Advance Q-Learning

CMPT 729 G100

Jason Peng

Overview

- Nonstationary Targets

- Overestimation

- Model Architecture

Overview

- Nonstationary Targets → Target Networks
- Overestimation
- Model Architecture

Overview

- Nonstationary Targets → Target Networks
- Overestimation → Pessimistic Estimates
- Model Architecture

Overview

- Nonstationary Targets → Target Networks
- Overestimation → Pessimistic Estimates
- Model Architecture

Moving Target

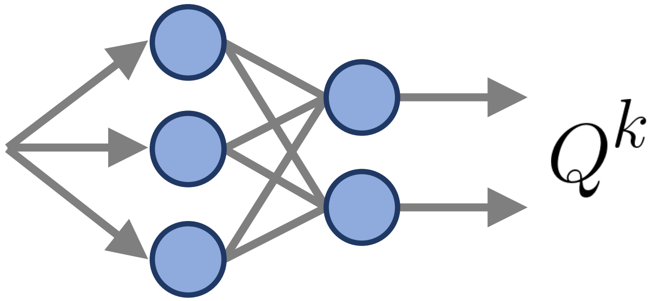
$$Q^{k+1} = \arg\min_{Q} \mathbb{E}_{(\mathbf{s}_i, \mathbf{a}_i, r_i, \mathbf{s}_i') \sim \mathcal{D}} \left[\left(\underline{y_i} - Q(\mathbf{s}_i, \mathbf{a}_i) \right)^2 \right]$$

$$y_i = r_i + \gamma \max_{\mathbf{a}'} \underline{Q^k}(\mathbf{s}', \mathbf{a}')$$

- Target values change every iteration
- Can lead to unstable learning dynamics

Target Network

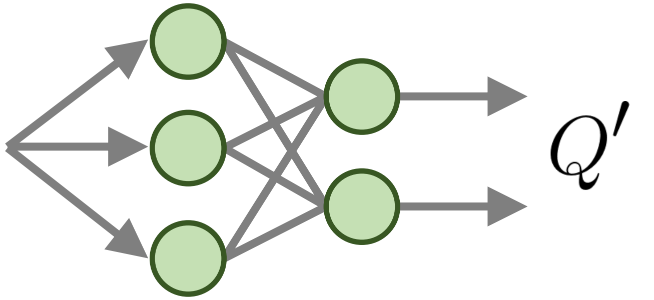
$$y_i = r_i + \gamma \max_{\mathbf{a}'} Q^k(\mathbf{s}', \mathbf{a}')$$



$$\Downarrow$$

$$y_i = r_i + \gamma \max_{\mathbf{a}'} \underline{Q'}(\mathbf{s}', \mathbf{a}')$$

target network



Human-Level Control Through Deep Reinforcement Learning
[Mnih et al. 2015]

Target Network

$$Q^{k+1} = \arg\min_{Q} \mathbb{E}_{(\mathbf{s}_i,\mathbf{a}_i,r_i,\mathbf{s}'_i)\sim\mathcal{D}} \left[(\underline{y_i} - Q(\mathbf{s}_i, \mathbf{a}_i))^2 \right]$$

$$y_i = r_i + \gamma \max_{\mathbf{a}'} \underline{Q'}(\mathbf{s}', \mathbf{a}')$$

- Target network is a delayed copy of the Q-function
- Every m iterations, copy parameters from Q-function to target network
- Works well in practice to stabilize Q-learning

Human-Level Control Through Deep Reinforcement Learning
[Mnih et al. 2015]

Deep Q-Network (DQN)

Experience Replay + Target Network

Human-Level Control Through Deep Reinforcement Learning
[Mnih et al. 2015]

Target Network

$$Q^{k+1} = \arg\min_{Q} \mathbb{E}_{(\mathbf{s}_i,\mathbf{a}_i,r_i,\mathbf{s}'_i)\sim\mathcal{D}}\left[\left(\underline{y_i} - Q(\mathbf{s}_i,\mathbf{a}_i)\right)^2\right]$$

$$y_i = r_i + \gamma \max_{\mathbf{a}'} \underline{Q'}(\mathbf{s}',\mathbf{a}')$$

- Every m iterations, copy parameters from Q-function to target network

✓ Works well in practice to stabilize Q-learning

✗ Abrupt changes to target values every m iterations

✗ Can cause some unstable learning dynamics

Human-Level Control Through Deep Reinforcement Learning
[Mnih et al. 2015]

Polyak Averaging

- Initialize target network with the *same* parameters a Q-function
- Every iteration, update target network:

$$\theta^{Q'} \leftarrow \underline{\alpha}\theta^{Q^k} + (1 - \underline{\alpha})\theta^{Q'}$$

step size

(e.g. $\alpha = 0.001$)

Continuous control with deep reinforcement learning
[Lillicrap et al. 2016]

Polyak Averaging

- Initialize target network with the *same* parameters a Q-function
- Every iteration, update target network:

$$\theta^{Q'} \leftarrow \underline{\alpha}\theta^{Q^k} + (1 - \underline{\alpha})\theta^{Q'}$$

step size

(e.g. $\alpha = 0.001$)

- Smoother changes to target values

Continuous control with deep reinforcement learning
[Lillicrap et al. 2016]

Target Network

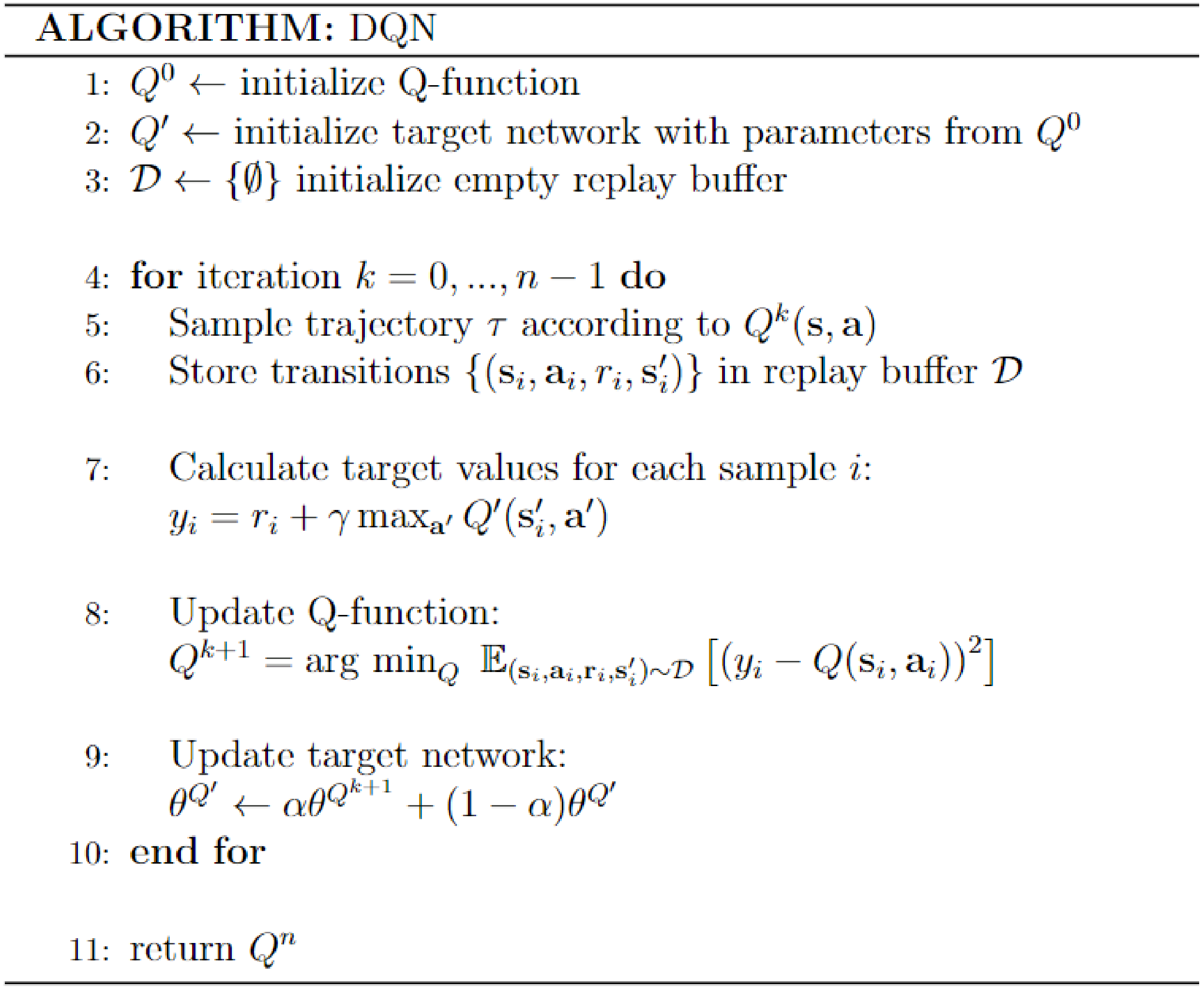
$$Q^{k+1} = \arg\min_{Q} \mathbb{E}_{(\mathbf{s}_i,\mathbf{a}_i,r_i,\mathbf{s}'_i)\sim\mathcal{D}}\left[\left(\underline{y_i} - Q(\mathbf{s}_i,\mathbf{a}_i)\right)^2\right]$$

$$y_i = r_i + \gamma \max_{\mathbf{a}'} \underline{Q'}(\mathbf{s}',\mathbf{a}')$$

Slowly moving target network

- Works very well in practice
- Nearly every modern Q-learning algorithms uses some kind of target network

DQN

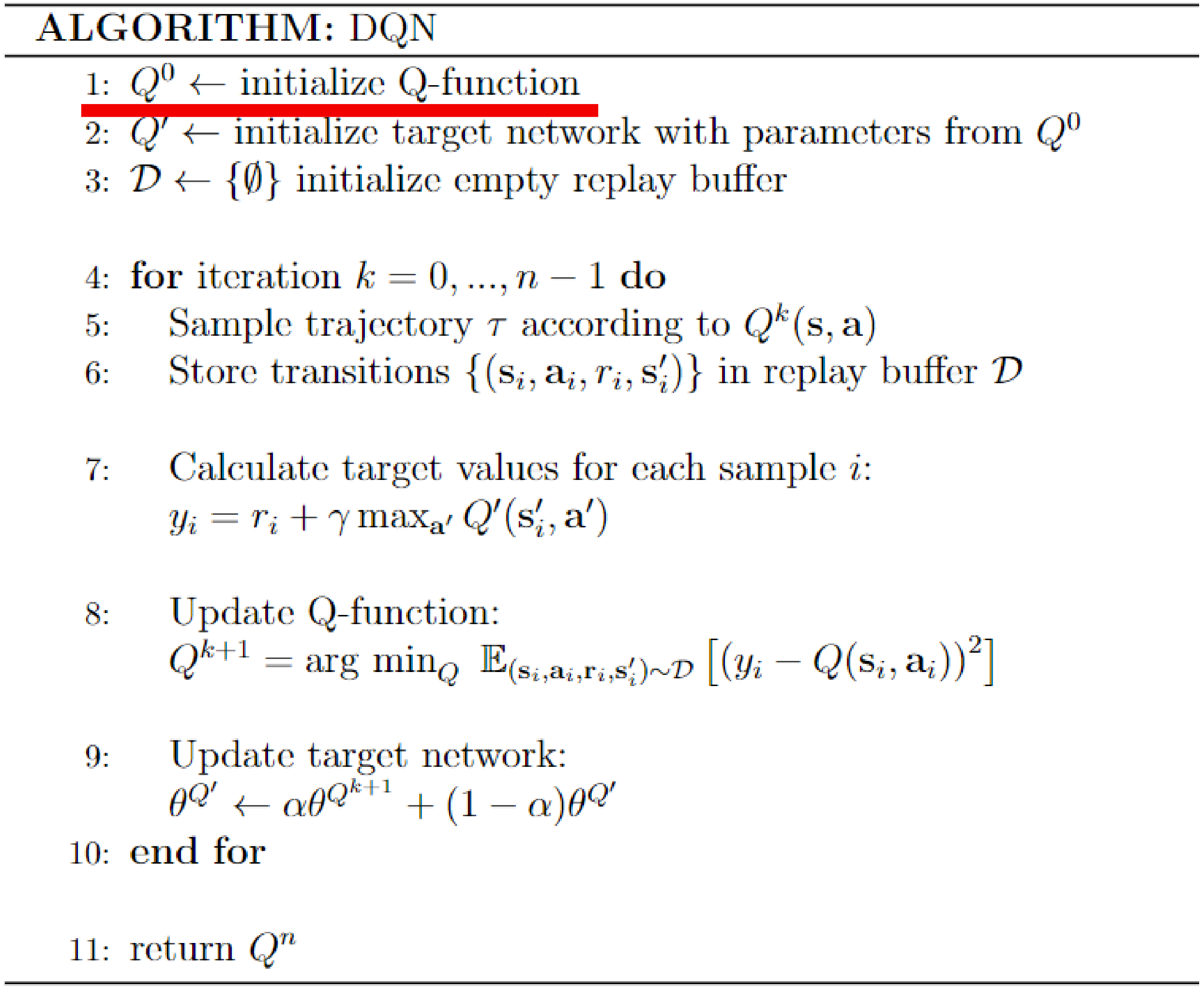


ALGORITHM: DQN

```
$Q^0 \leftarrow$ initialize Q-function
$Q' \leftarrow$ initialize target network with parameters from $Q^0$
$\mathcal{D} \leftarrow \{\emptyset\}$ initialize empty replay buffer

for iteration $k = 0, ..., n-1$ do
    Sample trajectory $\tau$ according to $Q^k(\mathbf{s}, \mathbf{a})$
    Store transitions $\{(\mathbf{s}_i, \mathbf{a}_i, r_i, \mathbf{s}_i')\}$ in replay buffer $\mathcal{D}$

    Calculate target values for each sample $i$:
    $y_i = r_i + \gamma \max_{\mathbf{a}'} Q'(\mathbf{s}_i', \mathbf{a}')$

    Update Q-function:
    $Q^{k+1} = \arg\min_Q \; \mathbb{E}_{(\mathbf{s}_i, \mathbf{a}_i, \mathbf{r}_i, \mathbf{s}_i') \sim \mathcal{D}} \left[ (y_i - Q(\mathbf{s}_i, \mathbf{a}_i))^2 \right]$

    Update target network:
    $\theta^{Q'} \leftarrow \alpha \theta^{Q^{k+1}} + (1-\alpha)\theta^{Q'}$
end for

return $Q^n$
```

Human-Level Control Through Deep Reinforcement Learning
[Mnih et al. 2015]

DQN

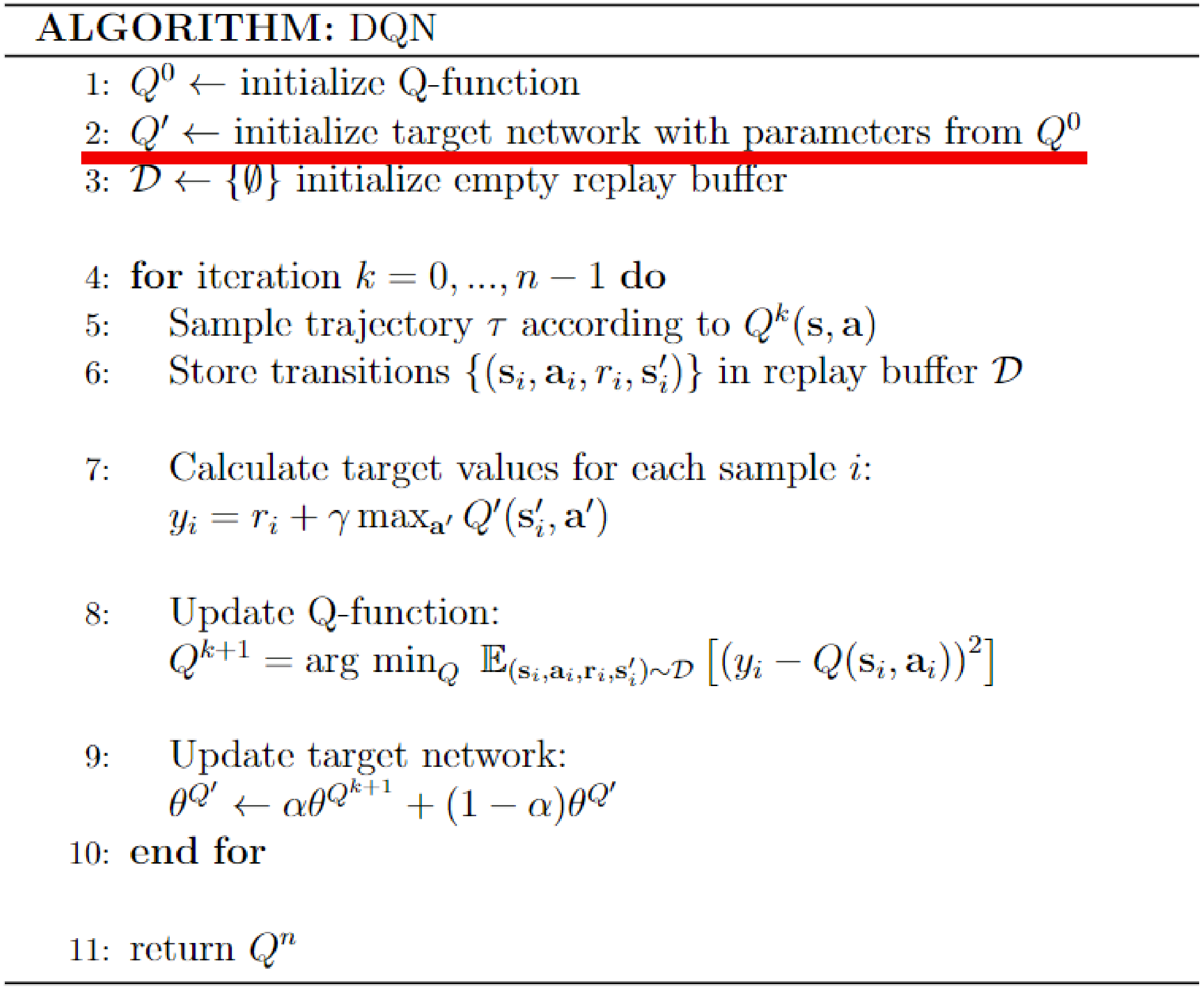


ALGORITHM: DQN

```
$Q^0 \leftarrow$ initialize Q-function
$Q' \leftarrow$ initialize target network with parameters from $Q^0$
$\mathcal{D} \leftarrow \{\emptyset\}$ initialize empty replay buffer

for iteration $k = 0, ..., n-1$ do
    Sample trajectory $\tau$ according to $Q^k(\mathbf{s}, \mathbf{a})$
    Store transitions $\{(\mathbf{s}_i, \mathbf{a}_i, r_i, \mathbf{s}'_i)\}$ in replay buffer $\mathcal{D}$

    Calculate target values for each sample $i$:
    $y_i = r_i + \gamma \max_{\mathbf{a}'} Q'(\mathbf{s}'_i, \mathbf{a}')$

    Update Q-function:
    $Q^{k+1} = \arg\min_Q \; \mathbb{E}_{(\mathbf{s}_i, \mathbf{a}_i, \mathbf{r}_i, \mathbf{s}'_i) \sim \mathcal{D}} \left[ (y_i - Q(\mathbf{s}_i, \mathbf{a}_i))^2 \right]$

    Update target network:
    $\theta^{Q'} \leftarrow \alpha \theta^{Q^{k+1}} + (1 - \alpha) \theta^{Q'}$
end for

return $Q^n$
```

Human-Level Control Through Deep Reinforcement Learning
[Mnih et al. 2015]

