General Principles of Human and Machine Learning

Lecture 4: Advances in RL

Dr Charline Tessereau

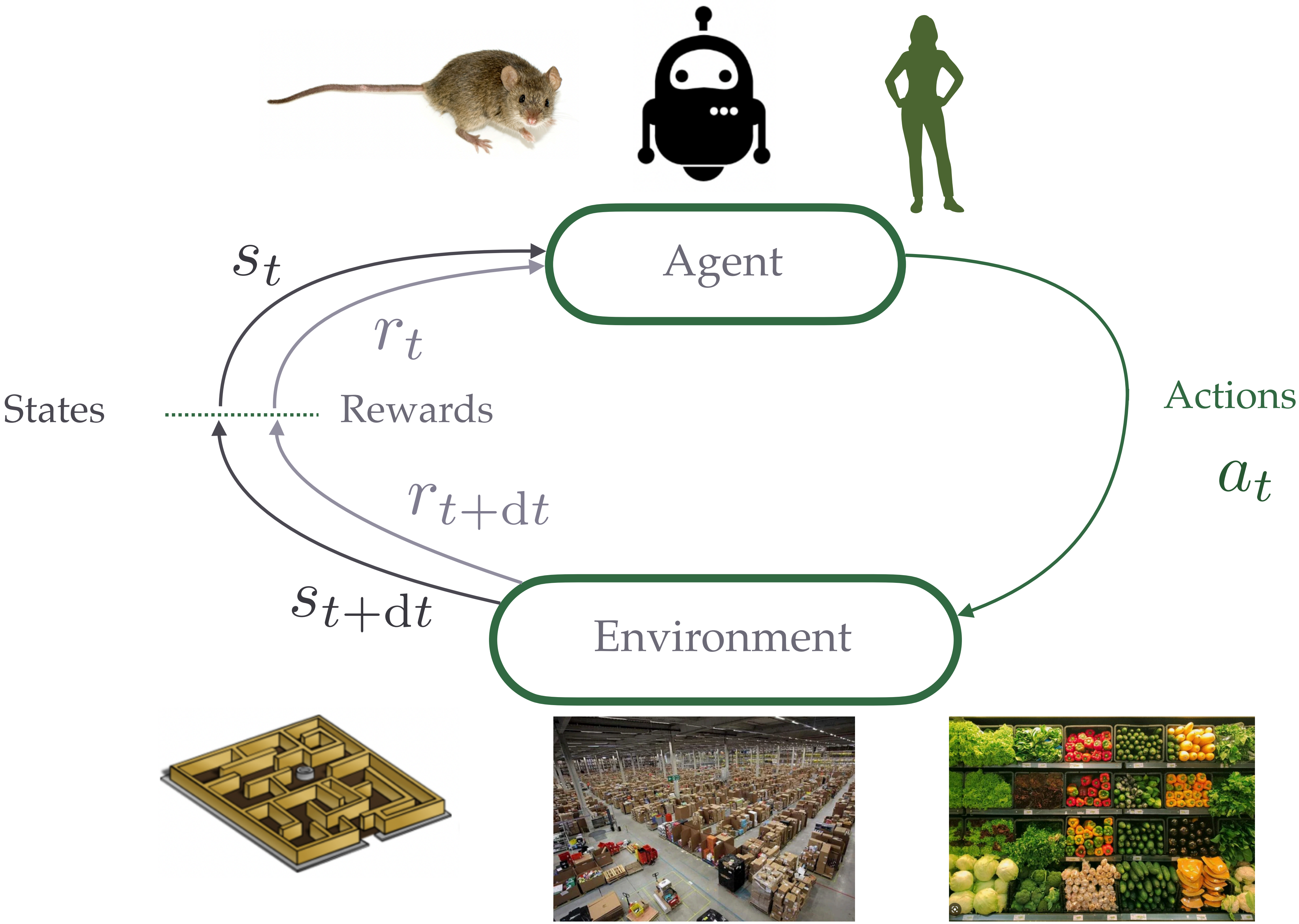
https://hmc-lab.com/GPHML.html

SUMMARY SO FAR

- ‘RL’ -> a more complete modern version of RW

Sutton and Barto. *Reinforcement learning: An introduction*. MIT press. (2018)

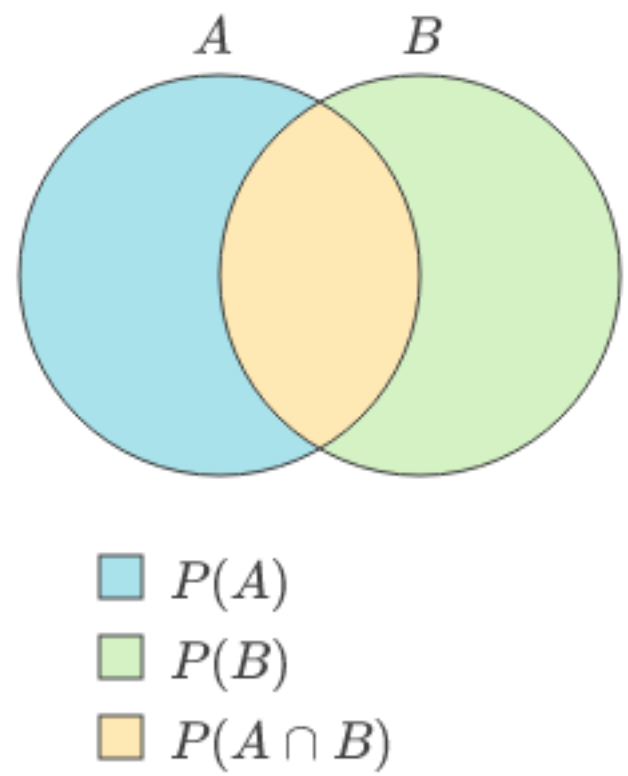
REINFORCEMENT LEARNING: LINK BETWEEN STATES AND ACTION VIA INTERACTION WITH THE ENVIRONMENT



Sutton and Barto. *Reinforcement learning: An introduction*. MIT press. (2018)

SUMMARY SO FAR

Conditional Probability and Markov property



Conditional Probability Formula

$$P(A \mid B) = \frac{P(A \cap B)}{P(B)}$$

Probability that A occurs given that B has already occured

Conditional probability is a way to measure the likelihood of an event happening, given that another event has already occurred. It helps us understand how the probability of one event changes when we have additional information.

