TD3

Course Structure

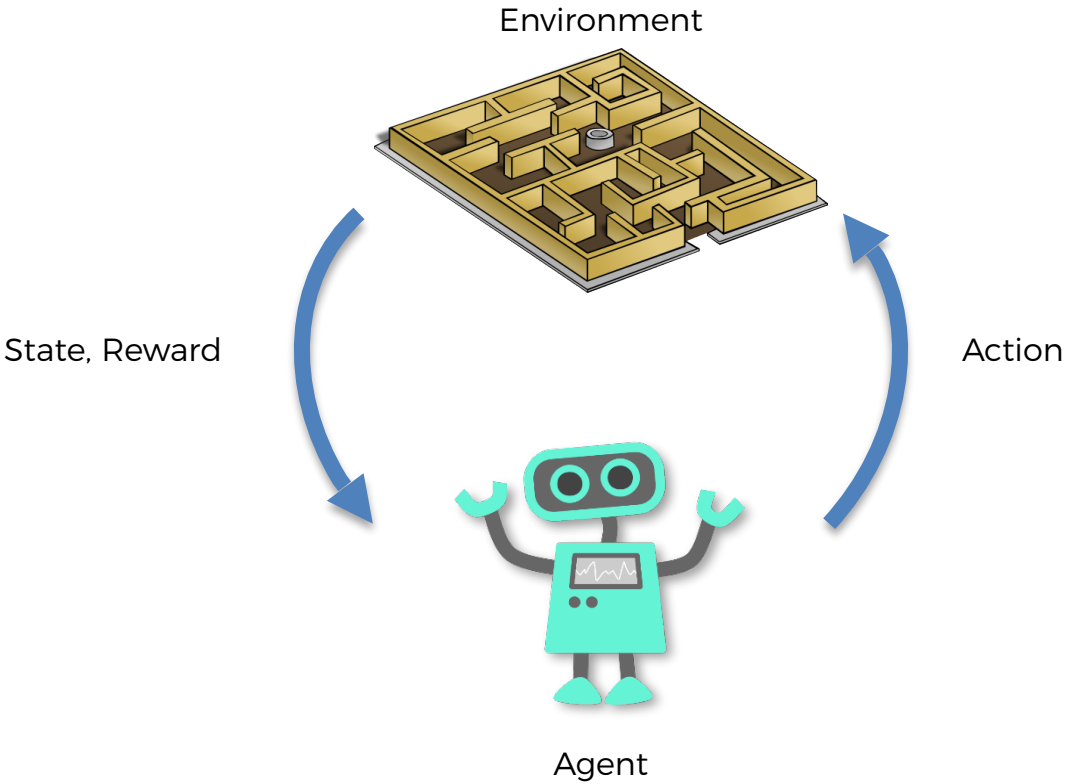
Part 1: Fundamentals

Part 2: Twin Delayed DDPG (TD3) - Theory

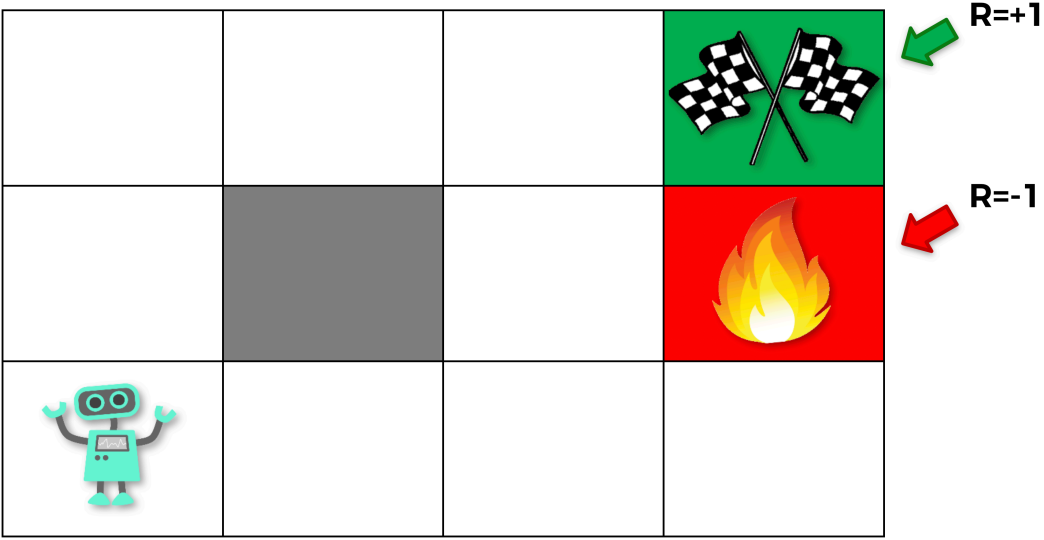
Part 3: Twin Delayed DDPG (TD3) - Implementation

Part 1
Fundamentals

Fundamentals: Reinforcement Learning



Fundamentals: Q-Learning



Initialization (First iteration):

For all couples of states s and actions a, the Q-values are initialized to 0.

Next iterations:

At each iteration t ≥ 1, we repeat the following steps:

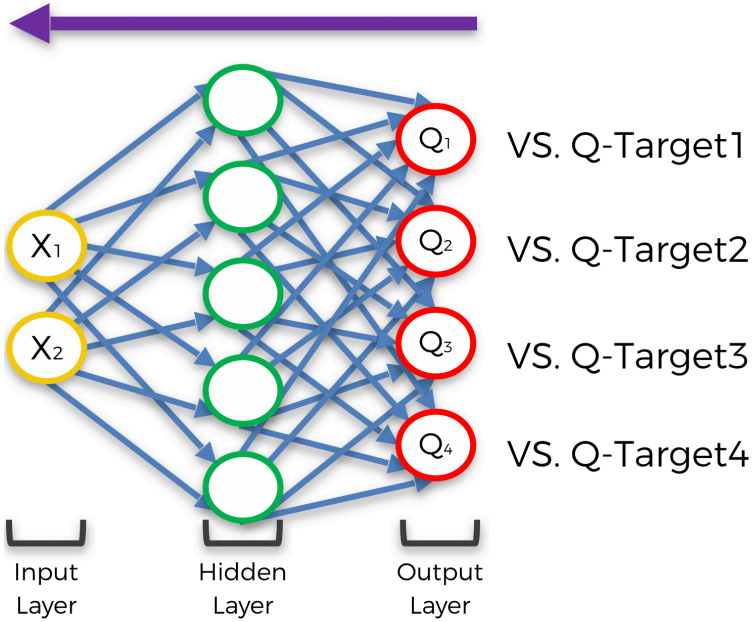
1. We select a random state s_t from the possible states.
2. From that state, we play a random action a_t.
3. We reach the next state s_{t+1} and we get the reward $R(s_t, a_t)$.
4. We compute the Temporal Difference $TD_t(s_t, a_t)$:

$$TD_t(s_t, a_t) = R(s_t, a_t) + \gamma \max_a(Q(s_{t+1}, a)) - Q(s_t, a_t)$$

5. We update the Q-value by applying the Bellman equation:

$$Q_t(s_t, a_t) = Q_{t-1}(s_t, a_t) + \alpha TD_t(s_t, a_t)$$

Fundamentals: Deep Q-Learning



$$L = \sum (Q\text{–}Target\ - Q)^2$$

Initialization:

1. The memory of the Experience Replay is initialized to an empty list M.
2. We choose a maximum size of the memory.

At each time t, we repeat the following process, until the end of the epoch:

1. We predict the Q-values of the current state s_t.
2. We play the action that has the highest Q-value: $a_t = \underset{a}{\text{argmax}}\{Q(s_t, a)\}$
3. We get the reward $R(s_t, a_t)$.
4. We reach the next state s_{t+1}.
5. We append the transition (s_t, a_t, r_t, s_{t+1}) in the memory M.
6. We take a random batch $B \subset M$ of transitions. For all the transitions $(s_{t_B}, a_{t_B}, r_{t_B}, s_{t_B+1})$ of the random batch B:

   We get the predictions: $Q(s_{t_B}, a_{t_B})$

   We get the targets: $R(s_{t_B}, a_{t_B}) + \gamma \underset{a}{\max}(Q(s_{t_B+1}, a))$

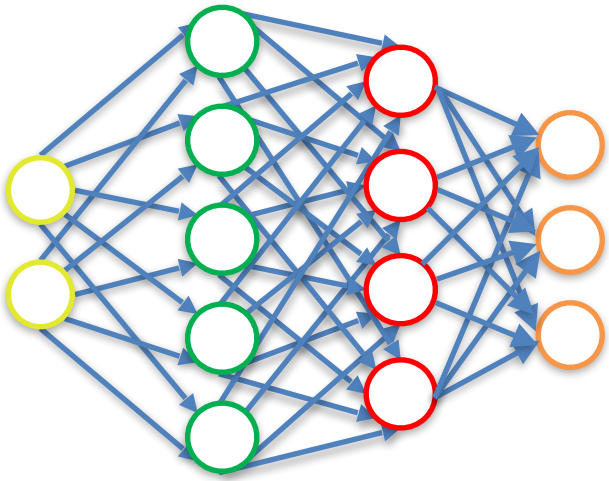
   We compute the loss between the predictions and the targets over the whole batch B:

$$\text{Loss} = \frac{1}{2}\sum_B \left(R(s_{t_B}, a_{t_B}) + \gamma \underset{a}{\max}(Q(s_{t_B+1}, a)) - Q(s_{t_B}, a_{t_B})\right)^2 = \frac{1}{2}\sum_B TD_{t_B}(s_{t_B}, a_{t_B})^2$$

   We backpropagate this loss error back into the neural network, and through stochastic gradient descent we update the weights according to how much they contributed to the loss error.

Fundamentals: Policy Gradient

States



Actions

Policy π_ϕ

Return: $R_t = \sum_{i=t}^{T} \gamma^{i-t} r(s_i, a_i)$

Goal: Maximize the expected return $J(\phi) = \mathbb{E}_{s_i \sim p_\pi, a_i \sim \pi}[R_0]$

Policy Gradient:

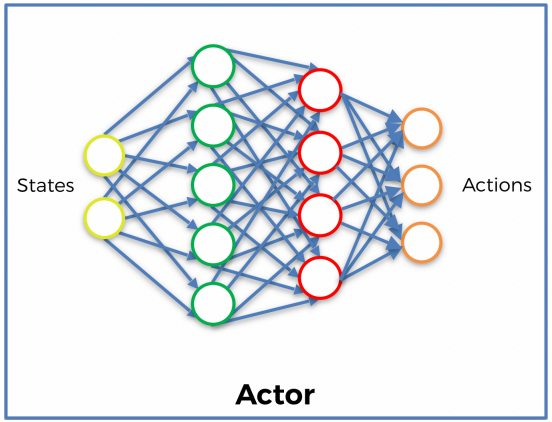
- We compute the gradient of the expected return with respect to the parameters ϕ

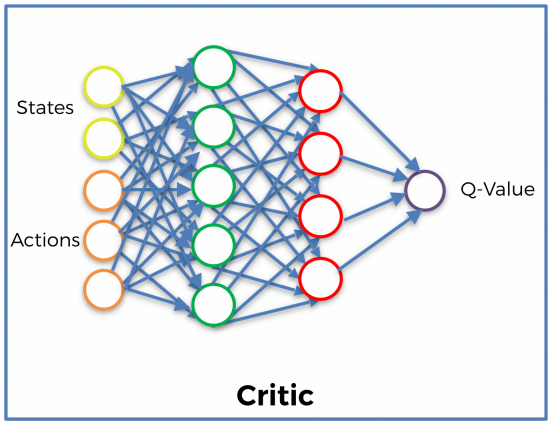
$$\nabla_\phi J(\phi)$$

- We update the parameters through gradient ascent:

$$\phi_{t+1} = \phi_t + \alpha \nabla_\phi J(\pi_\phi)|_{\phi_t}$$

Fundamentals: Actor-Critic





Return: $R_t = \sum_{i=t}^{T} \gamma^{i-t} r(s_i, a_i)$

Goal: Maximize the expected return $J(\phi) = \mathbb{E}_{s_i \sim p_\pi, a_i \sim \pi}[R_0]$

Deterministic Policy Gradient:

- In actor-critic methods, the policy, known as the actor, can be updated through the deterministic policy gradient algorithm:

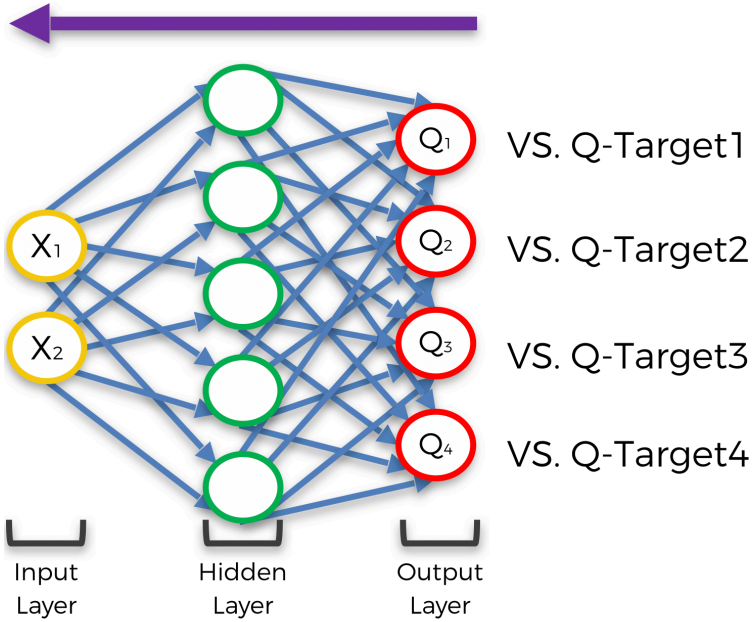
$$\nabla_\phi J(\phi) = \mathbb{E}_{s \sim p_\pi}\left[\nabla_a Q^\pi(s,a)|_{a=\pi(s)} \nabla_\phi \pi_\phi(s)\right]$$

- We update the policy parameters through gradient ascent:

$$\phi_{t+1} = \phi_t + \alpha \nabla_\phi J(\pi_\phi)|_{\phi_t}$$

Part 2
Twin Delayed DDPG (TD3)
Theory

Deep Q-Learning



$$L = \sum (Q\text{–}Target\ - Q)^2$$

Initialization:

1. The memory of the Experience Replay is initialized to an empty list M.
2. We choose a maximum size of the memory.

At each time t, we repeat the following process, until the end of the epoch:

1. We predict the Q-values of the current state s_t.
2. We play the action that has the highest Q-value: $a_t = \underset{a}{\text{argmax}}\{Q(s_t, a)\}$
3. We get the reward $R(s_t, a_t)$.
4. We reach the next state s_{t+1}.
5. We append the transition (s_t, a_t, r_t, s_{t+1}) in the memory M.
6. We take a random batch $B \subset M$ of transitions. For all the transitions $(s_{t_B}, a_{t_B}, r_{t_B}, s_{t_B+1})$ of the random batch B:

   We get the predictions: $Q(s_{t_B}, a_{t_B})$

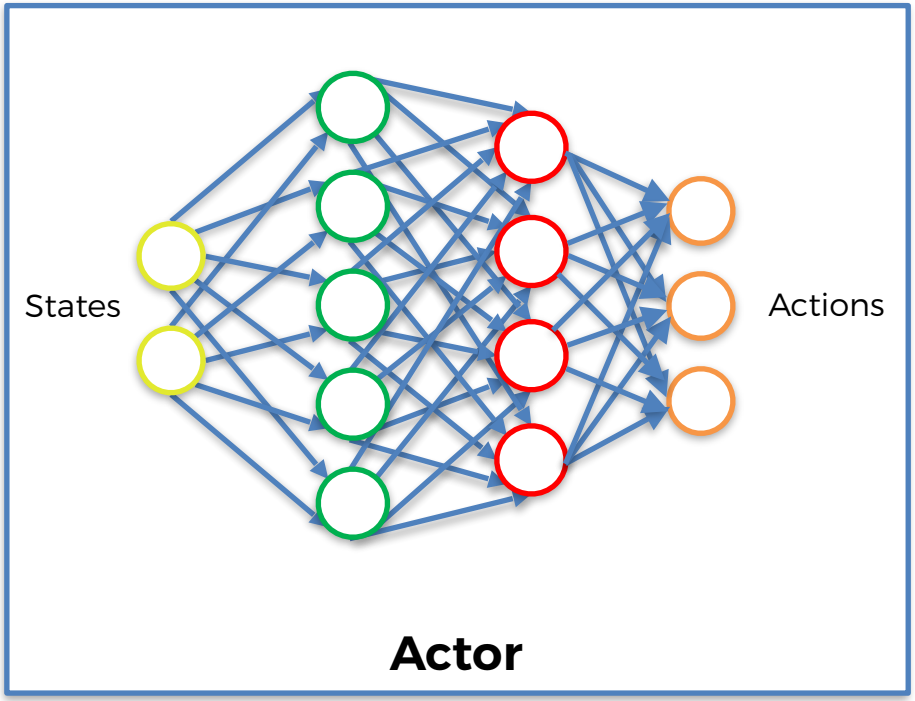
   We get the targets: $R(s_{t_B}, a_{t_B}) + \gamma \underset{a}{\max}(Q(s_{t_B+1}, a))$

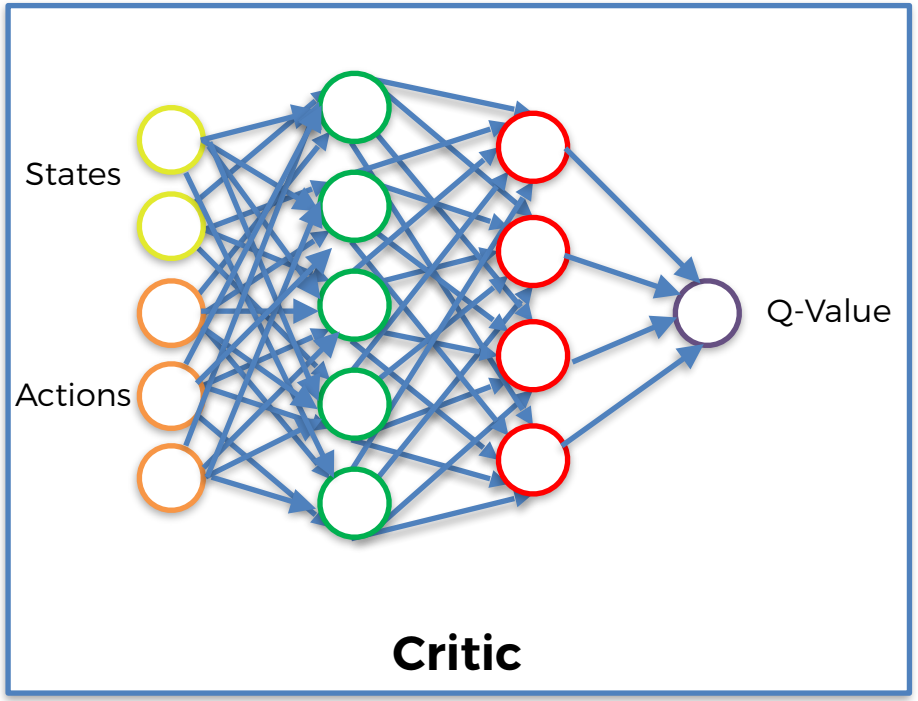
   We compute the loss between the predictions and the targets over the whole batch B:

$$\text{Loss} = \frac{1}{2}\sum_B \left(R(s_{t_B}, a_{t_B}) + \gamma \underset{a}{\max}(Q(s_{t_B+1}, a)) - Q(s_{t_B}, a_{t_B})\right)^2 = \frac{1}{2}\sum_B TD_{t_B}(s_{t_B}, a_{t_B})^2$$

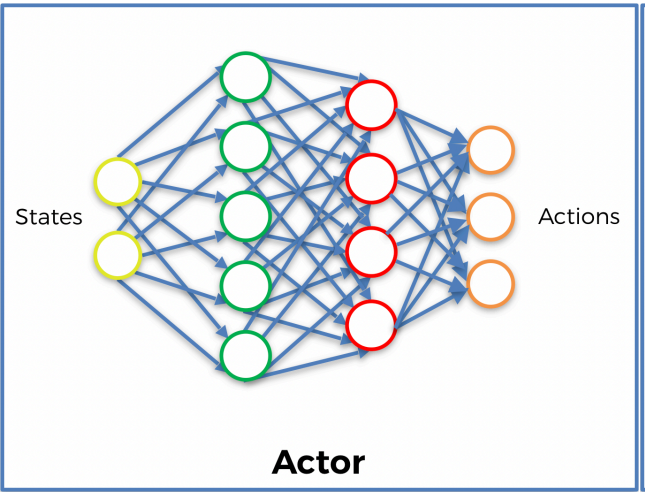
   We backpropagate this loss error back into the neural network, and through stochastic gradient descent we update the weights according to how much they contributed to the loss error.

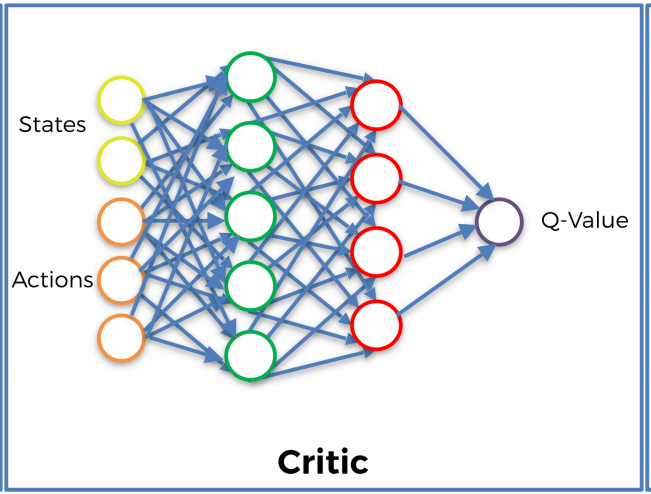
Actor-Critic

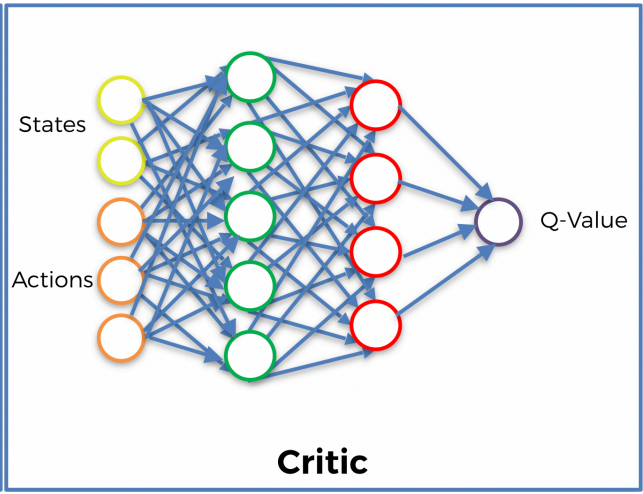




Twin-Delayed DDPG (TD3)







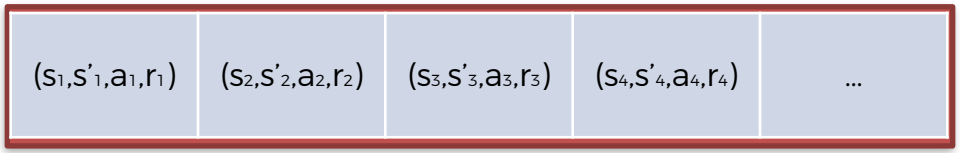
Twin-Delayed DDPG (TD3)

Twin-Delayed DDPG (TD3)

Initialization:

Step 1: We initialize the Experience Replay memory, with a size of 20000. We will populate it with each new transition.

