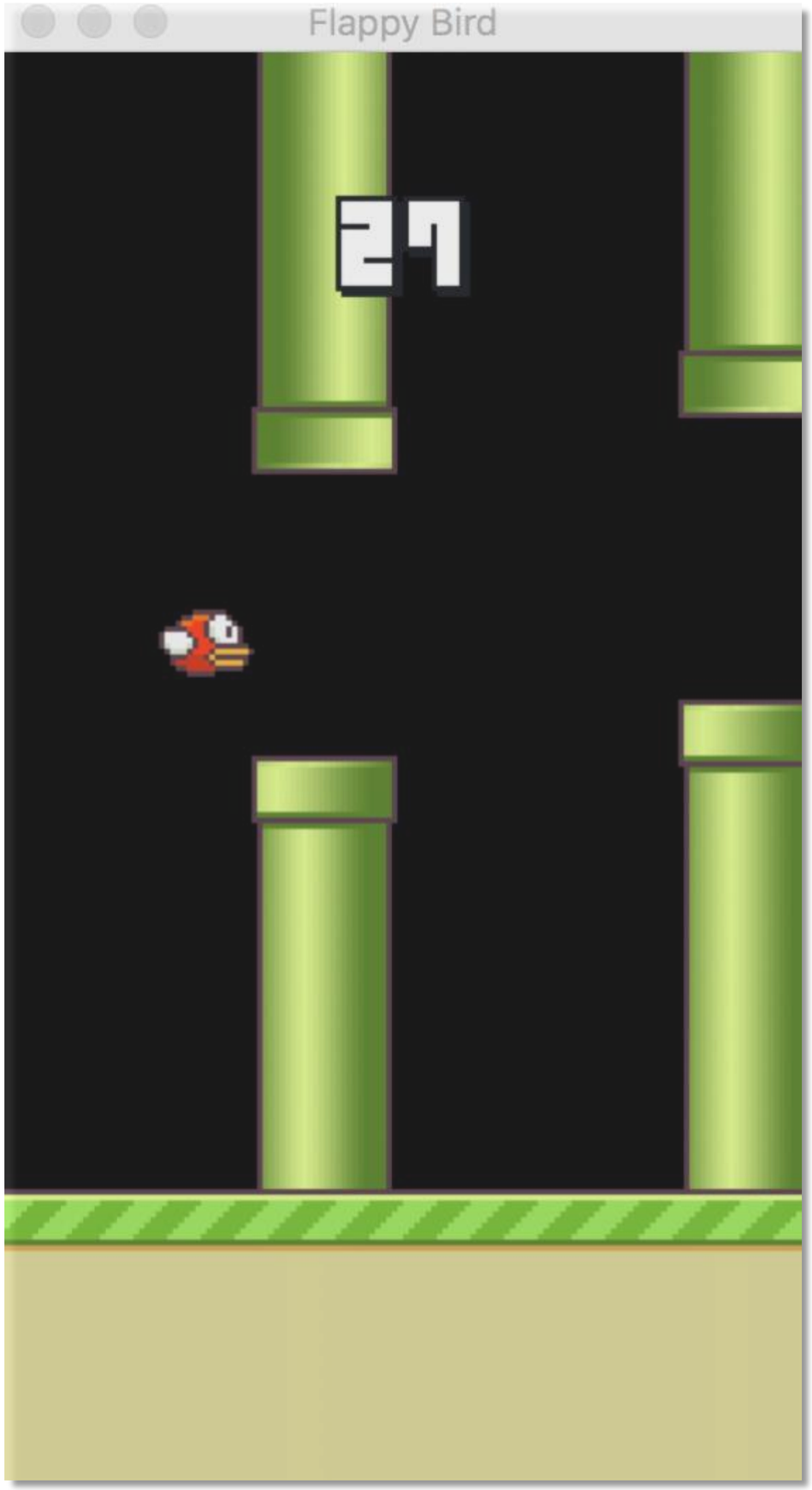
