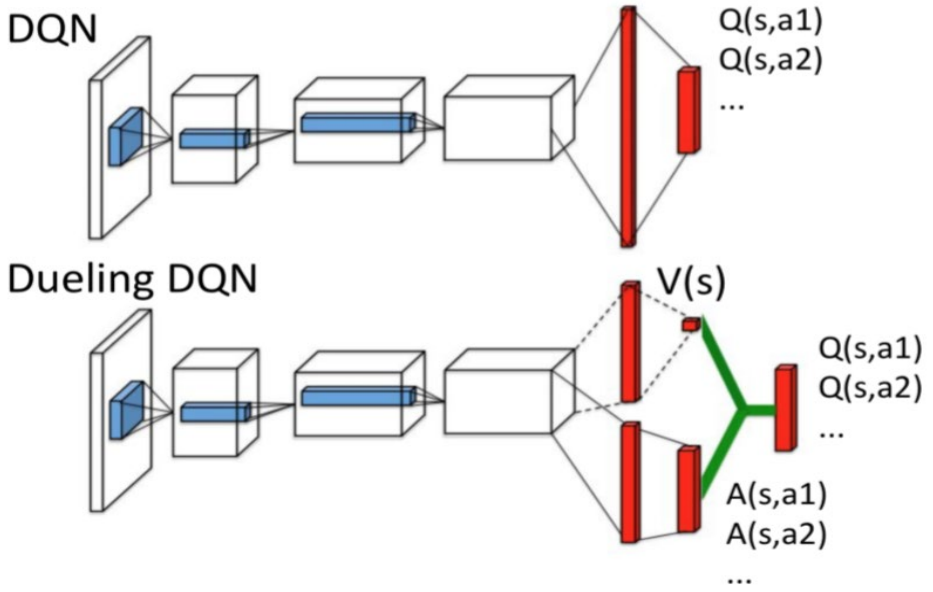
YAI 6th
강화 스터디

Turtlebot3 RL
중간 점검

2020-11-17

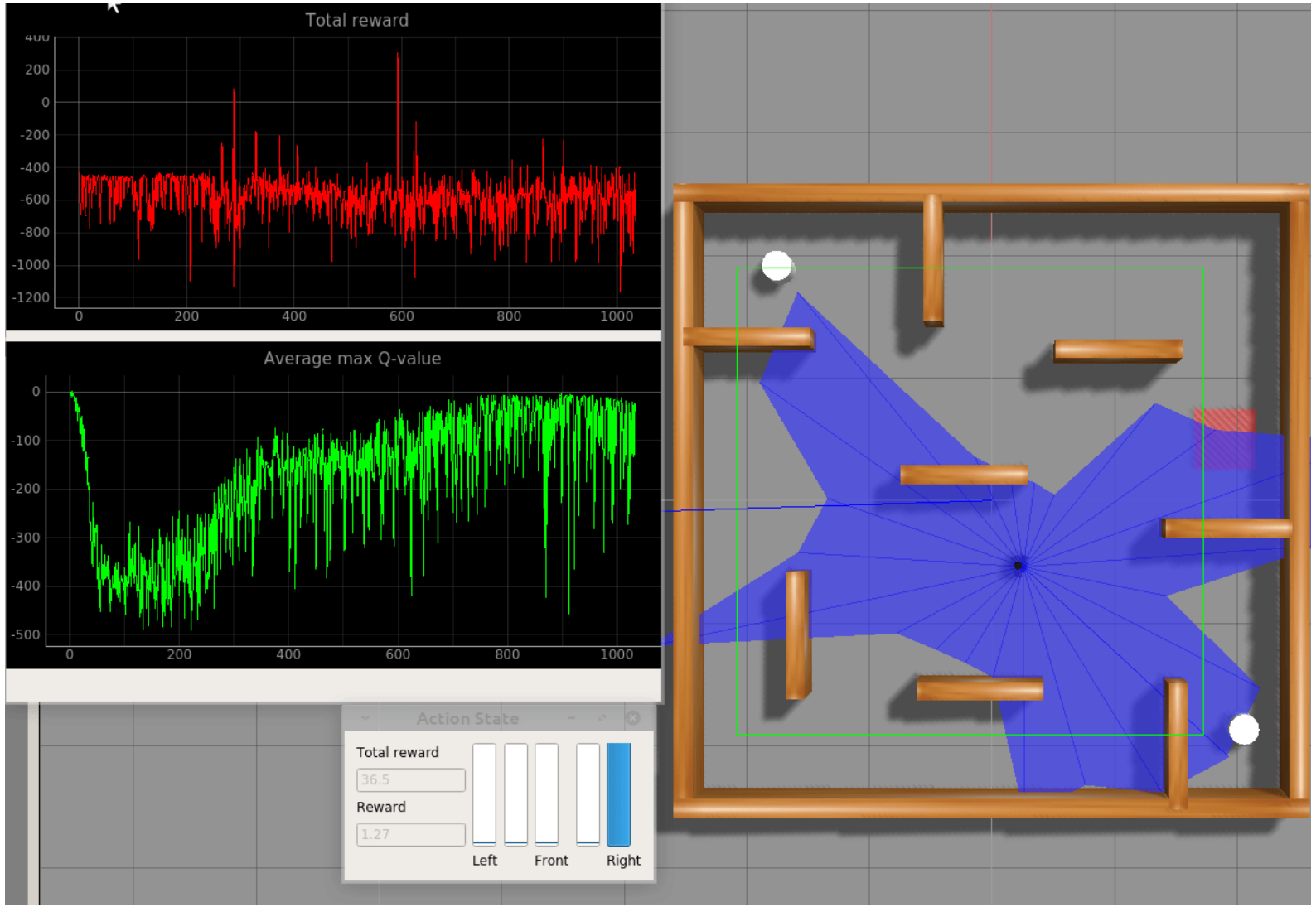
바로 시작하겠습니다~

01 PER-D3QN

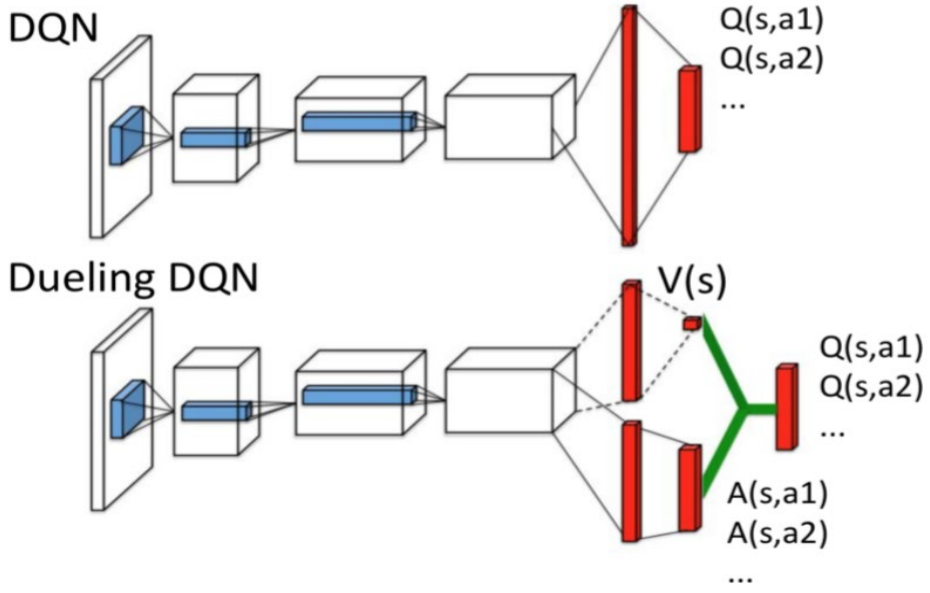


$$Q(s,a;\theta,\alpha,\beta) = V(s;\theta,\beta) + \left(A(s,a;\theta,\alpha) - \frac{1}{|\mathcal{A}|}\sum_{a'} A(s,a';\theta,\alpha) \right). \quad (9)$$

01 PER-D3QN


Total reward
400
200
0
-200
-400
-600
-800
-1000
-1200
0
200
400
600
800
1000
Average max Q-value
0
-100
-200
-300
-400
-500
0
200
400
600
800
1000
Action State
Total reward
36.5
Reward
1.27
Left
Front
Right

01 PER-D3QN



+

PER

$$Q(s,a;\theta,\alpha,\beta) = V(s;\theta,\beta) + \left(A(s,a;\theta,\alpha) - \frac{1}{|\mathcal{A}|}\sum_{a'} A(s,a';\theta,\alpha) \right). \quad (9)$$

