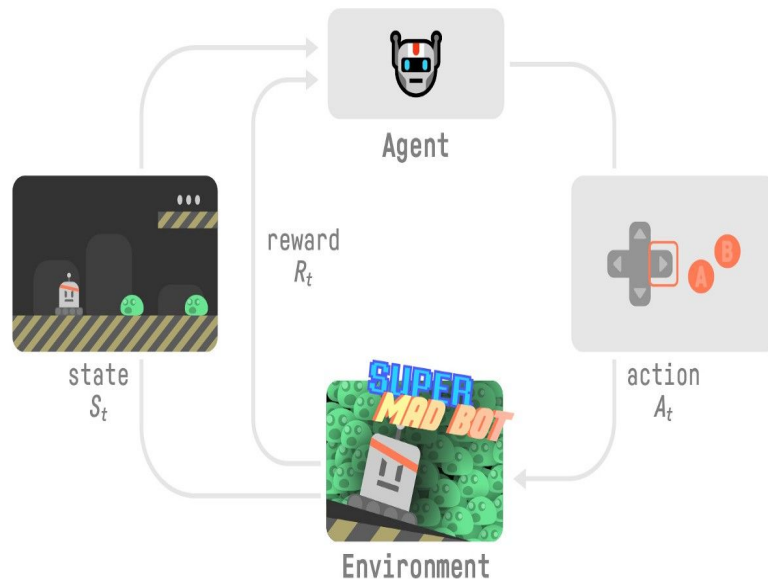


# **Introduction to Reinforcement Learning**

# Key Components of an RL System

Differs from supervised and unsupervised learning by labels.

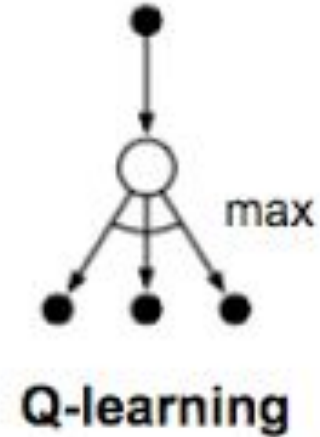
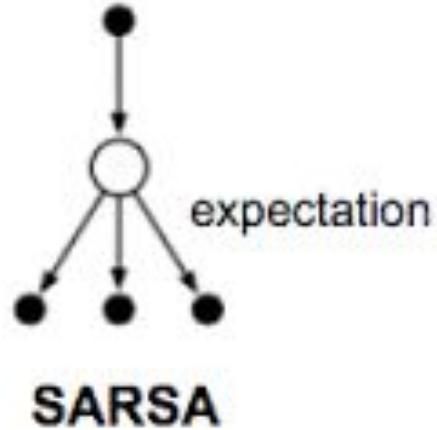
- In RL:
  - Environment
  - Agent
  - States
  - Actions
  - Rewards
- Types of RL
  - Episodic
  - Continuing



# Markov Decision Processes

- MDP is a framework for modeling the interaction between agents and environment.
- Agent needs only the current state to decide what action to take and not the history of all the states and actions they took before
- Outcomes are partly random, partly controlled by decision maker

# Sarsa vs Q-Learning



## SARSA

$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left[ R_{t+1} + \gamma Q(S_{t+1}, A_{t+1}) - Q(S_t, A_t) \right].$$