Sutton and Barto. *Reinforcement learning: An introduction*. MIT press. (2018)

SUMMARY SO FAR

Markov property

$$P(X_{n+1}|X_n, X_{n-1}, ..., X_1) = P(X_{n+1}|X_n)$$

represents the probability of the next event given the current event and all the previous events.

represents the probability of the next event given only the current event, without considering any of the previous events.

The future states of a stochastic process depends only on the current state and is independent of the past states, given the present state.

Sutton and Barto. *Reinforcement learning: An introduction*. MIT press. (2018)

SUMMARY SO FAR

TD update for Q value

$$Q_{new}(s,a) = \underbrace{Q(s,a)}_{\text{Current Q}} + \underbrace{\alpha}_{\text{Learning Rate}} \cdot \left[\underbrace{R(s,a)}_{\text{Reward}} + \underbrace{\gamma}_{\text{Discount rate}} \cdot \underbrace{maxQ'(s',a')}_{\text{Max Q}} - \underbrace{Q(s,a)}_{\text{Current Q}}\right]$$

[0, 1]
distant future
vs.
immediate future

SUMMARY SO FAR

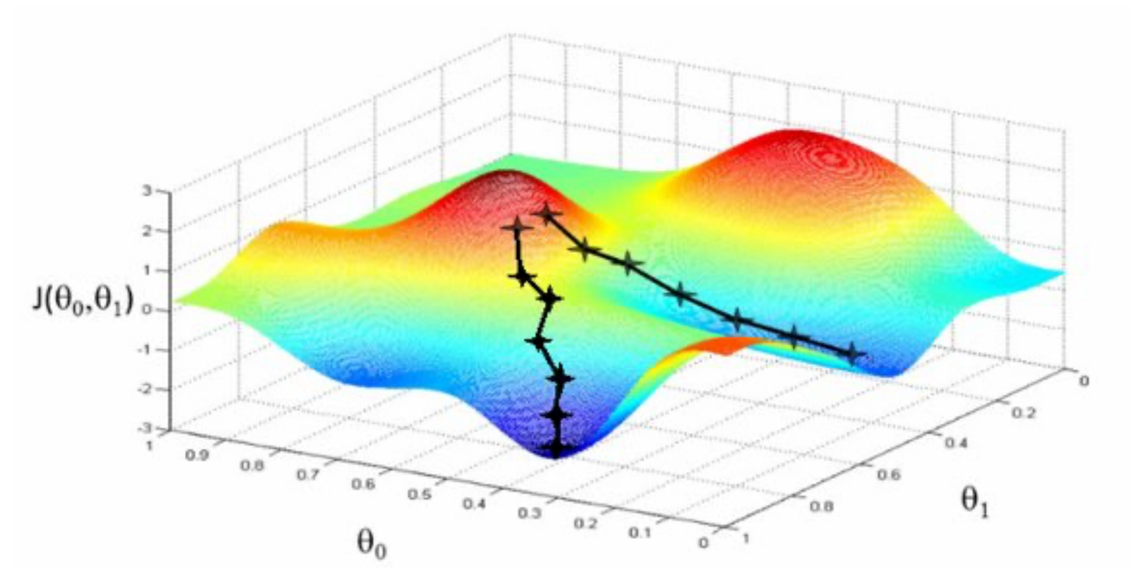
Q value : fails to scale up to large action spaces