Twin-Delayed DDPG (TD3)



Experience Replay Memory

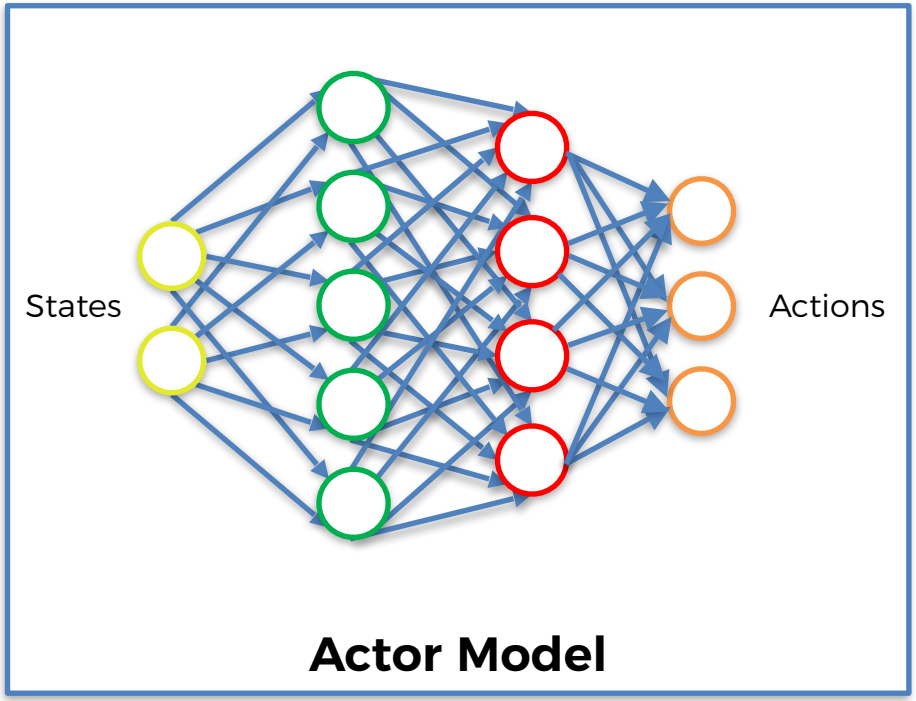
Twin-Delayed DDPG (TD3)

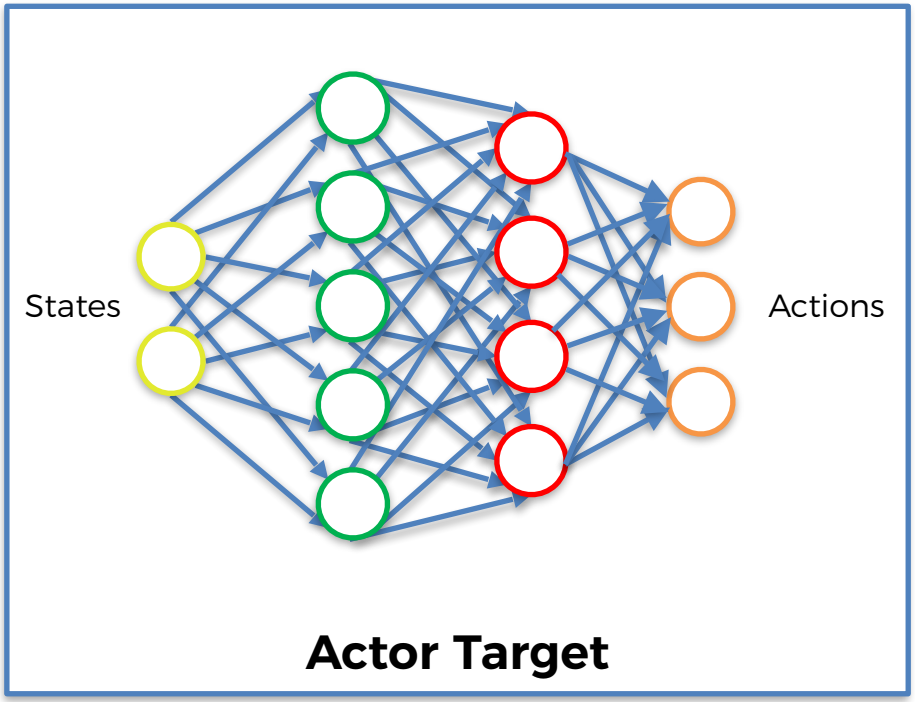
Initialization:

Step 1: We initialize the Experience Replay memory, with a size of 20000. We will populate it with each new transition.

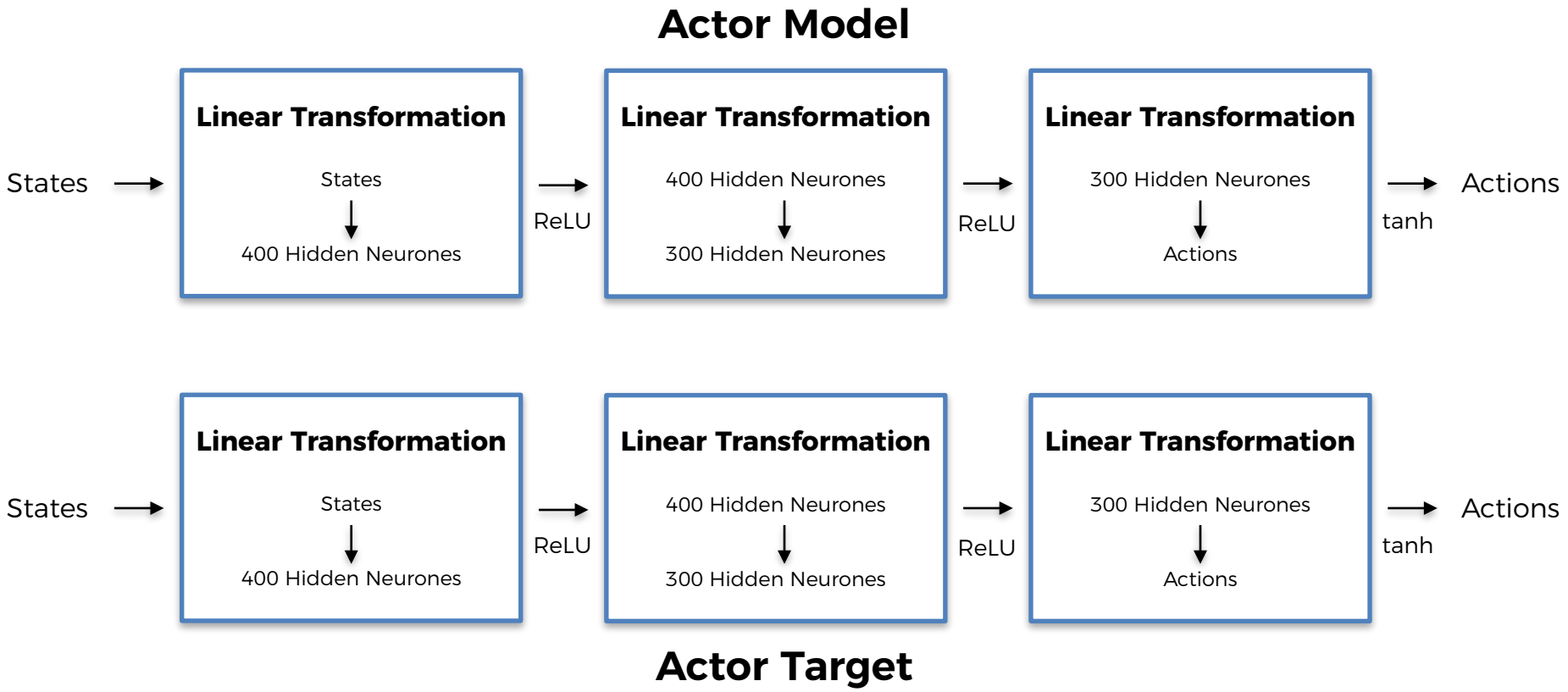
Step 2: We build one neural network for the Actor model and one neural network for the Actor target.

Twin-Delayed DDPG (TD3)





Twin-Delayed DDPG (TD3)

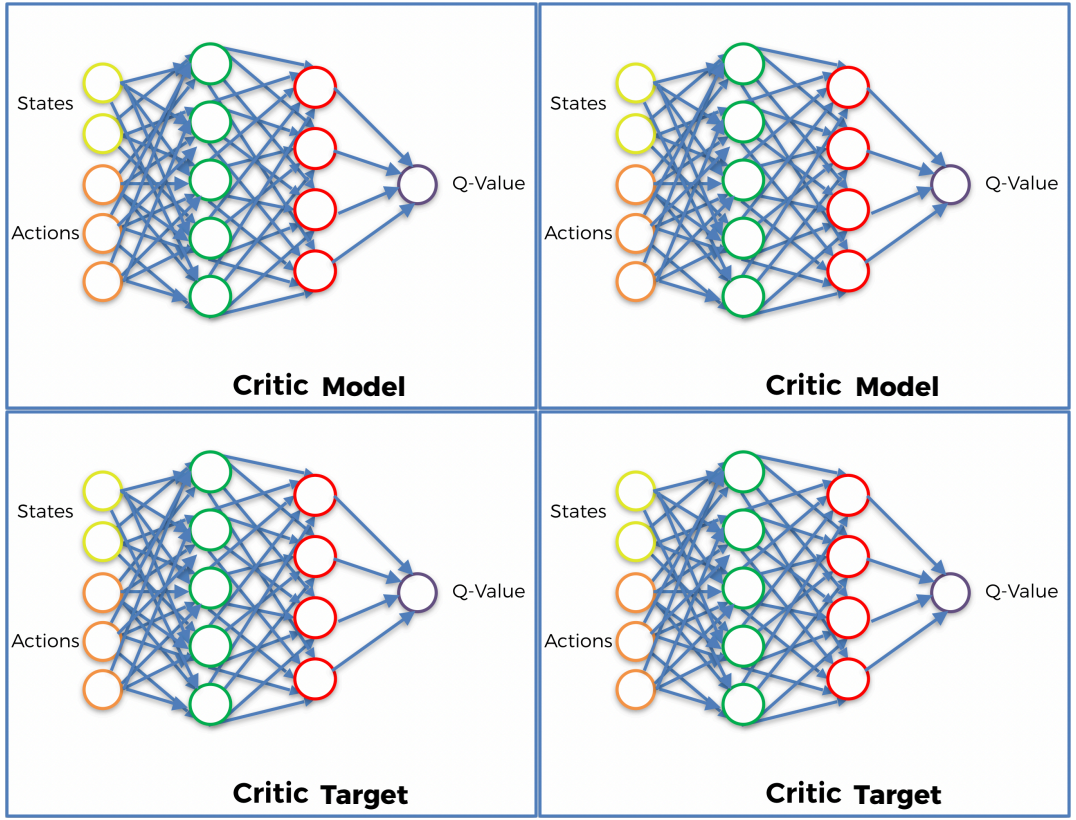


Twin-Delayed DDPG (TD3)

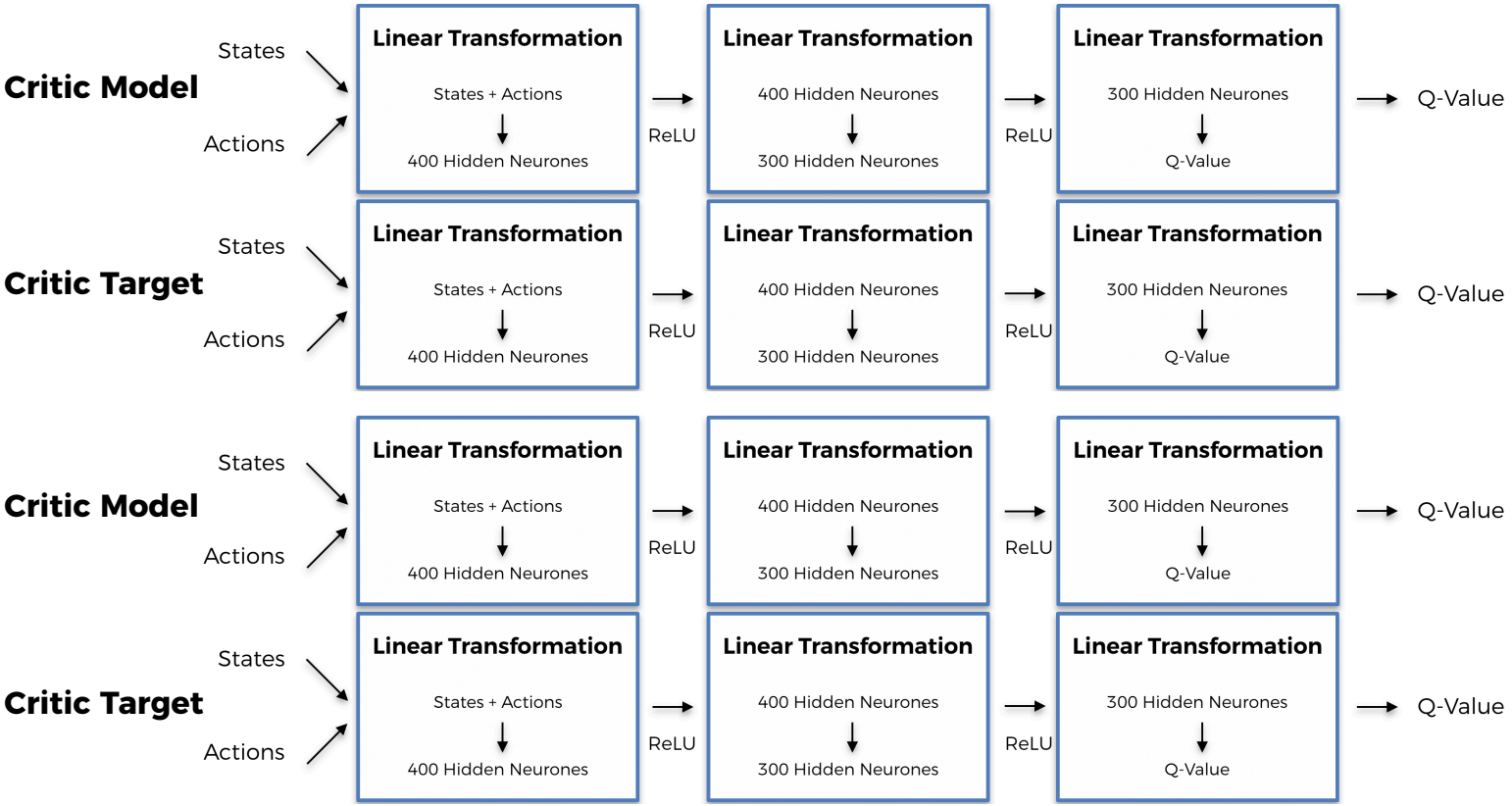
Initialization:

Step 1: We initialize the Experience Replay memory, with a size of 20000. We will populate it with each new transition.
Step 2: We build one neural network for the Actor model and one neural network for the Actor target.
Step 3: We build two neural networks for the two Critic models and two neural networks for the two Critic targets.