DQN

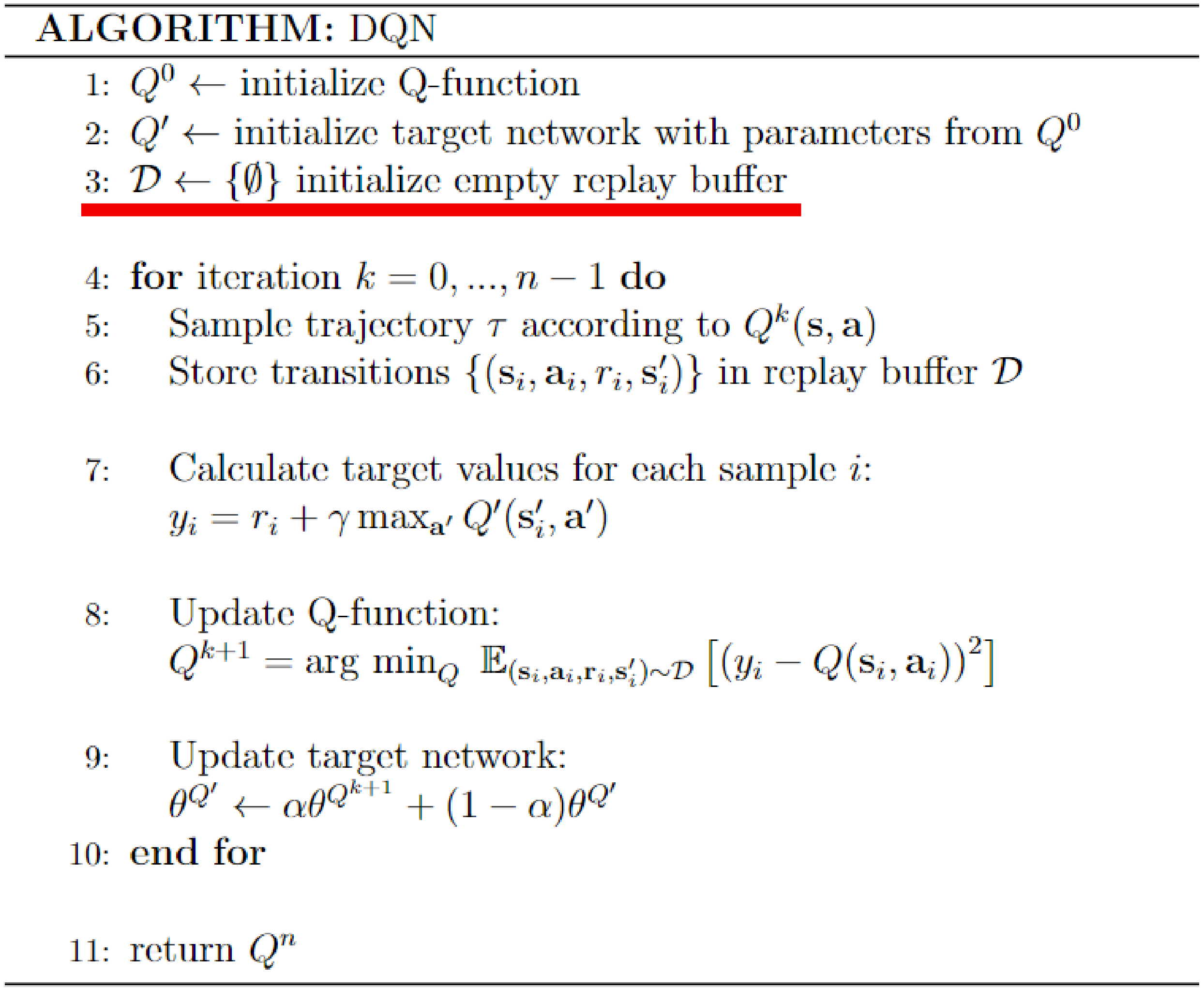


ALGORITHM: DQN

```
$Q^0 \leftarrow$ initialize Q-function
$Q' \leftarrow$ initialize target network with parameters from $Q^0$
$\mathcal{D} \leftarrow \{\emptyset\}$ initialize empty replay buffer

for iteration $k = 0, ..., n-1$ do
    Sample trajectory $\tau$ according to $Q^k(\mathbf{s}, \mathbf{a})$
    Store transitions $\{(\mathbf{s}_i, \mathbf{a}_i, r_i, \mathbf{s}'_i)\}$ in replay buffer $\mathcal{D}$

    Calculate target values for each sample $i$:
    $y_i = r_i + \gamma \max_{\mathbf{a}'} Q'(\mathbf{s}'_i, \mathbf{a}')$

    Update Q-function:
    $Q^{k+1} = \arg\min_Q \ \mathbb{E}_{(\mathbf{s}_i, \mathbf{a}_i, \mathbf{r}_i, \mathbf{s}'_i) \sim \mathcal{D}} \left[ (y_i - Q(\mathbf{s}_i, \mathbf{a}_i))^2 \right]$

    Update target network:
    $\theta^{Q'} \leftarrow \alpha \theta^{Q^{k+1}} + (1 - \alpha) \theta^{Q'}$
end for

return $Q^n$
```

Human-Level Control Through Deep Reinforcement Learning
[Mnih et al. 2015]

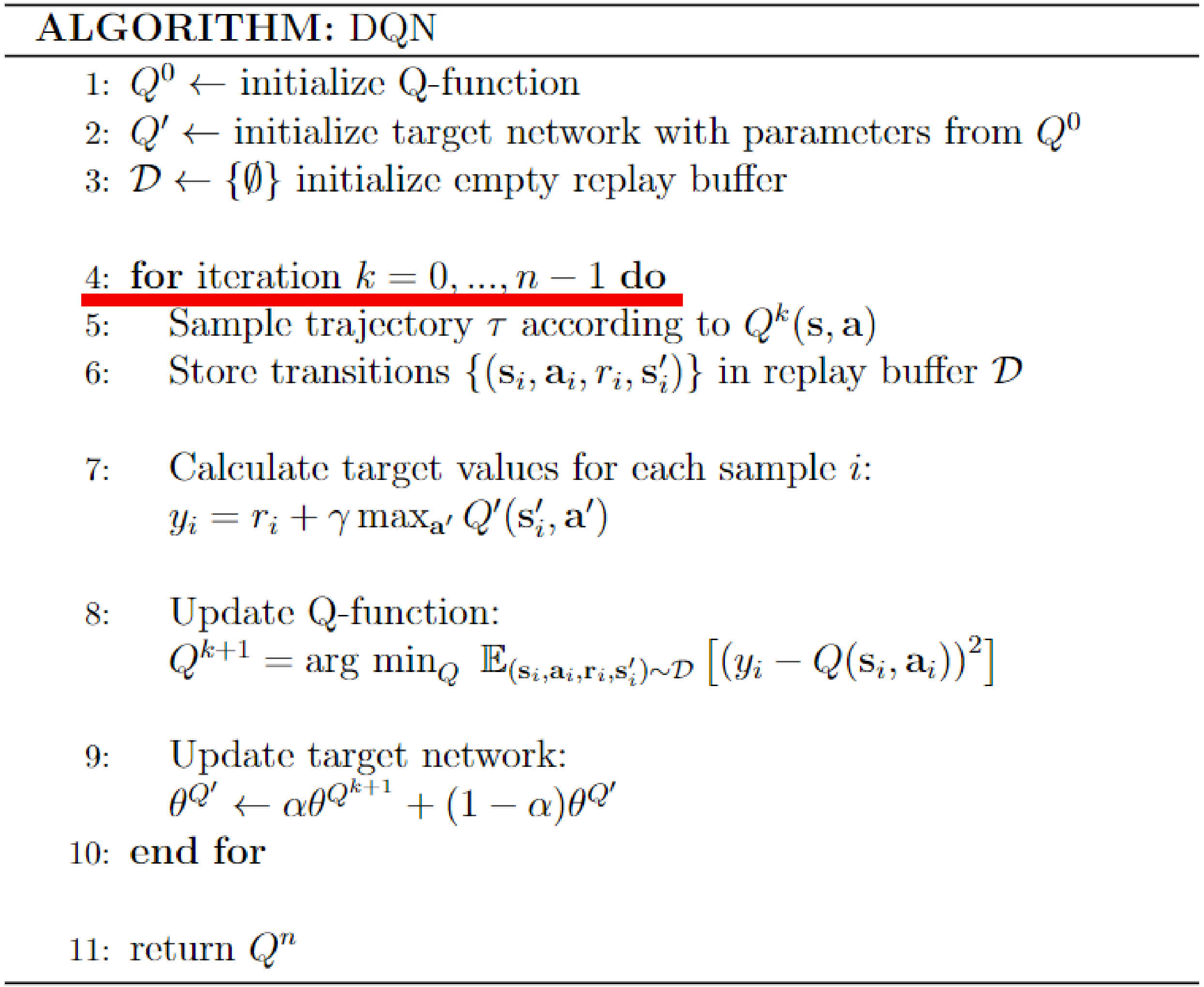
DQN



```
ALGORITHM: DQN
$Q^0 \leftarrow$ initialize Q-function
$Q' \leftarrow$ initialize target network with parameters from $Q^0$
$\mathcal{D} \leftarrow \{\emptyset\}$ initialize empty replay buffer

for iteration $k = 0, ..., n-1$ do
    Sample trajectory $\tau$ according to $Q^k(\mathbf{s}, \mathbf{a})$
    Store transitions $\{(\mathbf{s}_i, \mathbf{a}_i, r_i, \mathbf{s}'_i)\}$ in replay buffer $\mathcal{D}$

    Calculate target values for each sample $i$:
    $y_i = r_i + \gamma \max_{\mathbf{a}'} Q'(\mathbf{s}'_i, \mathbf{a}')$

    Update Q-function:
    $Q^{k+1} = \arg\min_Q \; \mathbb{E}_{(\mathbf{s}_i, \mathbf{a}_i, \mathbf{r}_i, \mathbf{s}'_i) \sim \mathcal{D}} \left[ (y_i - Q(\mathbf{s}_i, \mathbf{a}_i))^2 \right]$

    Update target network:
    $\theta^{Q'} \leftarrow \alpha \theta^{Q^{k+1}} + (1 - \alpha) \theta^{Q'}$
end for

return $Q^n$
```

Human-Level Control Through Deep Reinforcement Learning
[Mnih et al. 2015]

DQN

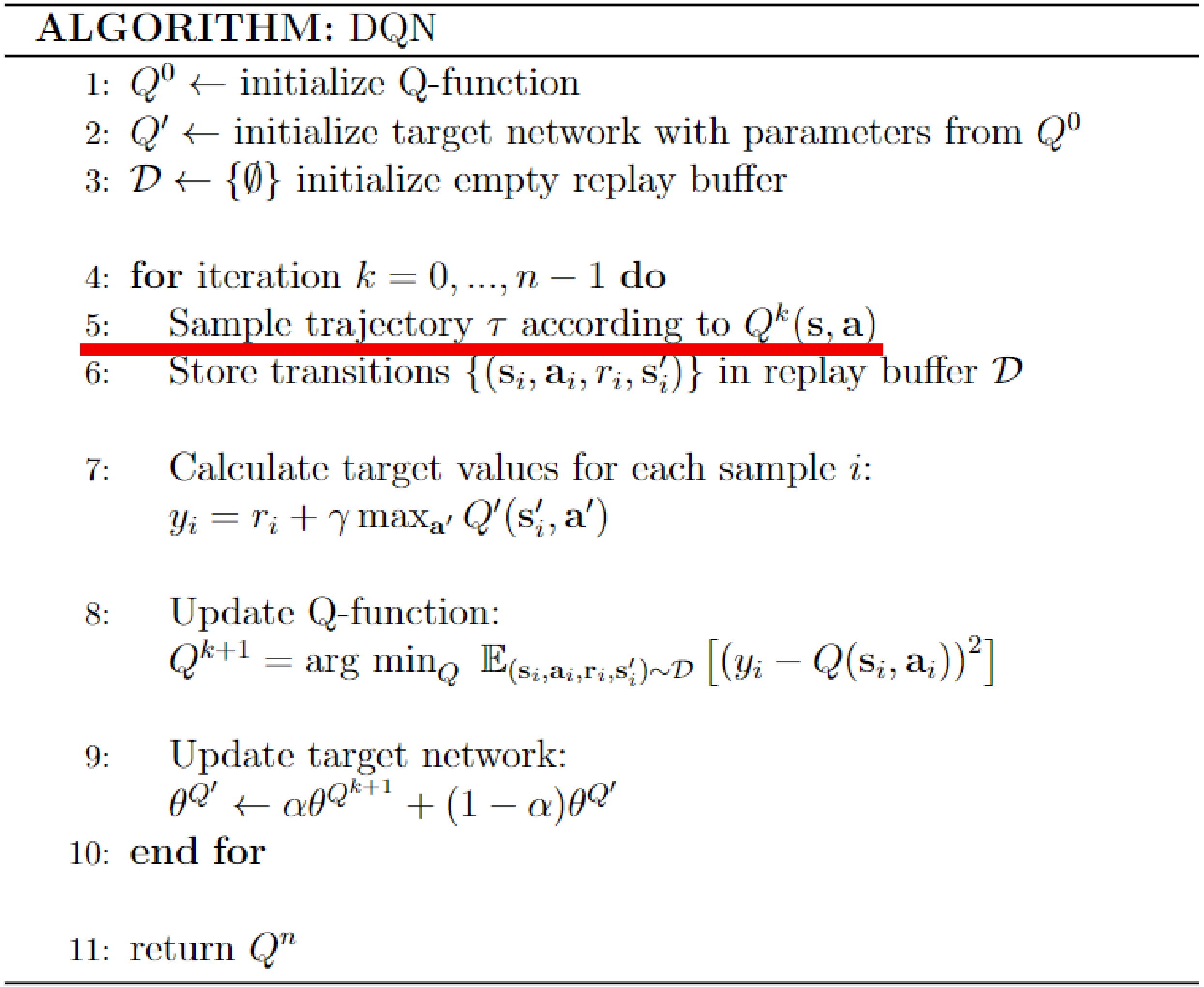


ALGORITHM: DQN

```
$Q^0 \leftarrow$ initialize Q-function
$Q' \leftarrow$ initialize target network with parameters from $Q^0$
$\mathcal{D} \leftarrow \{\emptyset\}$ initialize empty replay buffer

**for** iteration $k = 0, ..., n-1$ **do**
    Sample trajectory $\tau$ according to $Q^k(\mathbf{s}, \mathbf{a})$
    Store transitions $\{(\mathbf{s}_i, \mathbf{a}_i, r_i, \mathbf{s}'_i)\}$ in replay buffer $\mathcal{D}$

    Calculate target values for each sample $i$:
    $y_i = r_i + \gamma \max_{\mathbf{a}'} Q'(\mathbf{s}'_i, \mathbf{a}')$

    Update Q-function:
    $Q^{k+1} = \arg\min_Q \ \mathbb{E}_{(\mathbf{s}_i, \mathbf{a}_i, \mathbf{r}_i, \mathbf{s}'_i) \sim \mathcal{D}} \left[(y_i - Q(\mathbf{s}_i, \mathbf{a}_i))^2\right]$

    Update target network:
    $\theta^{Q'} \leftarrow \alpha \theta^{Q^{k+1}} + (1-\alpha)\theta^{Q'}$
**end for**

return $Q^n$
```

Human-Level Control Through Deep Reinforcement Learning
[Mnih et al. 2015]

DQN

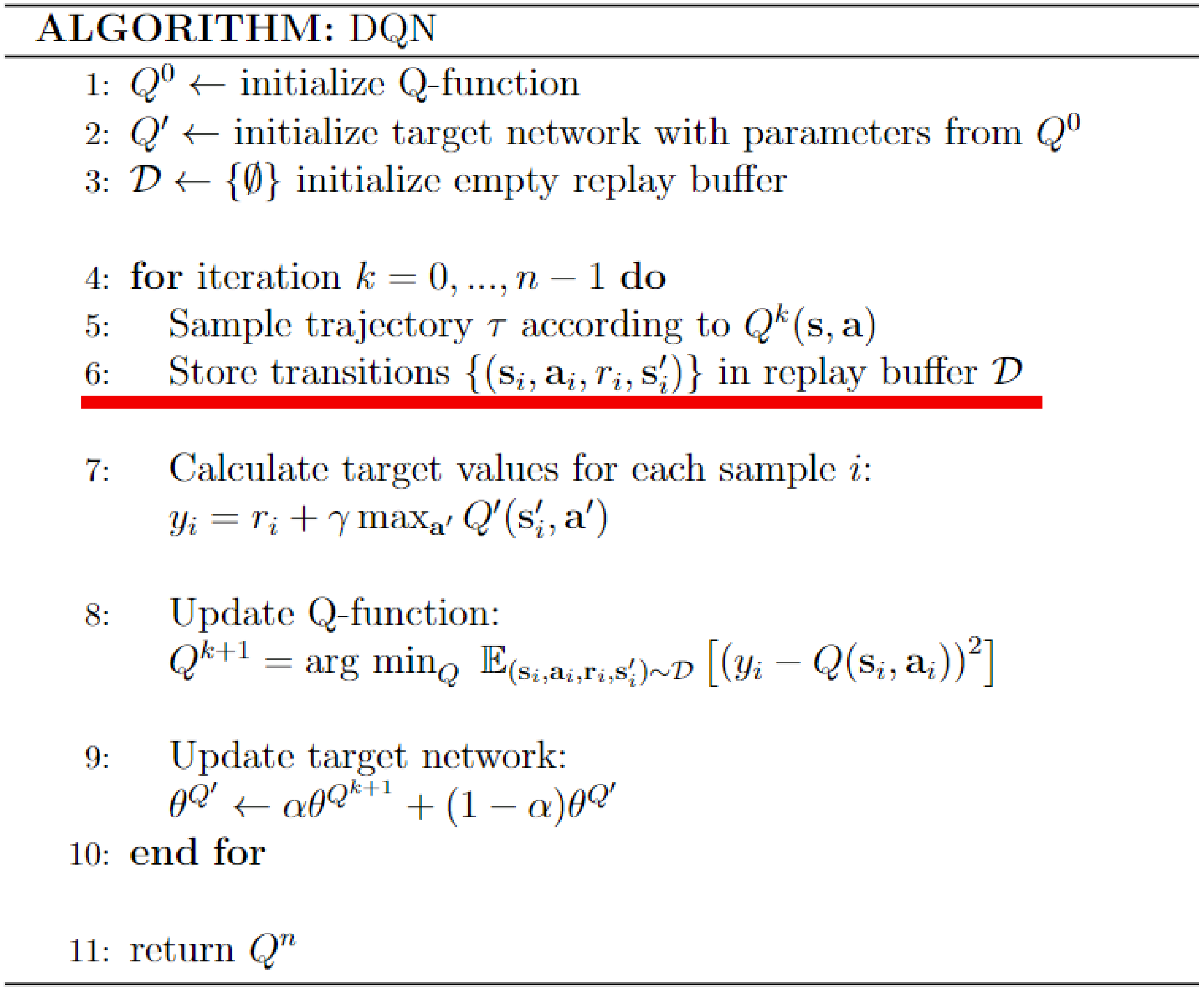


ALGORITHM: DQN

```
$Q^0 \leftarrow$ initialize Q-function
$Q' \leftarrow$ initialize target network with parameters from $Q^0$
$\mathcal{D} \leftarrow \{\emptyset\}$ initialize empty replay buffer

for iteration $k = 0, \ldots, n-1$ do
    Sample trajectory $\tau$ according to $Q^k(\mathbf{s}, \mathbf{a})$
    Store transitions $\{(\mathbf{s}_i, \mathbf{a}_i, r_i, \mathbf{s}_i')\}$ in replay buffer $\mathcal{D}$

    Calculate target values for each sample $i$:
    $y_i = r_i + \gamma \max_{\mathbf{a}'} Q'(\mathbf{s}_i', \mathbf{a}')$

    Update Q-function:
    $Q^{k+1} = \arg\min_Q \; \mathbb{E}_{(\mathbf{s}_i, \mathbf{a}_i, \mathbf{r}_i, \mathbf{s}_i') \sim \mathcal{D}} \left[ (y_i - Q(\mathbf{s}_i, \mathbf{a}_i))^2 \right]$

    Update target network:
    $\theta^{Q'} \leftarrow \alpha \theta^{Q^{k+1}} + (1-\alpha)\theta^{Q'}$
end for

return $Q^n$
```

