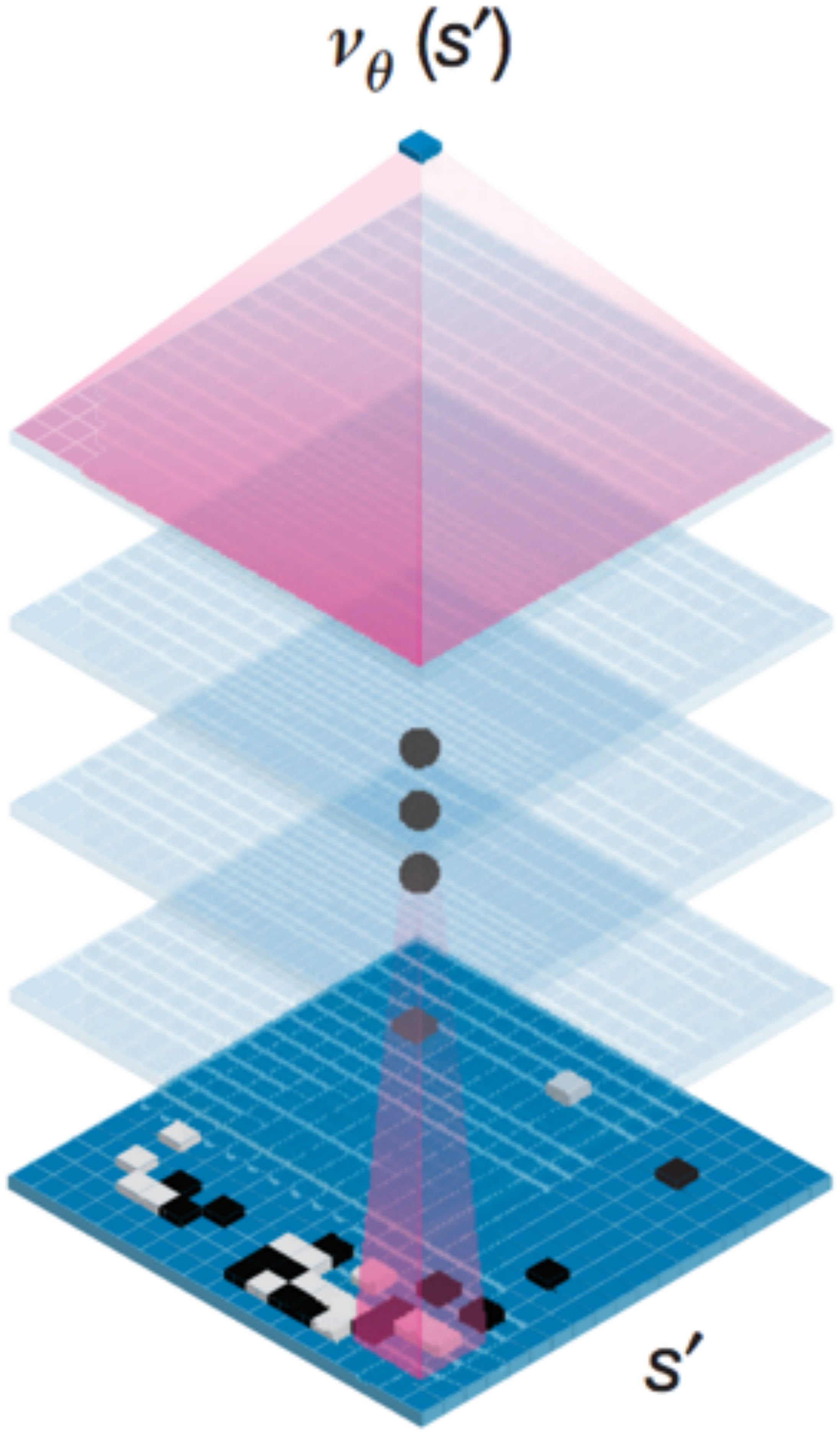


Initializing policy

- Predicting the wining possibility given a board configuration (reinforcement learning)