Twin-Delayed DDPG (TD3)



Twin-Delayed DDPG (TD3)

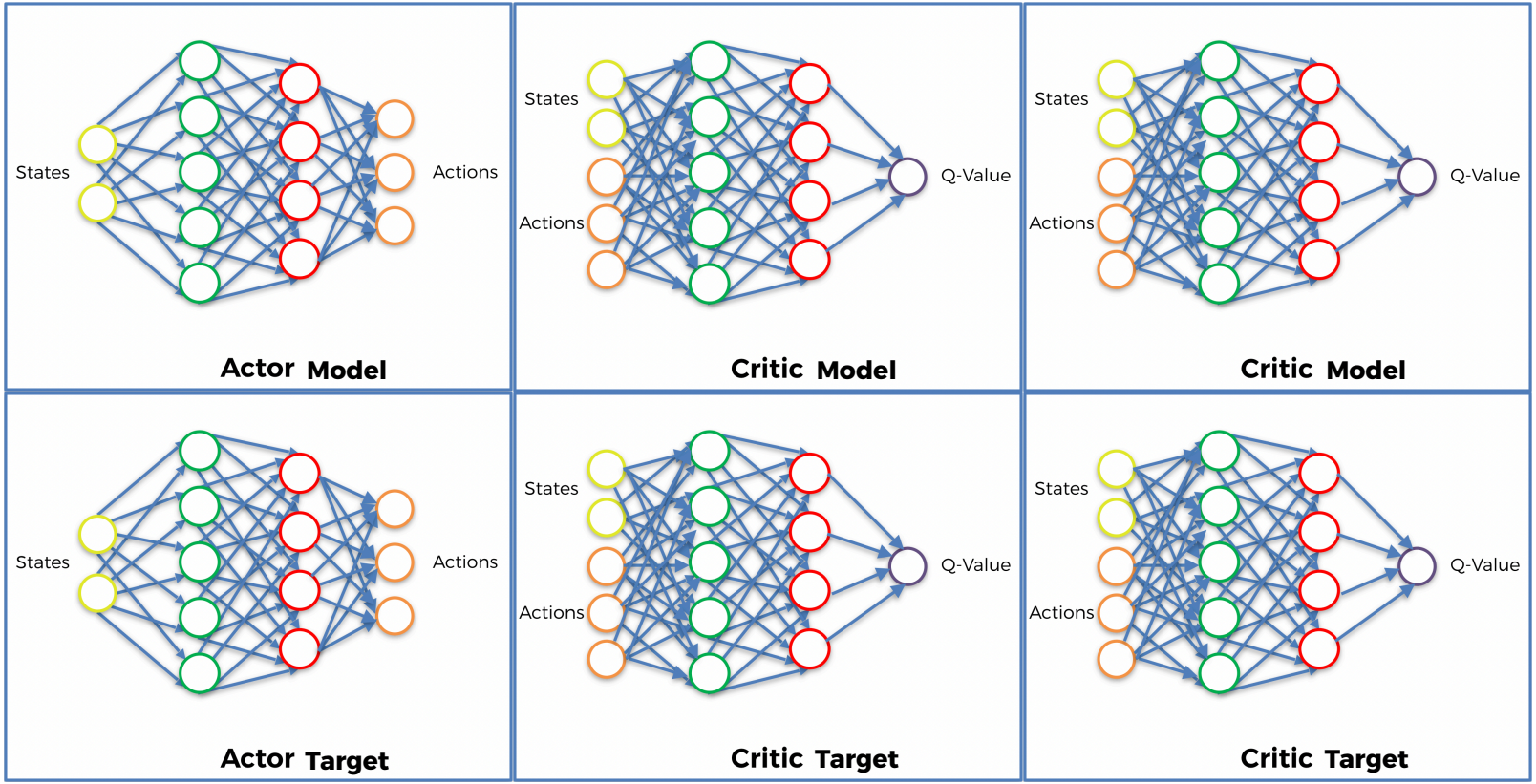


Twin-Delayed DDPG (TD3)

Initialization:

Step 1: We initialize the Experience Replay memory, with a size of 20000. We will populate it with each new transition.
Step 2: We build one neural network for the Actor model and one neural network for the Actor target.
Step 3: We build two neural networks for the two Critic models and two neural networks for the two Critic targets.

Twin-Delayed DDPG (TD3)



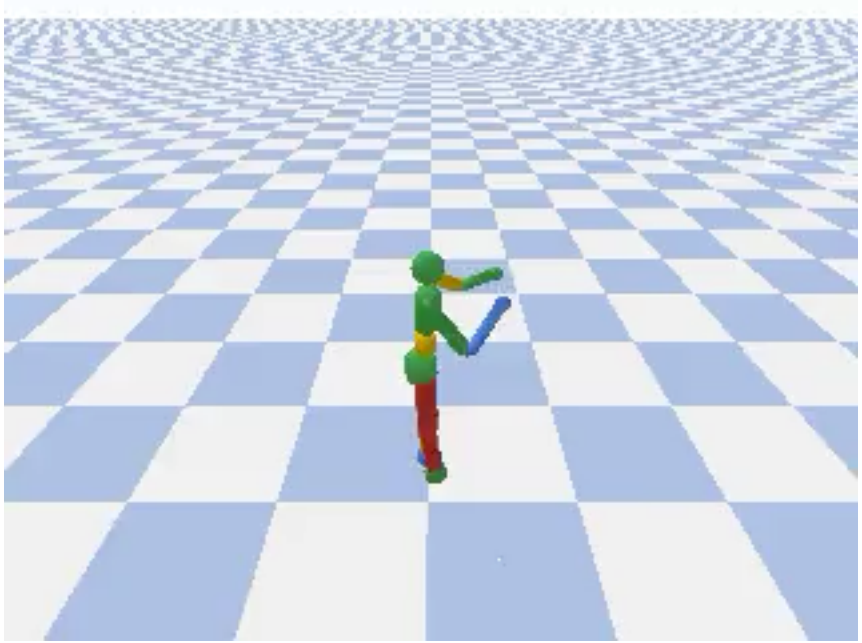
Twin-Delayed DDPG (TD3)

Initialization:

Step 1: We initialize the Experience Replay memory, with a size of 20000. We will populate it with each new transition.
Step 2: We build one neural network for the Actor model and one neural network for the Actor target.
Step 3: We build two neural networks for the two Critic models and two neural networks for the two Critic targets.

Training Process - We run a full episode with first 10,000 actions played randomly, and then with actions played by the Actor model.

Twin-Delayed DDPG (TD3)



Twin-Delayed DDPG (TD3)

Initialization:

Step 1: We initialize the Experience Replay memory, with a size of 20000. We will populate it with each new transition.
Step 2: We build one neural network for the Actor model and one neural network for the Actor target.
Step 3: We build two neural networks for the two Critic models and two neural networks for the two Critic targets.

Training Process - We run a full episode with first 10,000 actions played randomly, and then with actions played by the Actor model. Then we repeat the following steps:

Twin-Delayed DDPG (TD3)

Initialization:

Step 1: We initialize the Experience Replay memory, with a size of 20000. We will populate it with each new transition.
Step 2: We build one neural network for the Actor model and one neural network for the Actor target.
Step 3: We build two neural networks for the two Critic models and two neural networks for the two Critic targets.

Training Process - We run a full episode with first 10,000 actions played randomly, and then with actions played by the Actor model. Then we repeat the following steps:

Step 4: We sample a batch of transitions (s, s', a, r) from the memory. Then for each element of the batch:

Twin-Delayed DDPG (TD3)

Last States	Next States	Actions	Rewards
Last State 1	Next state 1	Action 1	Reward 1
Last State 2	Next state 2	Action 2	Reward 2
...	...	...	...
Last State 100	Next state 100	Action 100	Reward 100

Twin-Delayed DDPG (TD3)

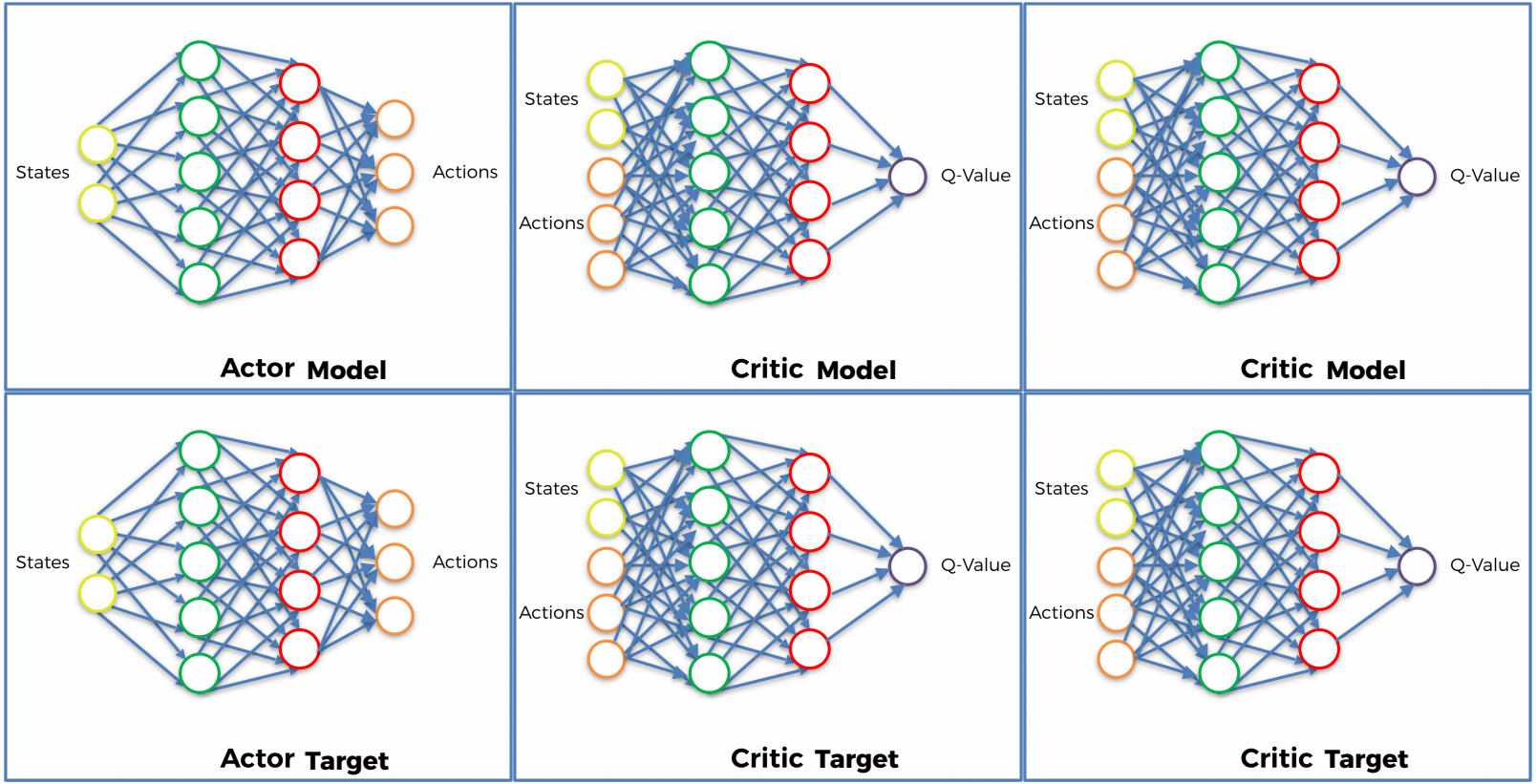
Batch of last states	Batch of next states	Batch of actions	Batch of rewards
Last State 1	Next state 1	Action 1	Reward 1
Last State 2	Next state 2	Action 2	Reward 2
...	...	...	...
Last State 100	Next state 100	Action 100	Reward 100