Human-Level Control Through Deep Reinforcement Learning
[Mnih et al. 2015]

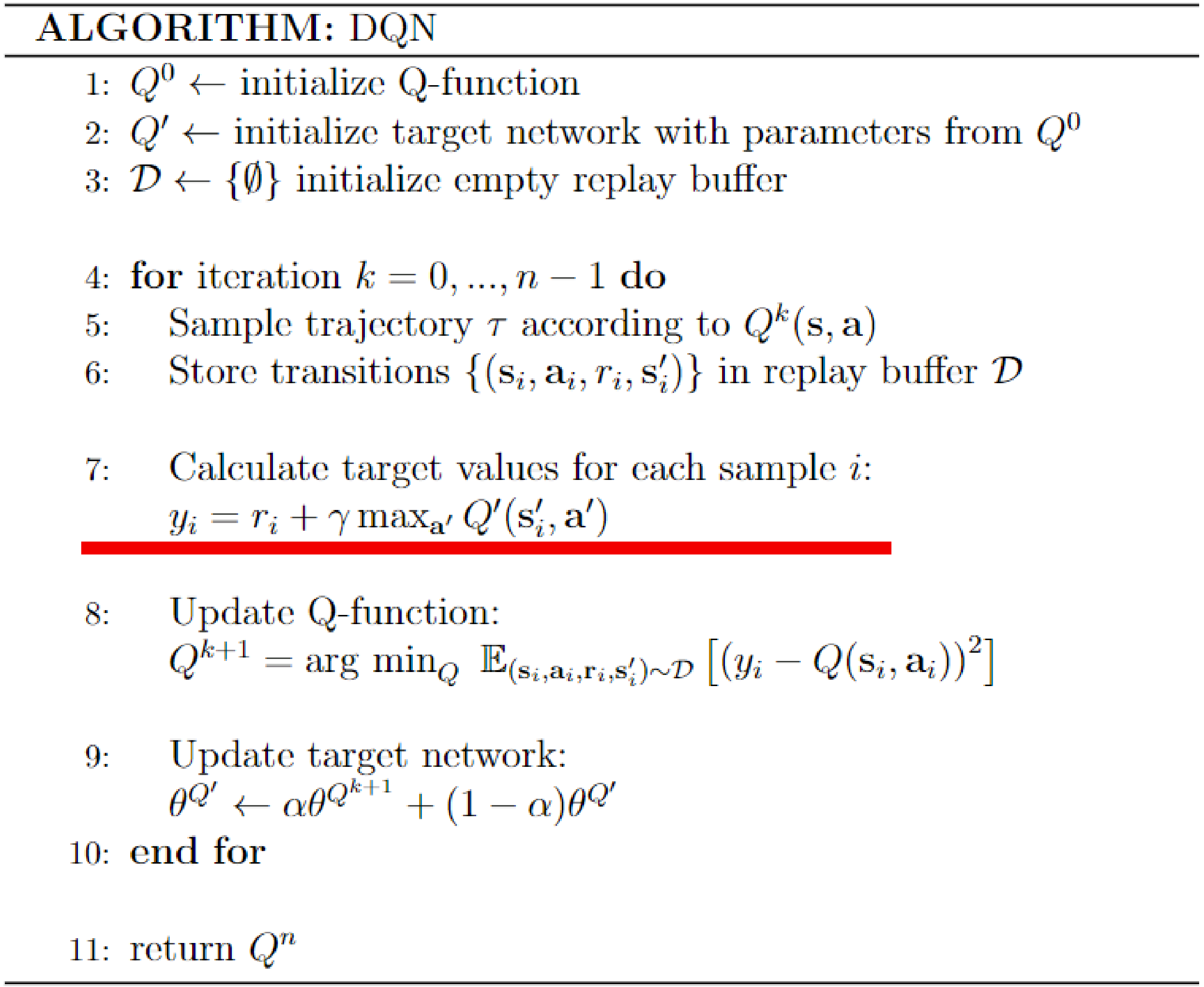
DQN



```
ALGORITHM: DQN
$Q^0 \leftarrow$ initialize Q-function
$Q' \leftarrow$ initialize target network with parameters from $Q^0$
$\mathcal{D} \leftarrow \{\emptyset\}$ initialize empty replay buffer

for iteration $k = 0, ..., n-1$ do
    Sample trajectory $\tau$ according to $Q^k(\mathbf{s}, \mathbf{a})$
    Store transitions $\{(\mathbf{s}_i, \mathbf{a}_i, r_i, \mathbf{s}'_i)\}$ in replay buffer $\mathcal{D}$

    Calculate target values for each sample $i$:
    $y_i = r_i + \gamma \max_{\mathbf{a}'} Q'(\mathbf{s}'_i, \mathbf{a}')$

    Update Q-function:
    $Q^{k+1} = \arg\min_Q \; \mathbb{E}_{(\mathbf{s}_i, \mathbf{a}_i, \mathbf{r}_i, \mathbf{s}'_i) \sim \mathcal{D}} \left[ (y_i - Q(\mathbf{s}_i, \mathbf{a}_i))^2 \right]$

    Update target network:
    $\theta^{Q'} \leftarrow \alpha \theta^{Q^{k+1}} + (1-\alpha)\theta^{Q'}$
end for

return $Q^n$
```

Human-Level Control Through Deep Reinforcement Learning
[Mnih et al. 2015]

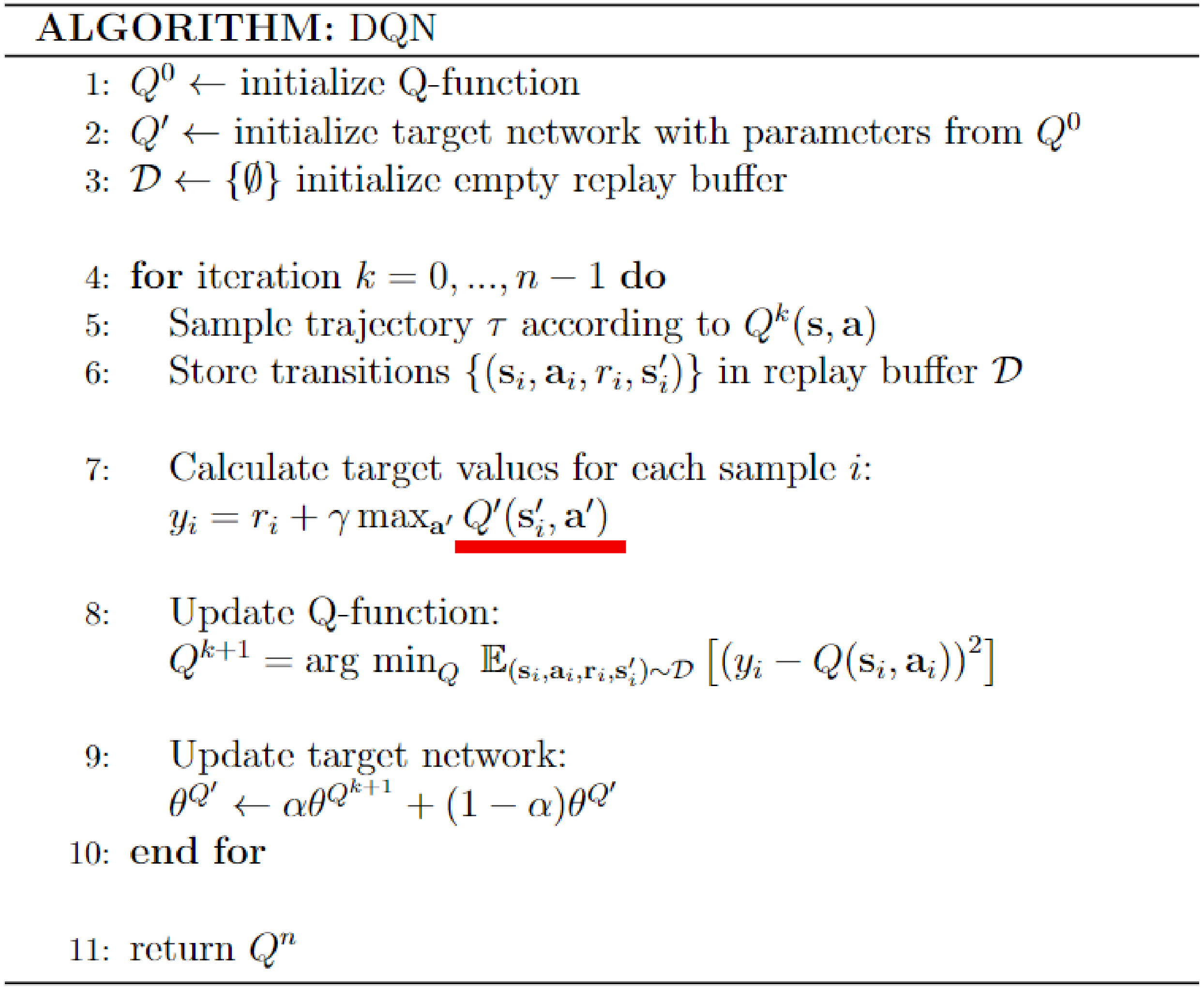
DQN



```
ALGORITHM: DQN
$Q^0 \leftarrow$ initialize Q-function
$Q' \leftarrow$ initialize target network with parameters from $Q^0$
$\mathcal{D} \leftarrow \{\emptyset\}$ initialize empty replay buffer

for iteration $k = 0, ..., n-1$ do
    Sample trajectory $\tau$ according to $Q^k(\mathbf{s}, \mathbf{a})$
    Store transitions $\{(\mathbf{s}_i, \mathbf{a}_i, r_i, \mathbf{s}'_i)\}$ in replay buffer $\mathcal{D}$

    Calculate target values for each sample $i$:
    $y_i = r_i + \gamma \max_{\mathbf{a}'} Q'(\mathbf{s}'_i, \mathbf{a}')$

    Update Q-function:
    $Q^{k+1} = \arg\min_Q \; \mathbb{E}_{(\mathbf{s}_i, \mathbf{a}_i, \mathbf{r}_i, \mathbf{s}'_i) \sim \mathcal{D}} \left[ (y_i - Q(\mathbf{s}_i, \mathbf{a}_i))^2 \right]$

    Update target network:
    $\theta^{Q'} \leftarrow \alpha \theta^{Q^{k+1}} + (1-\alpha)\theta^{Q'}$
end for

return $Q^n$
```

Human-Level Control Through Deep Reinforcement Learning
[Mnih et al. 2015]

DQN

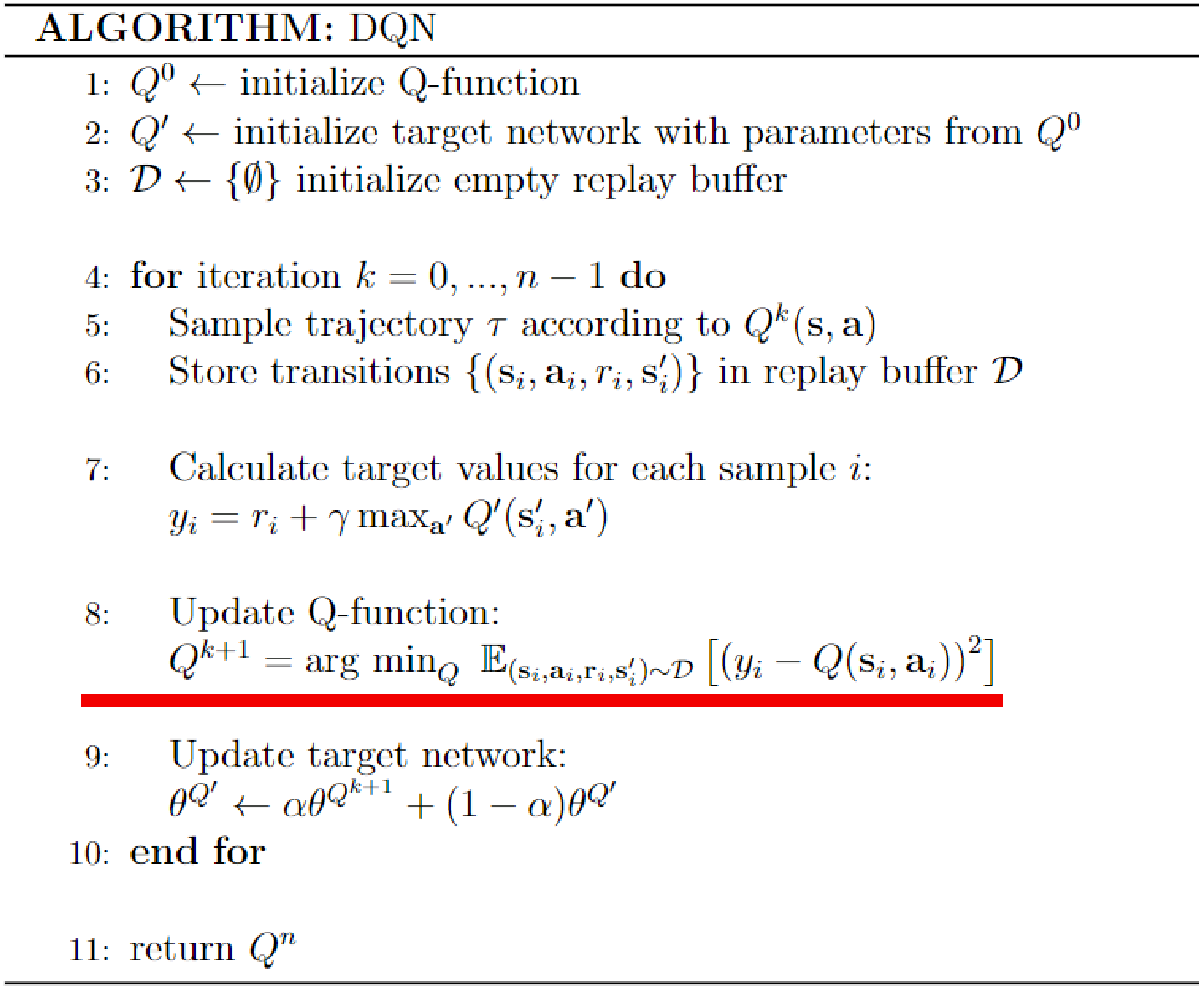


ALGORITHM: DQN

```
$Q^0 \leftarrow$ initialize Q-function
$Q' \leftarrow$ initialize target network with parameters from $Q^0$
$\mathcal{D} \leftarrow \{\emptyset\}$ initialize empty replay buffer

for iteration $k = 0, ..., n-1$ do
    Sample trajectory $\tau$ according to $Q^k(\mathbf{s}, \mathbf{a})$
    Store transitions $\{(\mathbf{s}_i, \mathbf{a}_i, r_i, \mathbf{s}'_i)\}$ in replay buffer $\mathcal{D}$

    Calculate target values for each sample $i$:
    $y_i = r_i + \gamma \max_{\mathbf{a}'} \underline{Q'(\mathbf{s}'_i, \mathbf{a}')}$

    Update Q-function:
    $Q^{k+1} = \arg\min_Q \ \mathbb{E}_{(\mathbf{s}_i, \mathbf{a}_i, \mathbf{r}_i, \mathbf{s}'_i) \sim \mathcal{D}} \left[ (y_i - Q(\mathbf{s}_i, \mathbf{a}_i))^2 \right]$

    Update target network:
    $\theta^{Q'} \leftarrow \alpha \theta^{Q^{k+1}} + (1-\alpha)\theta^{Q'}$
end for

return $Q^n$
```

Human-Level Control Through Deep Reinforcement Learning
[Mnih et al. 2015]

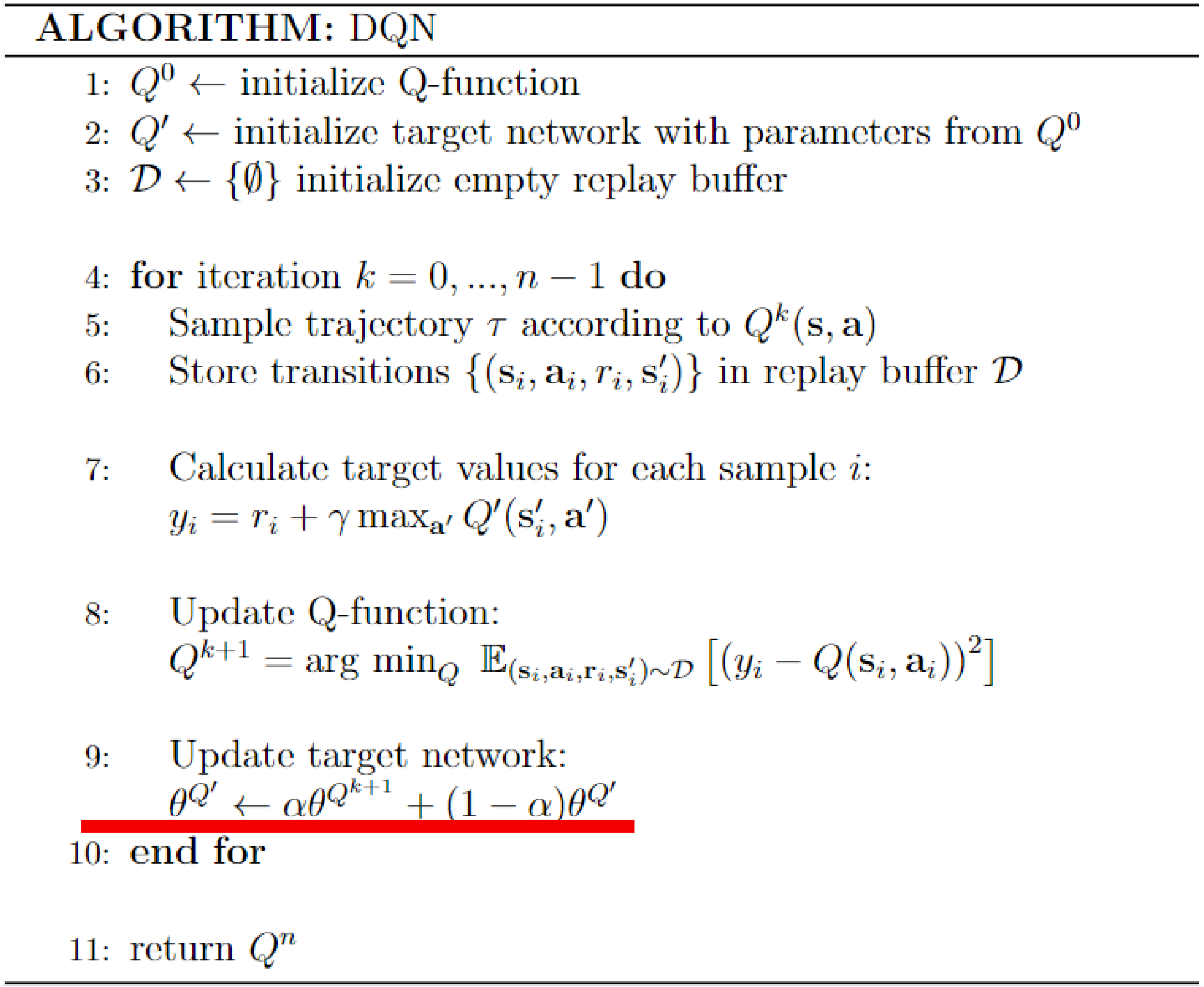
DQN



```
ALGORITHM: DQN
$Q^0 \leftarrow$ initialize Q-function
$Q' \leftarrow$ initialize target network with parameters from $Q^0$
$\mathcal{D} \leftarrow \{\emptyset\}$ initialize empty replay buffer

for iteration $k = 0, ..., n-1$ do
    Sample trajectory $\tau$ according to $Q^k(\mathbf{s}, \mathbf{a})$
    Store transitions $\{(\mathbf{s}_i, \mathbf{a}_i, r_i, \mathbf{s}'_i)\}$ in replay buffer $\mathcal{D}$

    Calculate target values for each sample $i$:
    $y_i = r_i + \gamma \max_{\mathbf{a}'} Q'(\mathbf{s}'_i, \mathbf{a}')$

    Update Q-function:
    $Q^{k+1} = \arg\min_Q \mathbb{E}_{(\mathbf{s}_i, \mathbf{a}_i, \mathbf{r}_i, \mathbf{s}'_i) \sim \mathcal{D}} \left[ (y_i - Q(\mathbf{s}_i, \mathbf{a}_i))^2 \right]$

    Update target network:
    $\theta^{Q'} \leftarrow \alpha \theta^{Q^{k+1}} + (1-\alpha)\theta^{Q'}$
end for

return $Q^n$
```