POLICY GRADIENT

- directly learn how to act
- not via learning the Q function

POLICY GRADIENT

$$\Delta\theta = \alpha \nabla \pi(s, \theta) * J(s, \theta)$$

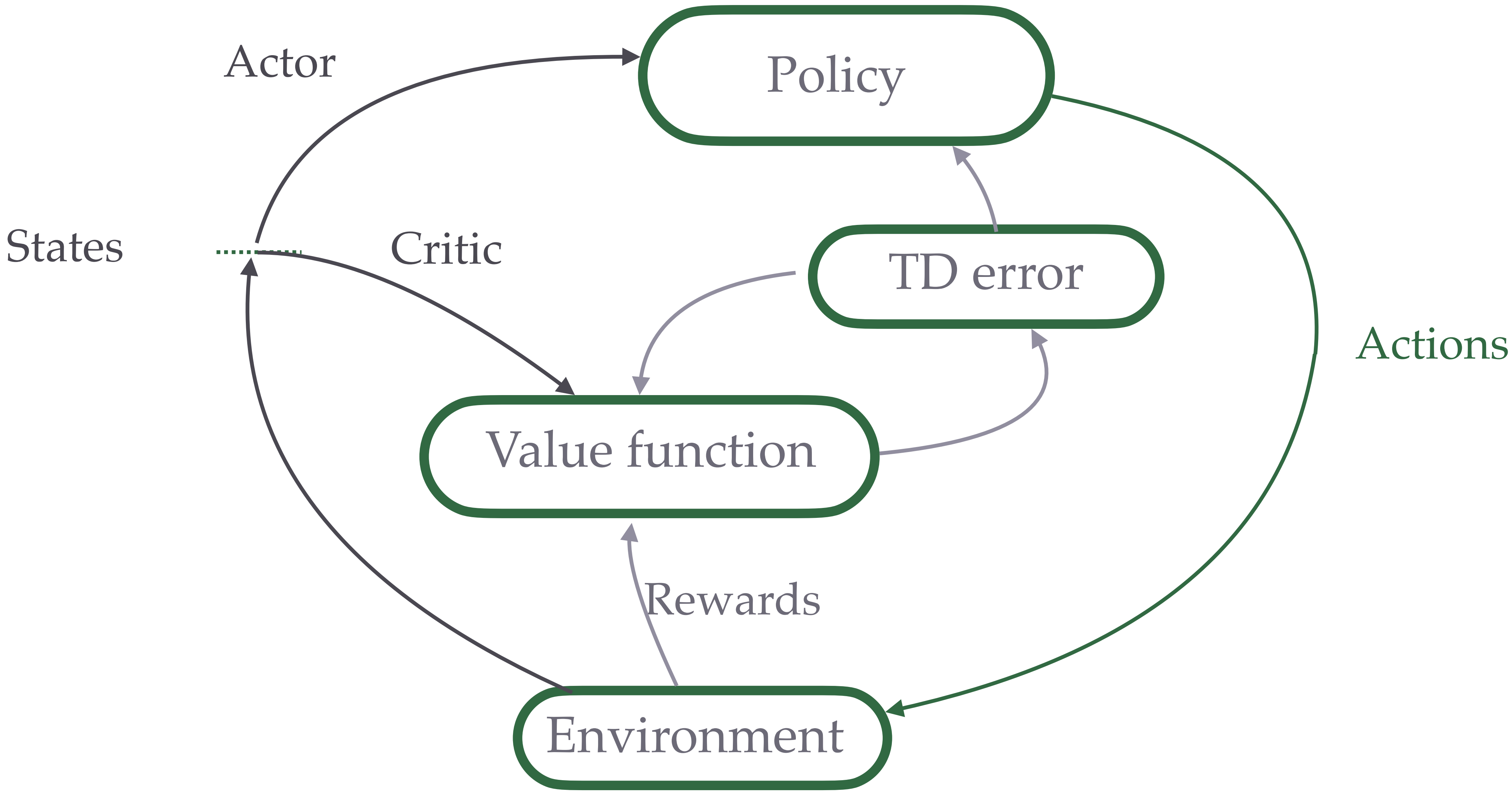


- $\Delta\theta$: update to the policy parameters.
- α learning rate
- $\nabla\pi$ (s, θ) : gradient of the policy with respect to the parameters (how the policy changes as the parameters θ change)
- J(s, θ) is the performance or objective function (i.e. expected return) = how good the policy is in state s under the parameter setting θ.

PROBLEM: POLICY GRADIENT DOES NOT DIFFERENTIATE THE STATE

But wait - this doesn’t allow us to learn to behave differently based on the state … not ideal - maybe one action is only not good at one state

ACTOR-CRITIC



ACTOR-CRITIC

- Basically two agents: one that learns the policy (via policy gradient) and one that learns the value (using TD errors)
- ‘The best of both worlds’
    - Contrary to standard TD learning, can scale to large action spaces
    - Contrary to policy gradient, still provides a policy that depends on the state

Q-LEARNING VS ACTOR-CRITIC

- Q-learning:
    - Well suited for situations w discrete action spaces and known state transitions
    - ‘Off-policy’: can use any value function to learn the optimal policy
- Actor-critic:
    - Well suited for situations in a continuous action and/or state space
    - ‘On policy’: the learnings are directly dependent on the behavior that is currently produced

SUMMARY SO FAR

Really good source of explanation - links to code etc:
https://mpatacchiola.github.io/blog/2017/02/11/dissecting-reinforcement-learning-4.html

FROM Q VALUE TO ACTION SELECTION:

ε - greedy policy:

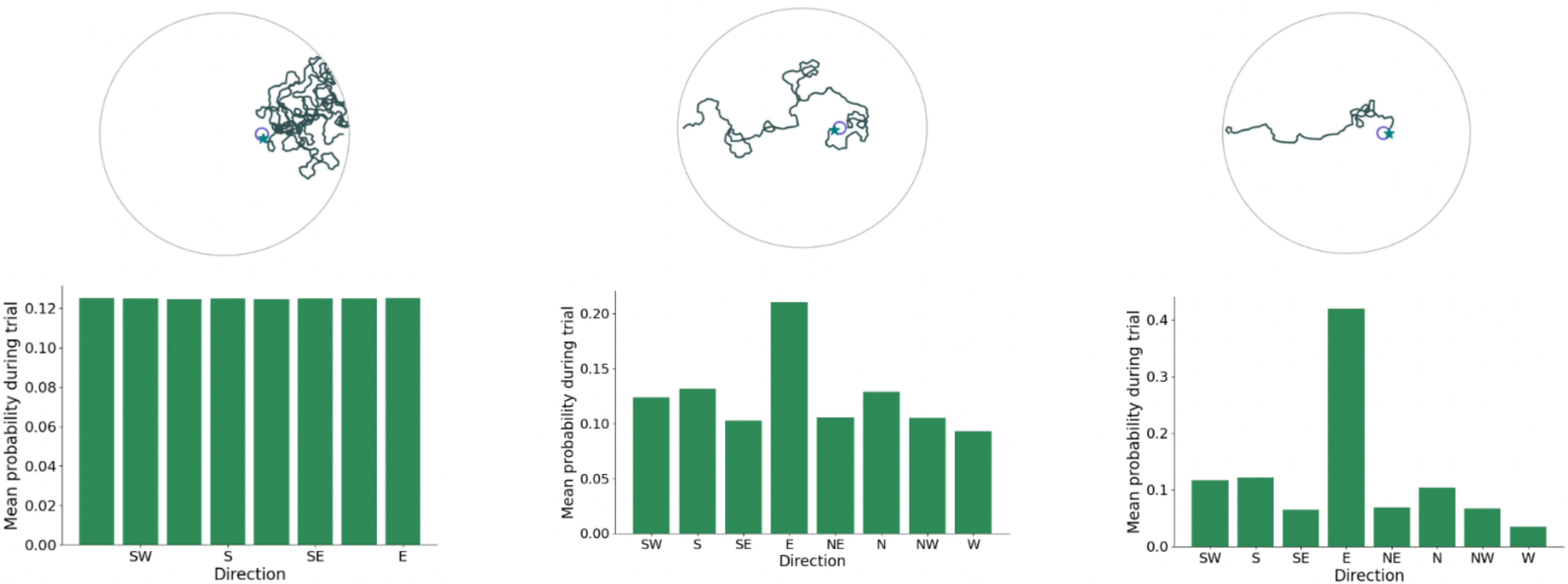
- $\pi(s, a_j) = \max_{a_j} Q(s, a_j)$ with probability 1- ε
- $\pi(s, a_j) = \text{Random}$ with probability ε

$$\pi(s, a_j) = \frac{\exp(\beta Q(s, a_j))}{\sum_{k=1}^{N_A} \exp\left(\beta Q(s, a_k)\right)}.$$