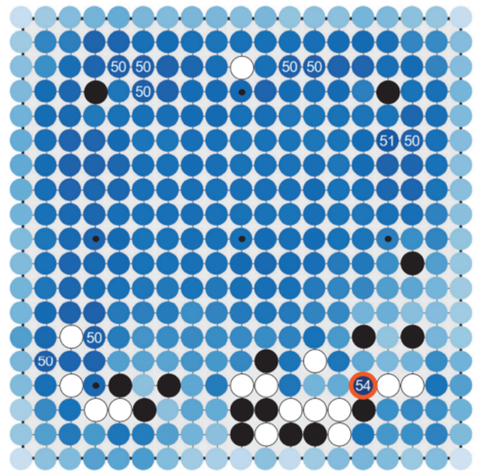


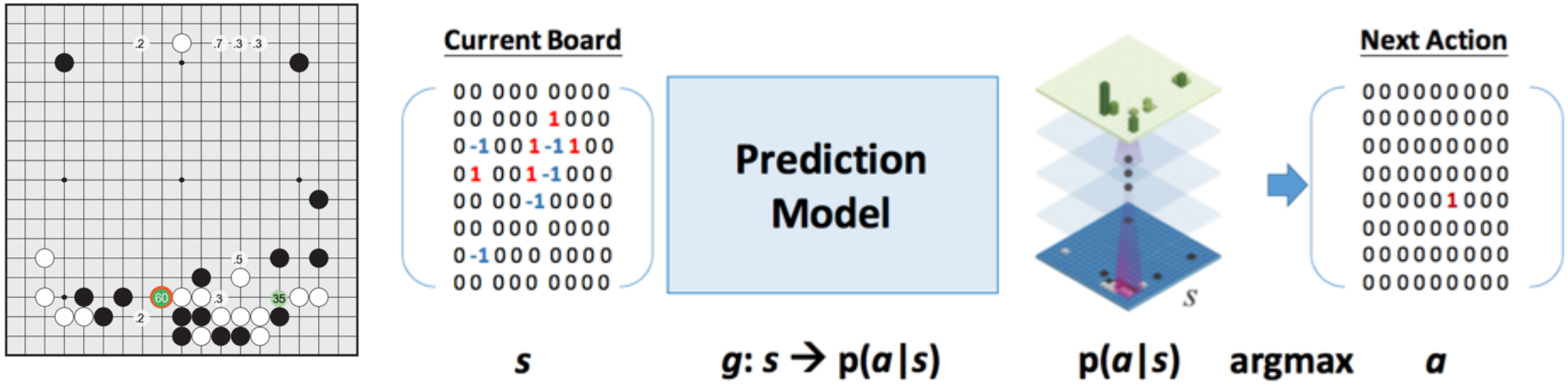


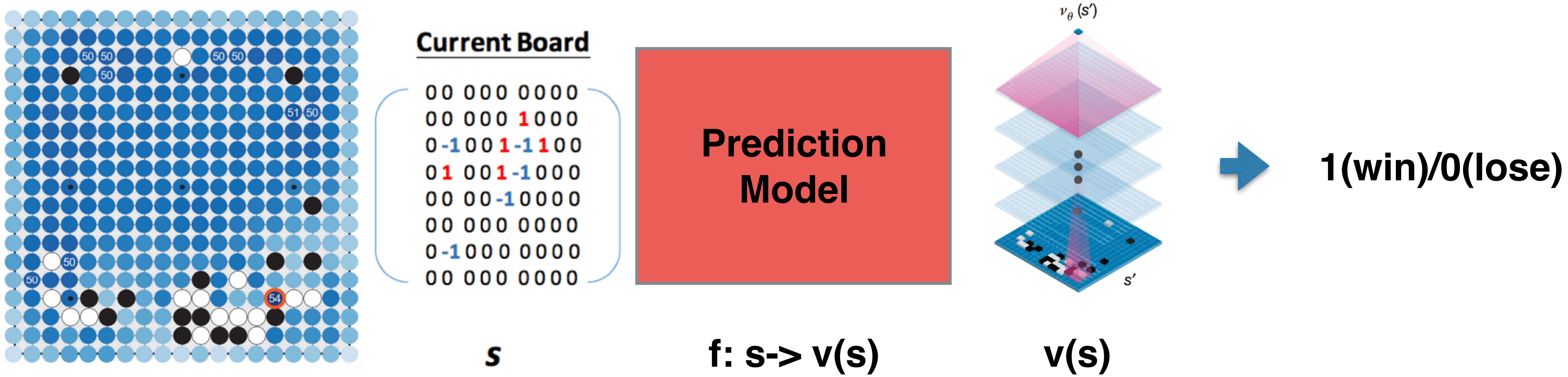


fitted value iteration algorithm

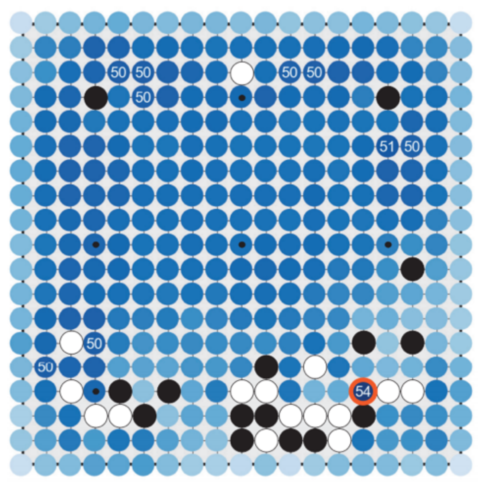
- Predicting the wining possibility given a board configuration (reinforcement learning)

