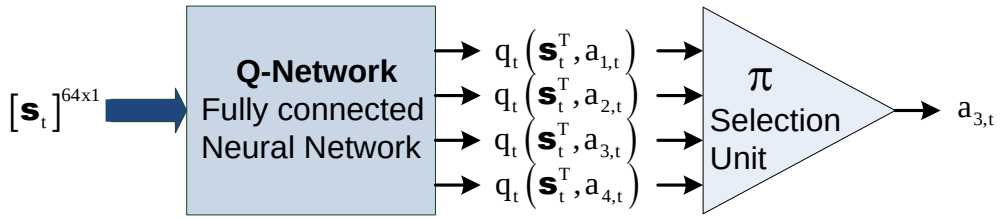
Chapter 1 gave an introduction of how the neural network is used to play the grid game. The outputs of the agent are the control commands to define the next step of the frog.

The agent itself is built of a neural network and a selection unit.

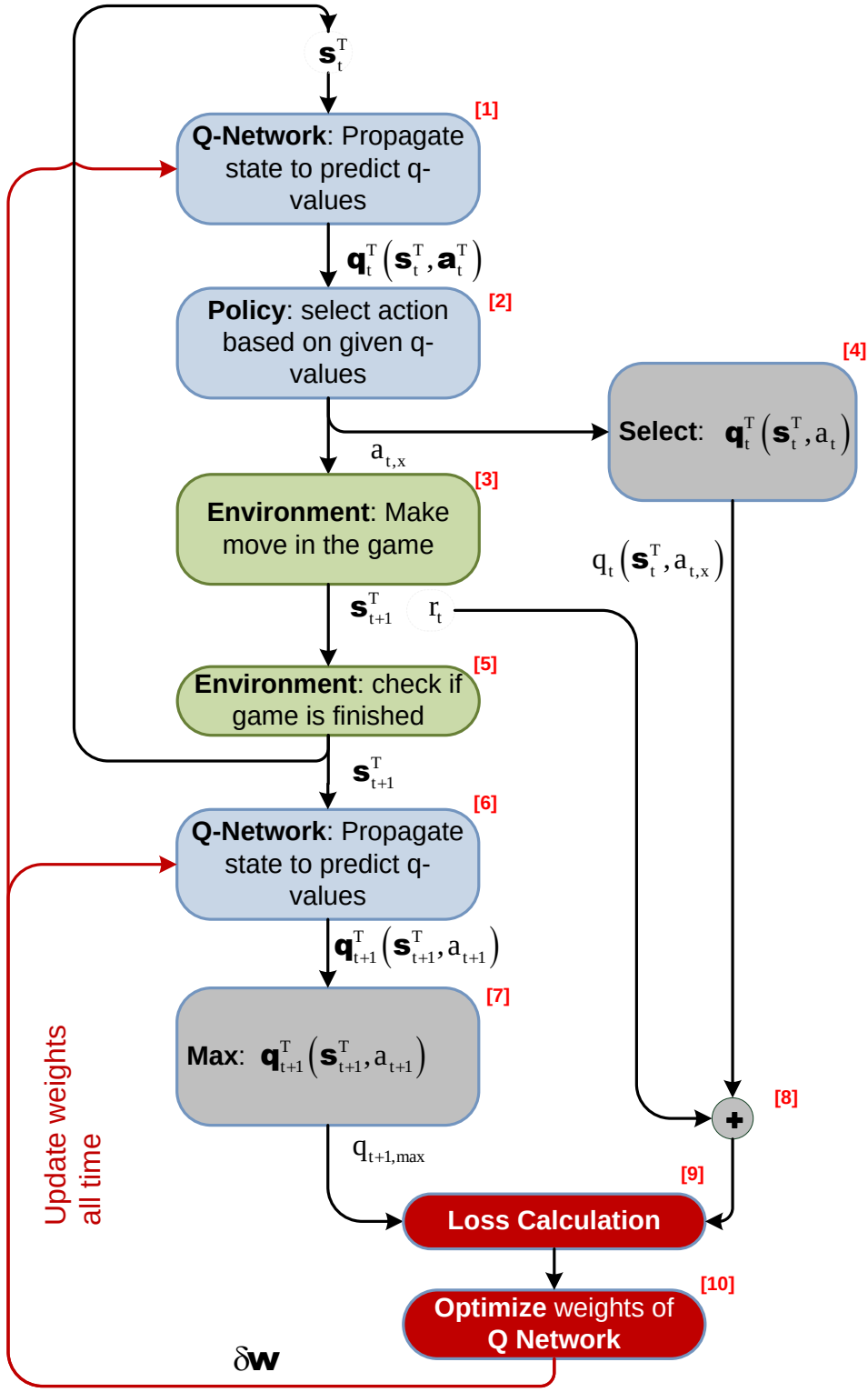


The actual game state at time t is given as $\mathbf{s}_t$ and outputs of the neural network are the predicted Q-values. In all states, 4 actions are possible. These actions are the possible moving directions of the frog on the grid game. For each of these directions, given that actual state, the neural network predicts the expected reward.

It is the task of the selection unit to select the action, based on the predicted Q-values. At first glance it would be the best, to choose the action belonging the highest reward. But this is deceptive. If you always would choose the action belonging to the highest reward, not all possible paths would be explored. Therefore, it is necessary sometimes, to ignore the neural network prediction and choose a random action. This whole process of action selection refers to the policy π. A common method to ensure exploring of yet undiscovered possibilities is calls the ε-greedy method. This method selects with a small probability ε after each time step a random action and with a probability of $1-\varepsilon$ the action according to the highest predicted Q-value.