Twin-Delayed DDPG (TD3)

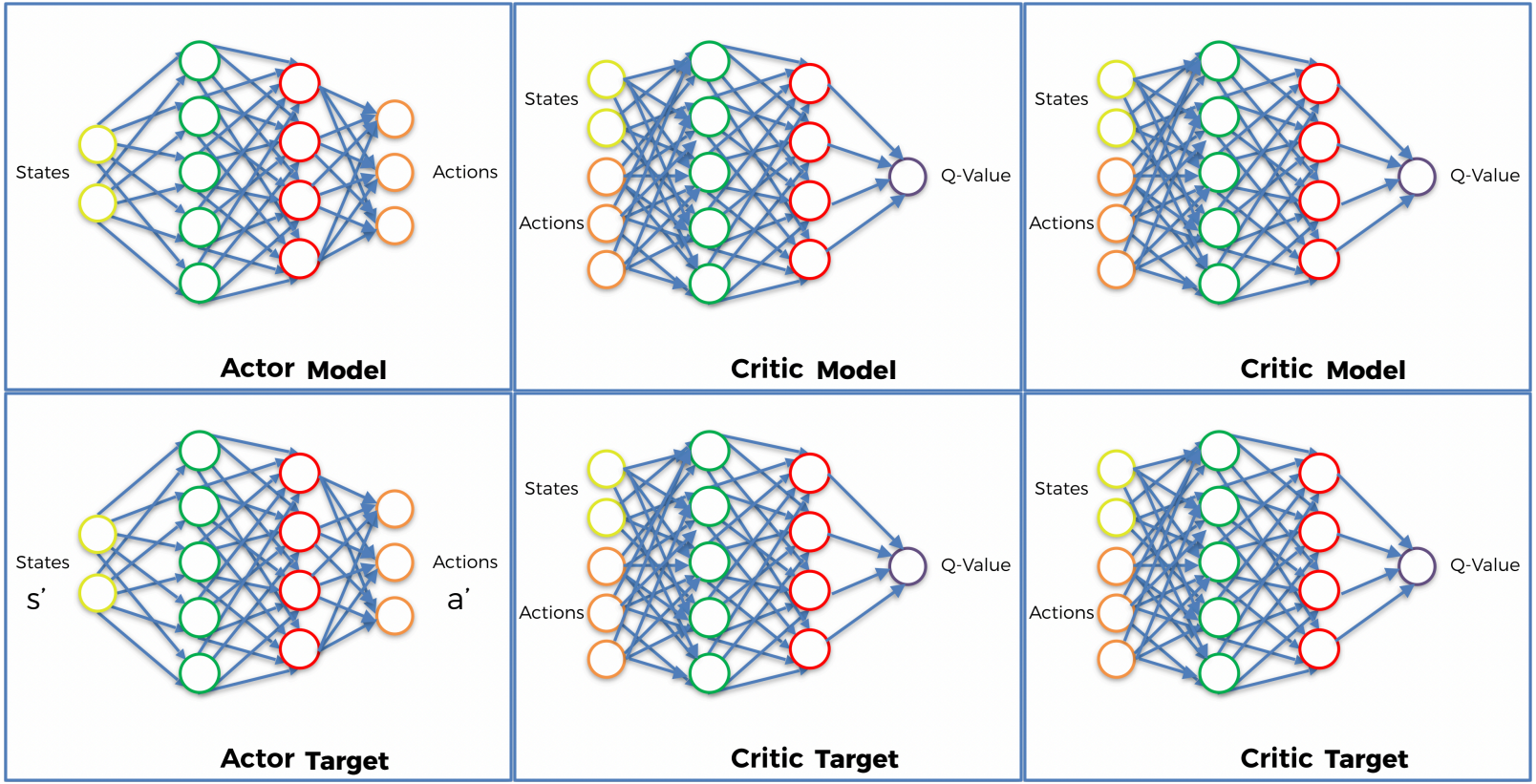
Initialization:
Step 1: We initialize the Experience Replay memory, with a size of 20000. We will populate it with each new transition.
Step 2: We build one neural network for the Actor model and one neural network for the Actor target.
Step 3: We build two neural networks for the two Critic models and two neural networks for the two Critic targets.

Training Process - We run a full episode with first 10,000 actions played randomly, and then with actions played by the Actor model. Then we repeat the following steps:
Step 4: We sample a batch of transitions (s, s', a, r) from the memory. Then for each element of the batch:
Step 5: From the next state s', the Actor target plays the next action a'.

Twin-Delayed DDPG (TD3)



Twin-Delayed DDPG (TD3)



Twin-Delayed DDPG (TD3)

Initialization:

Step 1: We initialize the Experience Replay memory, with a size of 20000. We will populate it with each new transition.
Step 2: We build one neural network for the Actor model and one neural network for the Actor target.
Step 3: We build two neural networks for the two Critic models and two neural networks for the two Critic targets.

Training Process - We run a full episode with first 10,000 actions played randomly, and then with actions played by the Actor model. Then we repeat the following steps:

Step 4: We sample a batch of transitions (s, s', a, r) from the memory. Then for each element of the batch:
Step 5: From the next state s', the Actor target plays the next action a'.
Step 6: We add Gaussian noise to this next action a' and we clamp it in a range of values supported by the environment.

Twin-Delayed DDPG (TD3)

$$\tilde{a} \leftarrow \pi_{\phi'}(s') + \epsilon, \quad \epsilon \sim \text{clip}(\mathcal{N}(0, \tilde{\sigma}), -c, c)$$

$$\tilde{a} \leftarrow \text{clip}(\tilde{a})$$

Twin-Delayed DDPG (TD3)

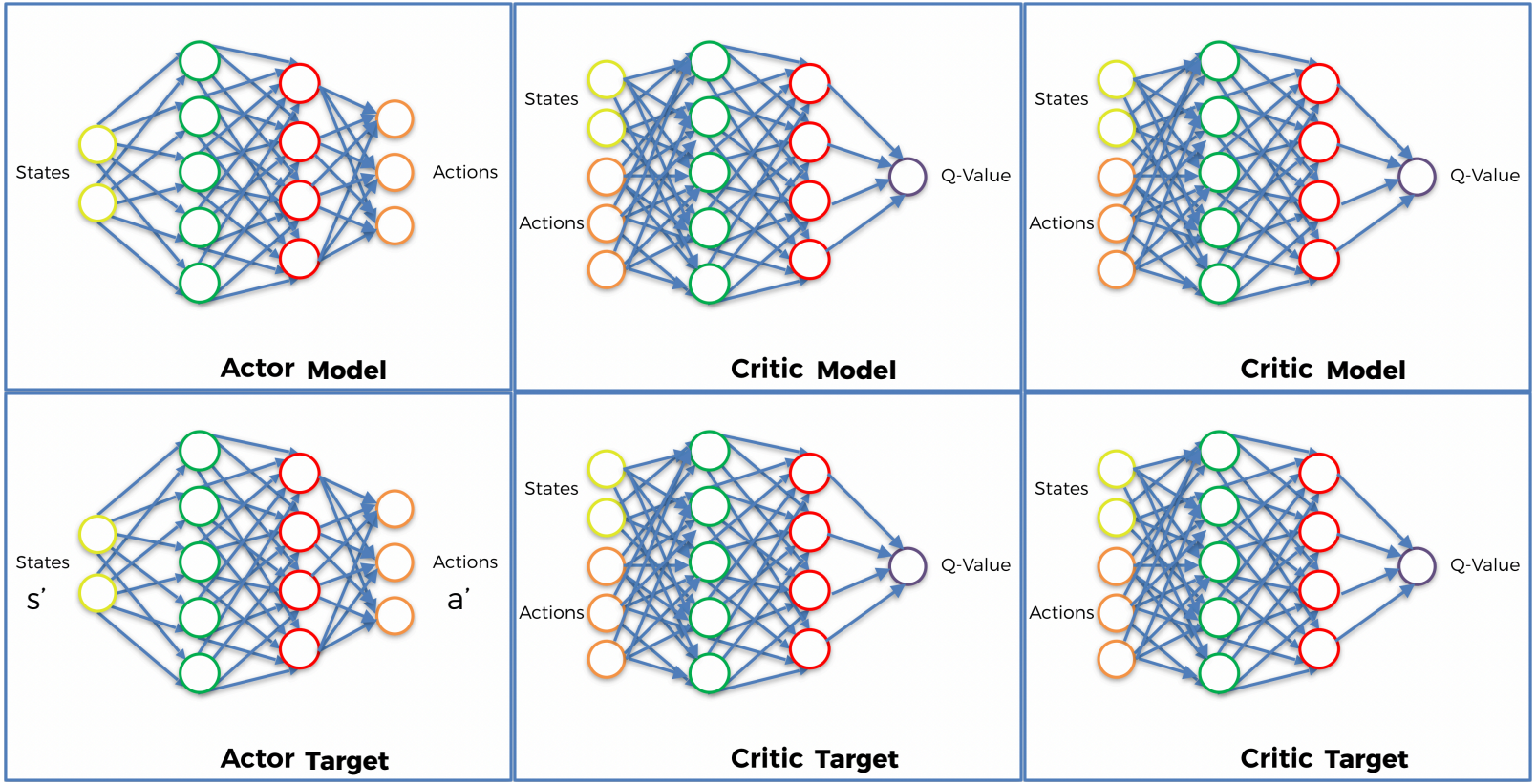
Initialization:

Step 1: We initialize the Experience Replay memory, with a size of 20000. We will populate it with each new transition.
Step 2: We build one neural network for the Actor model and one neural network for the Actor target.
Step 3: We build two neural networks for the two Critic models and two neural networks for the two Critic targets.

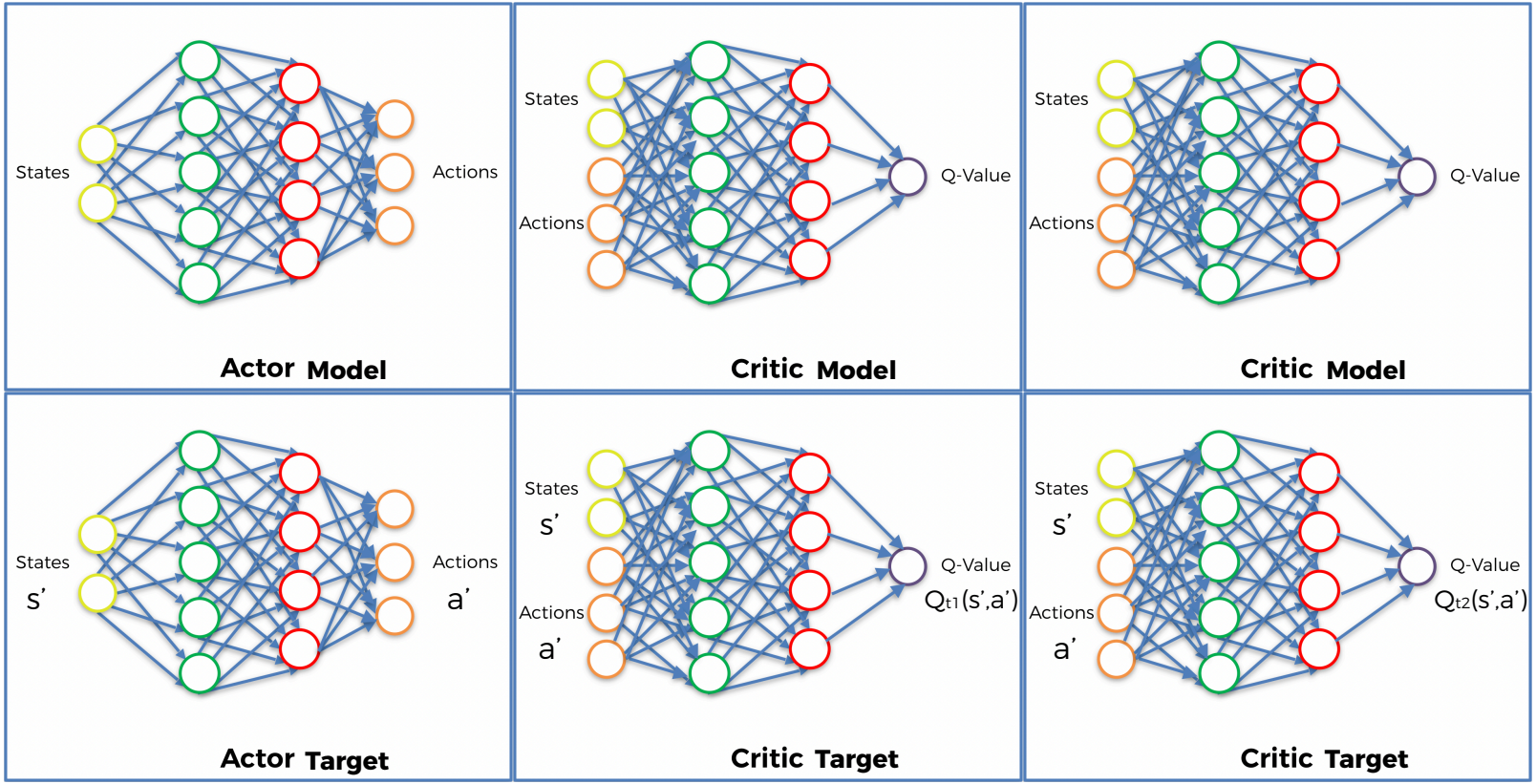
Training Process - We run a full episode with first 10,000 actions played randomly, and then with actions played by the Actor model. Then we repeat the following steps:

Step 4: We sample a batch of transitions (s, s', a, r) from the memory. Then for each element of the batch:
Step 5: From the next state s', the Actor target plays the next action a'.
Step 6: We add Gaussian noise to this next action a' and we clamp it in a range of values supported by the environment.
Step 7: The two Critic targets take each the couple (s', a') as input and return two Q-values $Q_{t1}(s',a')$ and $Q_{t2}(s',a')$ as outputs.

Twin-Delayed DDPG (TD3)



Twin-Delayed DDPG (TD3)



Twin-Delayed DDPG (TD3)

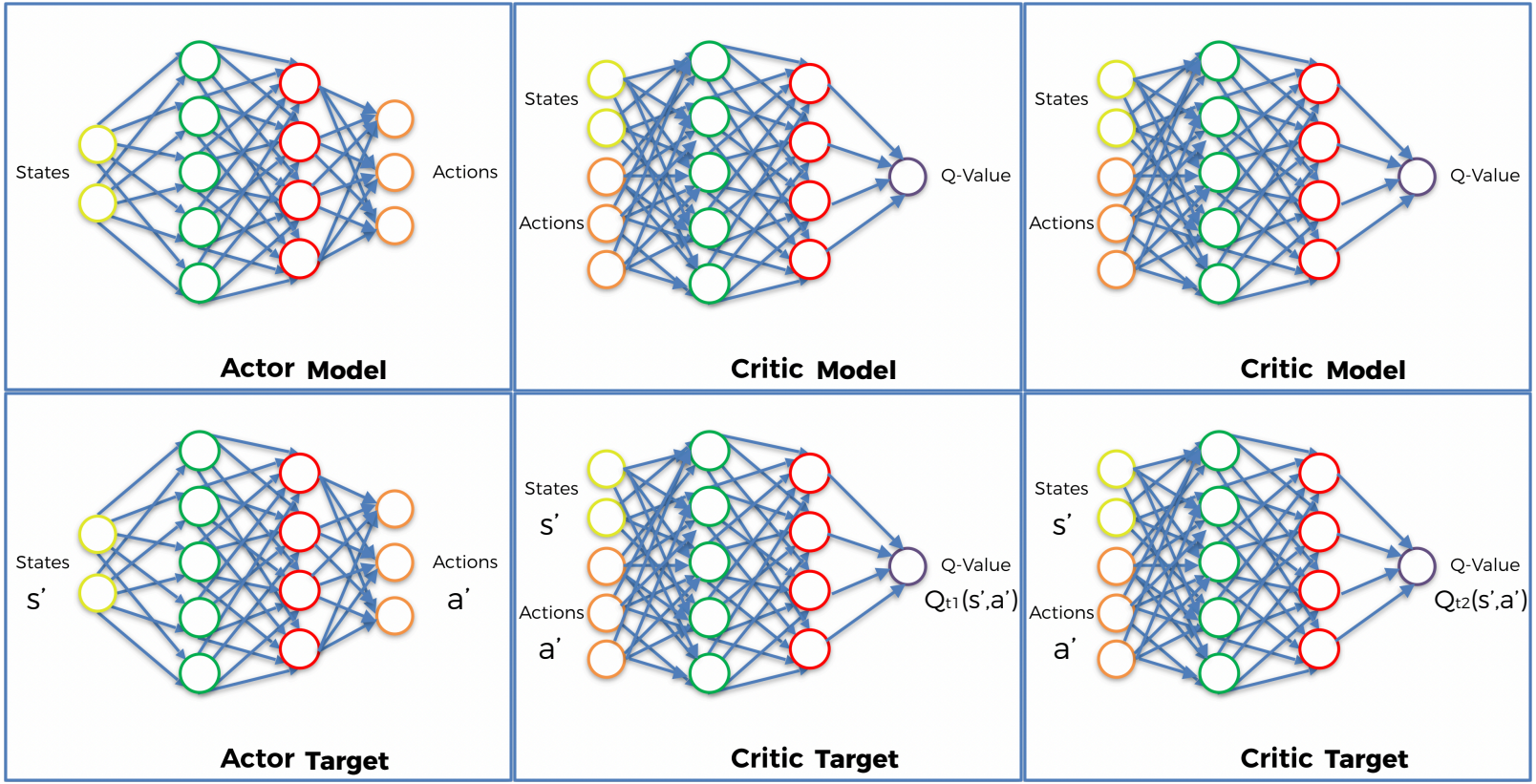
Initialization:

Step 1: We initialize the Experience Replay memory, with a size of 20000. We will populate it with each new transition.
Step 2: We build one neural network for the Actor model and one neural network for the Actor target.
Step 3: We build two neural networks for the two Critic models and two neural networks for the two Critic targets.

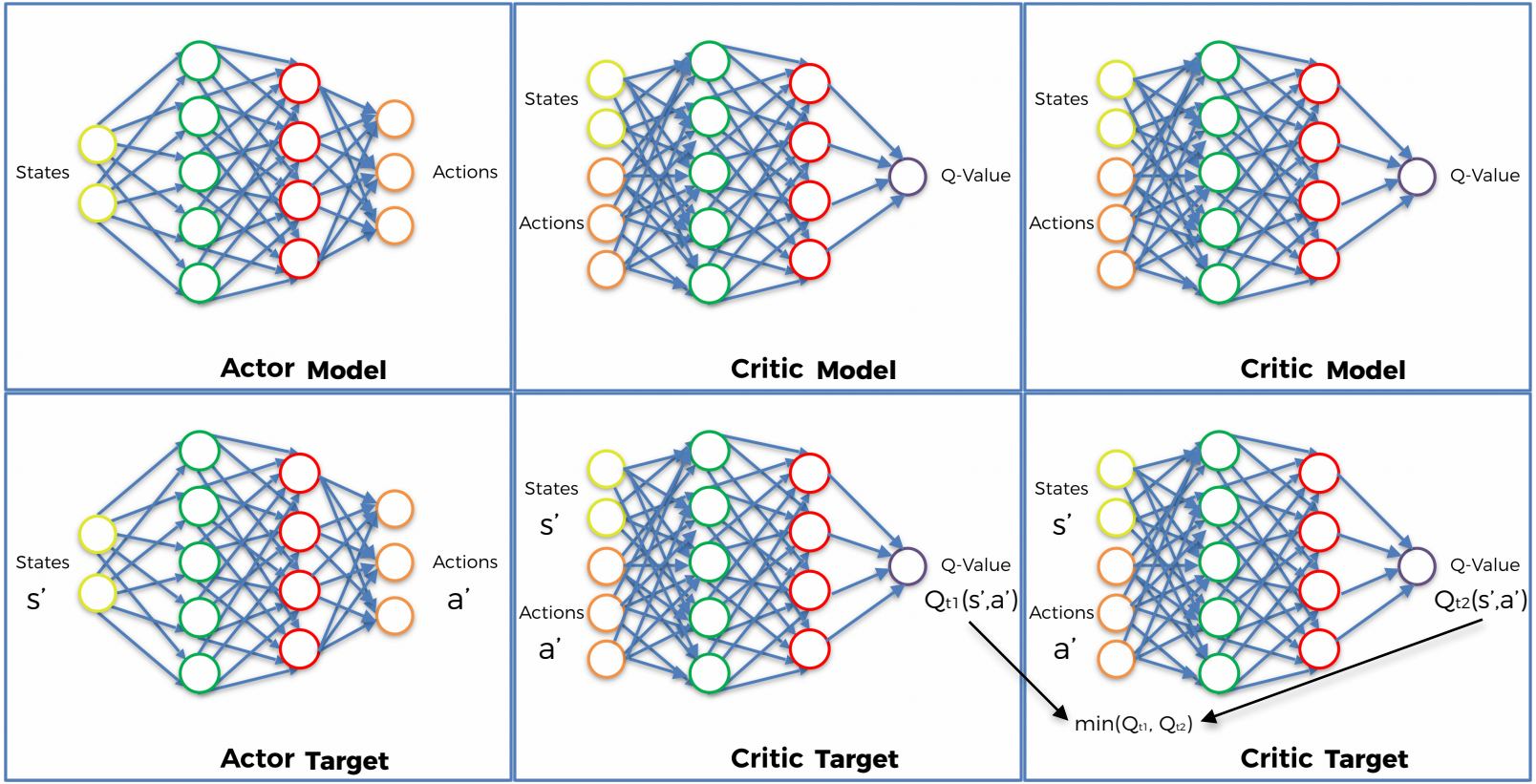
Training Process - We run a full episode with first 10,000 actions played randomly, and then with actions played by the Actor model. Then we repeat the following steps:

Step 4: We sample a batch of transitions (s, s', a, r) from the memory. Then for each element of the batch:
Step 5: From the next state s', the Actor target plays the next action a'.
Step 6: We add Gaussian noise to this next action a' and we clamp it in a range of values supported by the environment.
Step 7: The two Critic targets take each the couple (s', a') as input and return two Q-values $Q_{t1}(s',a')$ and $Q_{t2}(s',a')$ as outputs.
Step 8: We keep the minimum of these two Q-values: $\min(Q_{t1}, Q_{t2})$. It represents the approximated value of the next state.

Twin-Delayed DDPG (TD3)



Twin-Delayed DDPG (TD3)



Twin-Delayed DDPG (TD3)