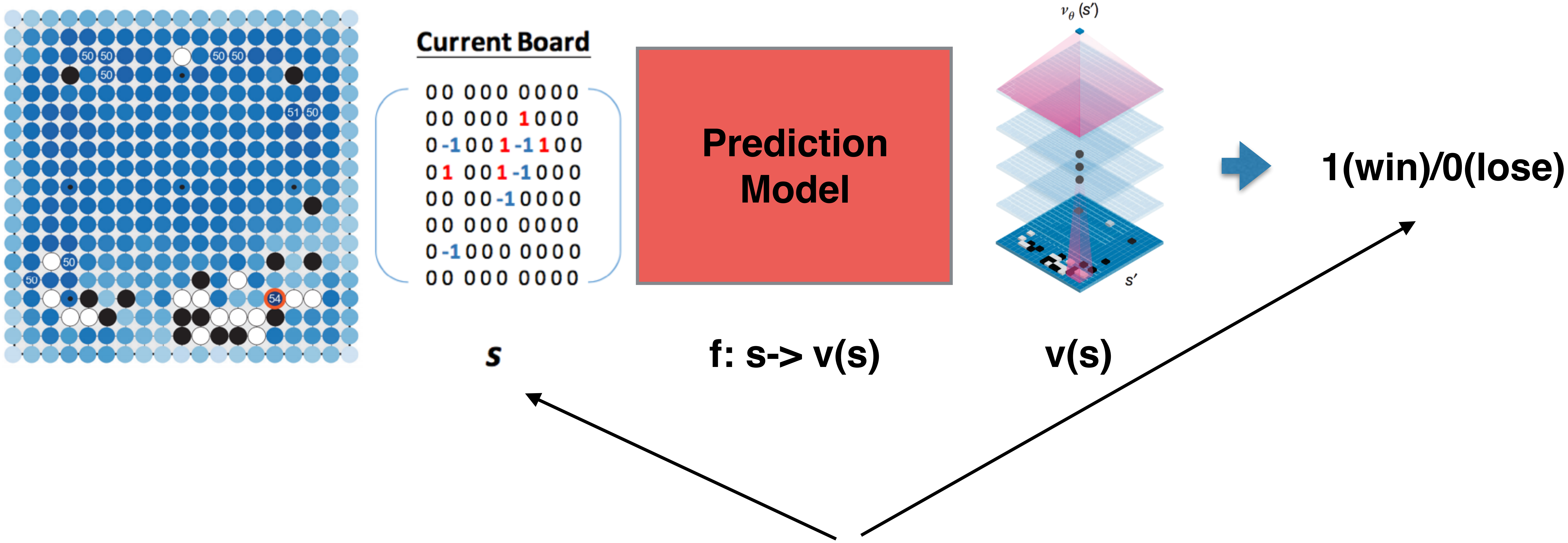






- Predicting the wining possibility given a board configuration (reinforcement learning)



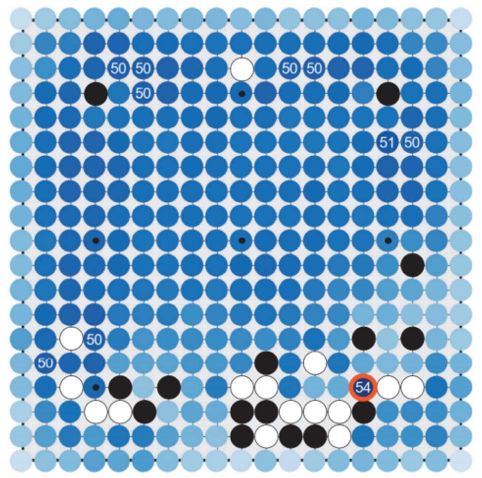


(S, Z) — training pair (x,y)