FROM Q VALUE TO ACTION SELECTION:

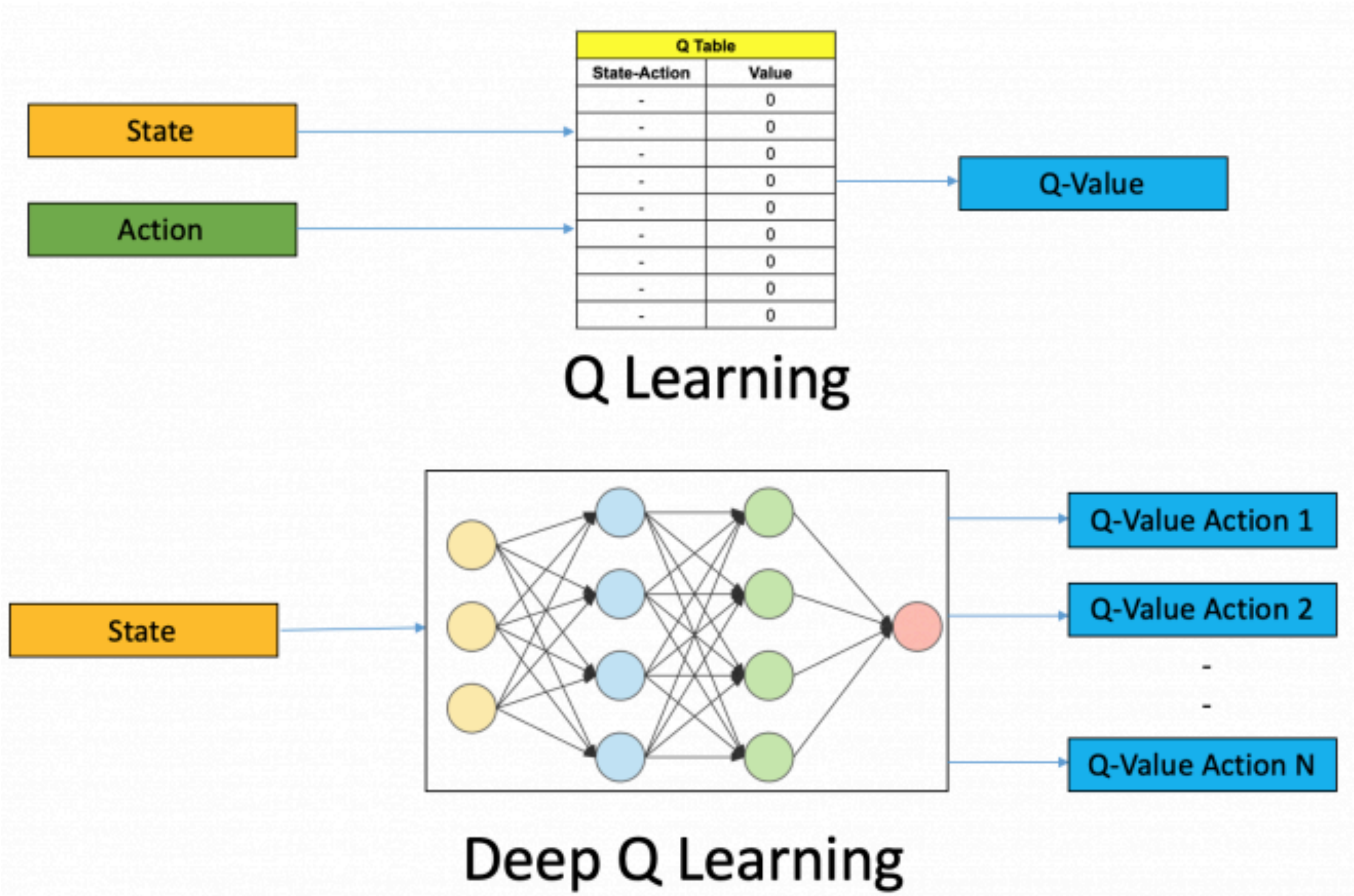
The effect of ε is straightforward: the higher the value the more random is the policy - the effect of β is actually the same.
Here are example trajectories (top) and corresponding policies (bottom) for different values of the inverse temperature:

β =0.01 β =0.2 β =0.6



DEEP Q LEARNING

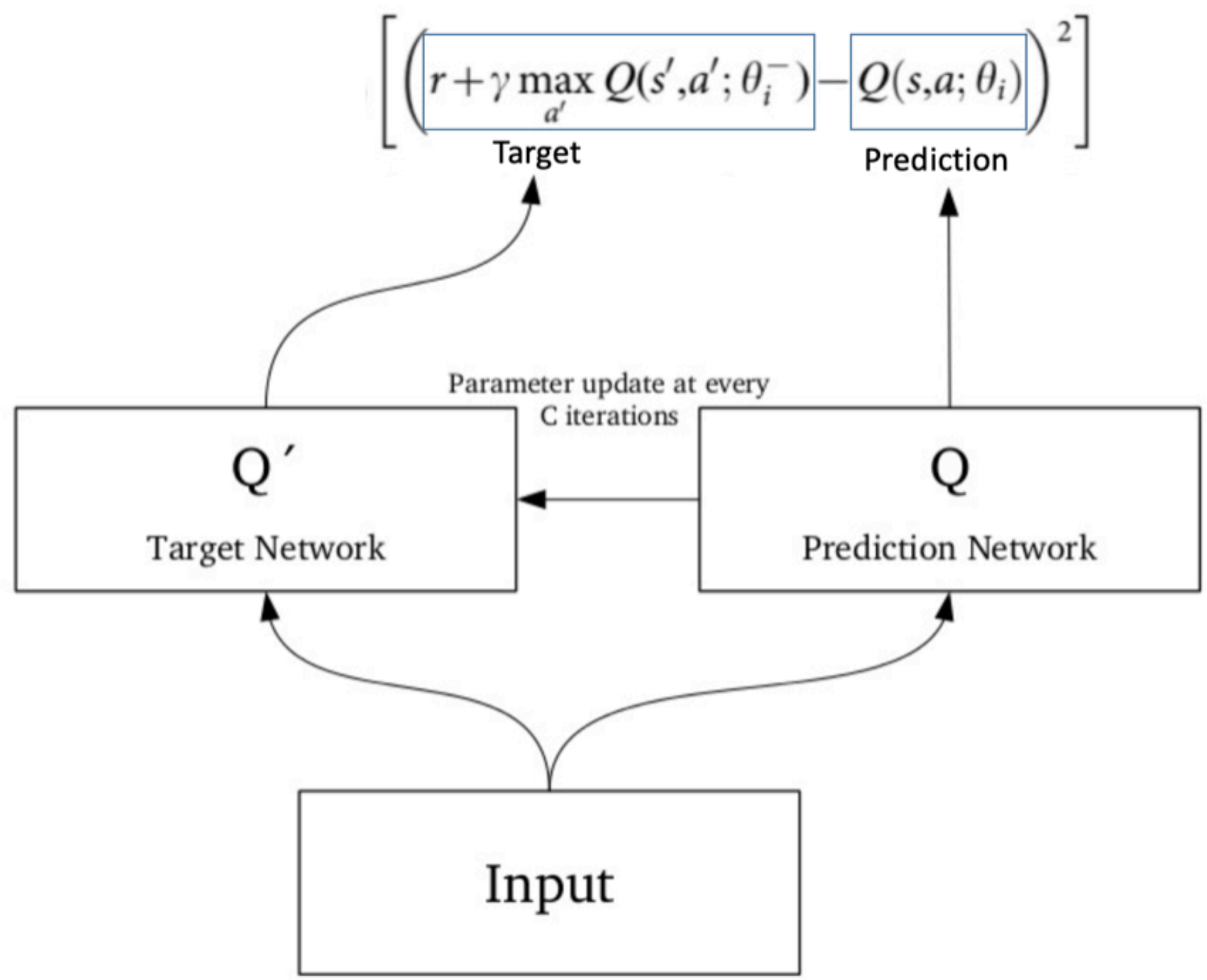
Instead of a Q-table (find the states and actions and read out the table), a neural network will approximate the Q value of every action from a state



State -> network -> Q values per actions (output dim number of actions)