Human-Level Control Through Deep Reinforcement Learning
[Mnih et al. 2015]

DQN



```
ALGORITHM: DQN
$Q^0 \leftarrow$ initialize Q-function
$Q' \leftarrow$ initialize target network with parameters from $Q^0$
$\mathcal{D} \leftarrow \{\emptyset\}$ initialize empty replay buffer

for iteration $k = 0, ..., n-1$ do
    Sample trajectory $\tau$ according to $Q^k(\mathbf{s}, \mathbf{a})$
    Store transitions $\{(\mathbf{s}_i, \mathbf{a}_i, r_i, \mathbf{s}'_i)\}$ in replay buffer $\mathcal{D}$

    Calculate target values for each sample $i$:
    $y_i = r_i + \gamma \max_{\mathbf{a}'} Q'(\mathbf{s}'_i, \mathbf{a}')$

    Update Q-function:
    $Q^{k+1} = \arg\min_Q \mathbb{E}_{(\mathbf{s}_i, \mathbf{a}_i, \mathbf{r}_i, \mathbf{s}'_i) \sim \mathcal{D}} \left[ (y_i - Q(\mathbf{s}_i, \mathbf{a}_i))^2 \right]$

    Update target network:
    $\underline{\theta^{Q'} \leftarrow \alpha \theta^{Q^{k+1}} + (1-\alpha)\theta^{Q'}}$
end for

return $Q^n$
```

Human-Level Control Through Deep Reinforcement Learning
[Mnih et al. 2015]

Overview

- Nonstationary Targets → Target Networks
- Overestimation → Pessimistic Estimates
- Model Architecture

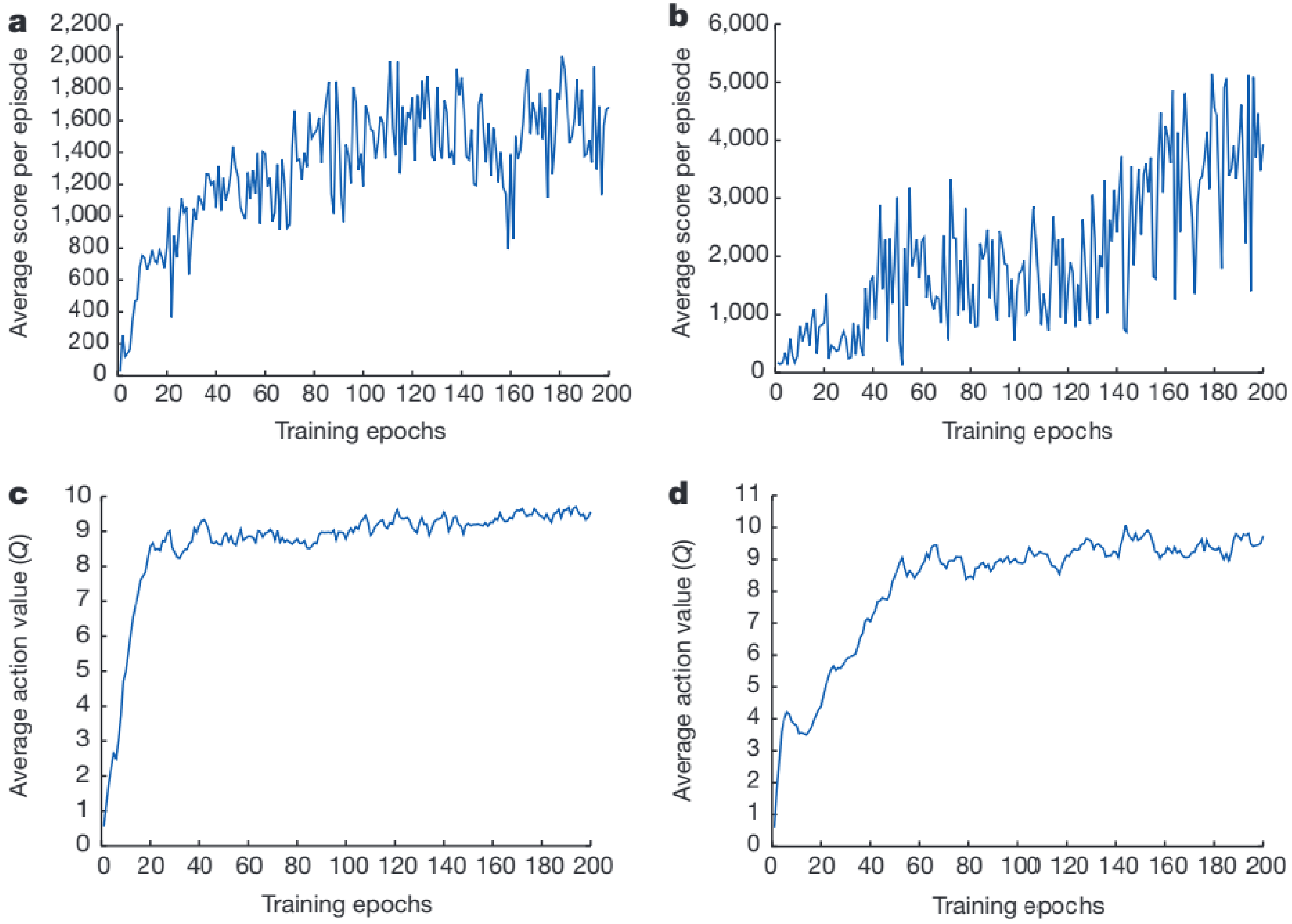
How Accurate is the Q-Function?

Space Invaders

Seaquest

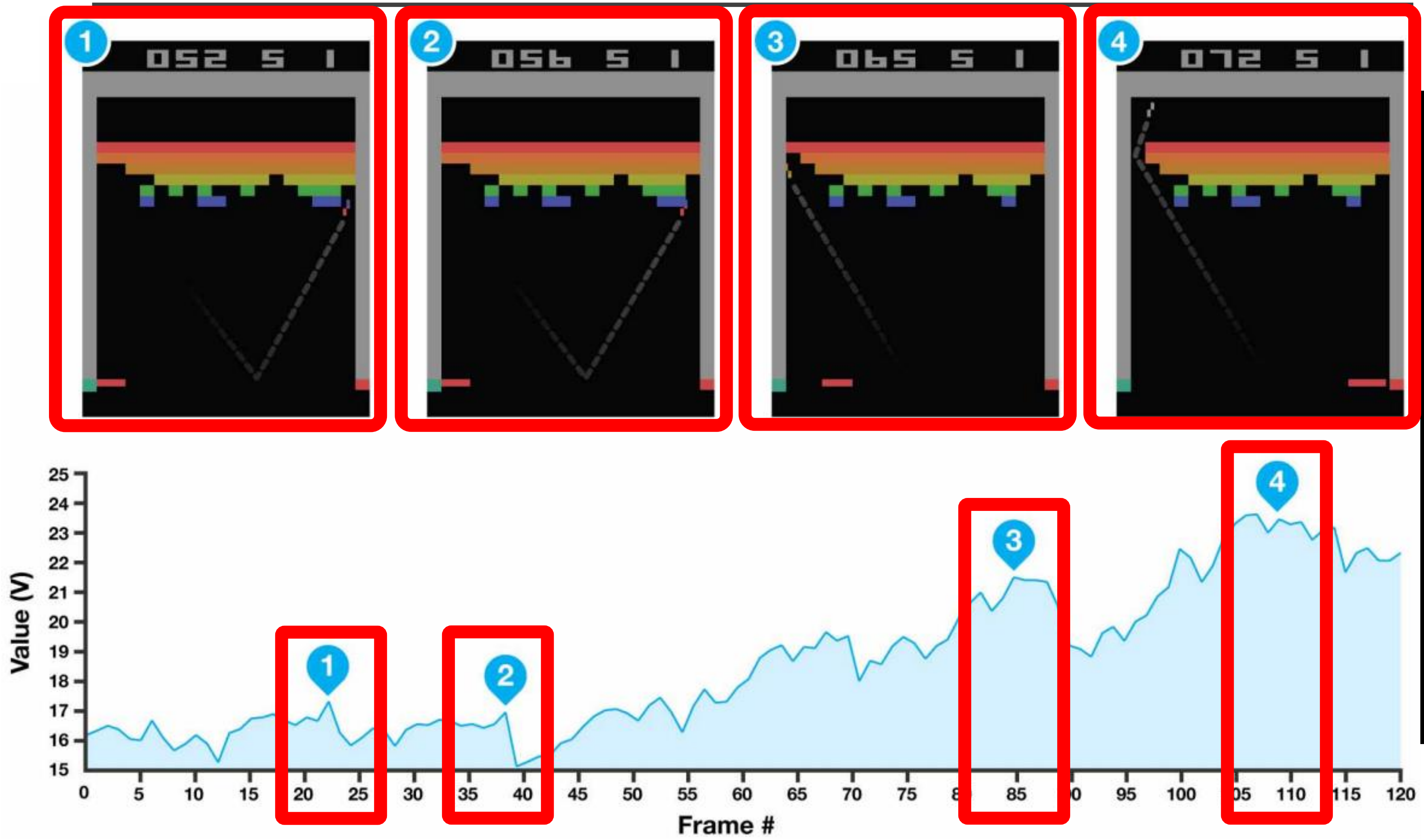
Real Score

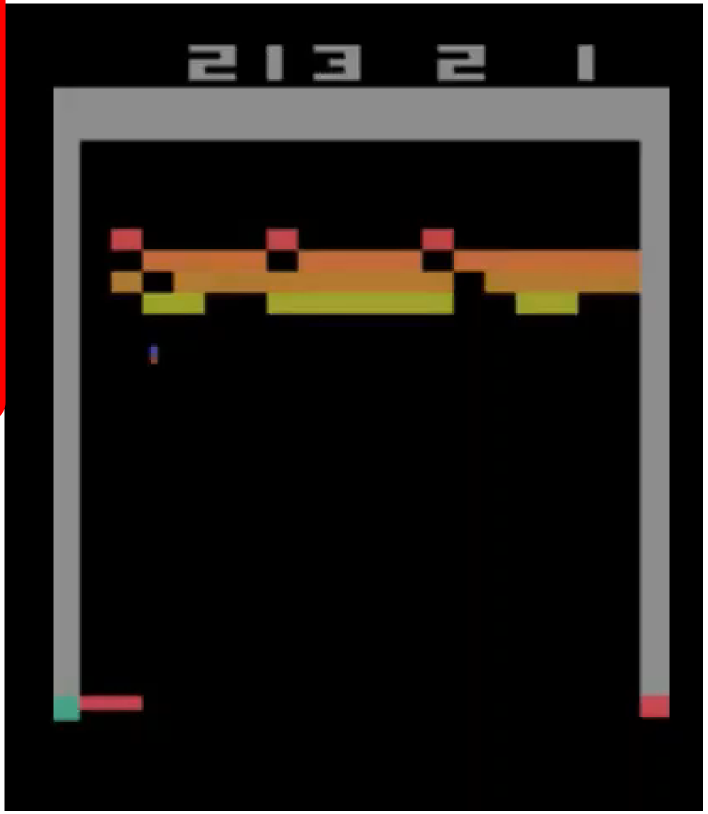
Predicted Value



Human-Level Control Through Deep Reinforcement Learning
[Mnih et al. 2015]

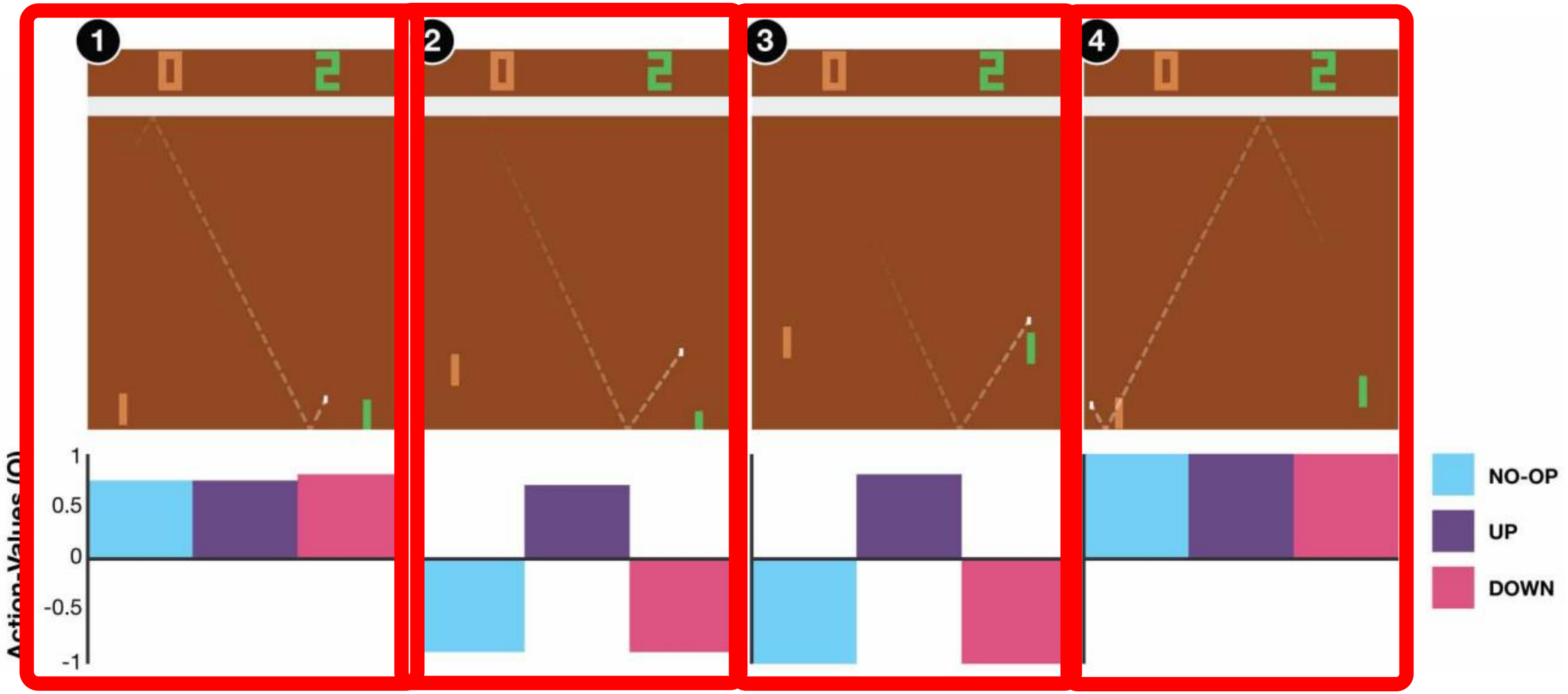
How Accurate is the Q-Function?

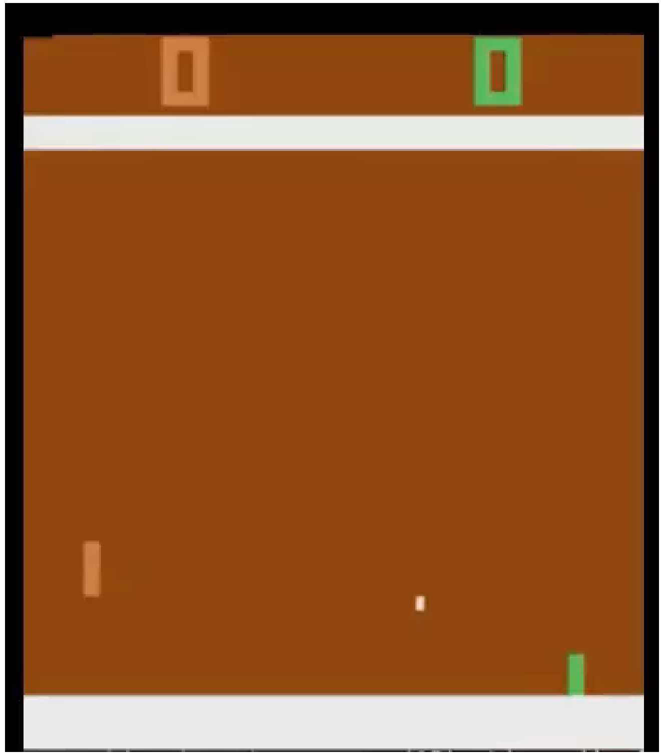




Human-Level Control Through Deep Reinforcement Learning
[Mnih et al. 2015]

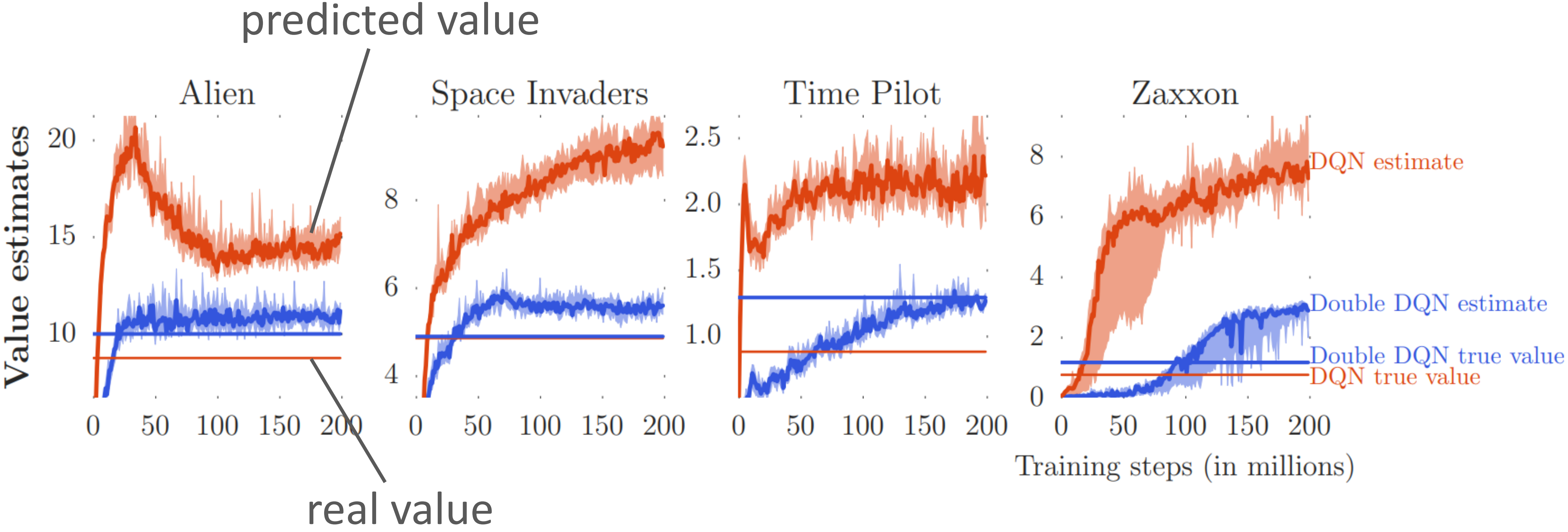
How Accurate is the Q-Function?