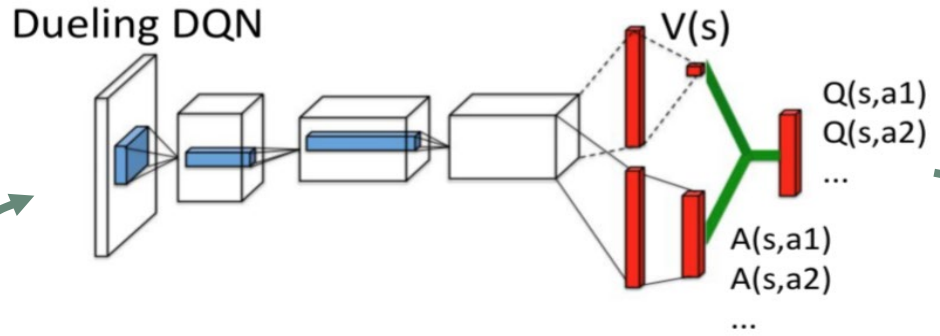
01 PER-D3QN

$$p_i = \left| r + \gamma \max_{a'} Q(s_{i+1}, a'; \boldsymbol{w}^-) - Q(s_i, a_i; \boldsymbol{w}) \right|$$

$$P(i) = \frac{p_i^\alpha}{\sum_k p_k^\alpha}$$

$$w_i = \left(\frac{1}{N} \cdot \frac{1}{P(i)} \right)$$

memory



1. Action

01 PER-D3QN

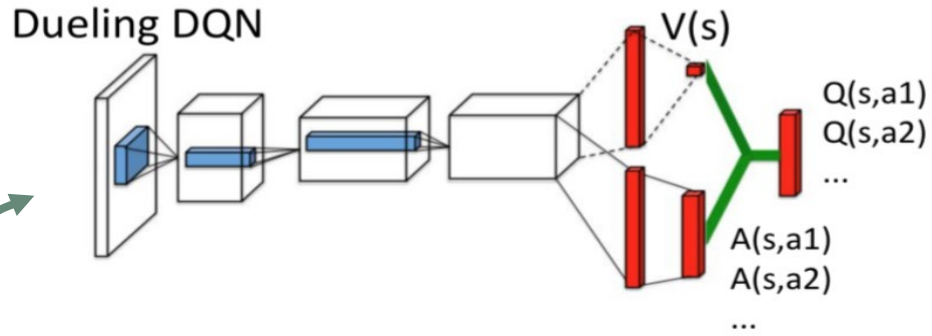
$$p_i = \left| r + \gamma \max_{a'} Q(s_{i+1}, a'; \boldsymbol{w}^-) - Q(s_i, a_i; \boldsymbol{w}) \right|$$

$$P(i) = \frac{p_i^\alpha}{\sum_k p_k^\alpha}$$

$$w_i = \left(\frac{1}{N} \cdot \frac{1}{P(i)} \right)$$

memory

2. Save memory



01 PER-D3QN

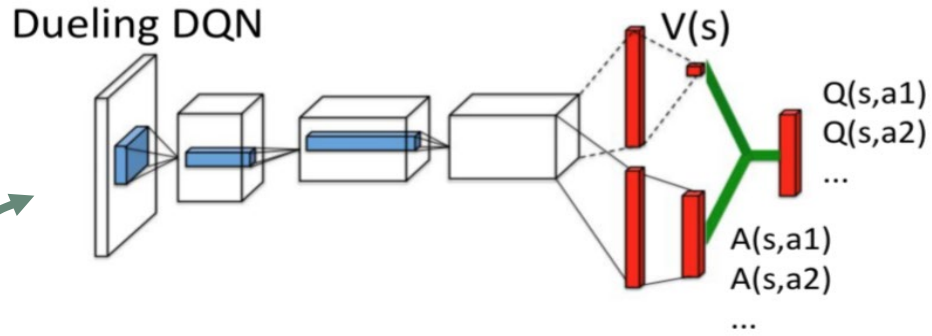
3. Pull out from memory with prioritization

$$p_i = \left| r + \gamma \max_{a'} Q(s_{i+1}, a'; \boldsymbol{w}^-) - Q(s_i, a_i; \boldsymbol{w}) \right|$$