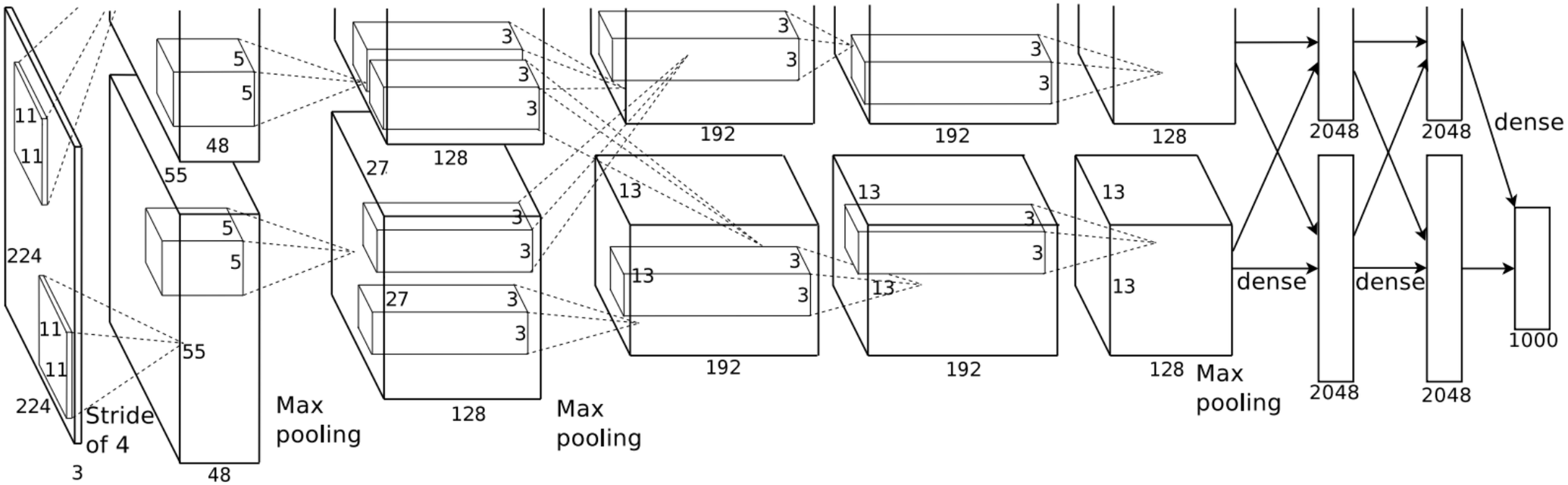
Welcome to Deep Reinforcement Learning