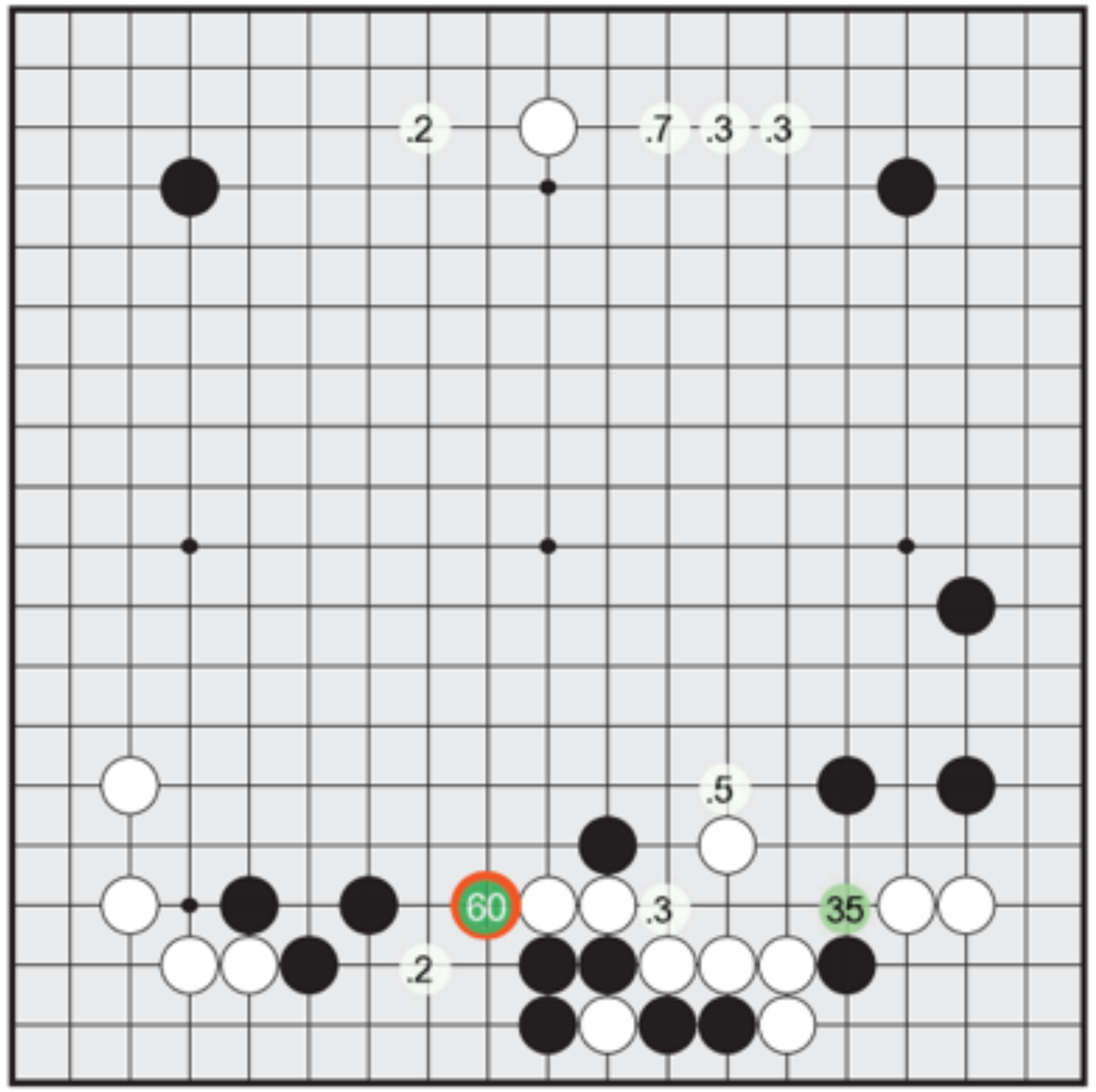
1 *"29,400,000 positions from 160,000 games played by KGS 6 to 9 dan human players"*

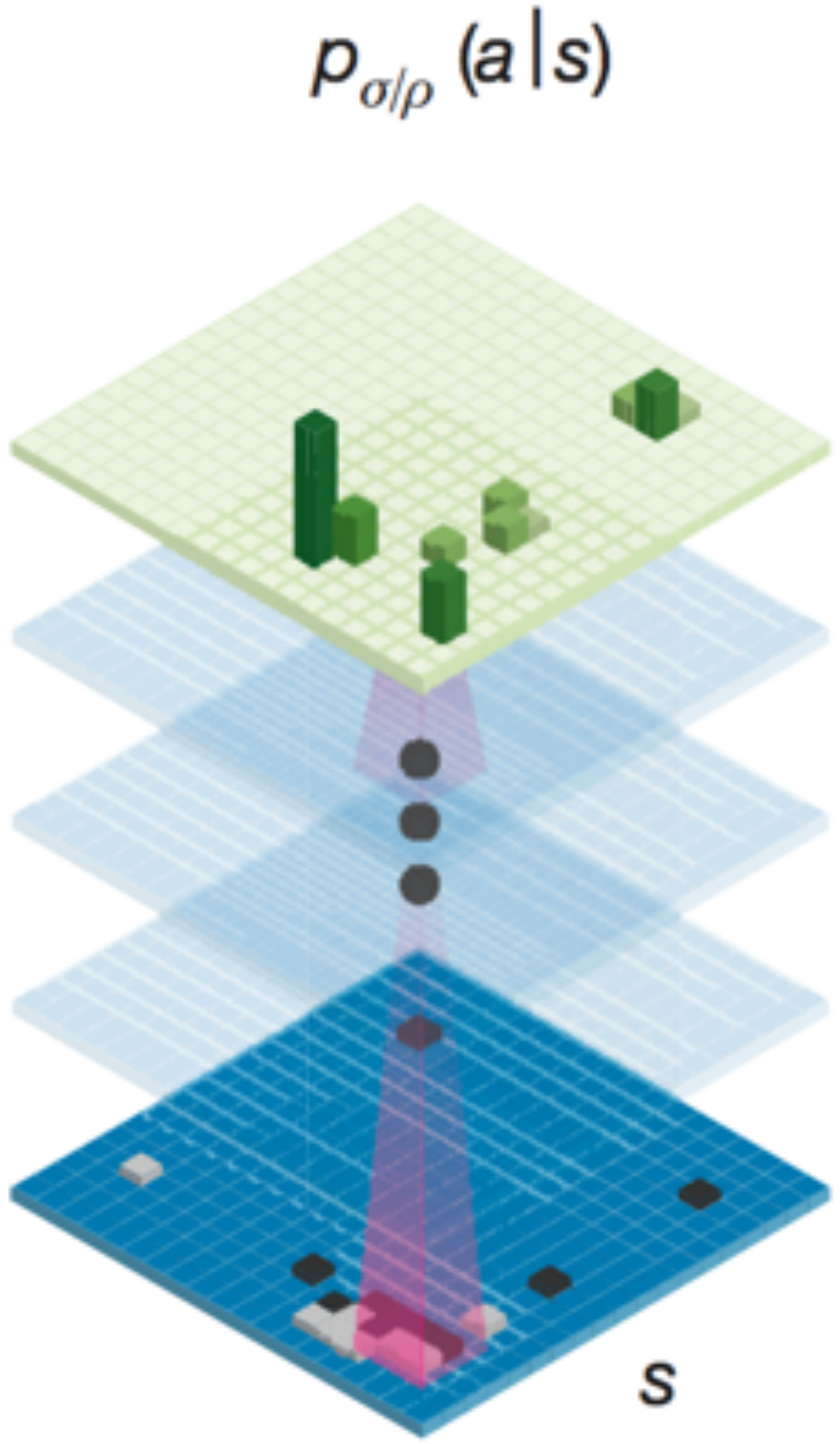
Overfitting, Training error rate 0.19, Test error rate 0.37