Human-Level Control Through Deep Reinforcement Learning
[Mnih et al. 2015]

Overestimation



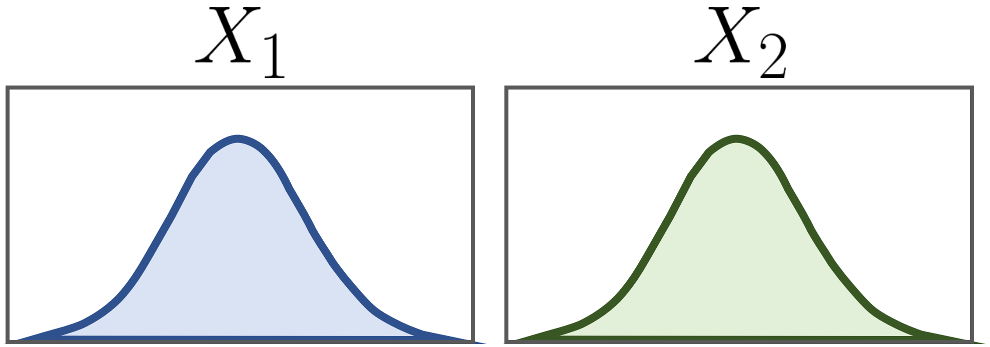
predicted value

real value

Deep Reinforcement Learning with Double Q-learning
[van Hasselt et al. 2016]

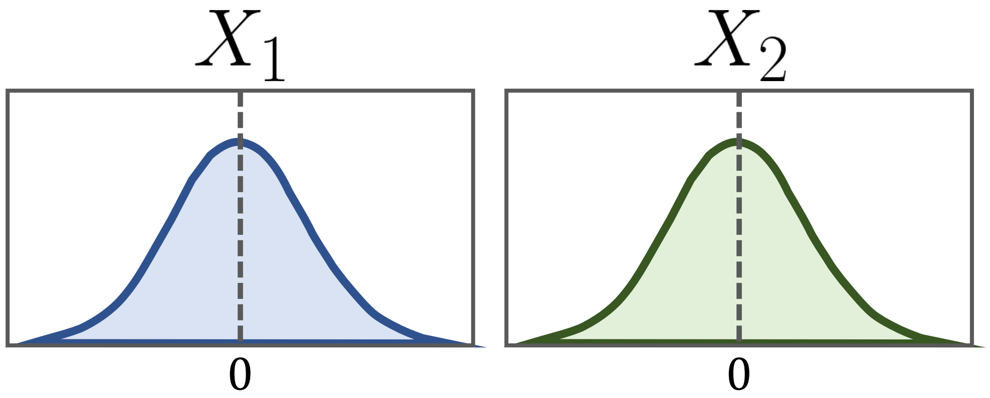
Overestimation

$$y_i = r_i + \gamma \underbrace{\max_{\mathbf{a}'}}_{\text{Bias towards positive errors}} Q^k(\mathbf{s}', \mathbf{a}')$$



Overestimation

$$y_i = r_i + \gamma \max_{\mathbf{a}'} \; Q^k(\mathbf{s}', \mathbf{a}')$$



$\mathbb{E}\left[X_1\right] = 0$ $\mathbb{E}\left[X_2\right] = 0$

$$\mathbb{E}\left[\max\left(X_1, X_2\right)\right] > 0$$

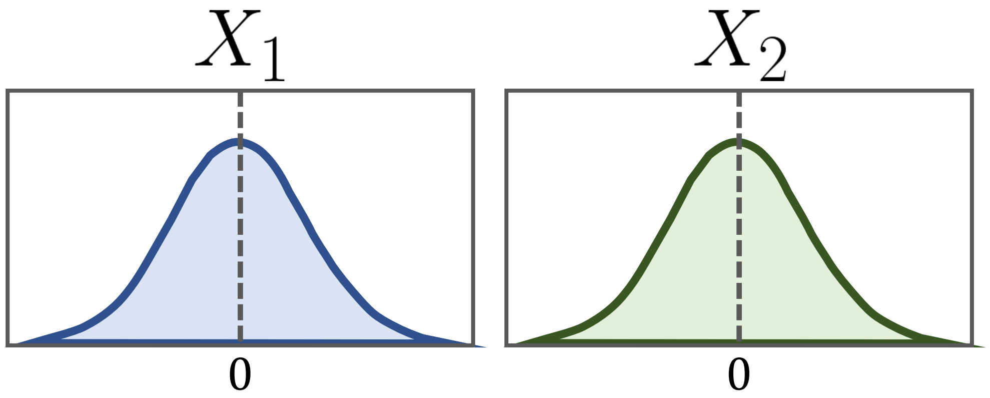
$$p(X_1 > 0) = p(X_2 > 0) = 0.5$$

$$p(\max\left(X_1, X_2\right) > 0)$$
$$= p(X_1 > 0 \text{ or } X_2 > 0) = \underline{0.75}$$

Overestimation

$$y_i = r_i + \gamma \max_{\mathbf{a}'} \underline{Q^k(\mathbf{s}', \mathbf{a}')}$$

Tends to be noisy



$\mathbb{E}[X_1] = 0$ $\mathbb{E}[X_2] = 0$

$$\mathbb{E}[\max(X_1, X_2)] > 0$$

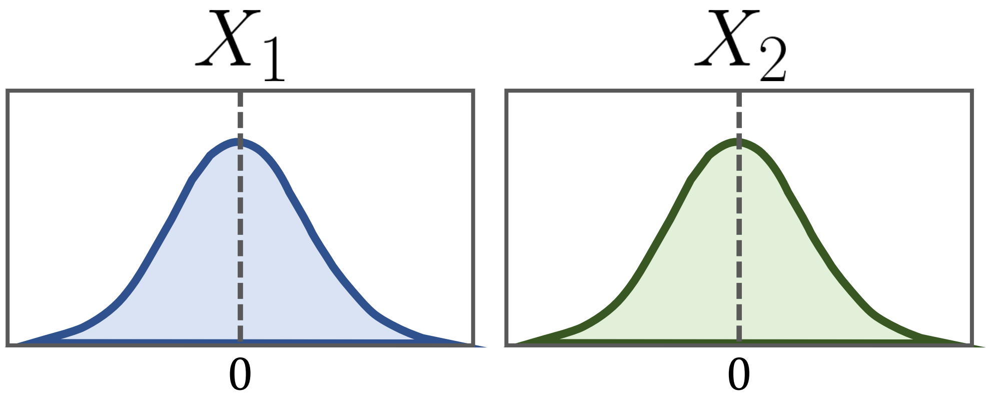
$$p(X_1 > 0) = p(X_2 > 0) = 0.5$$

$$p(\max(X_1, X_2) > 0)$$
$$= p(X_1 > 0 \text{ or } X_2 > 0) = 0.75$$

Overestimation

$$y_i = r_i + \gamma \underline{\max_{\mathbf{a}'} Q^k(\mathbf{s}', \mathbf{a}')}$$

More likely to *overestimate* next value



$\mathbb{E}\left[X_1\right] = 0$ $\mathbb{E}\left[X_2\right] = 0$

$$\mathbb{E}\left[\max\left(X_1, X_2\right)\right] > 0$$

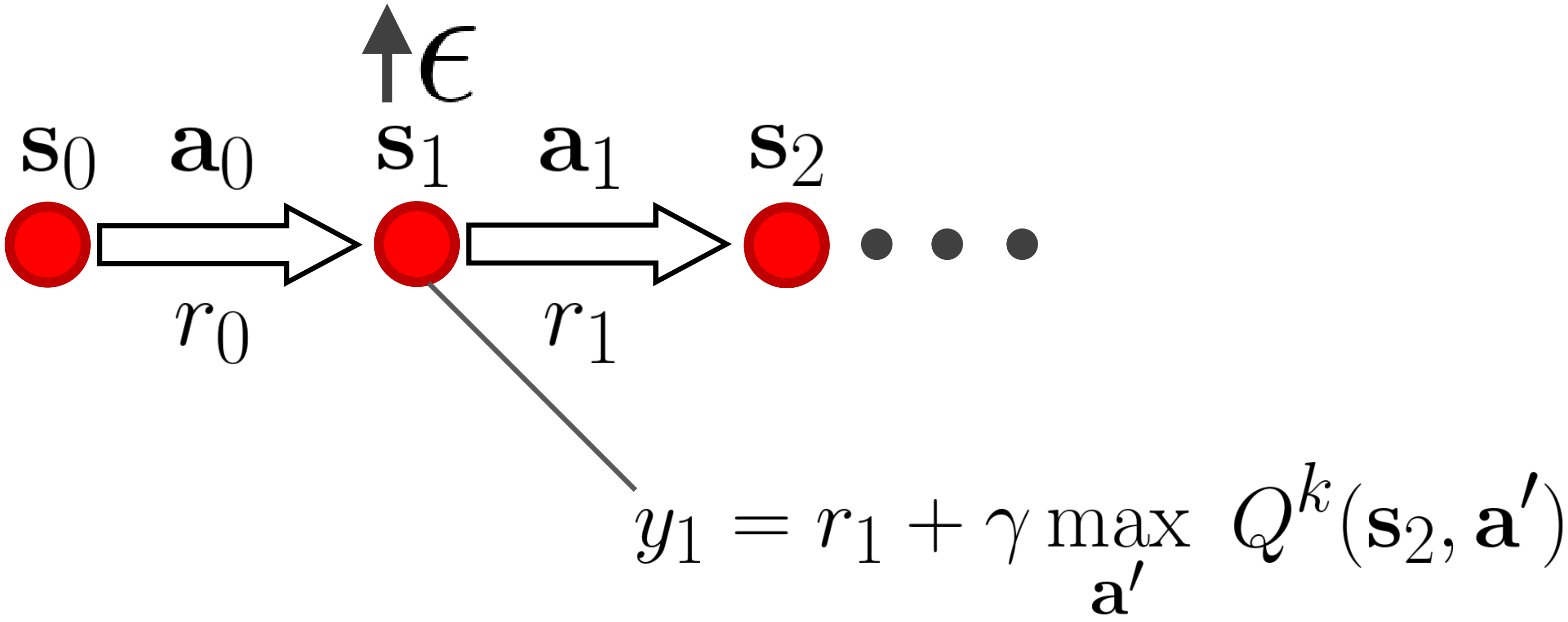
$$p(X_1 > 0) = p(X_2 > 0) = 0.5$$

$$p(\max\left(X_1, X_2\right) > 0)$$
$$= p(X_1 > 0 \text{ or } X_2 > 0) = 0.75$$

Overestimation

$$y_i = r_i + \gamma \max_{\mathbf{a}'} Q^k(\mathbf{s}', \mathbf{a}')$$

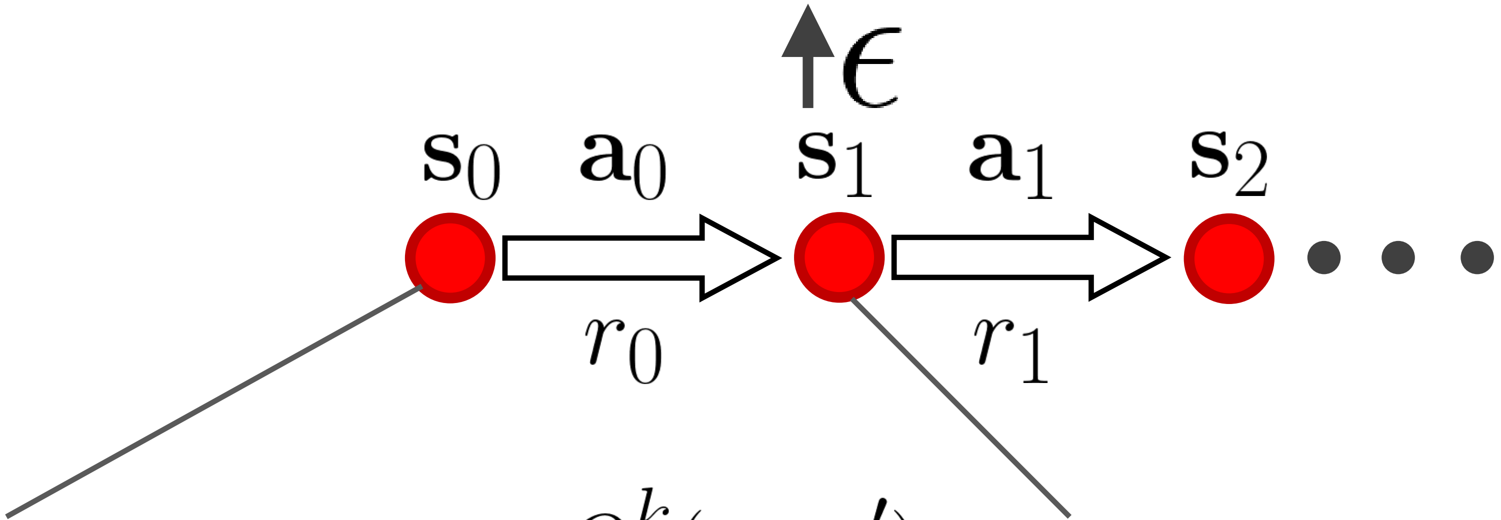
Bootstrapping can propagate overestimation errors



Overestimation

$$y_i = r_i + \gamma \max_{\mathbf{a}'} Q^k(\mathbf{s}', \mathbf{a}')$$

Bootstrapping can propagate overestimation errors



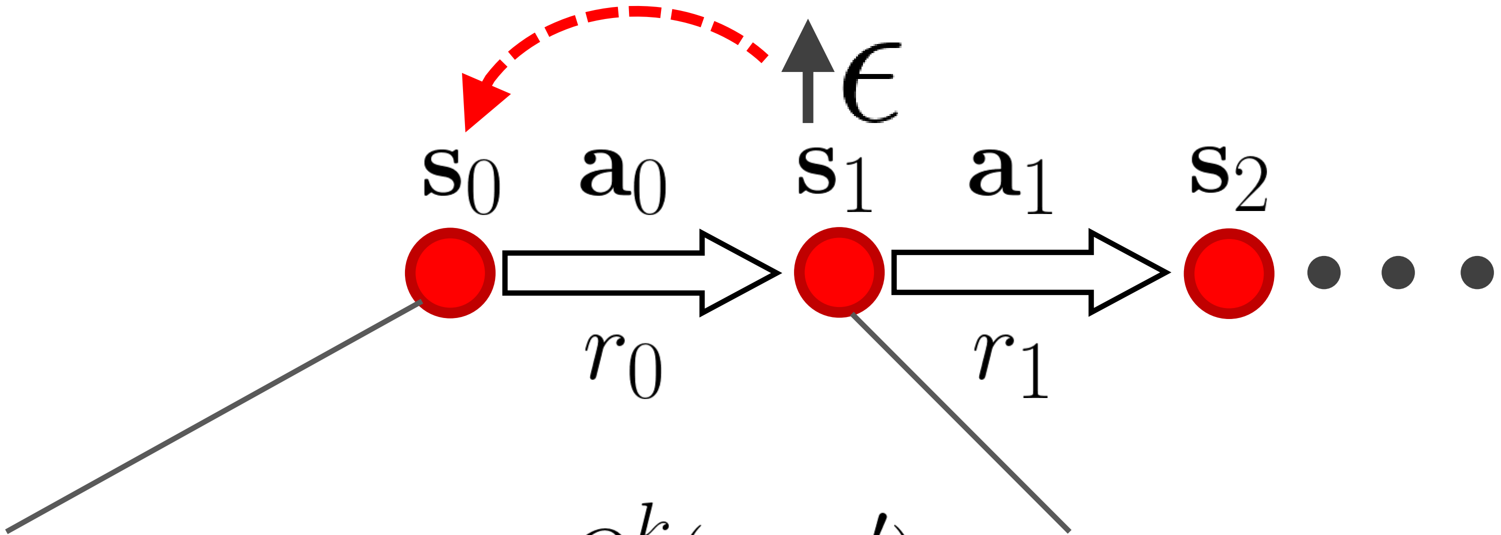
$$y_0 = r_0 + \gamma \max_{\mathbf{a}'} Q^k(\underline{\mathbf{s}_1}, \mathbf{a}')$$

$$y_1 = r_1 + \gamma \max_{\mathbf{a}'} Q^k(\mathbf{s}_2, \mathbf{a}')$$

Overestimation

$$y_i = r_i + \gamma \max_{\mathbf{a}'} \; Q^k(\mathbf{s}', \mathbf{a}')$$

Bootstrapping can propagate overestimation errors



$$y_0 = r_0 + \gamma \max_{\mathbf{a}'} \; Q^k(\mathbf{s}_1, \mathbf{a}')$$

$$y_1 = r_1 + \gamma \max_{\mathbf{a}'} \; Q^k(\mathbf{s}_2, \mathbf{a}')$$

Overestimation

$$y_i = r_i + \gamma \max_{\mathbf{a}'} \underline{Q^k(\mathbf{s}', \mathbf{a}')}$$

Target network can slow propagation of errors

Overestimation

$$y_i = r_i + \gamma \max_{\mathbf{a}'} \; Q^k(\mathbf{s}', \mathbf{a}')$$

$$\Downarrow$$

$$y_i = r_i + \gamma \underbrace{Q^k}_{\text{action evaluation}} \left(\mathbf{s}', \underbrace{\arg\,\max_{\mathbf{a}'} Q^k(\mathbf{s}', \mathbf{a}')}_{\text{action selection}} \right)$$

Double Q-Learning

Decouple selection from evaluation by using *different* Q-functions

$$y_i = r_i + \gamma \underbrace{Q^k}_{\text{action evaluation}} \left(\mathbf{s}', \underbrace{\arg\max_{\mathbf{a}'} Q^k(\mathbf{s}', \mathbf{a}')}_{\text{action selection}}\right)$$

Double Q-Learning

Decouple selection from evaluation by using *different* Q-functions

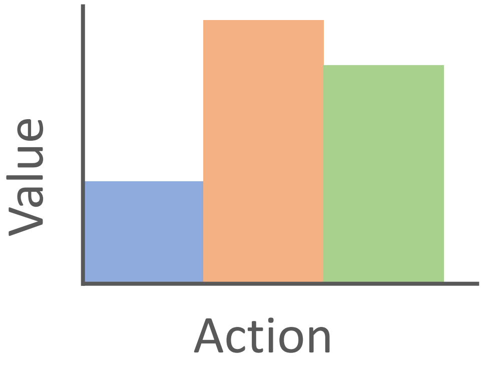
$$y_i = r_i + \gamma \underbrace{Q_B^k}_{\text{action evaluation}} \left(\mathbf{s}', \underbrace{\arg\max_{\mathbf{a}'} Q_A^k(\mathbf{s}', \mathbf{a}')}_{\text{action selection}}\right)$$

Double Q-Learning

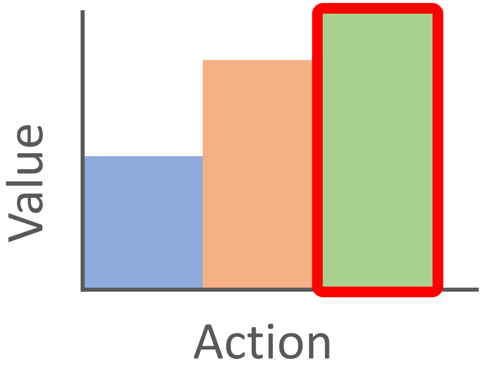
Decouple selection from evaluation by using *different* Q-functions

$$y_i = r_i + \gamma \, Q_B^k \left(\mathbf{s}', \arg\max_{\mathbf{a}'} Q_A^k(\mathbf{s}', \mathbf{a}')\right)$$