The state $[\mathbf{s}_t]^{64x1}$ represents in this case study the current state of the game environment.

The next figure shows a concrete realization of an online learning Q-Learning architecture.


$\mathbf{s}_t^T$
[1]
Q-Network: Propagate state to predict q-values
$\mathbf{q}_t^T(\mathbf{s}_t^T, \mathbf{a}_t^T)$
[2]
Policy: select action based on given q-values
[4]
Select: $\mathbf{q}_t^T(\mathbf{s}_t^T, a_t)$
$a_{t,x}$
[3]
Environment: Make move in the game
$q_t(\mathbf{s}_t^T, a_{t,x})$
$\mathbf{s}_{t+1}^T$
r_t
[5]
Environment: check if game is finished
$\mathbf{s}_{t+1}^T$
[6]
Q-Network: Propagate state to predict q-values
$\mathbf{q}_{t+1}^T(\mathbf{s}_{t+1}^T, a_{t+1})$
[7]
Max: $\mathbf{q}_{t+1}^T(\mathbf{s}_{t+1}^T, a_{t+1})$
[8]
$q_{t+1,max}$
[9]
Loss Calculation
[10]
Optimize weights of **Q Network**
$\delta\mathbf{w}$
Update weights all time

[1] Starting with an initial state $\mathbf{s}_t$ of the game environment at time t, the Q-network predicts for all possible actions the expected q-values $\mathbf{q}_t(\mathbf{s}_t, \mathbf{a}_t)$.

$$\mathbf{q}_t(\mathbf{s}_t, \mathbf{a}_t) = \left[q_{1,t}(\mathbf{s}_t, a_1), q_{2,t}(\mathbf{s}_t, a_2), q_{3,t}(\mathbf{s}_t, a_3), q_{4,t}(\mathbf{s}_t, a_4) \right]$$

Each Q-value reflects the anticipated cumulative reward from executing that action and following the policy thereafter, until the end of the episode.

[2] Action selection is governed by a policy, which determines how the predicted Q-values are used to choose the next action. This is a critical component in designing a reinforcement learning system. In this context, an ε-greedy policy is employed: in most cases, the action with the highest predicted Q-value is chosen, while occasionally a random action is selected to encourage exploration.

[3] The selected action $a_{\text{selected},t}$ is then applied to the environment (or game engine), which transitions to a new state and provides a reward t_t associated with the action taken. This new state reflects the current configuration of the environment after the move.

[4] While the environment generally proceeds with the action corresponding to the highest Q-value, for training purposes, it is essential to isolate the predicted Q-value associated with the actual action taken. This specific value is required for the computation of the loss during training.

[5] After executing the action, the system checks whether the episode has terminated—either by achieving a win or encountering a loss. If the episode continues, the new state $\mathbf{s}_{t+1}$ becomes the input to the Q-network, which again predicts Q-values for all possible actions in that state. This loop repeats until the game ends.

[6] As will be detailed in Section [9], loss computation at time step t requires an additional Q-value prediction for the *next* state. This extra prediction is performed *outside* the standard interaction loop, as it is not used for action selection but solely for learning purposes.

[7] Since this prediction serves only the loss calculation, no action is selected from it. Instead, only the maximum Q-value among the predictions is extracted and used.

[8],[9] Loss computation is a central aspect of training the Q-network. In this case study, the loss is computed as the difference between the predicted Q-value for the action taken at time t, and a target value incorporating the received reward and the maximum predicted Q-value from the next state. This target is more accurate than the network's initial prediction because it includes a real, observed reward.

The following equation illustrates why both the predicted and target terms correspond to Q-values at time step t, yet differ in their informational content:

$$\text{loss}_t = \left(q_t\left(\mathbf{s}_t, a_{\text{selected},t}\right) - \left(r_t + q_{t+1}\left(\mathbf{s}_{t+1}, a_{\text{selected},t+1}\right)\right)\right)^2$$

The variable $a_{\text{selected},t}$ represents the action, used at time t to make the next step in the game.

Having a look on the following equation will make it clearer, why both terms represent the Q-value at time t.

$$q_t = r_t + r_{t+1} + r_{t+2} + \ldots + r_{t+N} = r_t + q_{t+1} = r_t + r_{t+1} + q_{t+2}$$

[10] To optimize the network, the gradient of the loss function with respect to the neural network's weights is computed. Crucially, during this backpropagation step, the gradient is only derived from the part of the computation graph associated with the predicted Q-value — typically marked in blue in visualizations — ensuring stable and effective training.

$$\nabla_{\mathbf{w}}\left(q_t\left(\mathbf{s}_t, a_{\text{selected},t}\right) - \left(r_t + q_{t+1}\left(\mathbf{s}_{t+1}, a_{\text{selected},t+1}\right)\right)\right)^2$$

$$\nabla_{\mathbf{w}}\left(\text{loss}_t\right) = 2 \cdot \left(q_t\left(\mathbf{s}_t, a_{\text{selected},t}\right) - \left(r_t + q_{t+1}\left(\mathbf{s}_{t+1}, a_{\text{selected},t+1}\right)\right)\right) \cdot \nabla_{\mathbf{w}} q_t\left(\mathbf{s}_t, a_{\text{selected},t}\right)$$