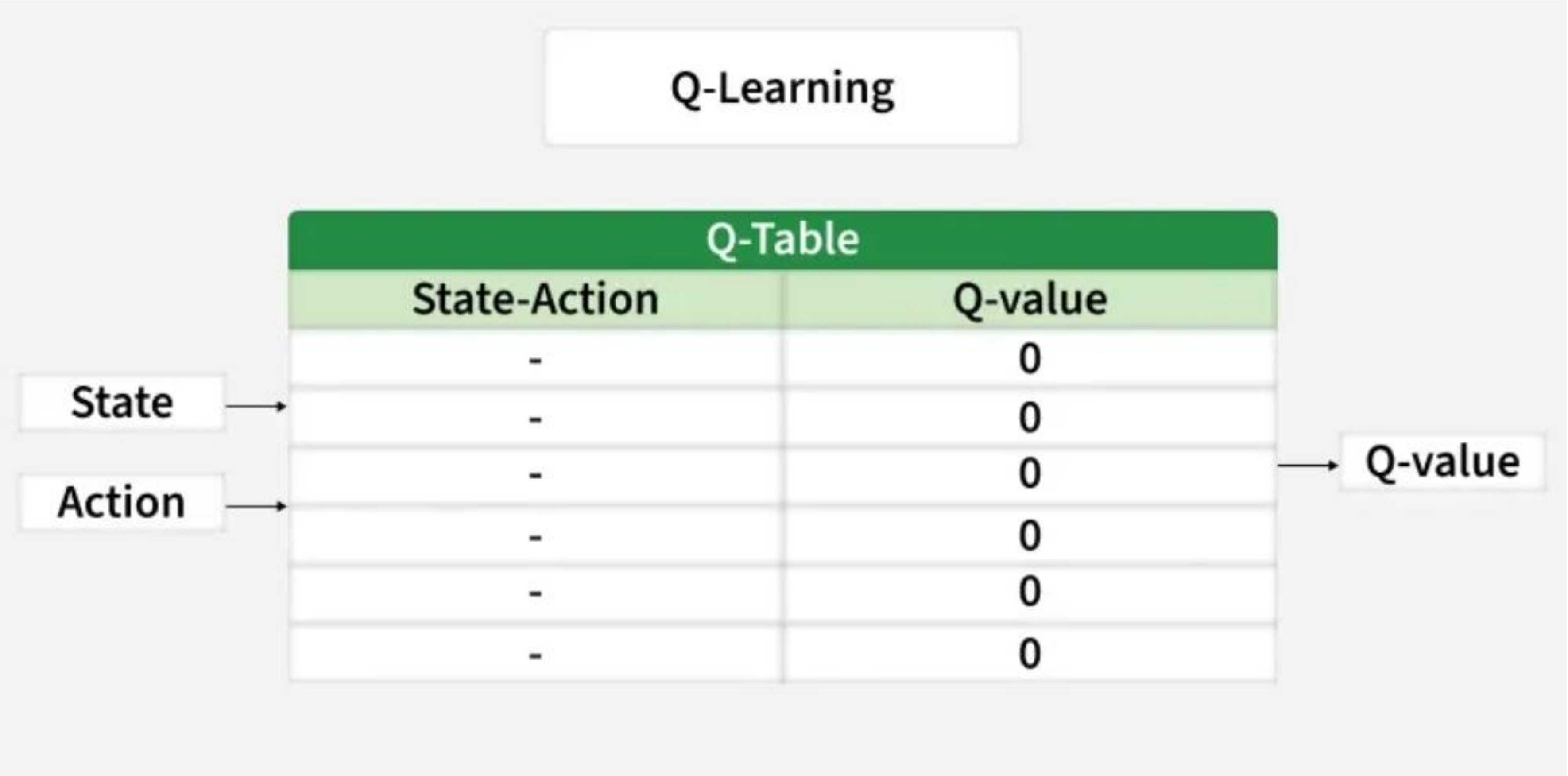
Deep Q-Network (Learning)

Introduction

- Deep Q-Learning is a method that uses deep learning to help machines make decisions in complicated situations.
- It’s especially useful in environments where the number of possible situations called states is very large like in video games or robotics.
- Before understanding Deep Q-Learning it’s important to understand the main concept of Q-Learning

- It is a model-free method that learns an optimal policy by estimating the Q-value function which tells how good it is to take a certain action in a certain situation.
- The goal is to find a plan that gives the highest total reward over time.