- Predicting the wining possibility given a board configuration (reinforcement learning)



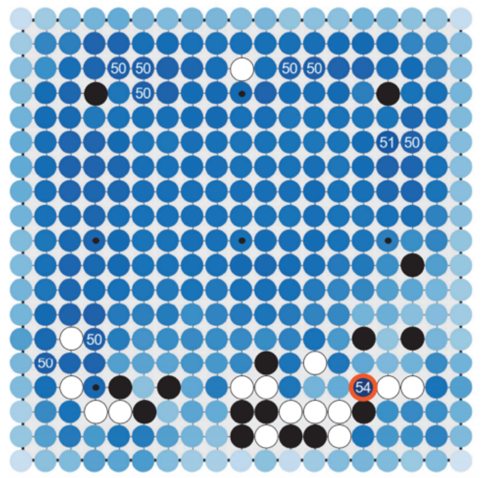
(S, Z) — training pair (x,y) — self play?



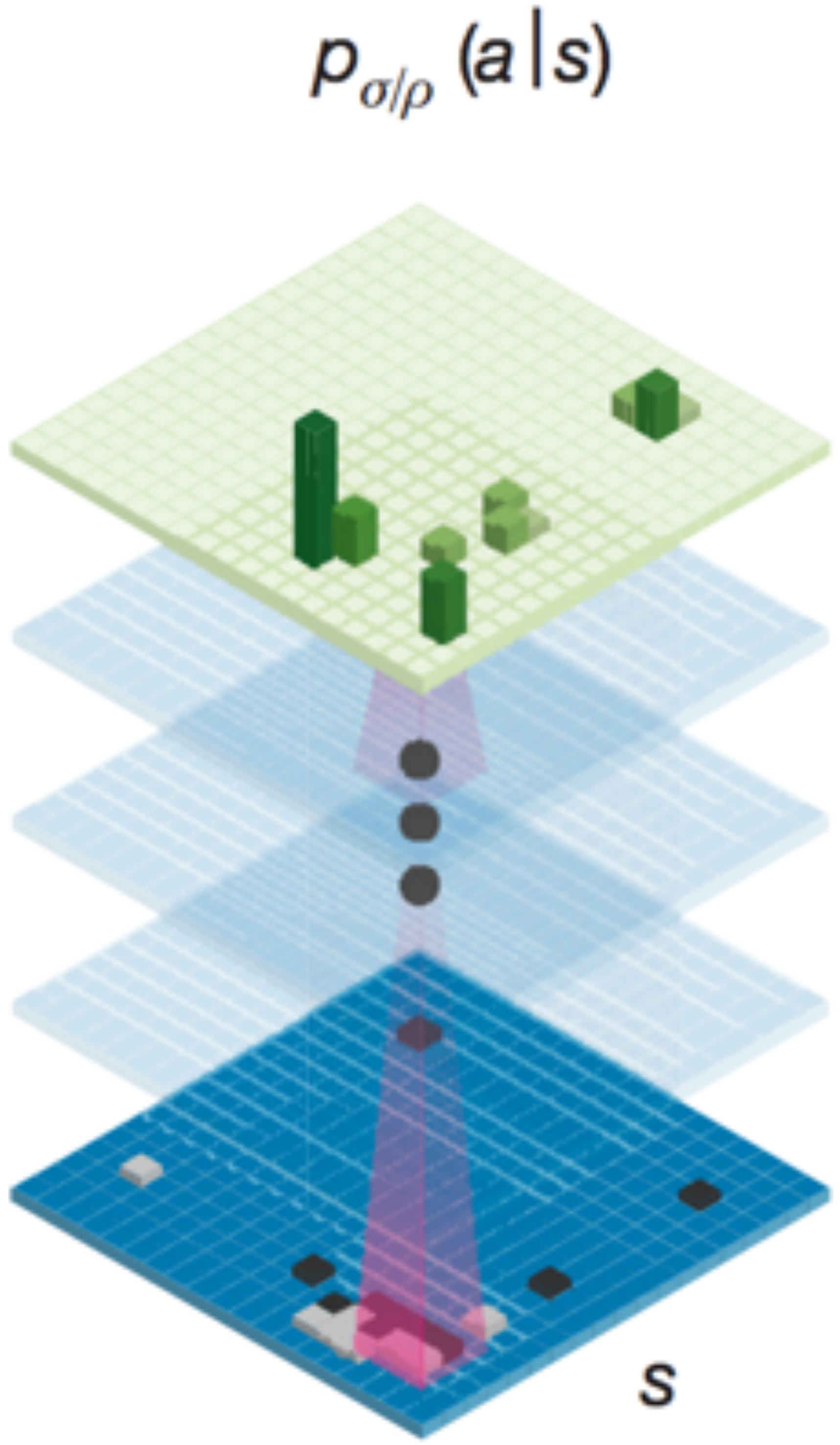


Supervised Learning policy (SL policy)

- Predicting the wining possibility given a board configuration (reinforcement learning)



(S, Z) — training pair (x,y) — Better policy?

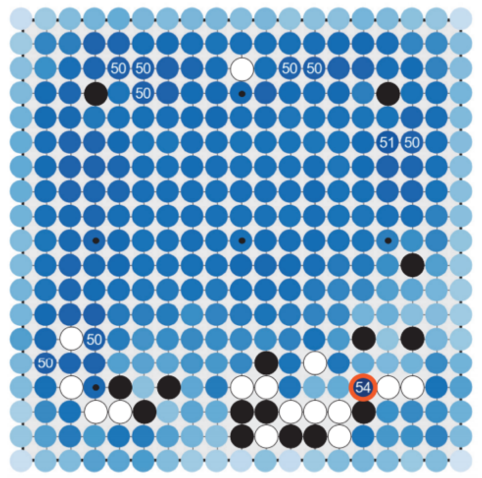


better policy → better estimation of the value