## Expected SARSA

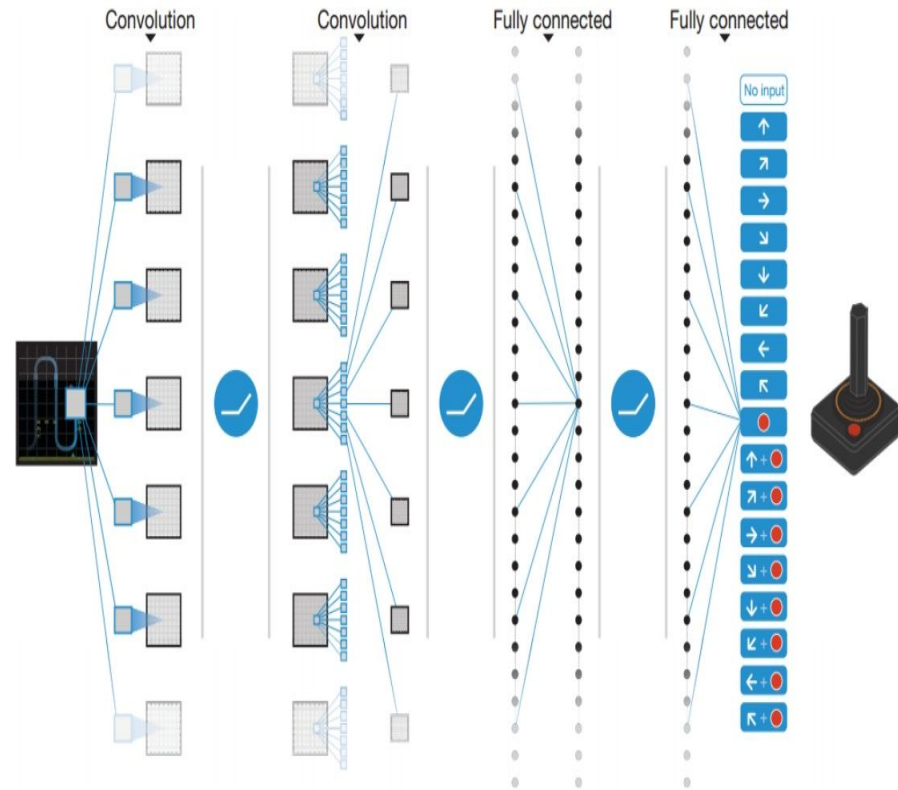
$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[ r_{t+1} + \gamma \sum_a \pi(s_{t+1}, a) Q(s_{t+1}, a) - Q(s_t, a_t) \right]$$

## Q-Learning

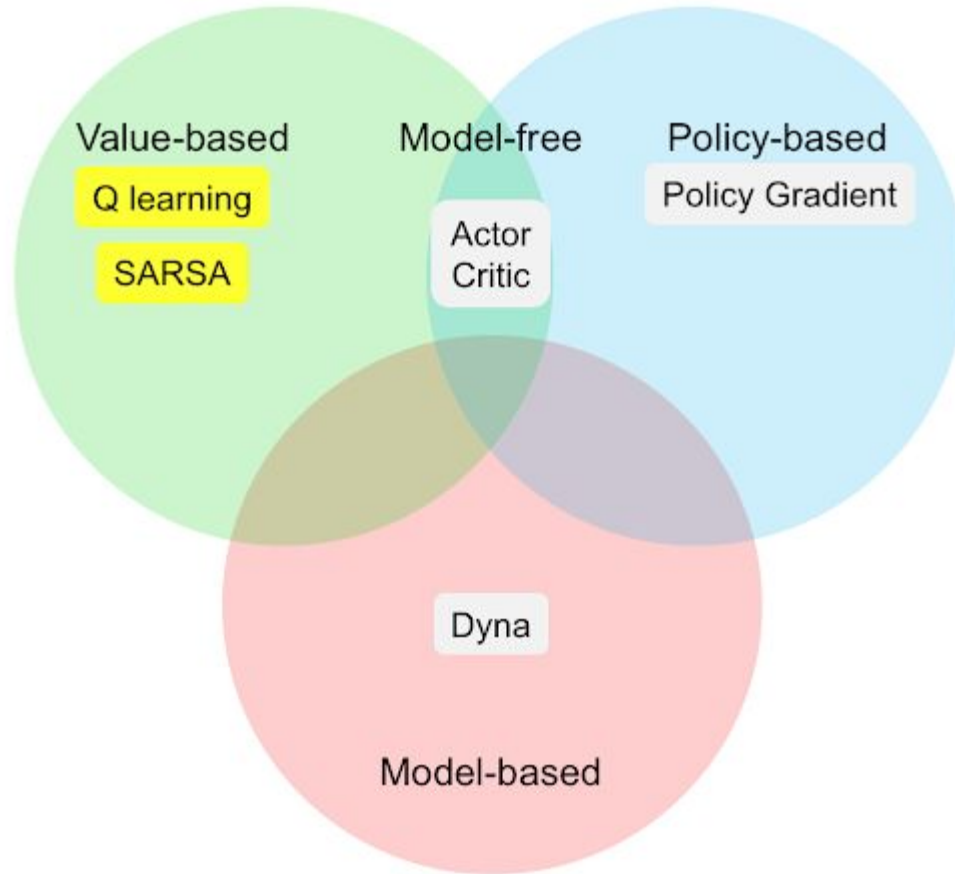
$$Q(S_t, A_t) \leftarrow Q(S_t, A_t) + \alpha \left[ R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t) \right]$$