Real Q



Q_A^k

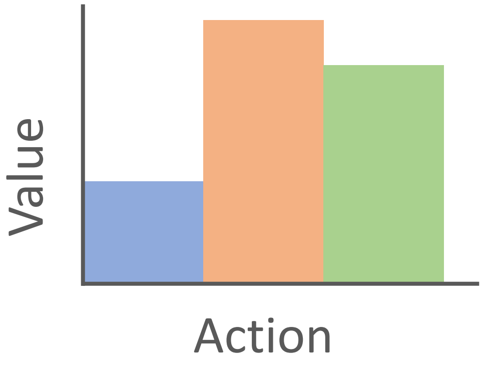


Double Q-Learning

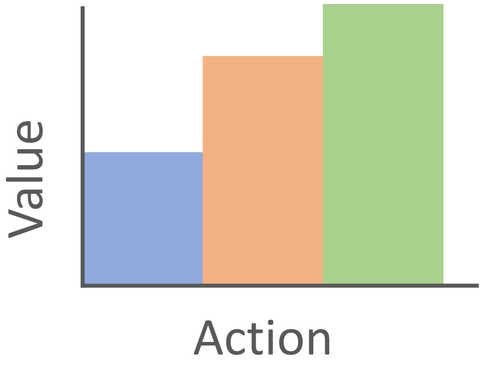
Decouple selection from evaluation by using *different* Q-functions

$$y_i = r_i + \gamma \, Q_B^k \left(\mathbf{s}', \arg\max_{\mathbf{a}'} Q_A^k(\mathbf{s}', \mathbf{a}') \right)$$

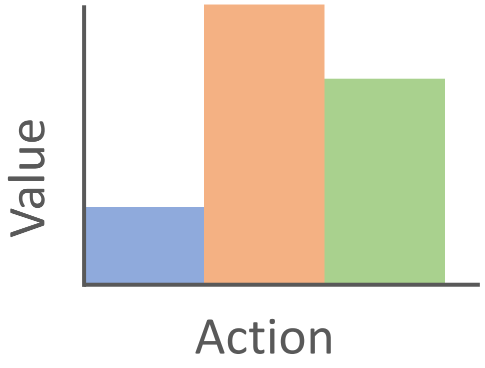
Real Q



Q_A^k



Q_B^k

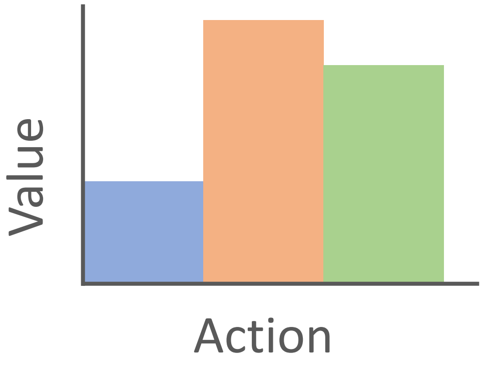


Double Q-Learning

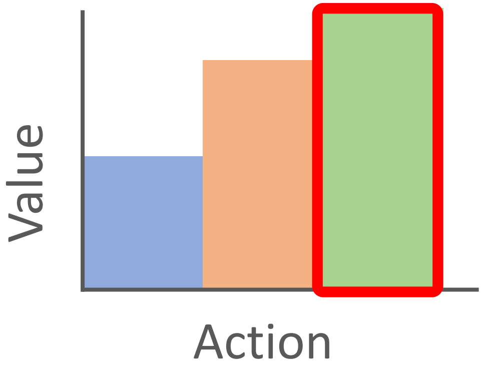
Decouple selection from evaluation by using *different* Q-functions

$$y_i = r_i + \gamma\, Q_B^k\left(\mathbf{s}', \underbrace{\arg\max_{\mathbf{a}'} Q_A^k(\mathbf{s}', \mathbf{a}')}_{\text{action selection}}\right)$$

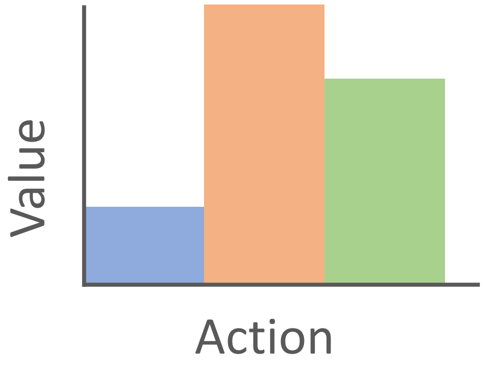
Real Q



Q_A^k



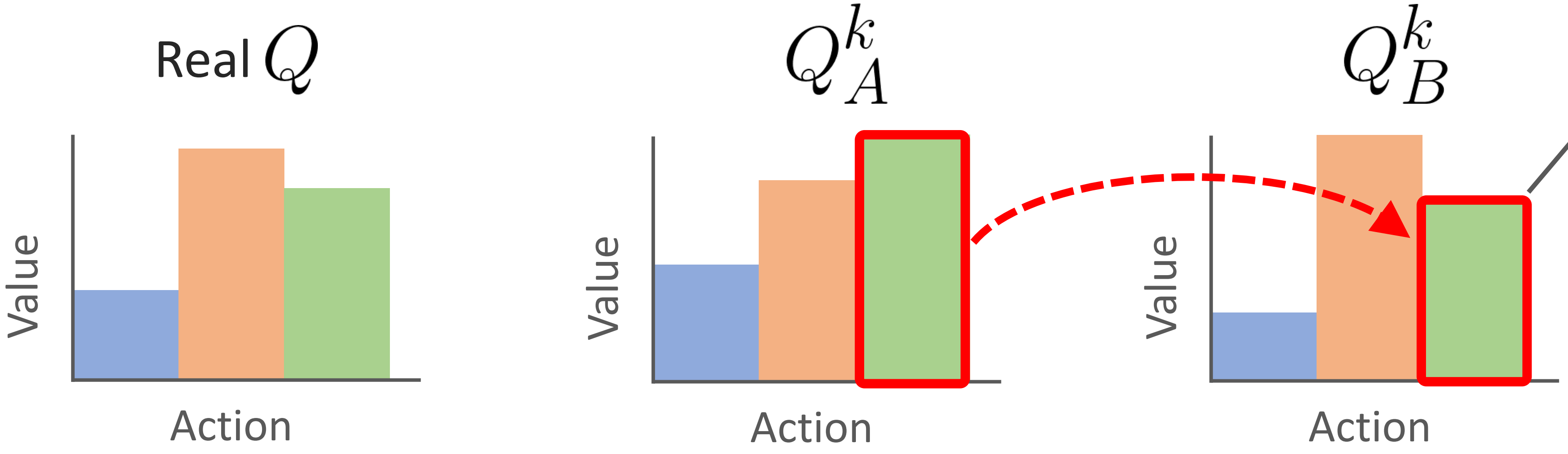
Q_B^k



Double Q-Learning

Decouple selection from evaluation by using *different* Q-functions

$$y_i = r_i + \gamma \underbrace{Q_B^k}_{\text{action evaluation}} \left(\mathbf{s}', \arg\max_{\mathbf{a}'} Q_A^k(\mathbf{s}', \mathbf{a}')\right)$$



use this value for bootstrap

Implementation

Option 1: Train two separate Q-functions

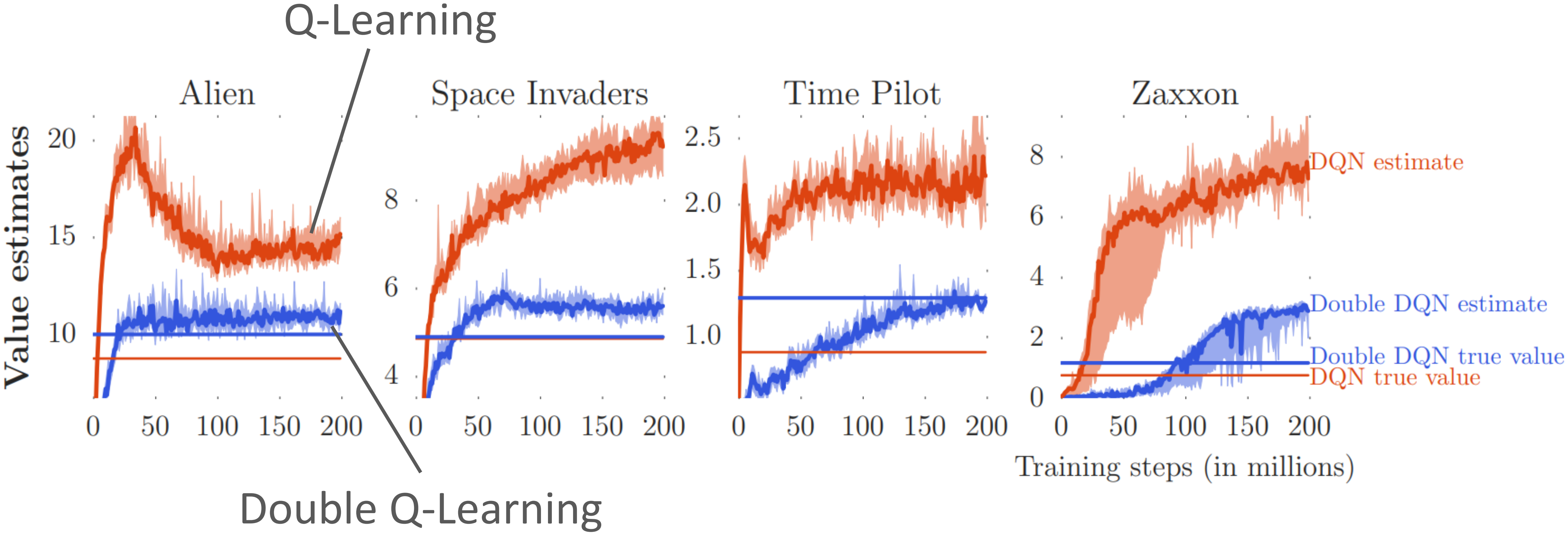
$$y_i = r_i + \gamma\, Q_B^k\left(\mathbf{s}', \arg\max_{\mathbf{a}'} Q_A^k(\mathbf{s}', \mathbf{a}')\right)$$

Option 2: Use target network

$$y_i = r_i + \gamma\, \underbrace{Q'}_{\text{target network}}\left(\mathbf{s}', \arg\max_{\mathbf{a}'} \underbrace{Q^k(\mathbf{s}', \mathbf{a}')}_{\text{main Q-network}}\right)$$

Deep Reinforcement Learning with Double Q-learning
[van Hasselt et al. 2016]

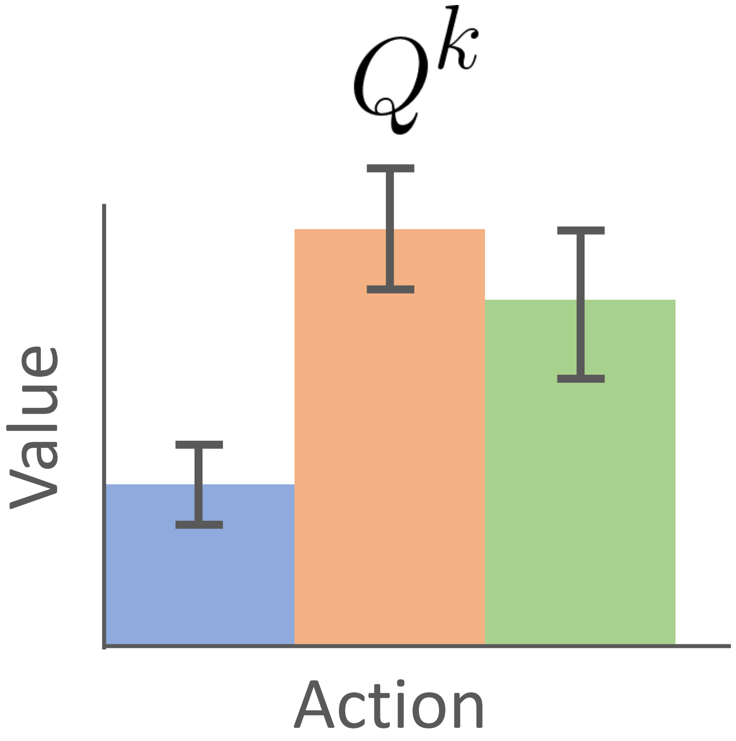
Double Q-Learning



Deep Reinforcement Learning with Double Q-learning
[van Hasselt et al. 2016]

Pessimistic Estimate

- Source of overestimation is model error
- Can we estimate model uncertainty for Q-function?

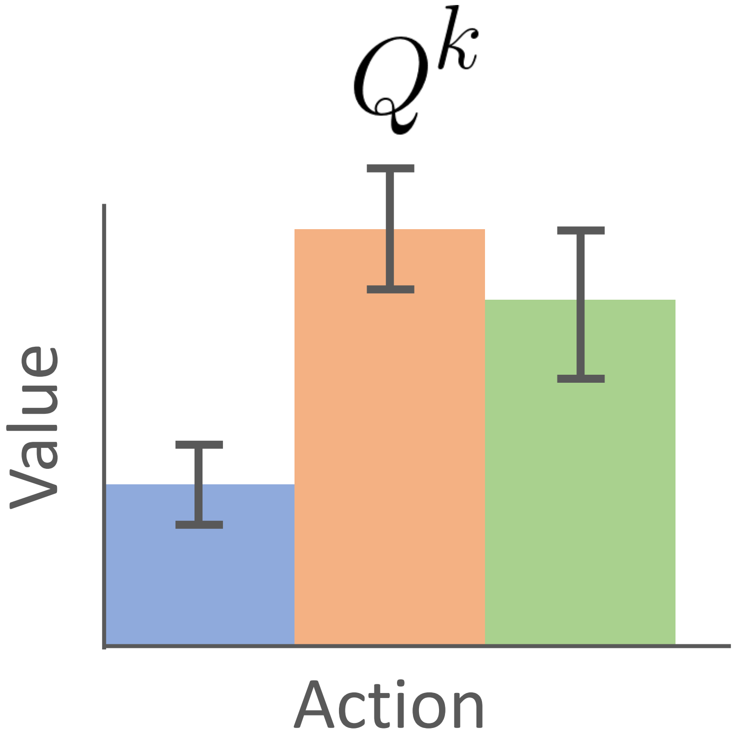


Pessimistic Estimate

- Source of overestimation is model error
- Can we estimate model uncertainty for Q-function?

Error: $[-\epsilon, \epsilon]$

$$y_i = r_i + \gamma \left(\max_{\mathbf{a}'} \; Q^k(\mathbf{s}', \mathbf{a}') - \underbrace{\epsilon}_{\text{lower bound}} \right)$$

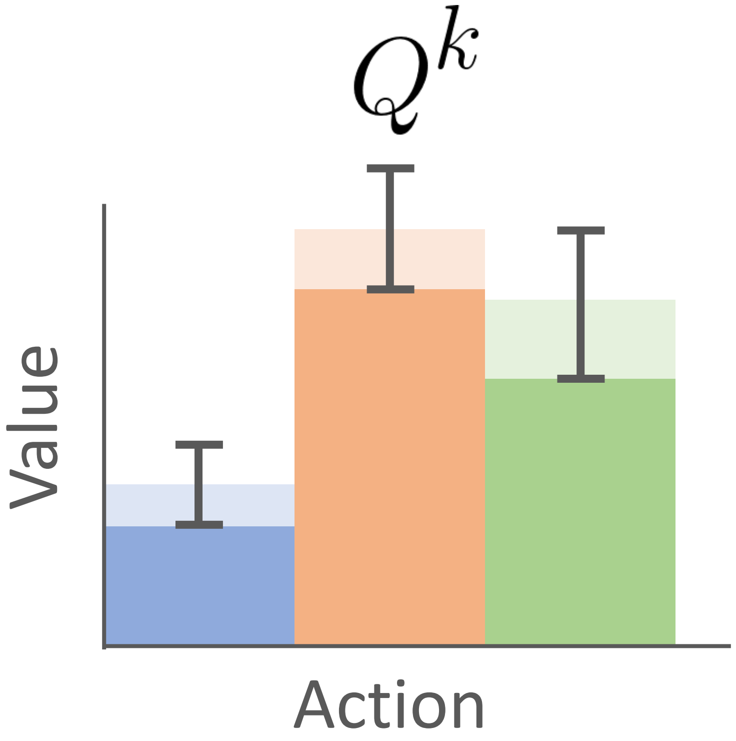


Pessimistic Estimate

- Source of overestimation is model error
- Can we estimate model uncertainty for Q-function?

Error: $[-\epsilon, \epsilon]$

$$y_i = r_i + \gamma\left(\max_{\mathbf{a}'} Q^k(\mathbf{s}',\mathbf{a}') - \underbrace{\epsilon}_{\text{lower bound}}\right)$$



Ensemble

- Estimate model uncertainty with an ensemble $\{Q_1, Q_2, ...\}$

$$y_i = r_i + \gamma \max_{\mathbf{a}'} \underline{\min_{j}}\, Q_j(\mathbf{s}', \mathbf{a}')$$

pessimistic value estimate

Maxmin Q-learning: Controlling the Estimation Bias of Q-learning
[Lan et al. 2020]

REDQ

- Compute minimum over a random subset of the ensemble $\mathcal{M} \subseteq \{Q_1, Q_2, ...\}$

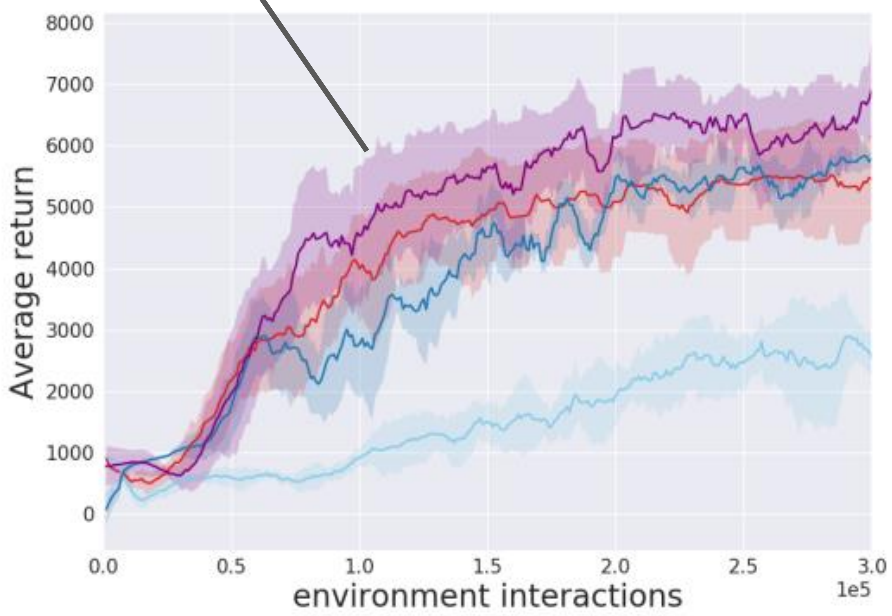
$$y_i = r_i + \gamma \max_{\mathbf{a}'} \min_{j \in \mathcal{M}} Q_j(\mathbf{s}', \mathbf{a}')$$

- in practice, randomly sampling 2 Q-functions work well

Randomized Ensembled Double Q-Learning: Learning Fast Without a Model
[Chen et al. 2021]