Recent years, many AI laboratories are working on studying deep reinforcement learning (DRL) which is expected to be a core technology in the future.

In this display, I introduce Deep Q-Network (DQN) that is the first deep reinforcement learning method proposed by DeepMind. After the paper was published on Nature in 2015, a lot of research institutes joined this field because deep neural network can empower RL to directly deal with high dimensional states like images, thanks to techniques used in DQN.

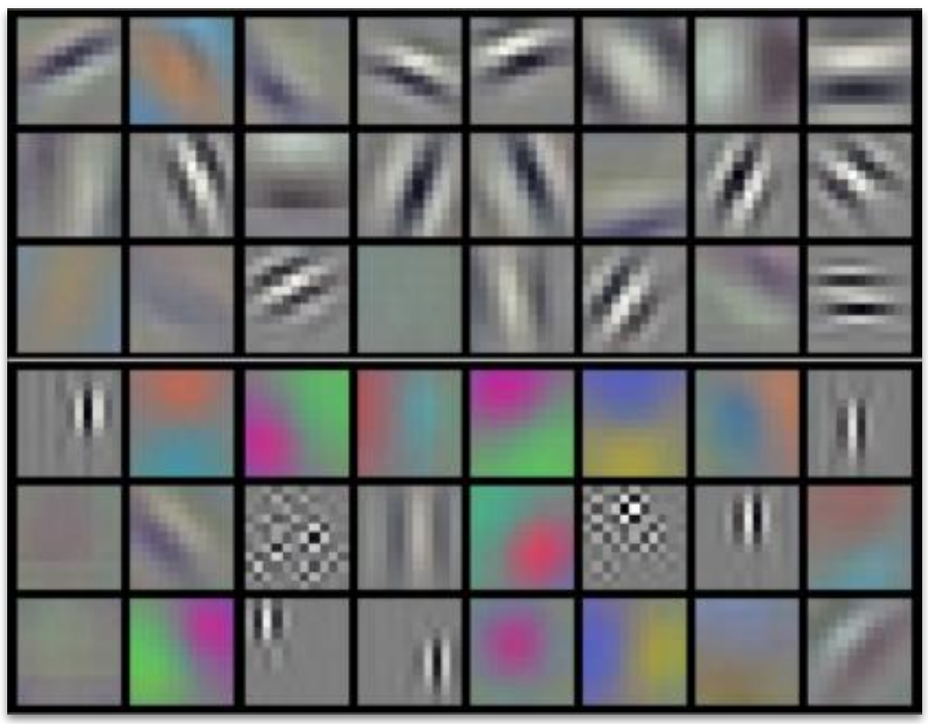


Deep Neural Network (DNN)



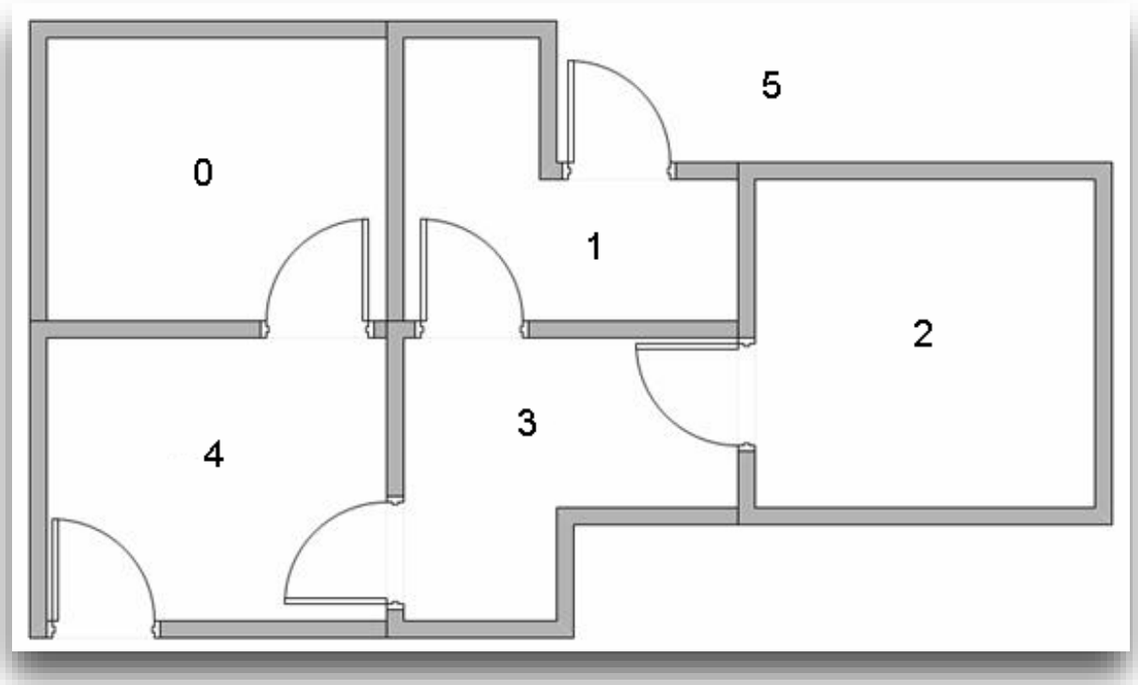
[AlexNet] has achieved an incredible score in ILSVRC 2012, image classification competition by using DNN.

The greatest thing of DNN is extracting feature representations through backpropagation.