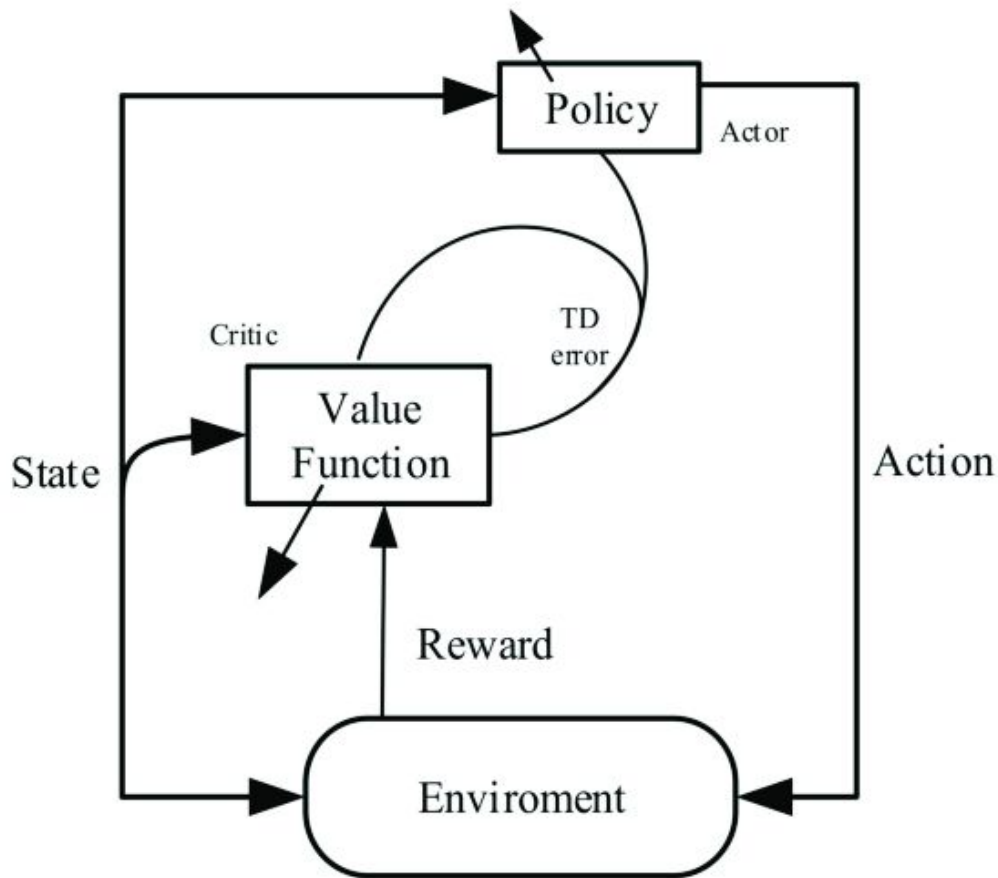
# Introduction to Deep Q-Networks

- Deep Q-Networks (DQNs) are an extension of Q-Learning that use neural networks to approximate Q-values,
- Enables them to handle large and complex state/action spaces.
- More suitable for tasks like playing video games where traditional Q-Learning struggles.