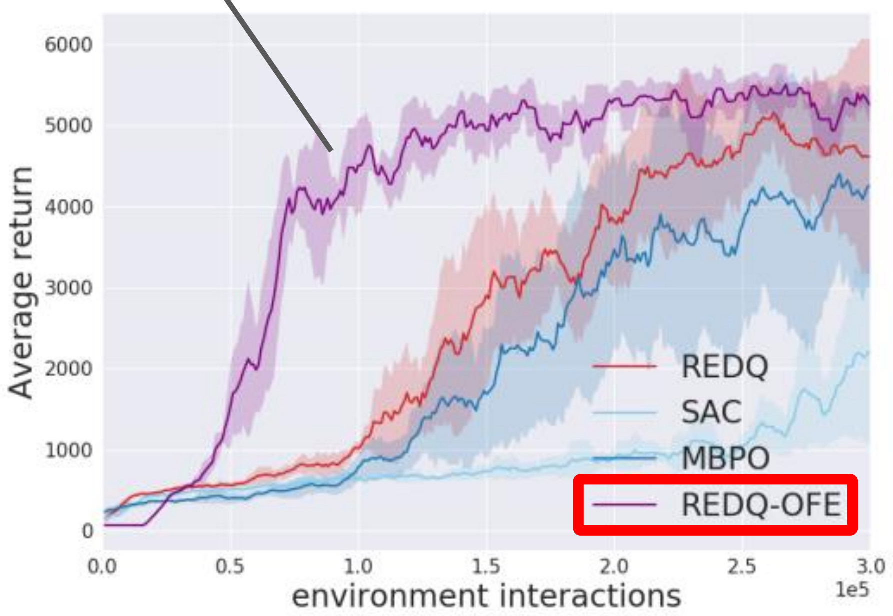
REDQ

REDQ



(a) Performance, Ant

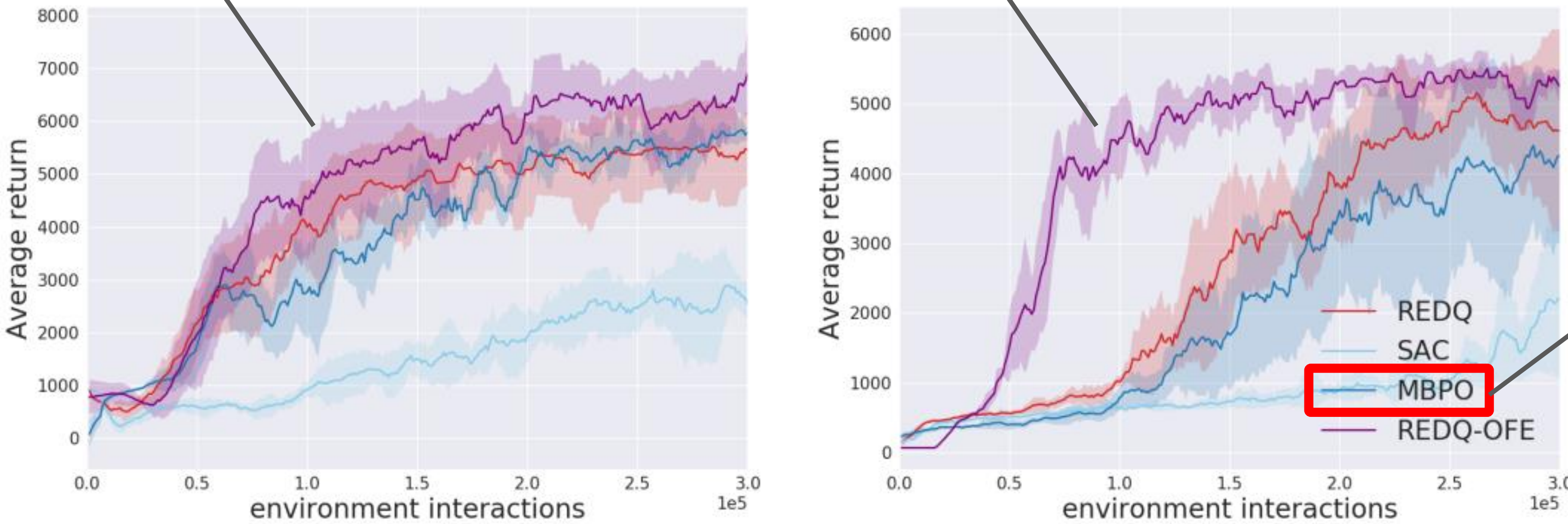
REDQ



(b) Performance, Humanoid

Randomized Ensembled Double Q-Learning: Learning Fast Without a Model
[Chen et al. 2021]

REDQ



REDQ

REDQ

Model-Based RL

(a) Performance, Ant

(b) Performance, Humanoid

Randomized Ensembled Double Q-Learning: Learning Fast Without a Model
[Chen et al. 2021]

REDQ

- Compute minimum over a random subset of the ensemble $\mathcal{M} \subseteq \{Q_1, Q_2, ...\}$

$$y_i = r_i + \gamma \max_{\mathbf{a}'} \min_{j \in \mathcal{M}} Q_j(\mathbf{s}', \mathbf{a}')$$

- in practice, randomly sampling 2 Q-functions work well

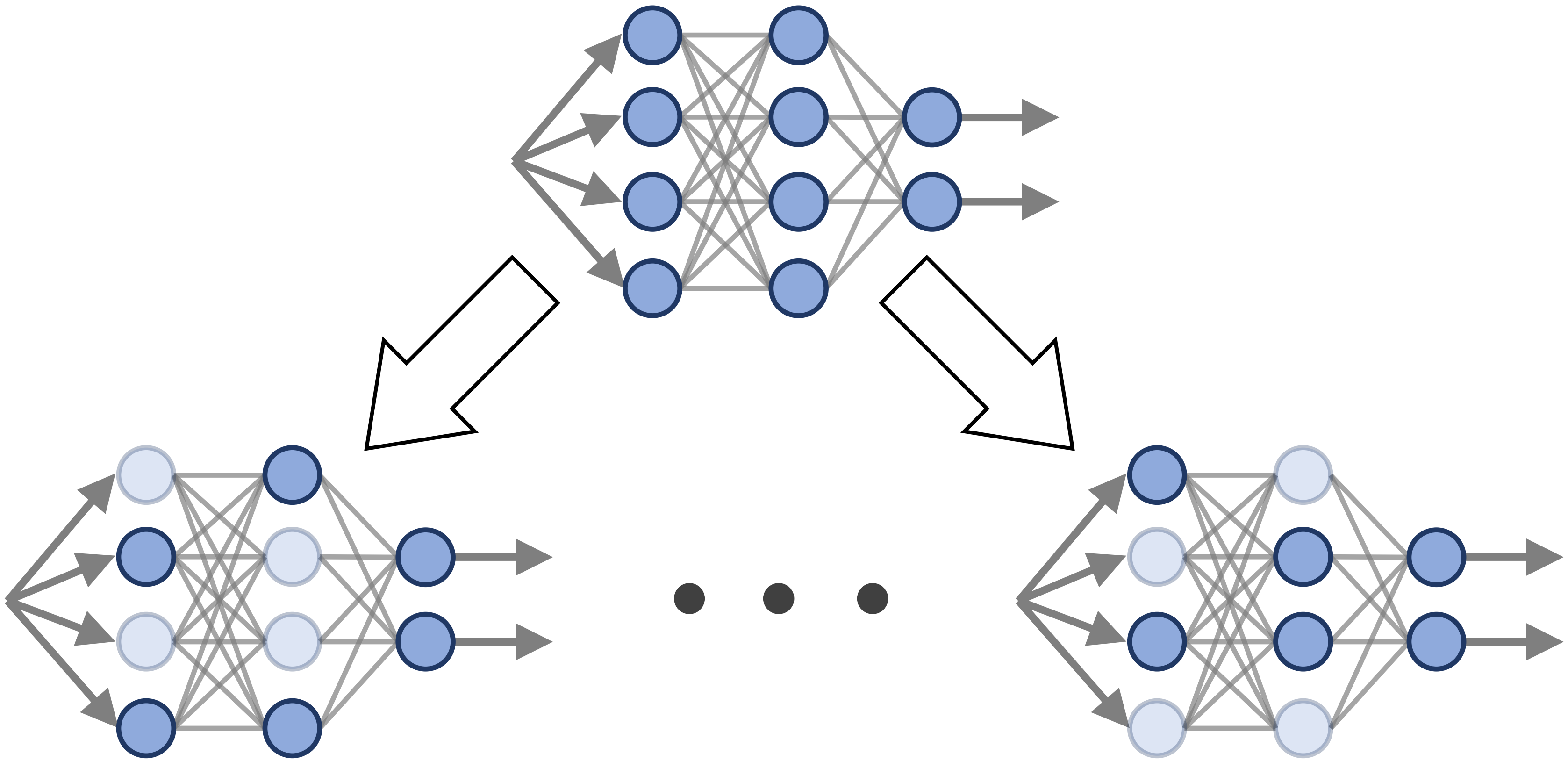
Drawback:

- Need to train multiple Q-functions

Randomized Ensembled Double Q-Learning: Learning Fast Without a Model
[Chen et al. 2021]

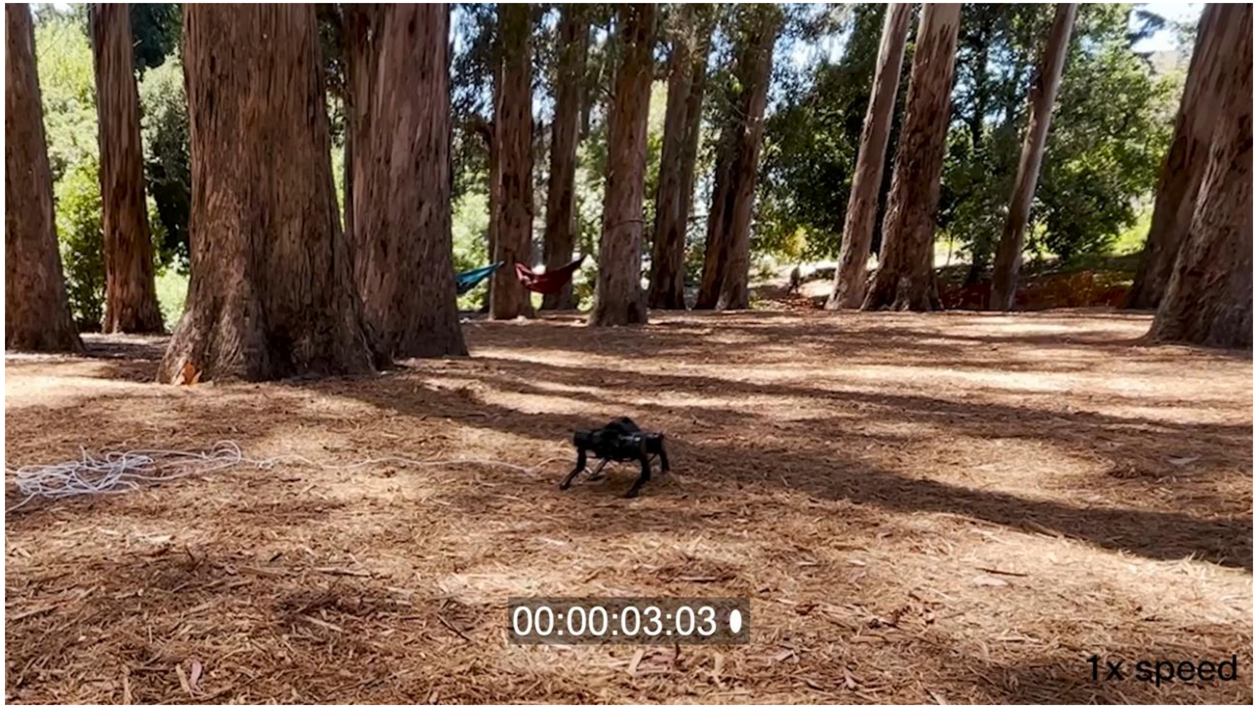
DroQ

- Instead of training an ensemble, emulate an ensemble using Dropout



Dropout Q-Functions for Doubly Efficient Reinforcement Learning
[Hiraoka et al. 2022]

DroQ

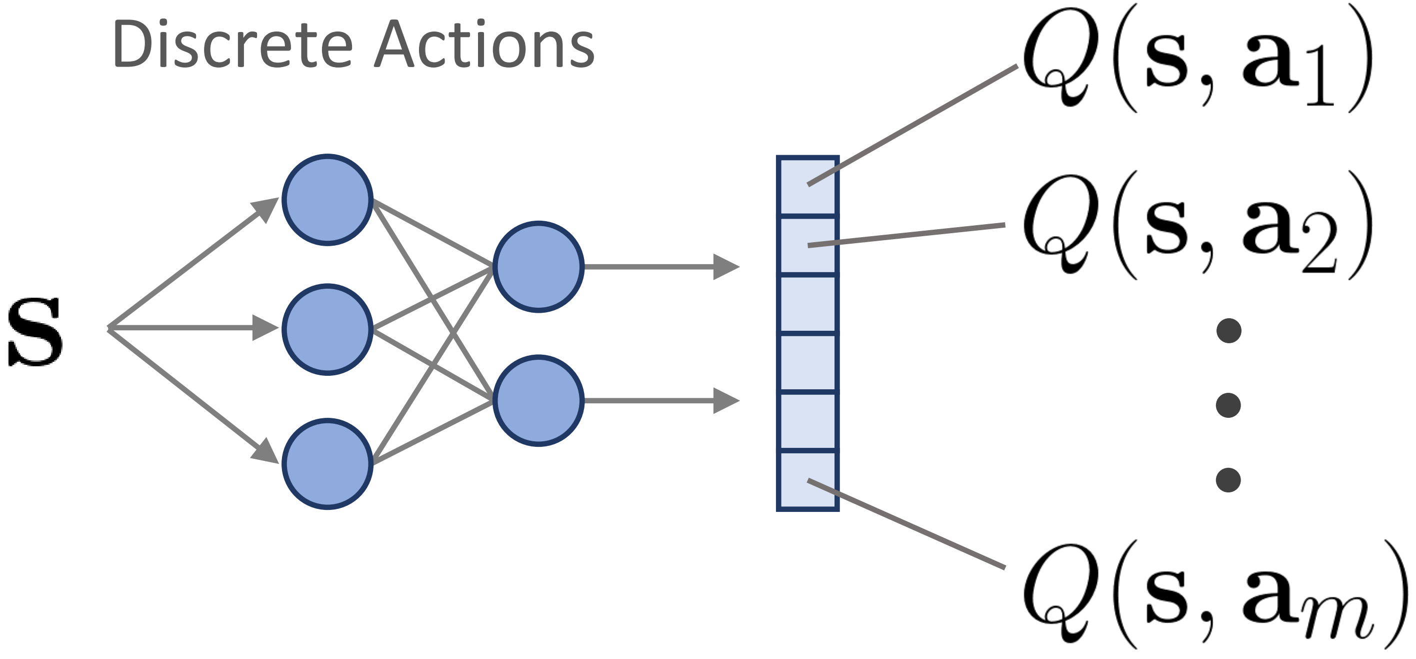


A Walk in the Park: Learning to Walk in 20 Minutes With Model-Free Reinforcement Learning
[Smith et al. 2022]

Overview

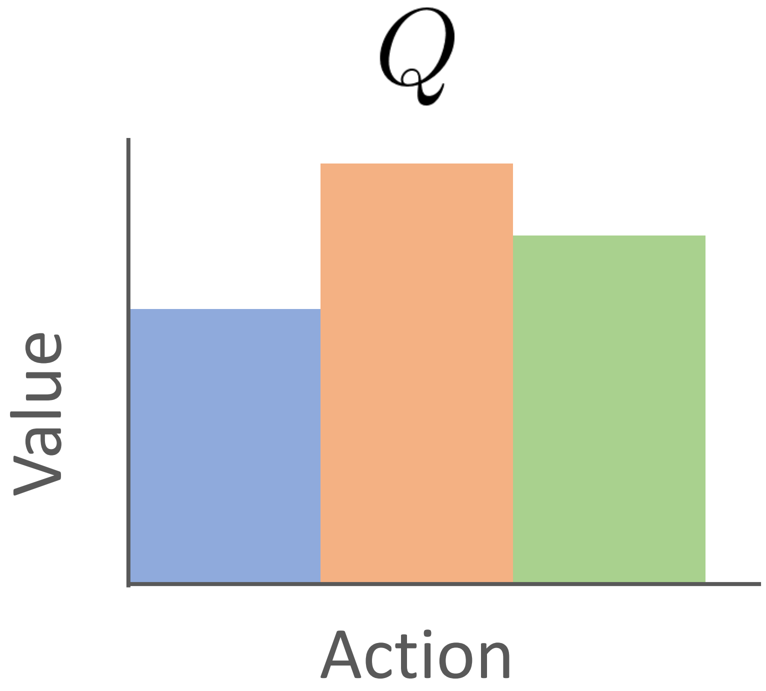
- Nonstationary Targets → Target Networks
- Overestimation → Pessimistic Estimates
- Model Architecture

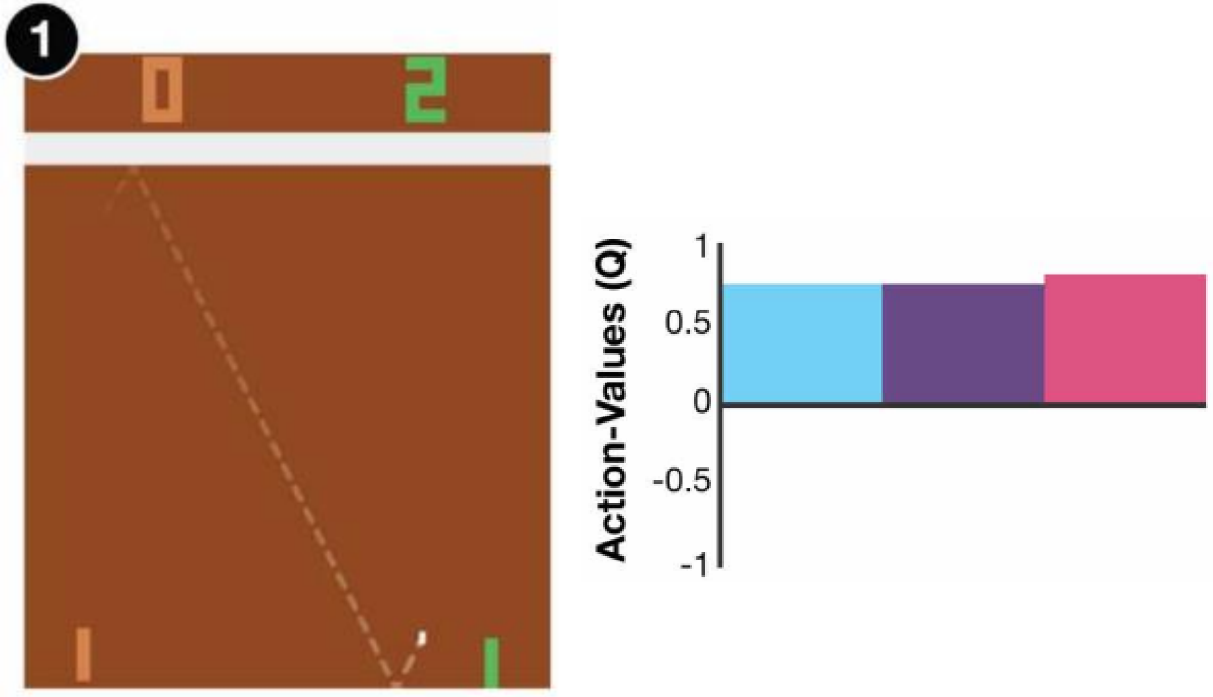
Model Architecture


Discrete Actions
$\mathbf{s}$
$Q(\mathbf{s}, \mathbf{a}_1)$
$Q(\mathbf{s}, \mathbf{a}_2)$
$Q(\mathbf{s}, \mathbf{a}_m)$

Model Architecture

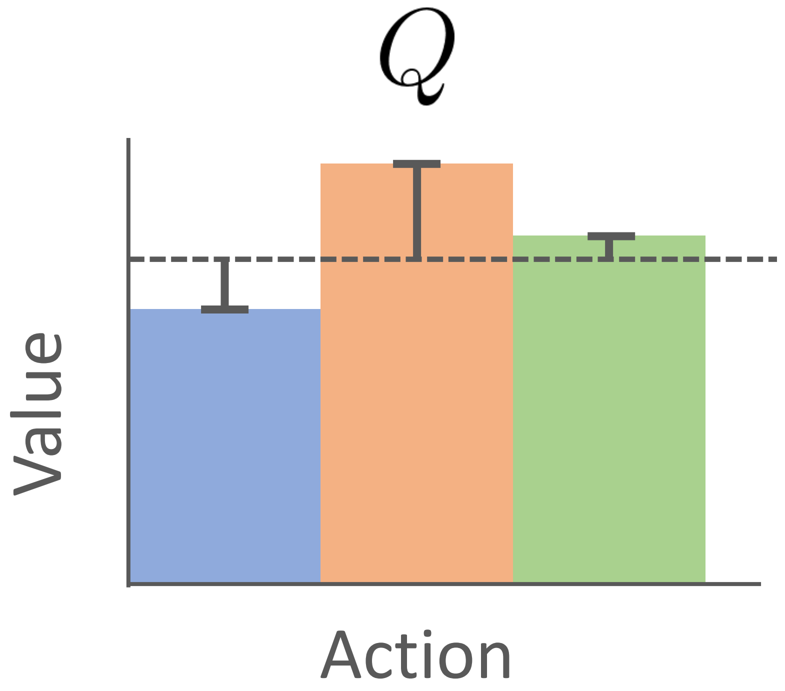
- Q-values at a particular state often do not vary that much