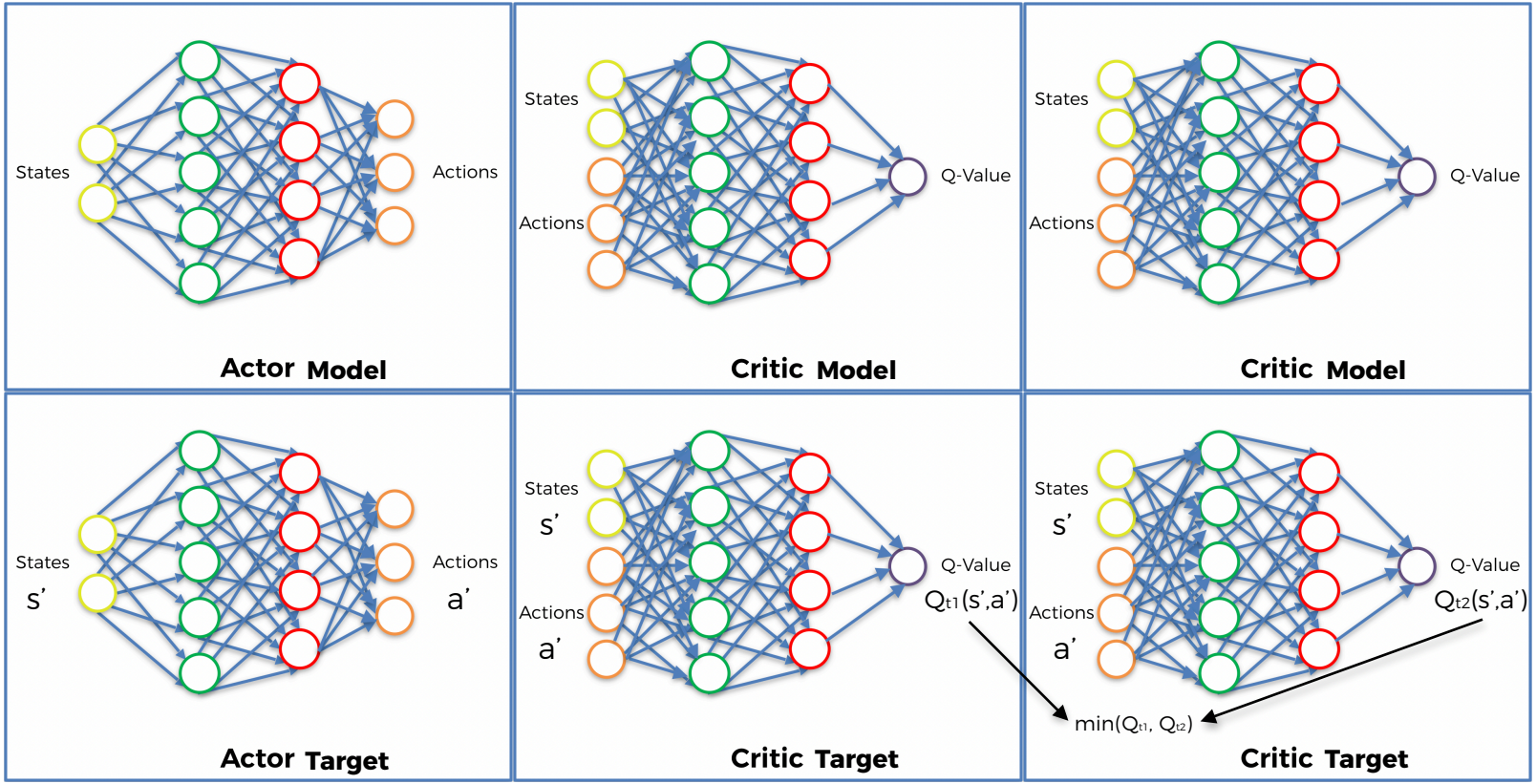
Initialization:

Step 1: We initialize the Experience Replay memory, with a size of 20000. We will populate it with each new transition.
Step 2: We build one neural network for the Actor model and one neural network for the Actor target.
Step 3: We build two neural networks for the two Critic models and two neural networks for the two Critic targets.

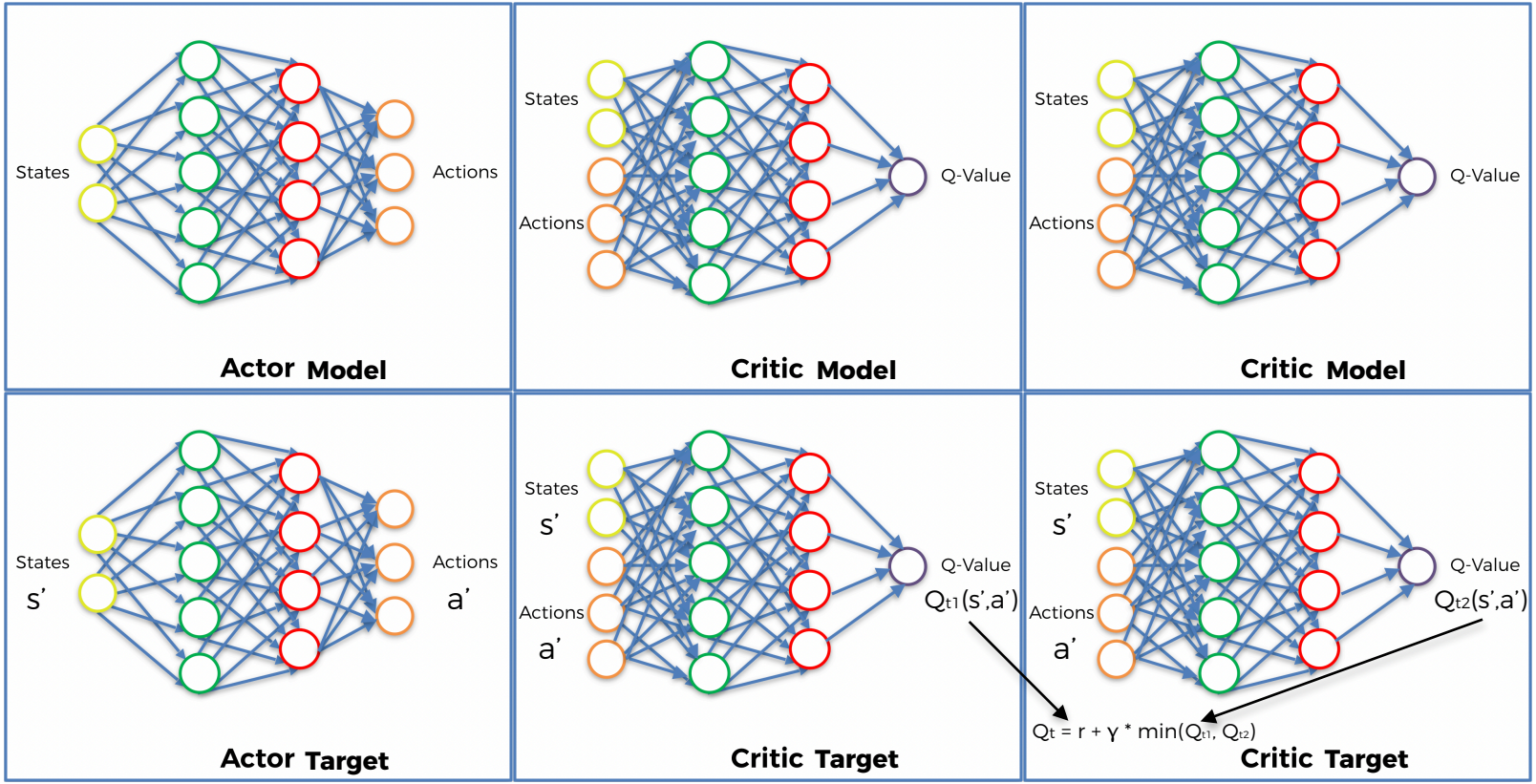
Training Process - We run a full episode with first 10,000 actions played randomly, and then with actions played by the Actor model. Then we repeat the following steps:

Step 4: We sample a batch of transitions (s, s', a, r) from the memory. Then for each element of the batch:
Step 5: From the next state s', the Actor target plays the next action a'.
Step 6: We add Gaussian noise to this next action a' and we clamp it in a range of values supported by the environment.
Step 7: The two Critic targets take each the couple (s', a') as input and return two Q-values $Q_{t1}(s',a')$ and $Q_{t2}(s',a')$ as outputs.
Step 8: We keep the minimum of these two Q-values: $\min(Q_{t1}, Q_{t2})$. It represents the approximated value of the next state.
Step 9: We get the final target of the two Critic models, which is: $Q_t = r + \gamma * \min(Q_{t1}, Q_{t2})$, where γ is the discount factor.

Twin-Delayed DDPG (TD3)



Twin-Delayed DDPG (TD3)



Twin-Delayed DDPG (TD3)

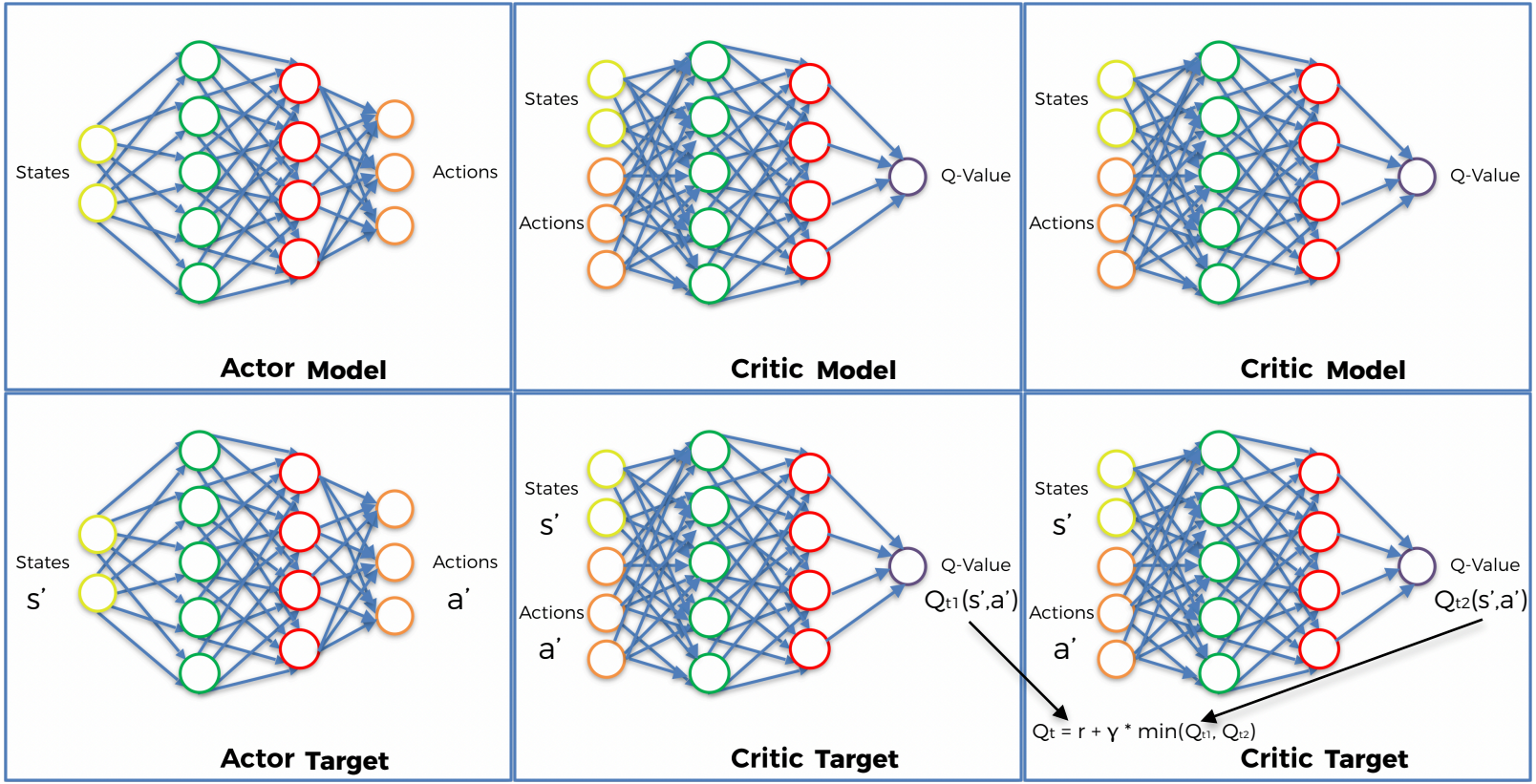
Initialization:

Step 1: We initialize the Experience Replay memory, with a size of 20000. We will populate it with each new transition.
Step 2: We build one neural network for the Actor model and one neural network for the Actor target.
Step 3: We build two neural networks for the two Critic models and two neural networks for the two Critic targets.