$$P(i) = \frac{p_i^\alpha}{\sum_k p_k^\alpha}$$

$$w_i = \left(\frac{1}{N} \cdot \frac{1}{P(i)} \right)$$

memory



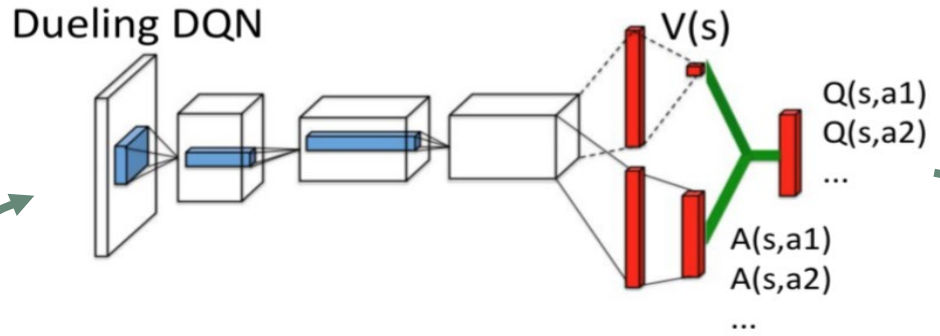
01 PER-D3QN

$$p_i = \left| r + \gamma \max_{a'} Q(s_{i+1}, a'; \boldsymbol{w}^-) - Q(s_i, a_i; \boldsymbol{w}) \right|$$

$$P(i) = \frac{p_i^\alpha}{\sum_k p_k^\alpha}$$

$$w_i = \left(\frac{1}{N} \cdot \frac{1}{P(i)} \right)$$ 4. Importance sampling

memory



01 PER-D3QN

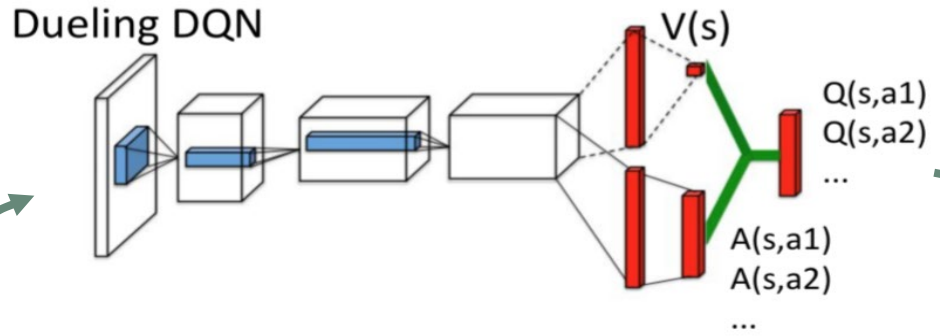
$$p_i = \left| r + \gamma \max_{a'} Q(s_{i+1}, a'; \boldsymbol{w}^-) - Q(s_i, a_i; \boldsymbol{w}) \right|$$

$$P(i) = \frac{p_i^\alpha}{\sum_k p_k^\alpha}$$

$$w_i = \left(\frac{1}{N} \cdot \frac{1}{P(i)} \right)$$

memory

5. Learn



01 PER-D3QN

$$p_i = \left| r + \gamma \max_{a'} Q(s_{i+1}, a'; \boldsymbol{w}^-) - Q(s_i, a_i; \boldsymbol{w}) \right|$$

$$P(i) = \frac{p_i^\alpha}{\sum_k p_k^\alpha}$$

$$w_i = \left(\frac{1}{N} \cdot \frac{1}{P(i)} \right)$$

=

```
def appendMemory(self, state, action, reward, next_state, done):
    #(edit5)
    td_error = reward + self.discount_factor * np.max(self.target_model.predict(next_state.reshape(1, len(next_state)))) - self.model.predict(state.reshape(1, len(state)))

    p = np.abs(td_error + 0.0000001) ** self.alpha
    #(edit5)

    self.memory.append((state, action, reward, next_state, done))

    #(edit5)
    self.priority.append(p)
    #(edit5)
```

```
def get_PER_batch(self):
    p_sum = np.sum(self.priority)
    prob = self.priority / p_sum

    sample_indices = choices(range(len(prob)), k=self.batch_size, weights=prob)

    importance = (1/prob) * (1/len(self.priority))
    importance = np.array(importance)[sample_indices]
    samples = np.array(self.memory)[sample_indices]

    return samples, importance
```

```
i = importance[i] ** self.beta

self.model.fit(X_batch, Y_batch, batch_size=self.batch_size, epochs=1, verbose=0, sample_weight=importance_batch)
```

01 PER-D3QN

```
def appendMemory(self, state, action, reward, next_state, done):
    #(edit5)
    td_error = reward + self.discount_factor * np.max(self.target_model.predict(next_st

    p = np.abs(td_error + 0.0000001) ** self.alpha
    #(edit5)

    self.memory.append((state, action, reward, next_state, done))

    #(edit5)
    self.priority.append(p)
    #(edit5)
```