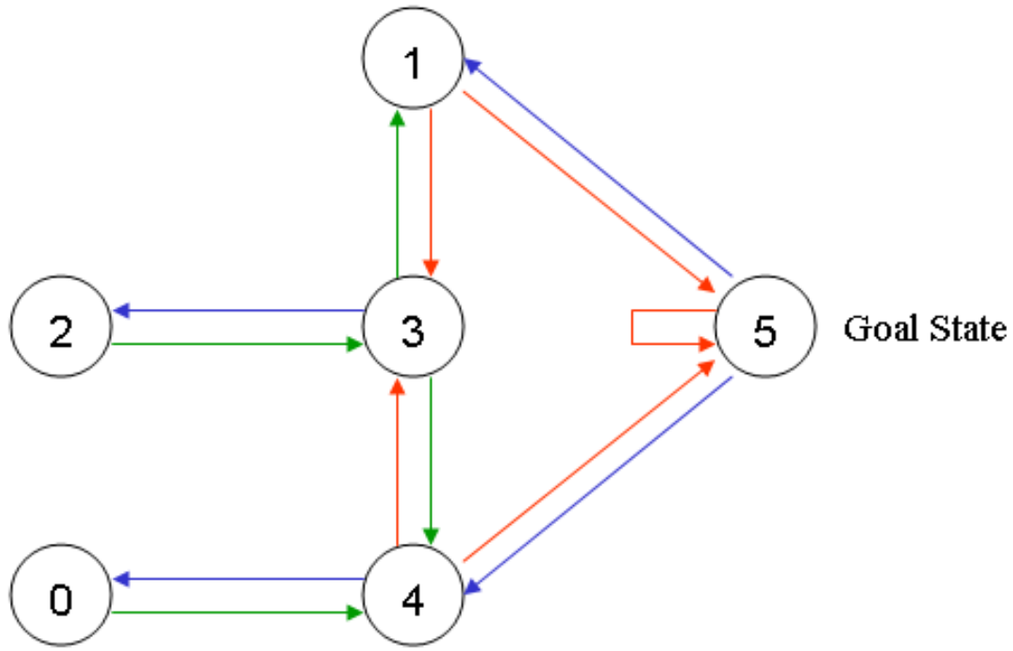


Q-Learning

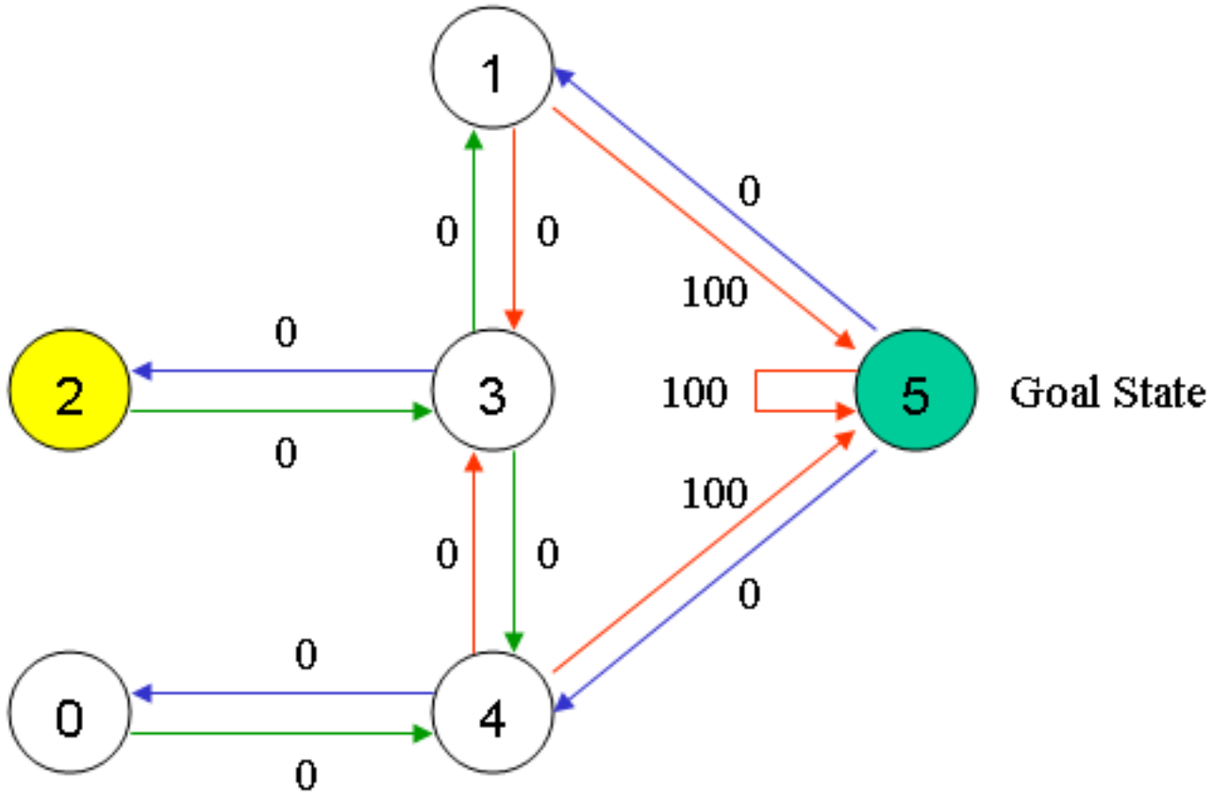
Suppose we have 5 rooms in a building connected by doors as shown in the figure below. We'll number each room 0 through 4. The outside of the building can be thought of as one big room (5). Notice that doors 1 and 4 lead into the building from room 5 (outside).