Reinforcement Learning policy (SL policy)

policy gradient

- Predicting the wining possibility given a board configuration (reinforcement learning)



ver. 0 = Supervised Learning policy (SL policy)

Policy Network ver.0 vs Policy Network ver.0 → Win/loss

......

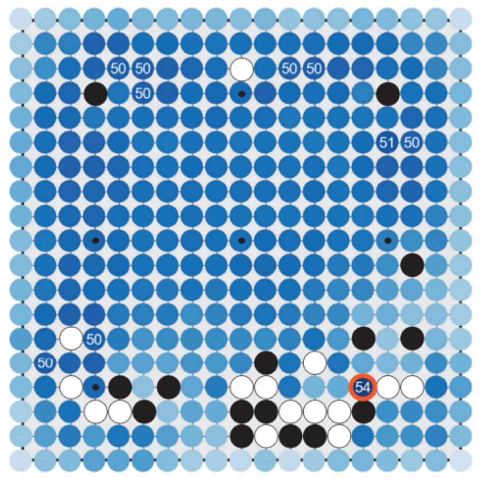
Policy Network ver. 200 vs Policy Network ver.153 → Win/loss

Policy Network ver. 201 vs Policy Network ver.78 → Win/loss

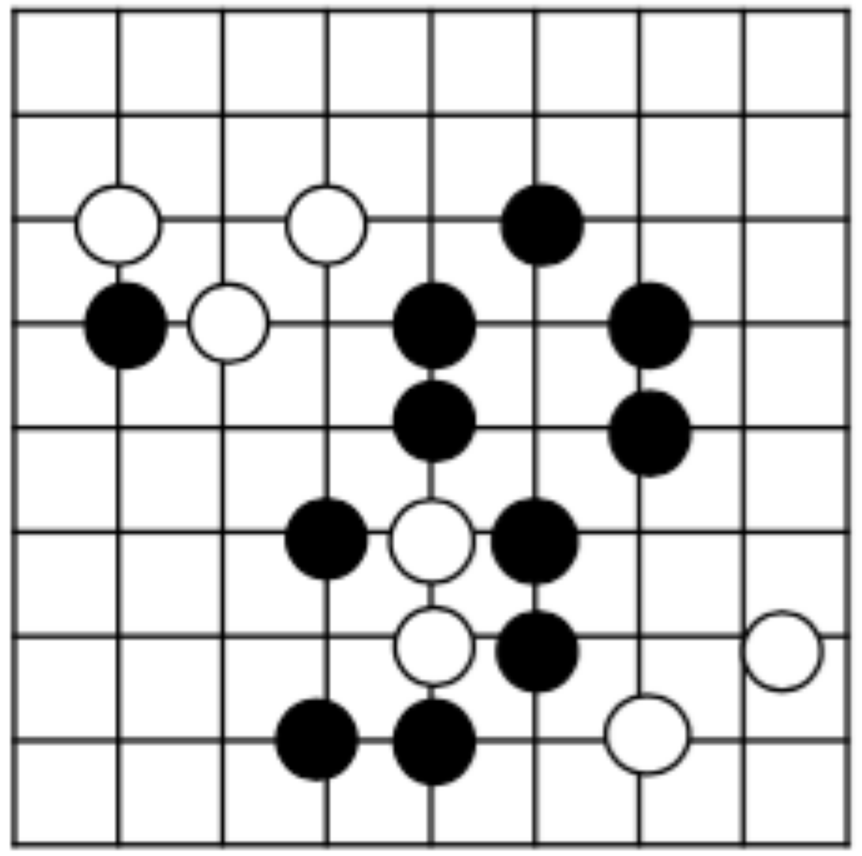
......