- Q-Learning works well for small problems but struggles with complex ones like images or many possible situations. Deep Q-Learning solves this by using a neural network to estimate values instead of a big table.

Key Challenges Addressed by Deep Q-Learning

- **High-Dimensional State Spaces:**

- Traditional Q-Learning uses a table to store values but this becomes impossible when there are too many situations.
- Neural networks can understand and work with many different situations at once so they are better for complex problems.

❑ Continuous Input Data:

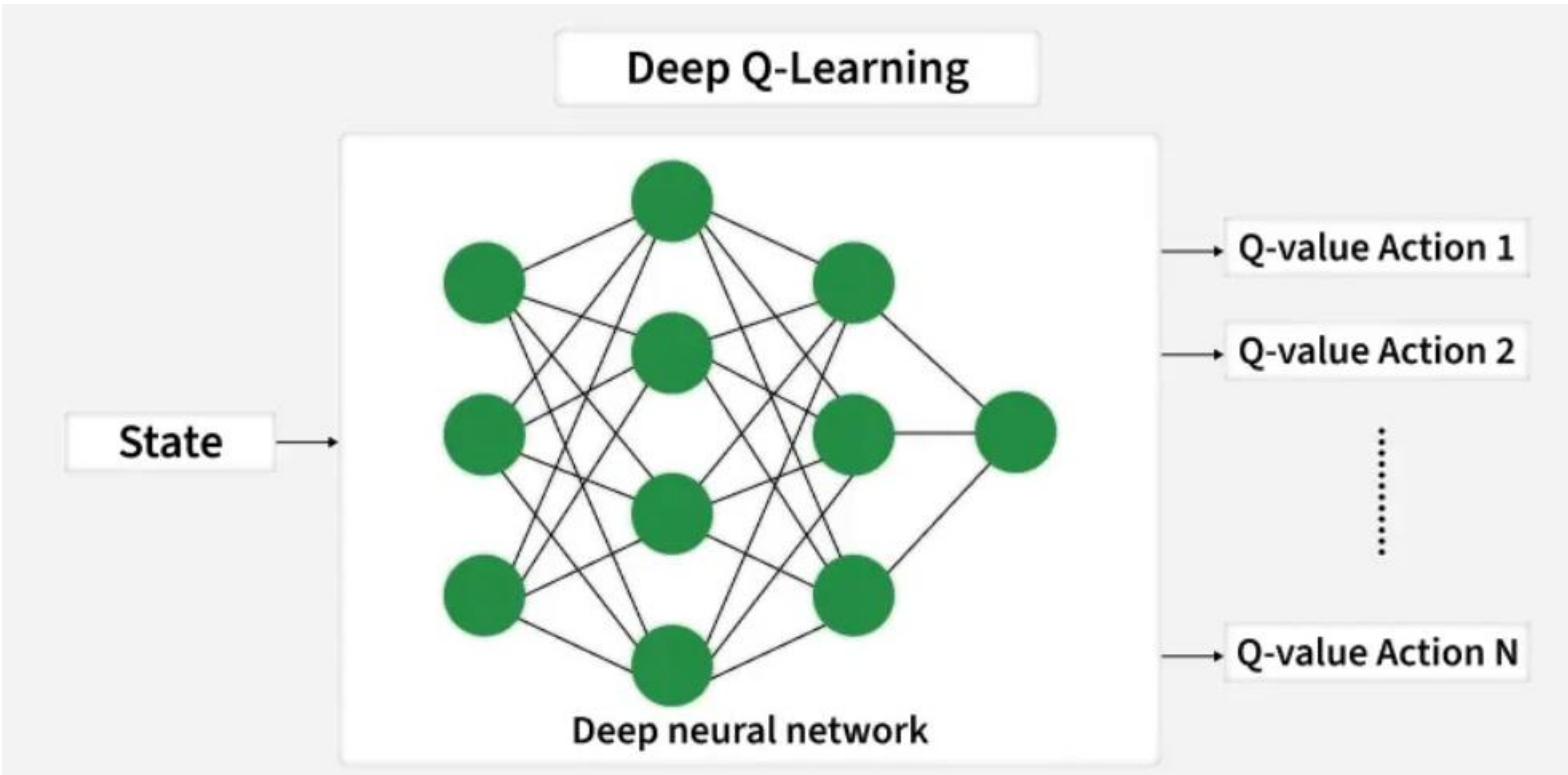
- ➢ Real-world problems often have continuous data like video images. Neural networks are good at handling this kind of information.