We can represent the rooms on a graph, each room as a node, and each door as a link.



Q-Learning



Action

$$R = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} -1 & -1 & -1 & -1 & 0 & -1 \\ -1 & -1 & -1 & 0 & -1 & 100 \\ -1 & -1 & -1 & 0 & -1 & -1 \\ -1 & 0 & 0 & -1 & 0 & -1 \\ 0 & -1 & -1 & 0 & -1 & 100 \\ -1 & 0 & -1 & -1 & 0 & 100 \end{bmatrix} \end{array}$$

Q(state, action) = R(state, action) + Gamma * Max[Q(next state, all actions)]

Q-Learning Example By Hand

Now let's imagine what would happen if our agent were in state 5. Look at the sixth row of the reward matrix R (i.e. state 5). It has 3 possible actions: go to state 1, 4 or 5.

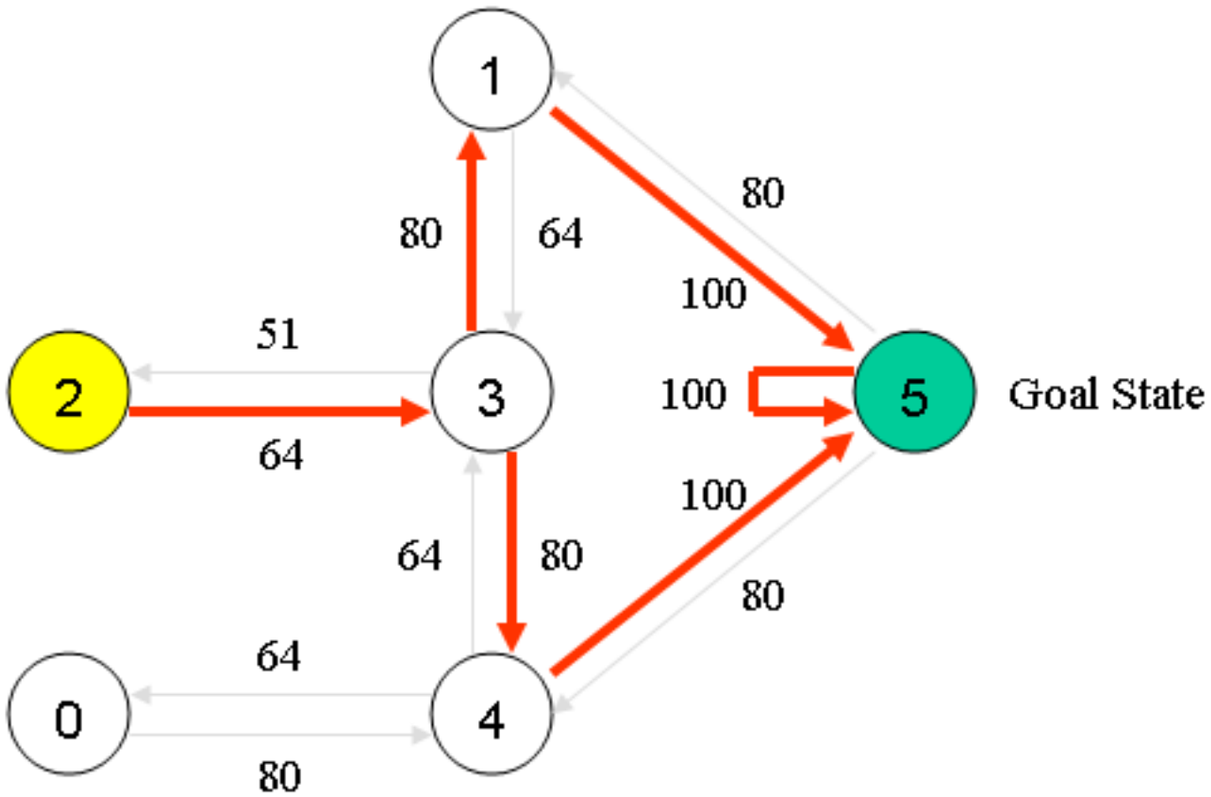
Q(state, action) = R(state, action) + Gamma * Max[Q(next state, all actions)]
Q(1, 5) = R(1, 5) + 0.8 * Max[Q(5, 1), Q(5, 4), Q(5, 5)] = 100 + 0.8 * 0 = 100