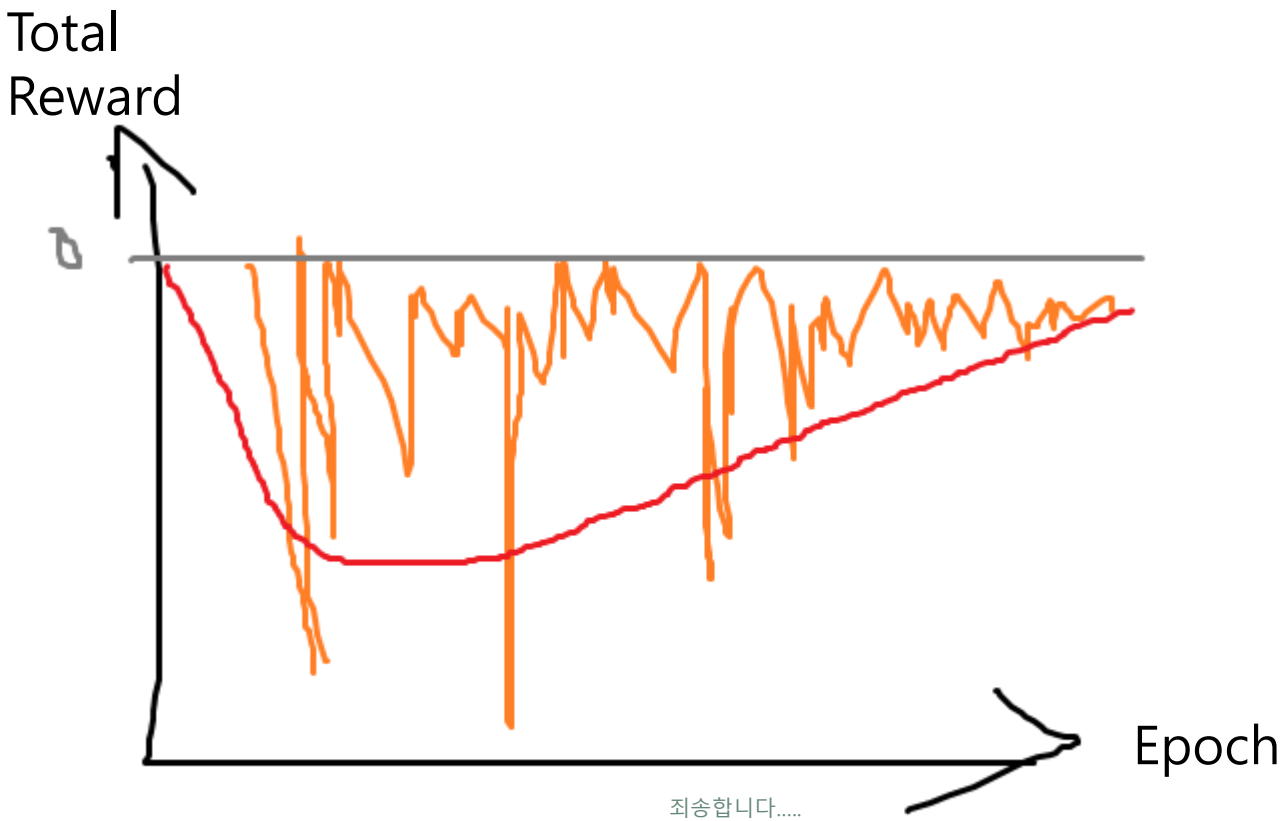
01 PER-D3QN

```
def get_PER_batch(self):
    p_sum = np.sum(self.priority)
    prob = self.priority / p_sum

    sample_indices = choices(range(len(prob)), k=self.batch_size, weights=prob)

    importance = (1/prob) * (1/len(self.priority))
    importance = np.array(importance)[sample_indices]
    samples = np.array(self.memory)[sample_indices]

    return samples, importance
```

01 PER-D3QN

```
i = importance[i] ** self.beta

self.model.fit(X_batch, Y_batch, batch_size=self.batch_size, epochs=1, verbose=0, sample_weight=importance_batch)
```

01 PER-D3QN_result



02 Agent & Environment

1. Rewards = {'right_direction' = +1,
'wrong_direction' = -4
'far_away_from_goal' = -2**distance_ratio
'collision' = -150
'goal' = +200
'near_obstacle' = -5
}