❑ Scalability:

- ➢ Deep learning helps Q-Learning grow and handle bigger, harder tasks that regular Q-Learning couldn't solve before.

Architecture of Deep Q-Networks

A DQN consists of the following components:



❑ Neural Network

- ➢ The network approximates the Q-value function $Q(s, a, \theta)$ where θ represents the trainable parameters.
- ➢ For example, in Atari games the input might be raw pixels from the game screen and the output is a vector of Q-values corresponding to each possible action.

❑ Experience Replay

- To stabilize training, DQNs store past experiences (s, a, r, s') in a replay buffer.
- During training, mini-batches of experiences are sampled randomly from the buffer, breaking the correlation between consecutive experiences and improving generalization.

❑ Target Network

- ➢ A separate target network with parameters θ^{-} is used to compute the target Q-values during updates. The target network is periodically updated with the weights of the main network to ensure stability.

$$L(\theta) = E[(r + \gamma \max_{a'} Q(s', a'; \theta^{-}) - Q(s, a; \theta))^2]$$

❑ Loss Function :

- The loss function measures the difference between the predicted Q-values and the target Q-values:

$$L(\theta) = E[(r + \gamma \max_{a'} Q(s', a'; \theta^-) - Q(s, a; \theta))^2]$$

Training Process

- **Initialization** :

- Initialize the replay buffer, main network (θ) and target network (θ^{-}).
- Set hyperparameters such as learning rate (α), discount factor (γ) and exploration rate (ϵ).

- **Exploration vs. Exploitation** : Use an ϵ-greedy policy to balance exploration and exploitation:
  - With probability ϵ, select a random action to explore.
  - Otherwise, choose the action with the highest Q-value according to the current network.

- **Experience Collection** : Interact with the environment, collect experiences (s, a, r, s') and store them in the replay buffer.

- **Training Updates** :

  - Sample a mini-batch of experiences from the replay buffer.
  - Compute the target Q-values using the target network.
  - Update the main network by minimizing the loss function using gradient descent.

- **Target Network Update**: Periodically copy the weights of the main network to the target network to ensure stability.

- **Decay Exploration Rate**: Gradually decrease ϵ over time to shift from exploration to exploitation.

Applications

- **Atari Games:** It can learn to play old video games very well even better than humans by looking at the screen pixels.
- **Robotics:** It helps robots to learn how to pick objects, move around and do tasks with their hands.

- **Self-Driving Cars:** It helps cars to make decisions like changing lanes and avoiding obstacles safely.
- **Finance:** It is used to find the best ways to trade stocks, manage money and reduce risks.
- **Healthcare:** It helps with planning treatments, discovering new medicines and personalizing care for patients.

As this technology improves Deep Q-Learning will help build even smarter systems to solve more complex real-life problems.