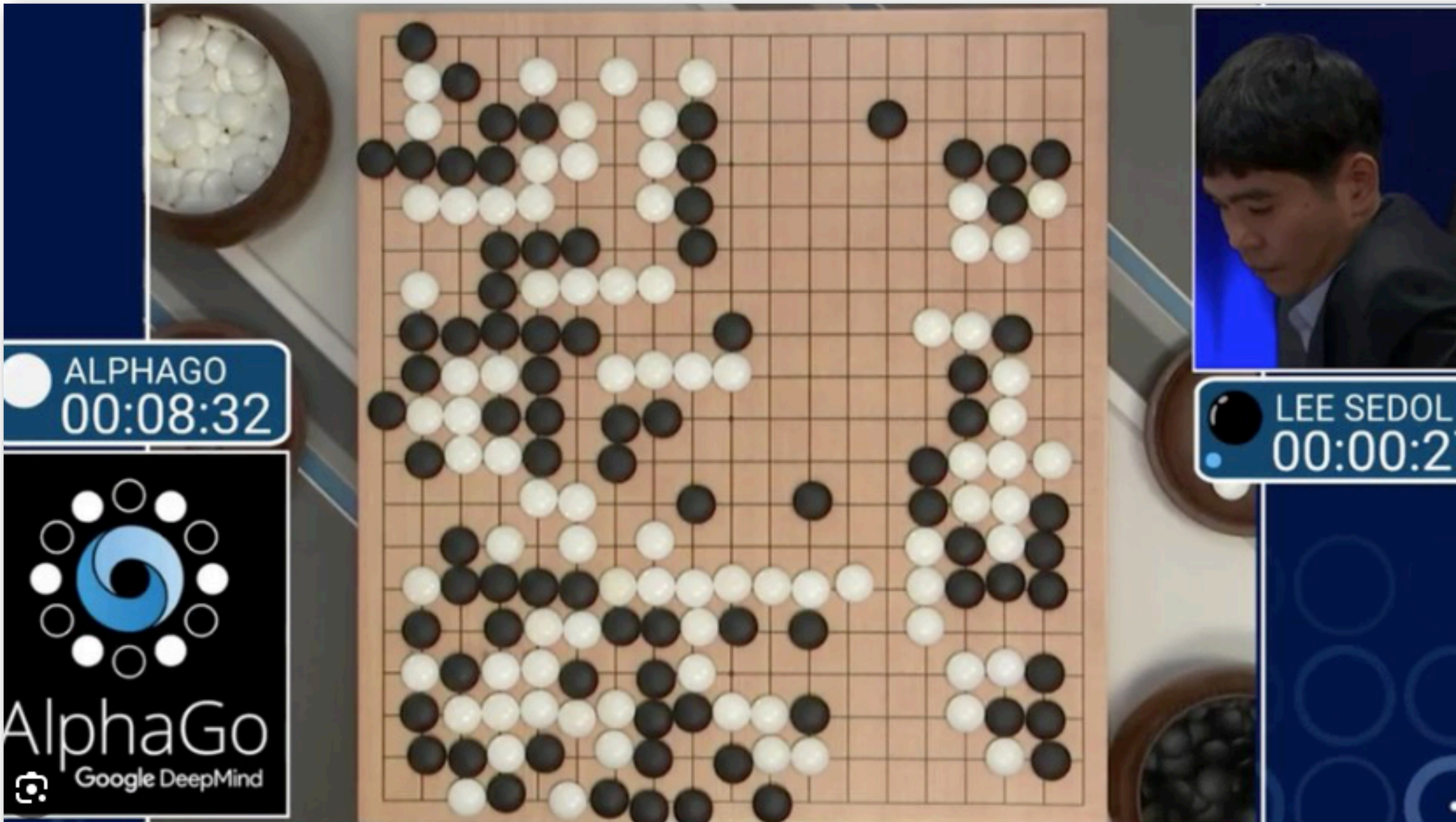
DEEP Q LEARNING

Algorithm :
Init
Choosing action
Updating weights using Bellman equation

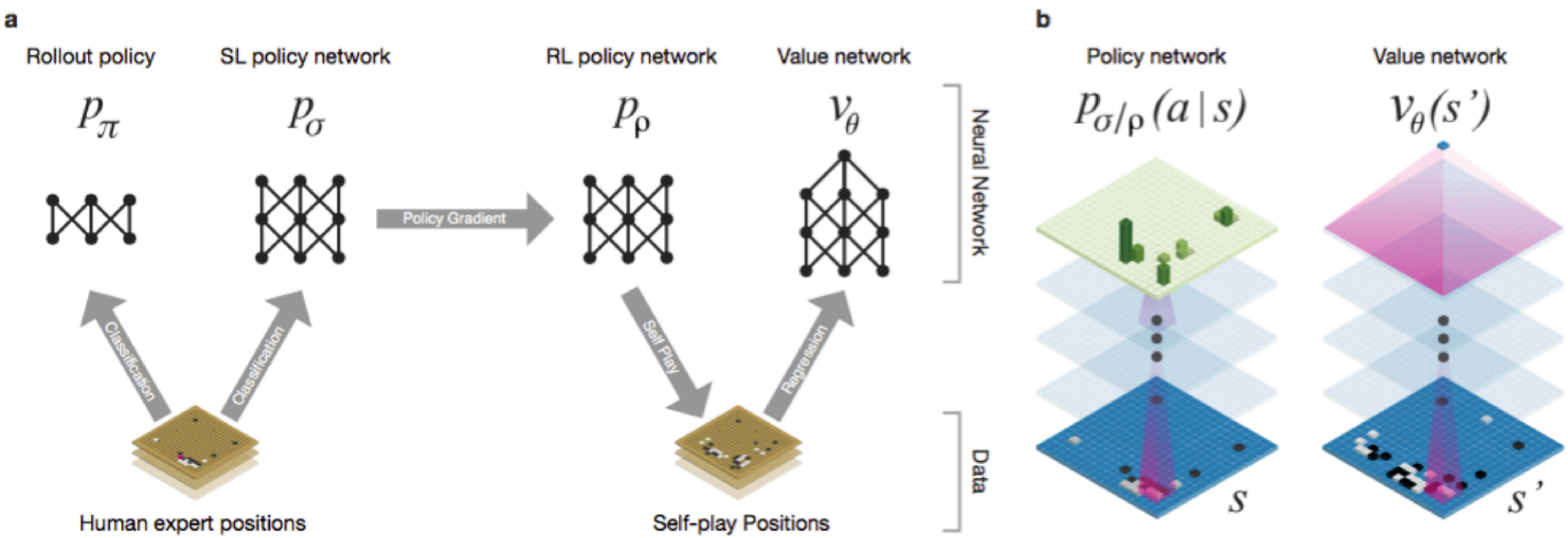


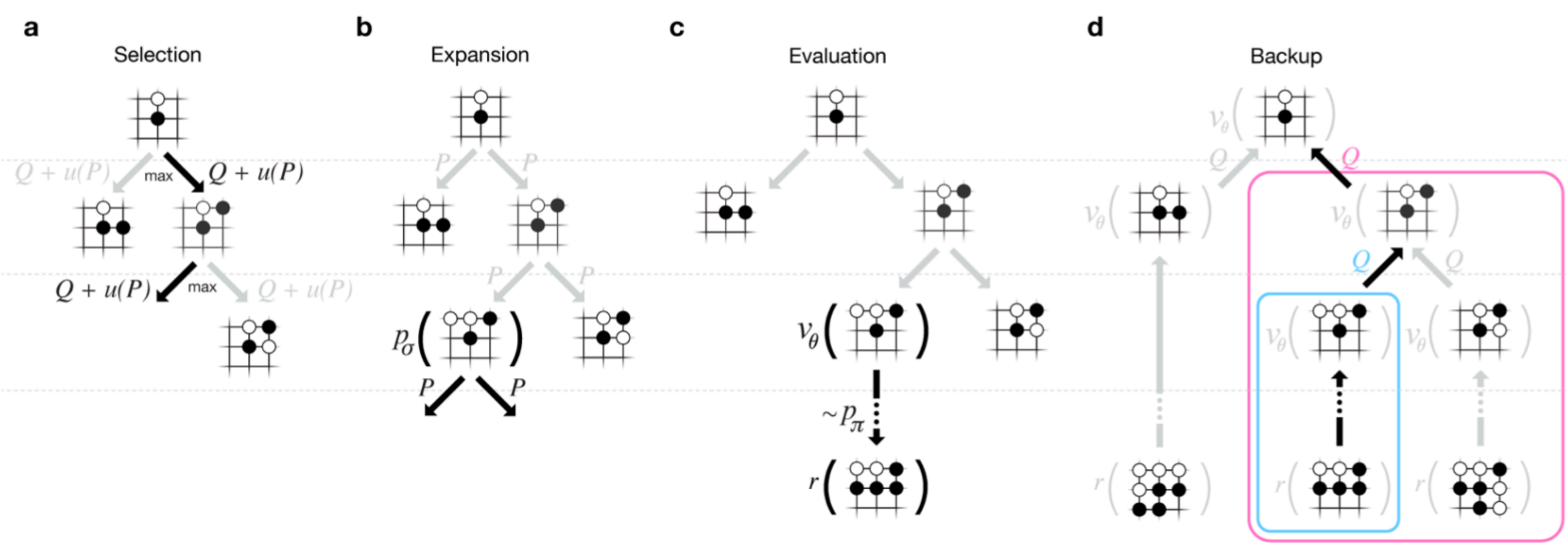
DEEP Q LEARNING

Alpha Go



DEEP Q LEARNING





DEEP Q LEARNING

AlphaGo has both a value network and a policy network to address different aspects of the game:

- The value network predicts the outcome of the game state, providing an estimate of the winning probabilities.
- The policy network, on the other hand, suggests the best move by assigning probabilities to different actions, aiding in the exploration and decision-making process.