2. Actions = rotation right? rotate left?

02 Agent & Environment

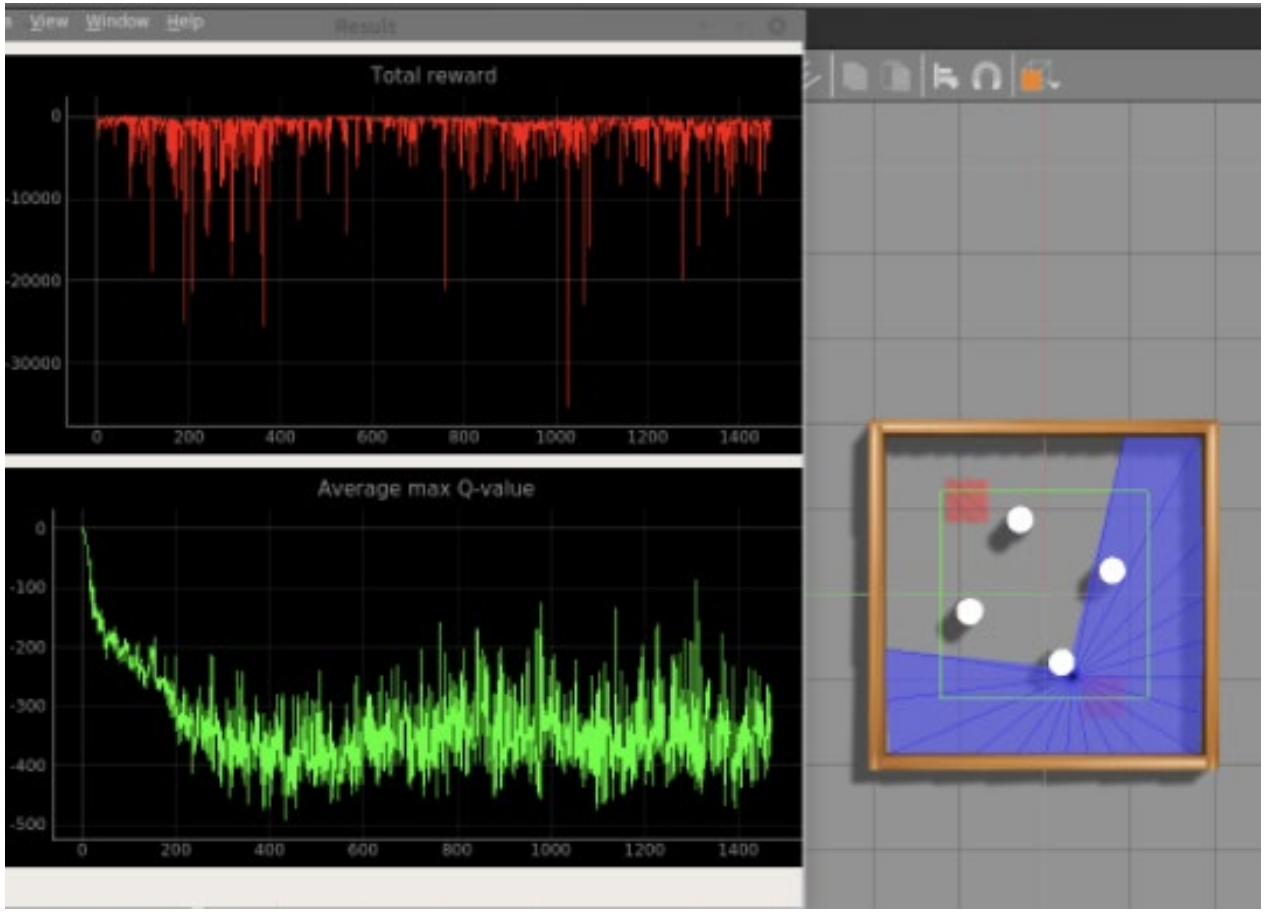
1. Rewards = {'right_direction' = +1,
'wrong_direction' = -4
'far_away_from_goal' = -2**distance_ratio
'collision' = -150
'goal' = +200
'near_obstacle' = -5
'time_since_goal' -= 1
}

2. Actions = rotation right? rotate left? move faster!

3. Easier environment

4. Back to DDQN

02 Agent & Environment



03 Building

jwill1994/ros2_ml ☆

By jwill1994 • Updated 6 days ago

Container

laphisboy/xx_ros_tb3 ☆

By laphisboy • Updated 13 days ago

ubuntu 16:04 with ROS kinetic and Turtlebot3 with referencing ROBOTIS e-Manual

Container

06 Q’s?

Fin

감사합니다~