$$Q = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 100 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 80 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix} \end{array} \qquad Q = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 400 & 0 \\ 0 & 0 & 0 & 320 & 0 & 500 \\ 0 & 0 & 0 & 320 & 0 & 0 \\ 0 & 400 & 256 & 0 & 400 & 0 \\ 320 & 0 & 0 & 320 & 0 & 500 \\ 0 & 400 & 0 & 0 & 400 & 500 \end{bmatrix} \end{array}$$

$$Q = \begin{array}{c} \\ 0 \\ 1 \\ 2 \\ 3 \\ 4 \\ 5 \end{array} \begin{array}{c} \begin{array}{cccccc} 0 & 1 & 2 & 3 & 4 & 5 \end{array} \\ \begin{bmatrix} 0 & 0 & 0 & 0 & 80 & 0 \\ 0 & 0 & 0 & 64 & 0 & 100 \\ 0 & 0 & 0 & 64 & 0 & 0 \\ 0 & 80 & 51 & 0 & 80 & 0 \\ 64 & 0 & 0 & 64 & 0 & 100 \\ 0 & 80 & 0 & 0 & 80 & 100 \end{bmatrix} \end{array}$$

Q-Learning Example By Hand

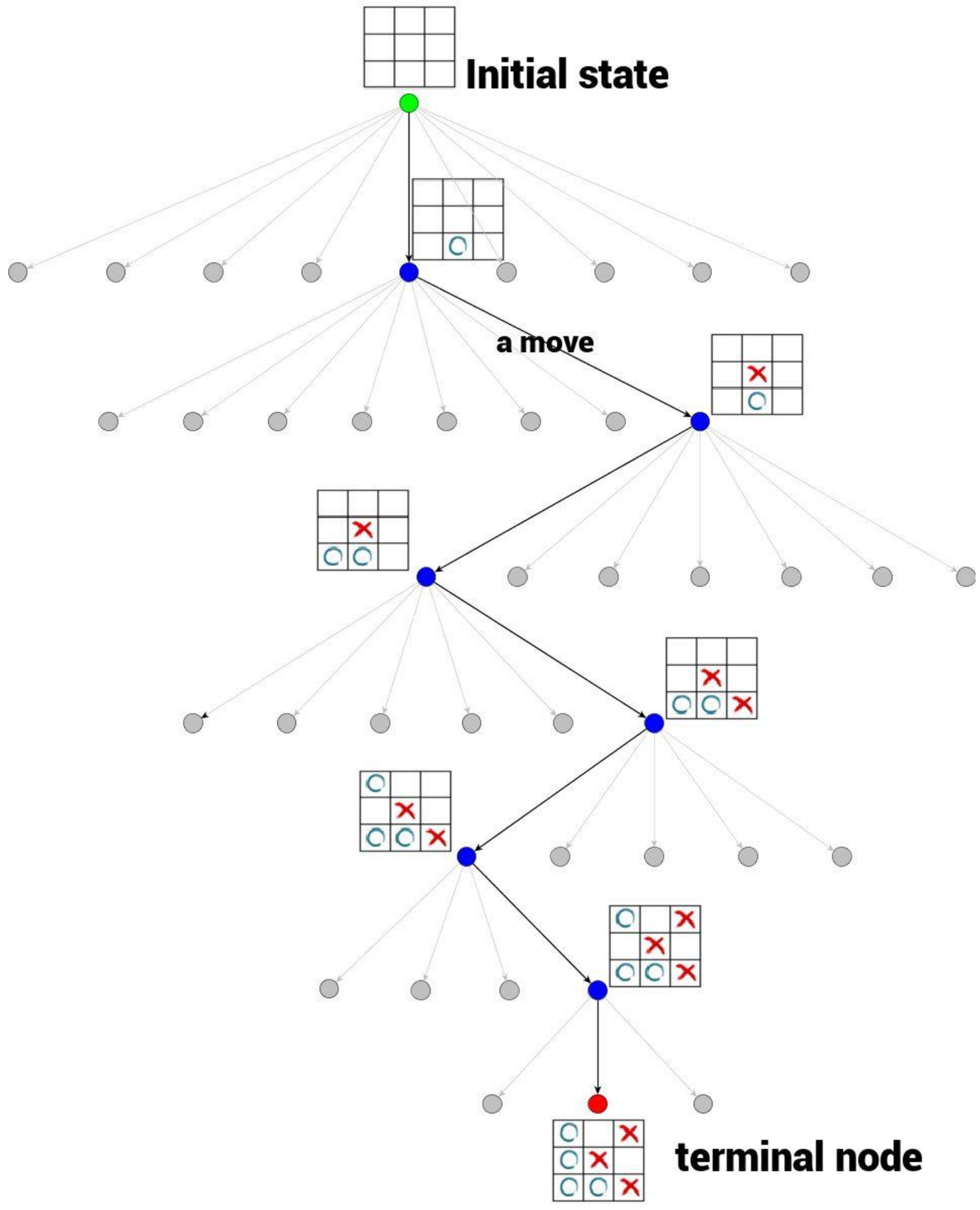
Once the matrix Q gets close enough to a state of convergence, we know our agent has learned the most optimal paths to the goal state. Tracing the best sequences of states is as simple as following the links with the highest values at each state.



For example, from initial State 2, the agent can use the matrix Q as a guide:

- From State 2 the maximum Q values suggests the action to go to state 3.
- From State 3 the maximum Q values suggest two alternatives: go to state 1 or 4. Suppose we arbitrarily choose to go to 1.
- From State 1 the maximum Q values suggests the action to go to state 5.

Thus the sequence is 2 - 3 - 1 - 5.


Initial state
a move
terminal node

Deep Q-Network

DQN is introduced in 2 papers, Playing Atari with Deep Reinforcement Learning on NIPS in 2013 and Human-level control through deep reinforcement learning on Nature in 2015.

DQN overcomes unstable learning by mainly 3 techniques.

Experience Replay

DNN is easily overfitting current episodes. Once DNN is overfitted, it’s hard to produce various experiences.

To solve this problem, Experience Replay stores experiences including state transitions, rewards and actions, which are necessary data to perform Q learning, and makes mini-batches to update neural networks. This technique expects the following merits.

Target Network

In TD error calculation, target function is changed frequently with DNN. Unstable target function makes training difficult. So Target Network technique fixes parameters of target function and replaces them with the latest network every thousands steps.