Model Architecture

- Q-values at a particular state often do not vary that much
- Only the *relative* values are needed to select actions



Model Architecture

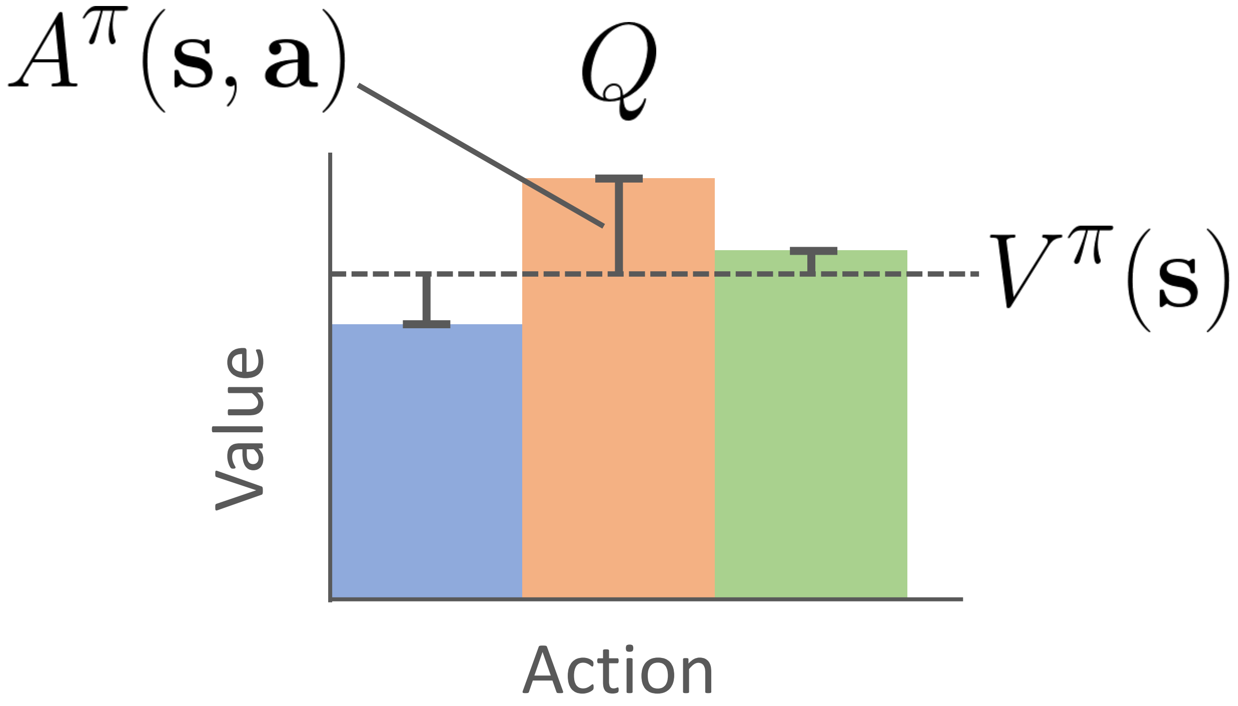
$$A^{\pi}(\mathbf{s},\mathbf{a}) = Q^{\pi}(\mathbf{s},\mathbf{a}) - \underline{V^{\pi}(\mathbf{s})}$$

Model Architecture

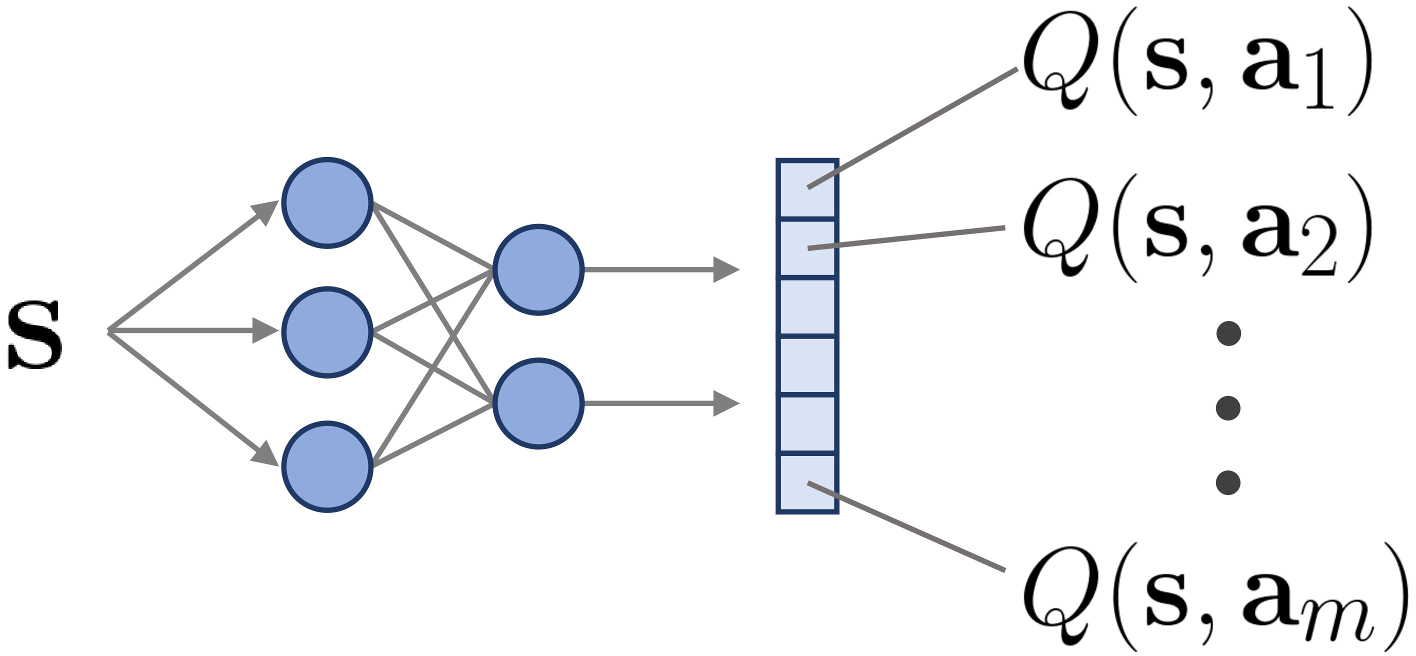
$$Q^{\pi}(\mathbf{s},\mathbf{a}) = \underbrace{V^{\pi}(\mathbf{s})}_{\substack{\text{Action-independent}\\ \text{value function}}} + \underbrace{A^{\pi}(\mathbf{s},\mathbf{a})}_{\substack{\text{Action-dependent}\\ \text{advantage function}}}$$

Model Architecture

$$Q^{\pi}(\mathbf{s}, \mathbf{a}) = \underline{V^{\pi}(\mathbf{s})} + \underline{A^{\pi}(\mathbf{s}, \mathbf{a})}$$

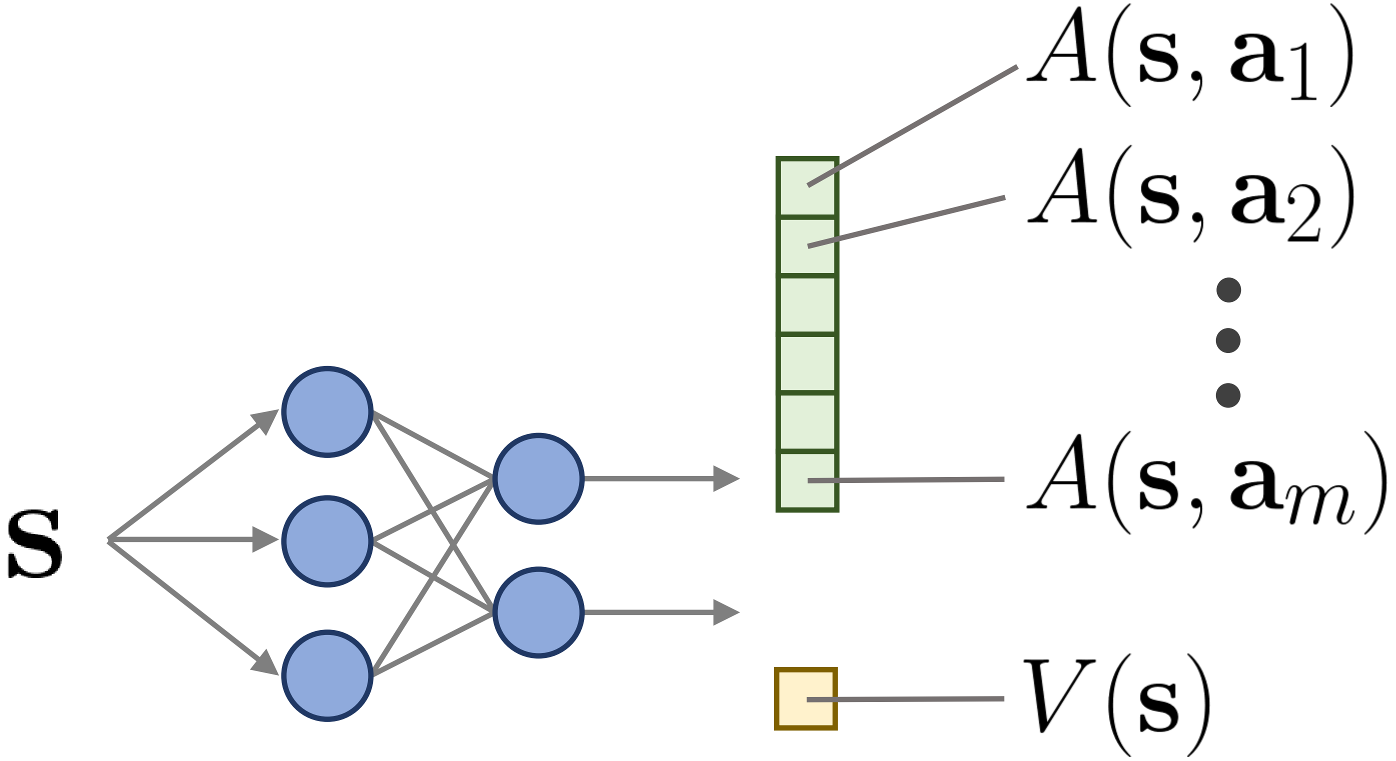


Dueling Q-Networks


$\mathbf{s}$
$Q(\mathbf{s}, \mathbf{a}_1)$
$Q(\mathbf{s}, \mathbf{a}_2)$
$Q(\mathbf{s}, \mathbf{a}_m)$

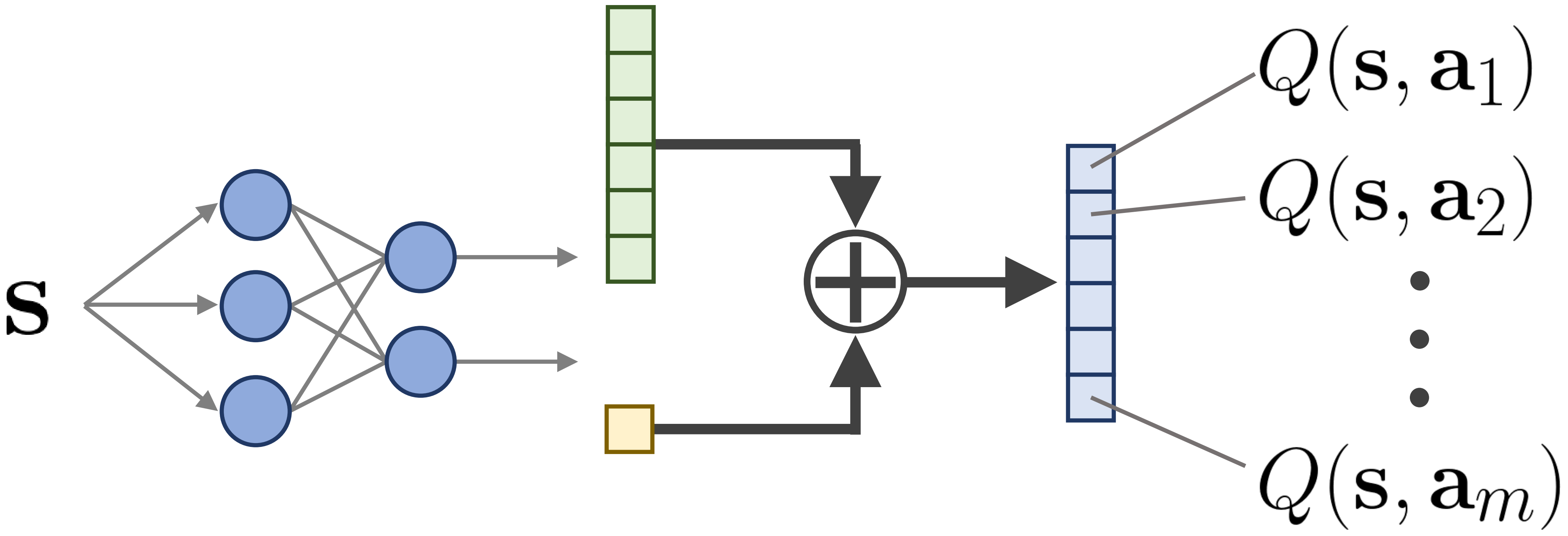
Dueling Network Architectures for Deep Reinforcement Learning
[Wang et al. 2016]

Dueling Q-Networks


$\mathbf{s}$
$A(\mathbf{s}, \mathbf{a}_1)$
$A(\mathbf{s}, \mathbf{a}_2)$
$A(\mathbf{s}, \mathbf{a}_m)$
$V(\mathbf{s})$

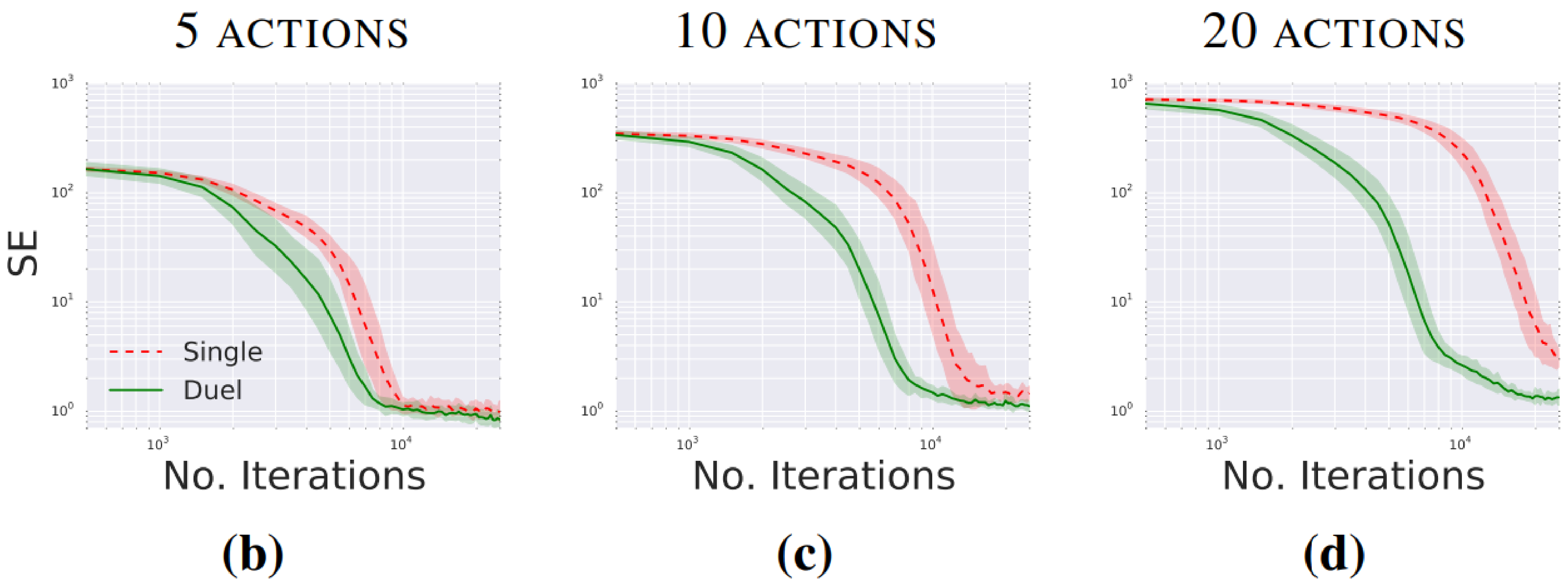
Dueling Network Architectures for Deep Reinforcement Learning
[Wang et al. 2016]

Dueling Q-Networks


$\mathbf{s}$
$Q(\mathbf{s}, \mathbf{a}_1)$
$Q(\mathbf{s}, \mathbf{a}_2)$
$Q(\mathbf{s}, \mathbf{a}_m)$

Dueling Network Architectures for Deep Reinforcement Learning
[Wang et al. 2016]

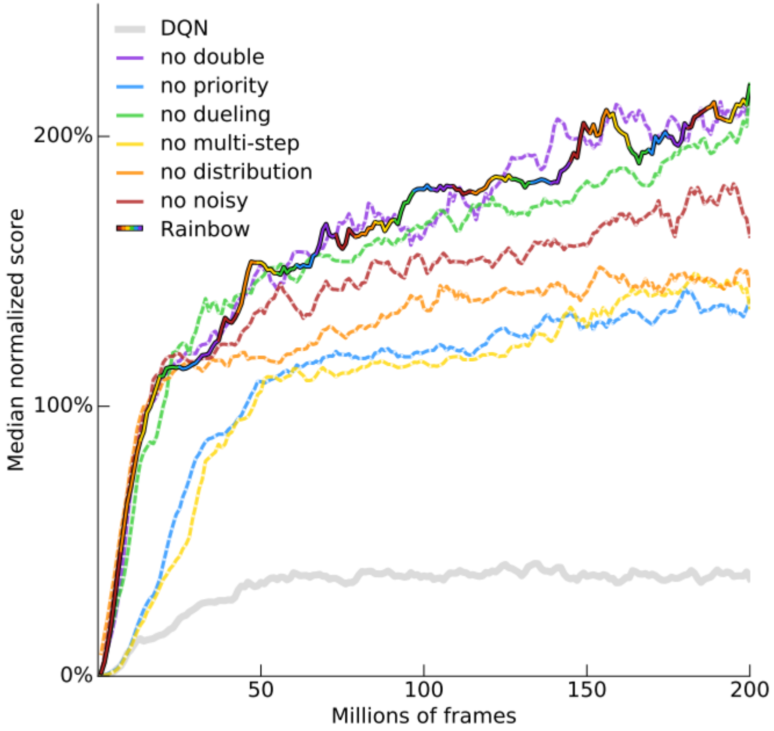
Dueling Q-Networks



Dueling Network Architectures for Deep Reinforcement Learning
[Wang et al. 2016]

Lots of Tricks

- Prioritized Replay
- Multi-Step Returns
- Distributional RL
- Noisy Nets
- Etc…



Note: techniques for improving Q-Learning can also be applied to other algorithms (e.g. DDPG, SAC, TD3, MPO, etc.)

Rainbow: Combining Improvements in Deep Reinforcement Learning
[Hessel et al. 2018]

Summary

- Target Networks
- Overestimation
- Model Architecture