Q-learning and deep Q-learning

- **Definition**

Q-learning is a model-free reinforcement learning algorithm to learn the value of an action in a particular state. It does not require a model of the environment (hence "model-free"), and it can handle problems with stochastic transitions and rewards without requiring adaptations.

For any finite Markov decision process (FMDP), Q-learning finds an optimal policy in the sense of maximizing the expected value of the total reward over any and all successive steps, starting from the current state.

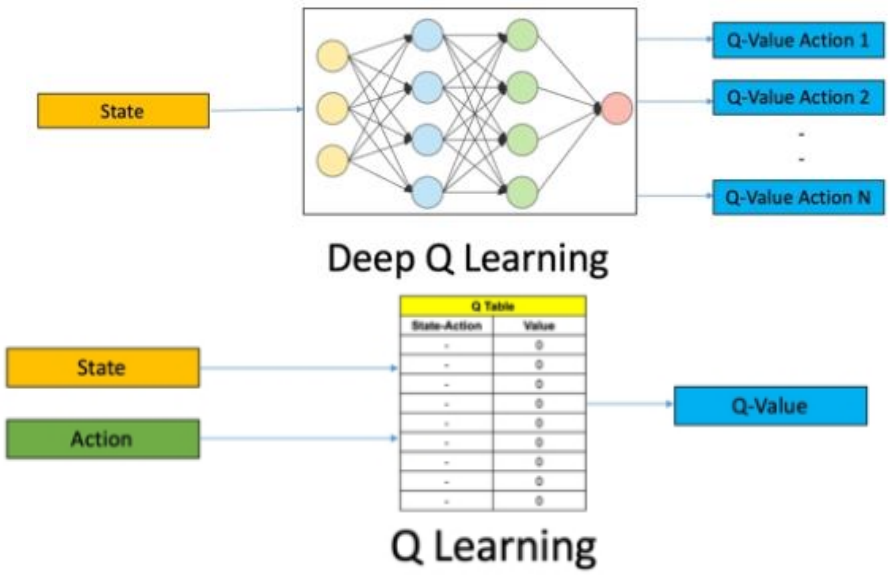
Deep Q-Learning uses experience replay to learn in small batches in order to avoid skewing the dataset distribution of different states, actions, rewards, and the next states that the neural network will see. Importantly, the agent doesn't need to train after each step.

- **The differences in implementation of Q-learning and deep Q-learning**

In implementing Q-Learning, we have a mapping from State and Action to Q, which in the simplest case, is stored in a table called Q-Table. Now if the number of actions and states is a lot, this method of estimating Q values (quality or value) will not be optimal. For example, if we want to act in this way for a self-driving car that can observe thousands of states of the environment and make dozens of decisions, we will have problems.

One of the solutions to this problem is the use of artificial neural networks (ANN). Deep Neural Network (DNN) can generalize its results if it trains properly. In this situation, instead of storing Q in a table, we can use the prediction model for Q.

The differences between these two strategies are shown in below figure:



Along with the many similarities between Q-Learning and Deep Q-Learning, there are very important differences in implementing these two methods, which are discussed below:

I. In the Q-Learning method, we used the following formula to train Q:

$$Q(s,a)^{\text{new}} \leftarrow Q(s,a)^{\text{old}} + \alpha \left[r + \gamma \max_{a'} Q(s',a')^{\text{old}} - Q(s,a)^{\text{old}} \right]$$

By updating these values, agent learning is completed, but in Deep Q-Learning, after updating the Q values, the deep model must also be trained on these data.
In the Q-Learning method, often small values for the learning rate α are suitable, but in Deep Q-Learning, often high values of α are desirable. In most cases, the value of the learning rate is considered equal to 1, in which case the relationship of updating the Q values is simplified as follows:

$$Q(s,a)^{new} \leftarrow r + \gamma \max_{a'} Q(s',a')^{\text{old}}$$

The following points should also be kept in mind when implementing Deep Q-Learning

II. In the Q-Learning method, all the Q values were stored and ready, but in Deep Q-Learning, we must enter the current conditions into the model so that it produces the Q values at the output.
III. Agent training in Deep Q-Learning takes more time than Q-Learning.
IV. In Deep Q-Network training, by updating the weights with respect to one data (including the initial condition, the action performed, the reward received and the subsequent condition), the Q values for other conditions also change, while this did not happen in Q-Table. This causes the model to constantly chase some Q value and never reach it, and we should notice this problem in designing the algorithm.
V. In Q-Learning, epsilon values (ϵ) used to decrease linearly during the episodes, but for Deep Q-Learning, there are often two phases; in the first phase, we see the epsilon decrease linearly or exponentially, but in the second phase, the epsilon values are fixed and equal to the final value.
VI. In Q-Learning, because there is no model for training, optimizer algorithms are not used, but for Deep Q-Network training, we use optimizer algorithms, so the inputs and outputs of the model must be in a certain interval (often in the interval [-1,+1] or [0,+1]) or their mean and variance are equal to 0 and 1, respectively. For this reason, State and Q values should be scaled if needed.