By combining the two networks, AlphaGo can effectively evaluate game states and make informed strategic moves, enhancing its gameplay performance.

DEEP Q LEARNING

AlphaGo: combination of deep neural networks and Monte Carlo Tree Search (MCTS):

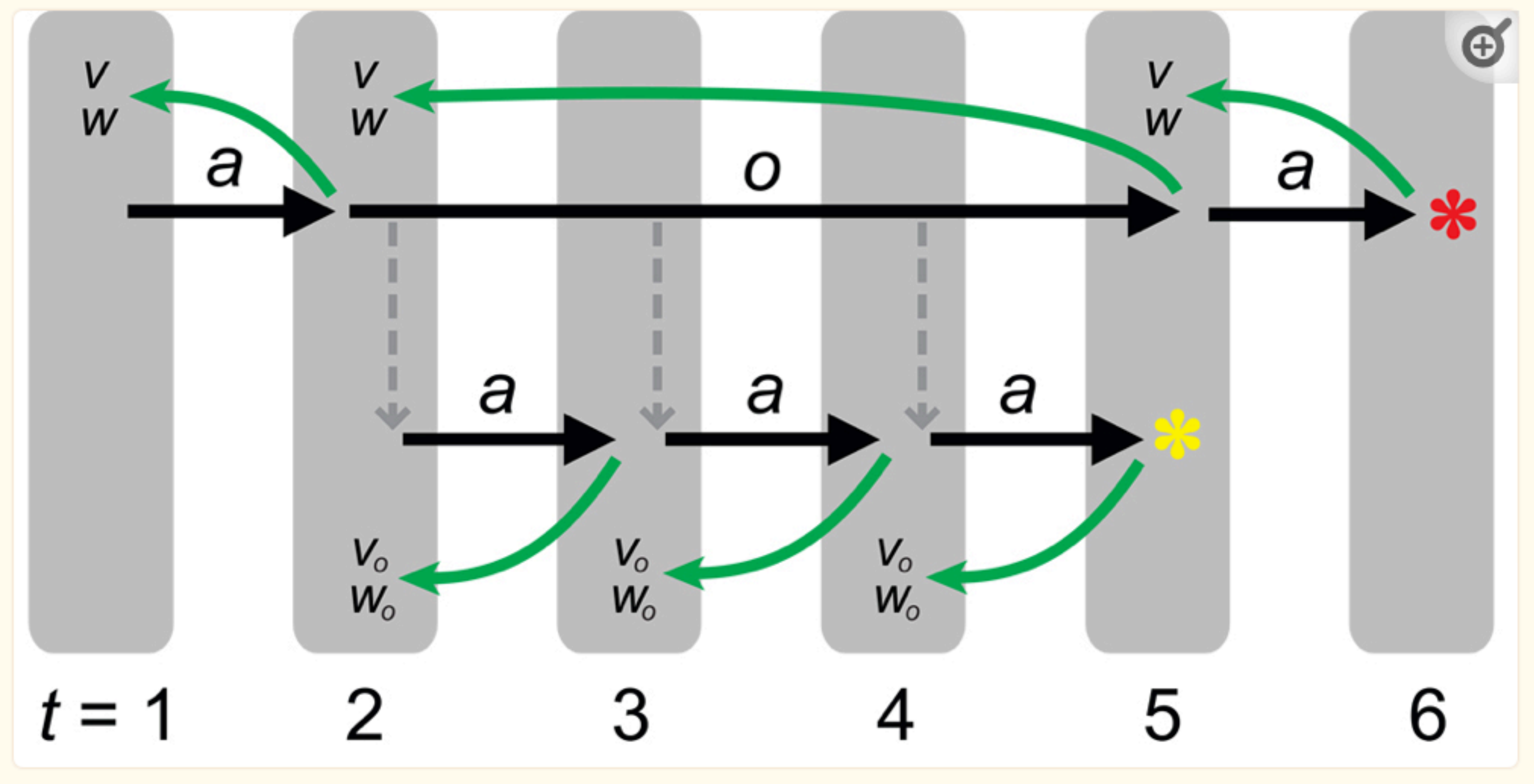
- It first trains a value network on expert human moves to estimate the outcome of game states.
- Then, it trains a policy network using reinforcement learning to suggest moves.
- During gameplay, AlphaGo performs MCTS simulations to explore possible moves and their outcomes, using the value and policy networks to guide the search.