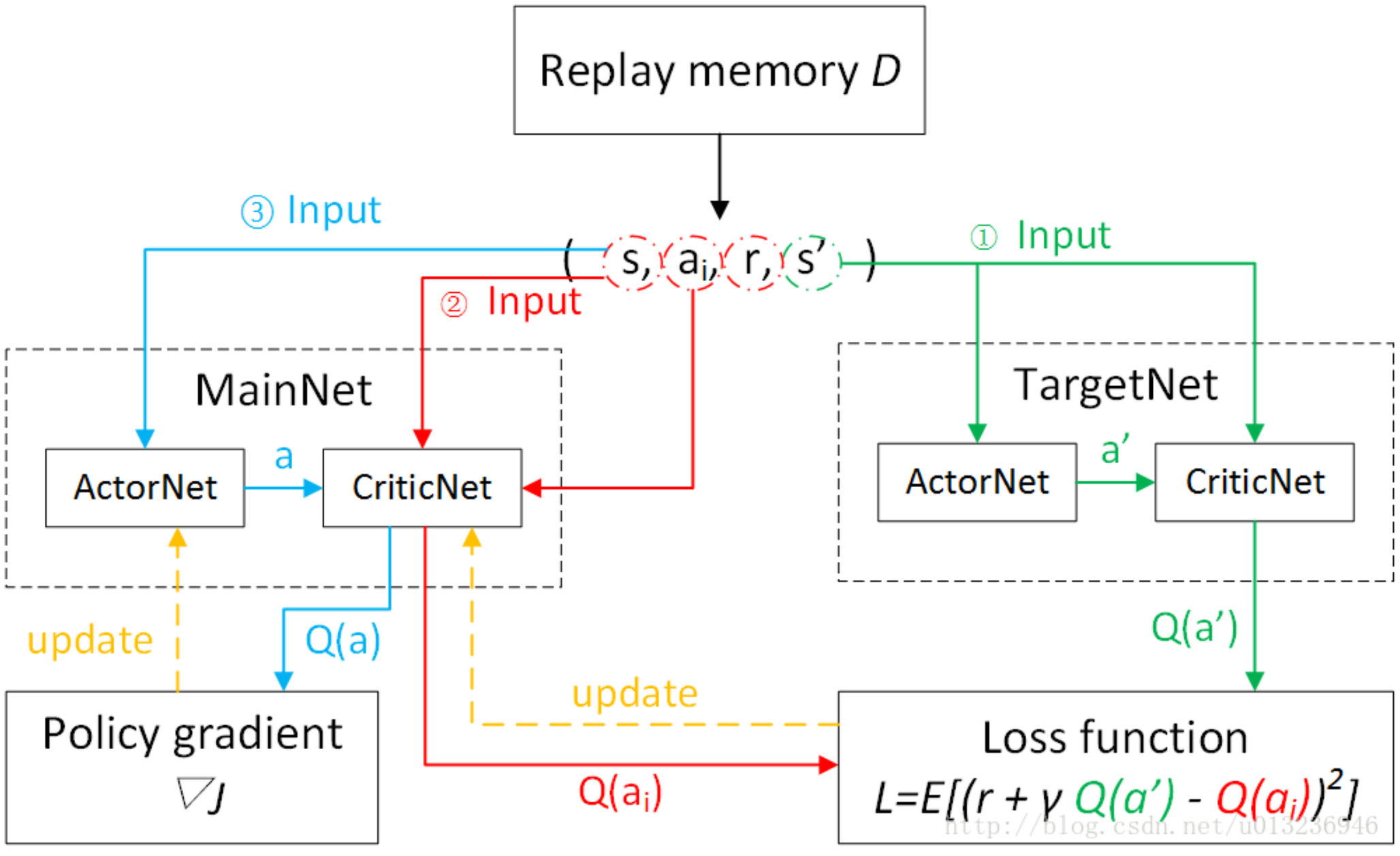
$$Q(s_t, a) \leftarrow Q(s_t, a) + \alpha \left[r_{t+1} + \gamma \max_p \boxed{Q(s_{t+1}, p)} - Q(s_t, a) \right]$$

target Q function in the red rectangular is fixed

Clipping Rewards

Each game has different score scales. For example, in Pong, players can get 1 point when wining the play. Otherwise, players get -1 point. However, in SpaceInvaders, players get 10~30 points when defeating invaders. This difference would make training unstable. Thus Clipping Rewards technique clips scores, which all positive rewards are set +1 and all negative rewards are set -1.

Deep Deterministic Policy Gradient (DDPG)



Normalized Advantage Functions (NAF)