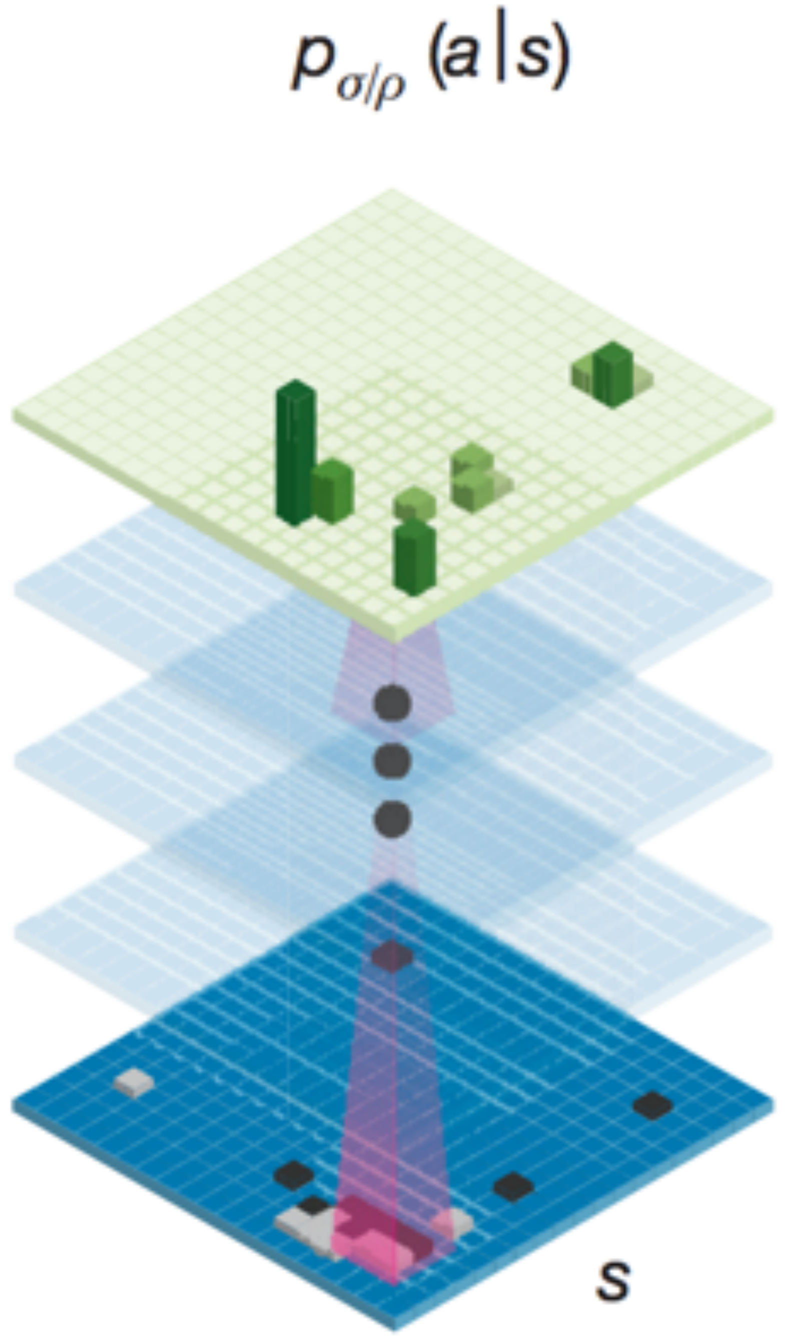
Policy Network ver. 20000 vs Policy Network ver. 12531 → Win/loss

- Predicting the wining possibility given a board configuration (reinforcement learning)