# Main Approaches





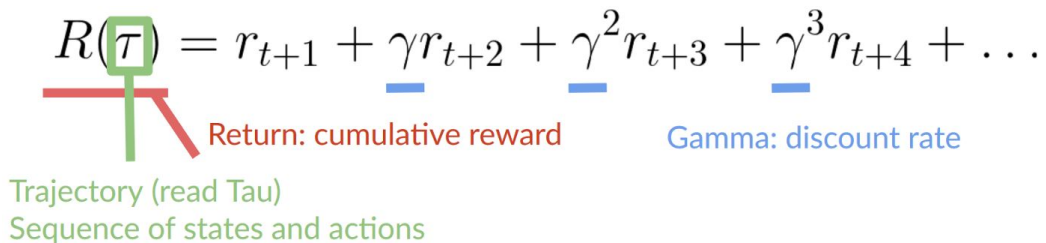
## Policy Gradient Methods

Policy Gradient Methods focus on optimizing the policy directly rather than the Q-values. These methods are particularly effective in environments with complex action spaces, such as video games, allowing for robust and dynamic policy learning.



# Influence of Reward Functions

- Reward functions play a pivotal role in shaping the behavior of RL agents.
- Proper design of reward functions ensures the agent aligns with desired outcomes, guiding its learning and decision-making process in game environments.

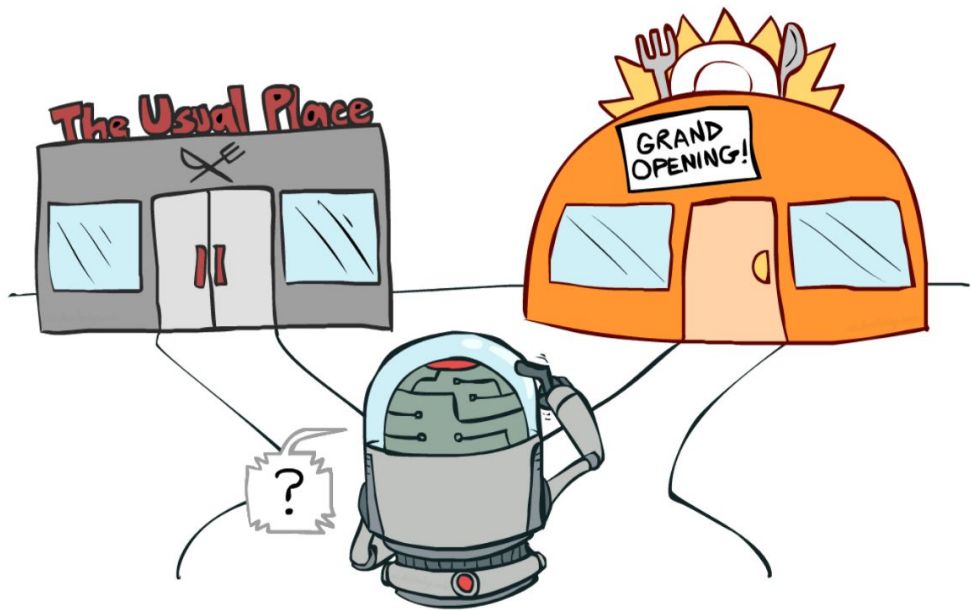

$$R(\tau) = r_{t+1} + \gamma r_{t+2} + \gamma^2 r_{t+3} + \gamma^3 r_{t+4} + \dots$$

Return: cumulative reward

Gamma: discount rate

Trajectory (read Tau)  
Sequence of states and actions

$$R(\tau) = \sum_{k=0}^{\infty} \gamma^k r_{t+k+1}$$



## Exploration vs. Exploitation Dilemma

The exploration vs. exploitation dilemma in RL involves deciding between exploring new actions to discover more rewarding outcomes or exploiting known actions that yield the highest reward.



## Tools and Frameworks for RL

- OpenAI Gym
- PyTorch RL
- TensorFlow Agents
- Ray RLlib
- Stable-Baselines

# Challenges in Implementing RL

- Implementing RL in real-time or complex games poses challenges like computational demand, learning efficiency, and scalability.

# RL Applications in Computer Games

- Reinforcement Learning is widely used in computer games to create intelligent and adaptive agents.
- Examples include atari games, board games and real-time strategy games.

# AlphaGo and AlphaZero

AlphaGo and AlphaZero are prominent models utilizing RL techniques to master board games like Go and Chess. These systems have demonstrated the immense potential of RL by defeating world champions, showcasing advanced decision-making and strategic planning polished through RL.