The combination of deep learning and search algorithms is the strength

DEEP Q LEARNING

- Advantages:
  - Can handle high-dimensional state spaces
  - Generalization and Transfer Learning: has the ability to generalize learned knowledge to unseen or similar states. By capturing and representing the underlying structure of the environment in the neural network weights, it can facilitate transfer learning, where knowledge acquired in one task can be transferred to related tasks or domains.
- Shortcomings:
  - Sample Efficiency and Data Requirements: Deep Q-learning often requires a large amount of training data to effectively learn the action-value function
  - Lack of temporal abstraction: Deep Q Learning operates in one single timescale

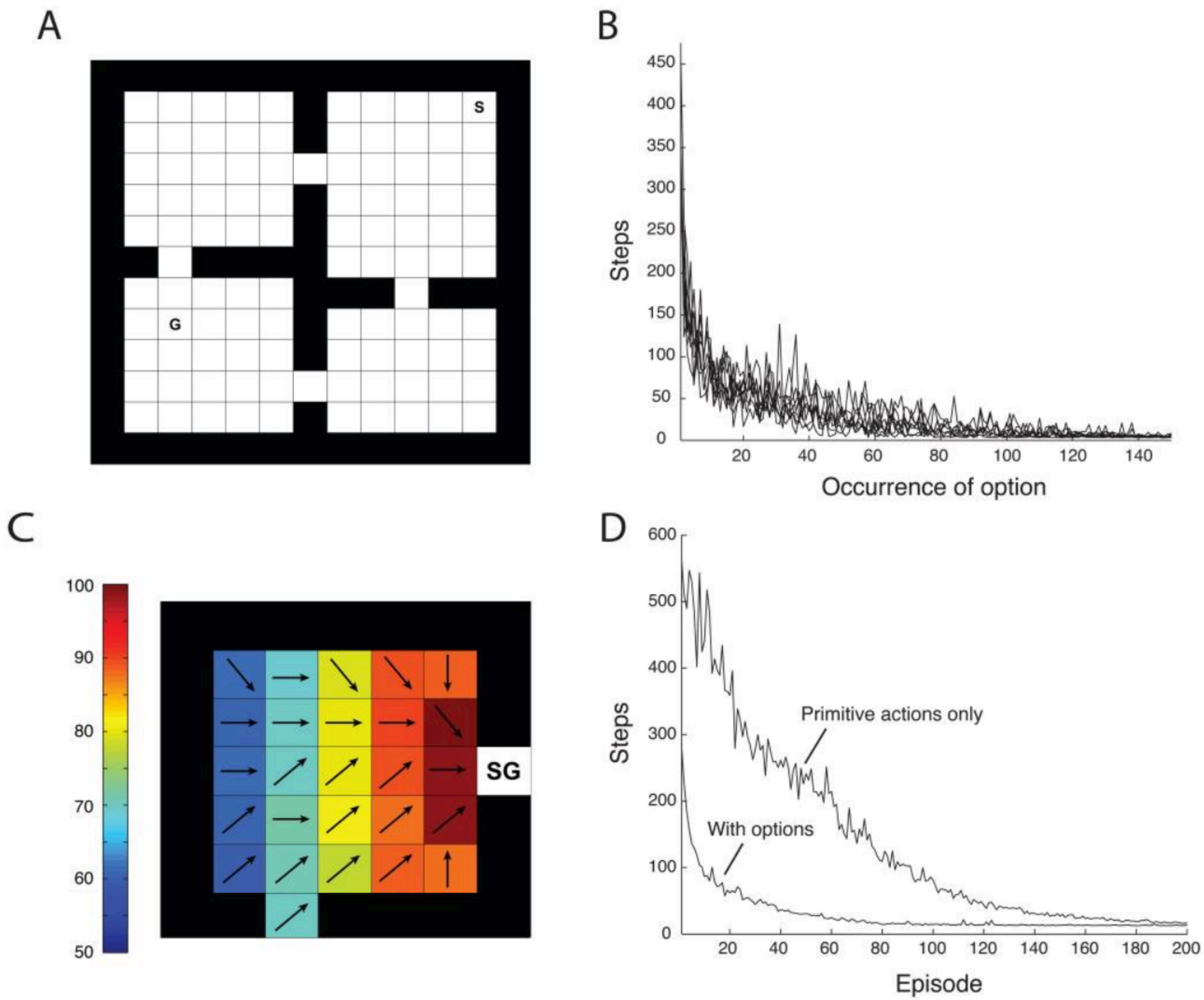
OPTION FRAMEWORK



- Hierarchical representation of actions
- An 'option' is basically a sequence of actions.

Botvinick, Niv and Barto. *Cognition*. (20109)

OPTION FRAMEWORK



Botvinick, Niv and Barto. *Cognition*. (20109)

OPTION FRAMEWORK

- Shortcomings:
    - Option Discovery: challenge of effectively discovering meaningful and useful options for a given task or environment.
    - Hierarchical Structure: Designing an optimal hierarchical structure of options that balances granularity and complexity can be difficult and may require domain expertise
    - Credit Assignment: Assigning credit to the options within the framework and properly attributing rewards or penalties to the appropriate levels of the hierarchy can be nontrivial and may require careful design and implementation.

OPTION FRAMEWORK

- Advantages:
  - enables the representation of temporally extended actions or behaviors, allowing agents to perform more complex and efficient actions in a hierarchical manner.
  - Reusability: Options can be learned and reused across different states and tasks, promoting faster learning and improved performance by leveraging previously acquired sub-policies or skills
  - Efficient Exploration: Hierarchical architectures, such as the option framework, provide higher-level exploration strategies, guiding the agent to explore in a more purposeful and efficient manner, leading to more effective exploration and learning.

SUMMARY MODEL-FREE RL

- For ‘small’ tasks:
    - Value or Q-learning: basically quick and dirty - there is a table that tells you what is the best action to perform in that state - still needs comparison between all action values - so not scalable to big action and state spaces
    - Policy gradient: basically repeating previously successful actions in that states
    - Actor-critic: 2 cooperating agents: one learns the value of states, the other one the policy
- For ‘bigger-scaled’ problems:
    - Deep Q-learning: better transfer learning across states - so good for large states and action spaces than traditional Q-learning because the network can make generalization between states
    - Hierarchical RL: enables extended action sequences + reusing options across tasks