Board position





win z = 1

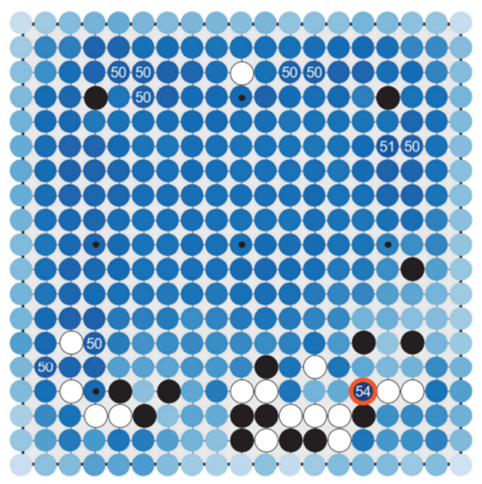
lose z = -1

Update

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

policy gradient

- Predicting the wining possibility given a board configuration (reinforcement learning)



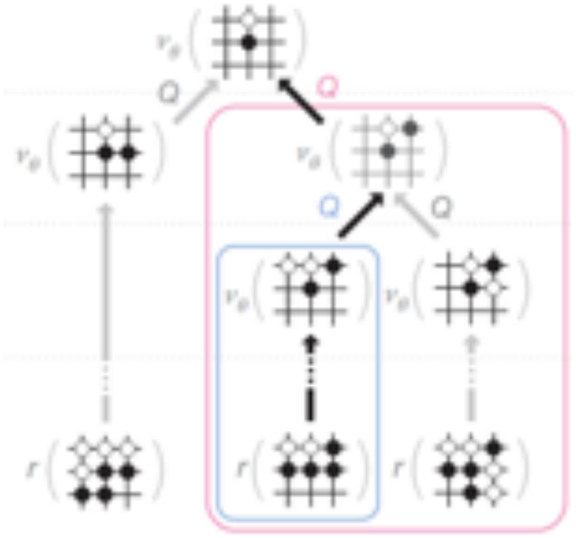
Policy Network ver. 20000

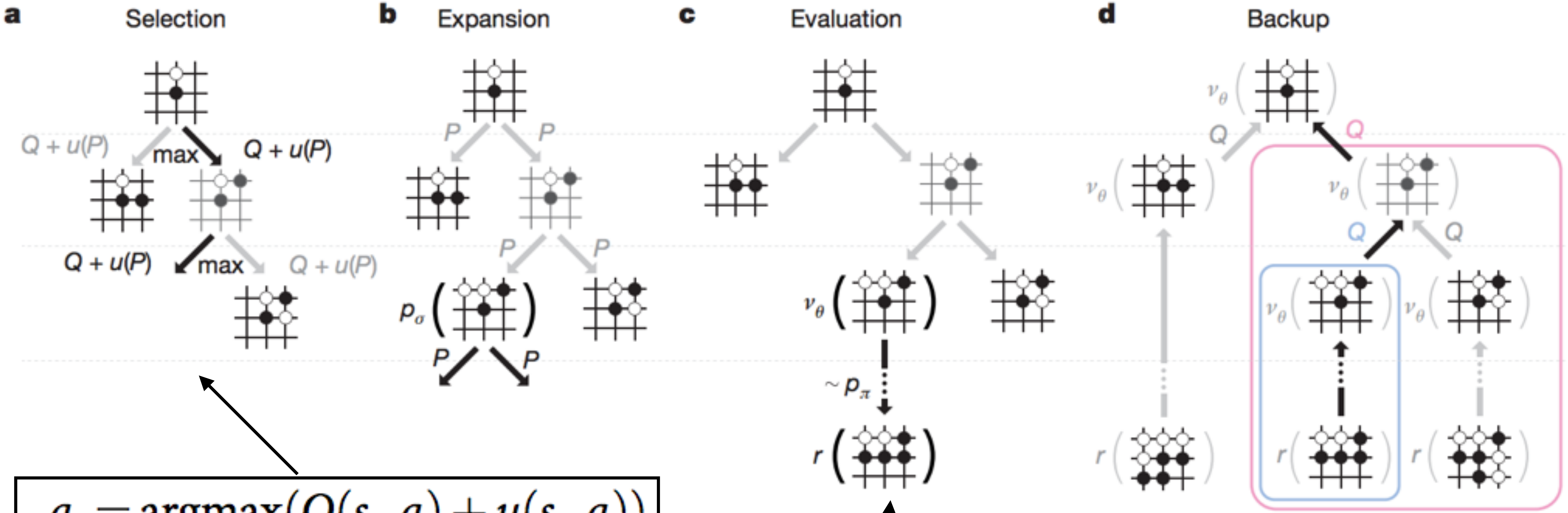
Reinforcement Learning policy (RL policy)

2

“a new self-play data set consisting of 30,000,000 positions, each sampled from a separate game”

- Select a move, more wisely (Monte Carlo tree search)





$$a_t = \operatorname*{argmax}_a \left(Q(s_t, a) + u(s_t, a)\right)$$

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

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

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

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

Infinite Monkey Theorem