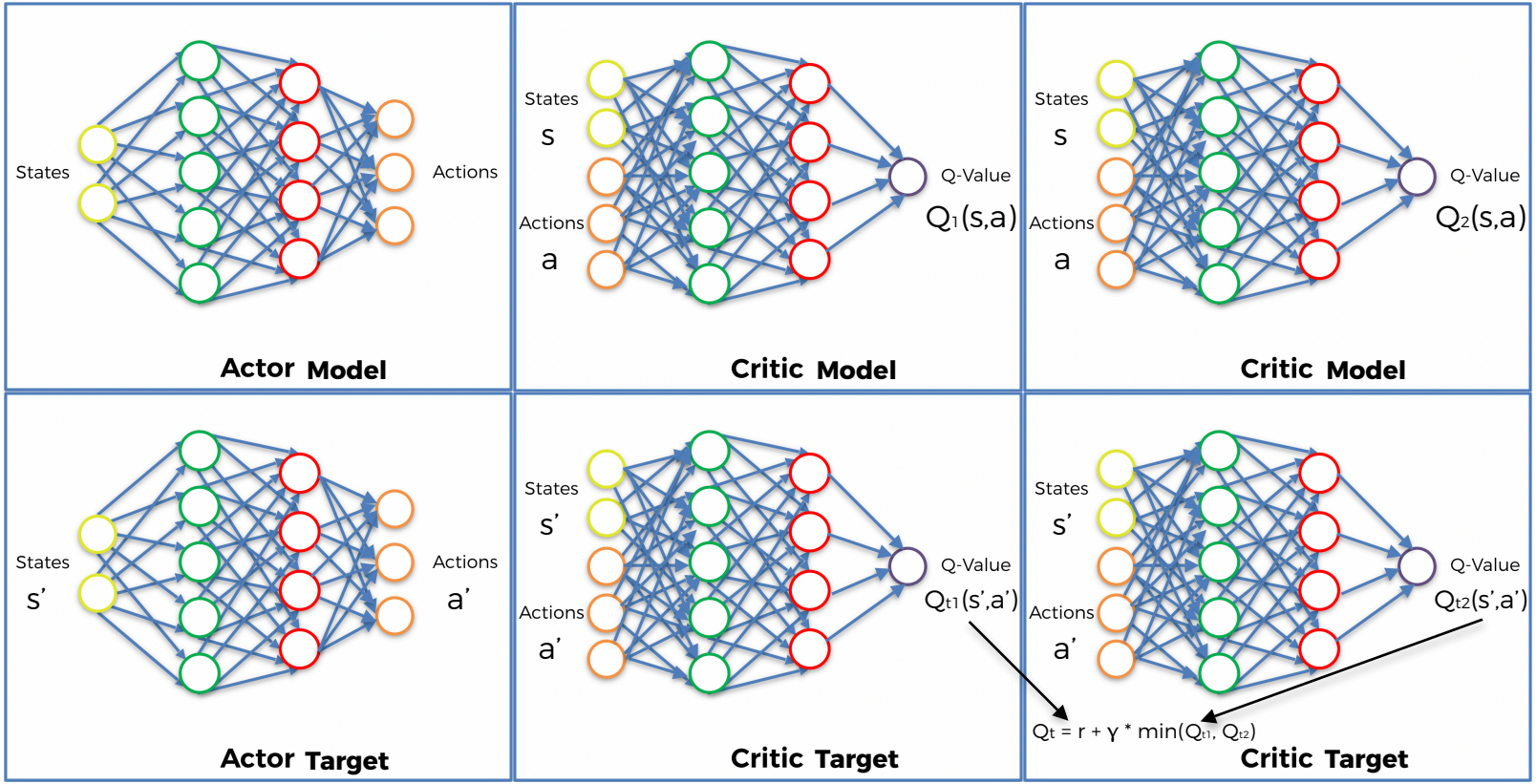
Training Process - We run a full episode with first 10,000 actions played randomly, and then with actions played by the Actor model. Then we repeat the following steps:

Step 4: We sample a batch of transitions (s, s', a, r) from the memory. Then for each element of the batch:
Step 5: From the next state s', the Actor target plays the next action a'.
Step 6: We add Gaussian noise to this next action a' and we clamp it in a range of values supported by the environment.
Step 7: The two Critic targets take each the couple (s', a') as input and return two Q-values $Q_{t1}(s',a')$ and $Q_{t2}(s',a')$ as outputs.
Step 8: We keep the minimum of these two Q-values: $\min(Q_{t1}, Q_{t2})$. It represents the approximated value of the next state.
Step 9: We get the final target of the two Critic models, which is: $Q_t = r + \gamma * \min(Q_{t1}, Q_{t2})$, where γ is the discount factor.
Step 10: The two Critic models take each the couple (s, a) as input and return two Q-values $Q_1(s,a)$ and $Q_2(s,a)$ as outputs.

Twin-Delayed DDPG (TD3)



Twin-Delayed DDPG (TD3)



Twin-Delayed DDPG (TD3)

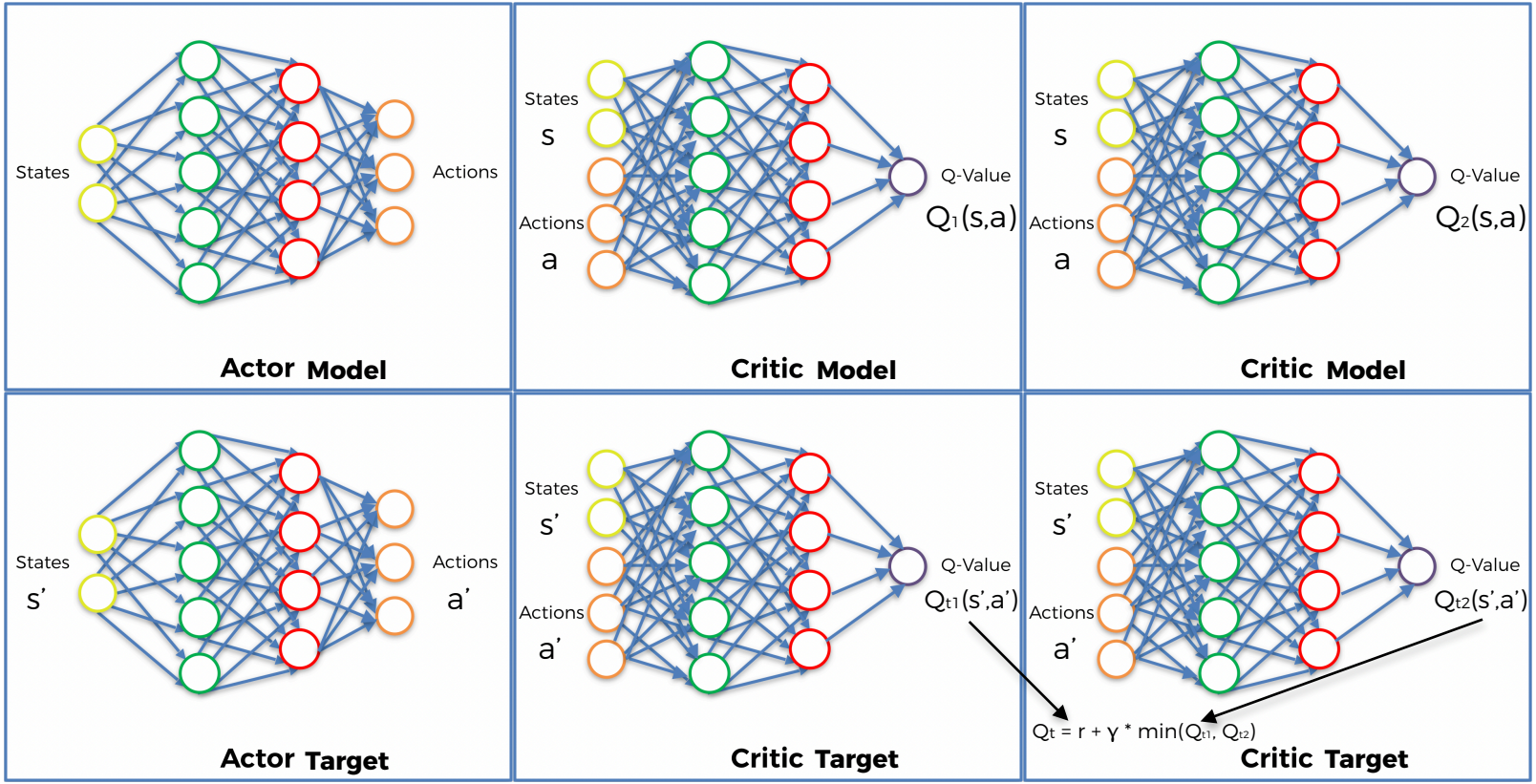
Initialization:

Step 1: We initialize the Experience Replay memory, with a size of 20000. We will populate it with each new transition.
Step 2: We build one neural network for the Actor model and one neural network for the Actor target.
Step 3: We build two neural networks for the two Critic models and two neural networks for the two Critic targets.

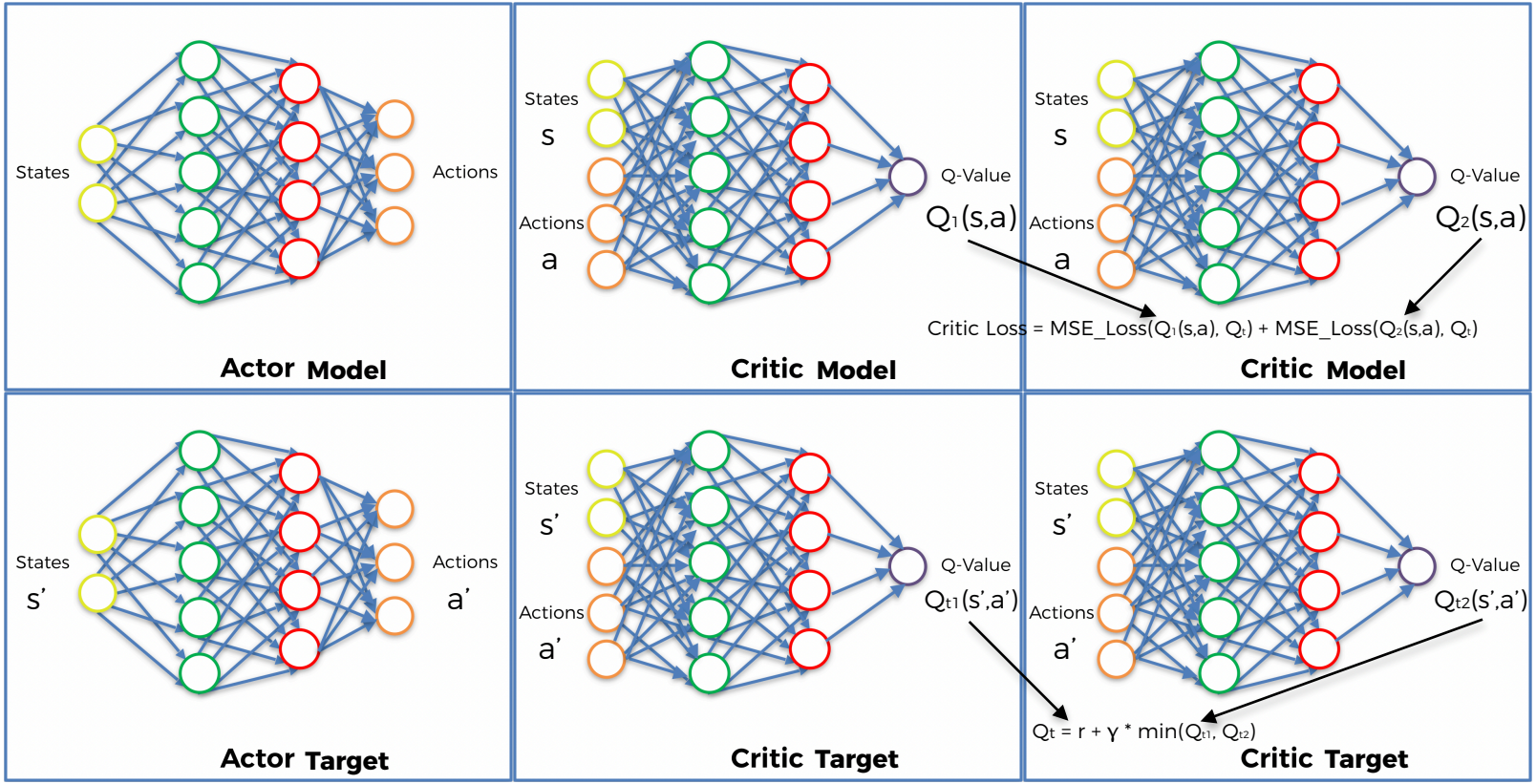
Training Process - We run a full episode with first 10,000 actions played randomly, and then with actions played by the Actor model. Then we repeat the following steps:

Step 4: We sample a batch of transitions (s, s', a, r) from the memory. Then for each element of the batch:
Step 5: From the next state s', the Actor target plays the next action a'.
Step 6: We add Gaussian noise to this next action a' and we clamp it in a range of values supported by the environment.
Step 7: The two Critic targets take each the couple (s', a') as input and return two Q-values $Q_{t1}(s',a')$ and $Q_{t2}(s',a')$ as outputs.
Step 8: We keep the minimum of these two Q-values: $\min(Q_{t1}, Q_{t2})$. It represents the approximated value of the next state.
Step 9: We get the final target of the two Critic models, which is: $Q_t = r + \gamma * \min(Q_{t1}, Q_{t2})$, where γ is the discount factor.
Step 10: The two Critic models take each the couple (s, a) as input and return two Q-values $Q_1(s,a)$ and $Q_2(s,a)$ as outputs.
Step 11: We compute the loss coming from the two Critic models: Critic Loss = $MSE\_Loss(Q_1(s,a), Q_t) + MSE\_Loss(Q_2(s,a), Q_t)$

Twin-Delayed DDPG (TD3)



Twin-Delayed DDPG (TD3)



Twin-Delayed DDPG (TD3)

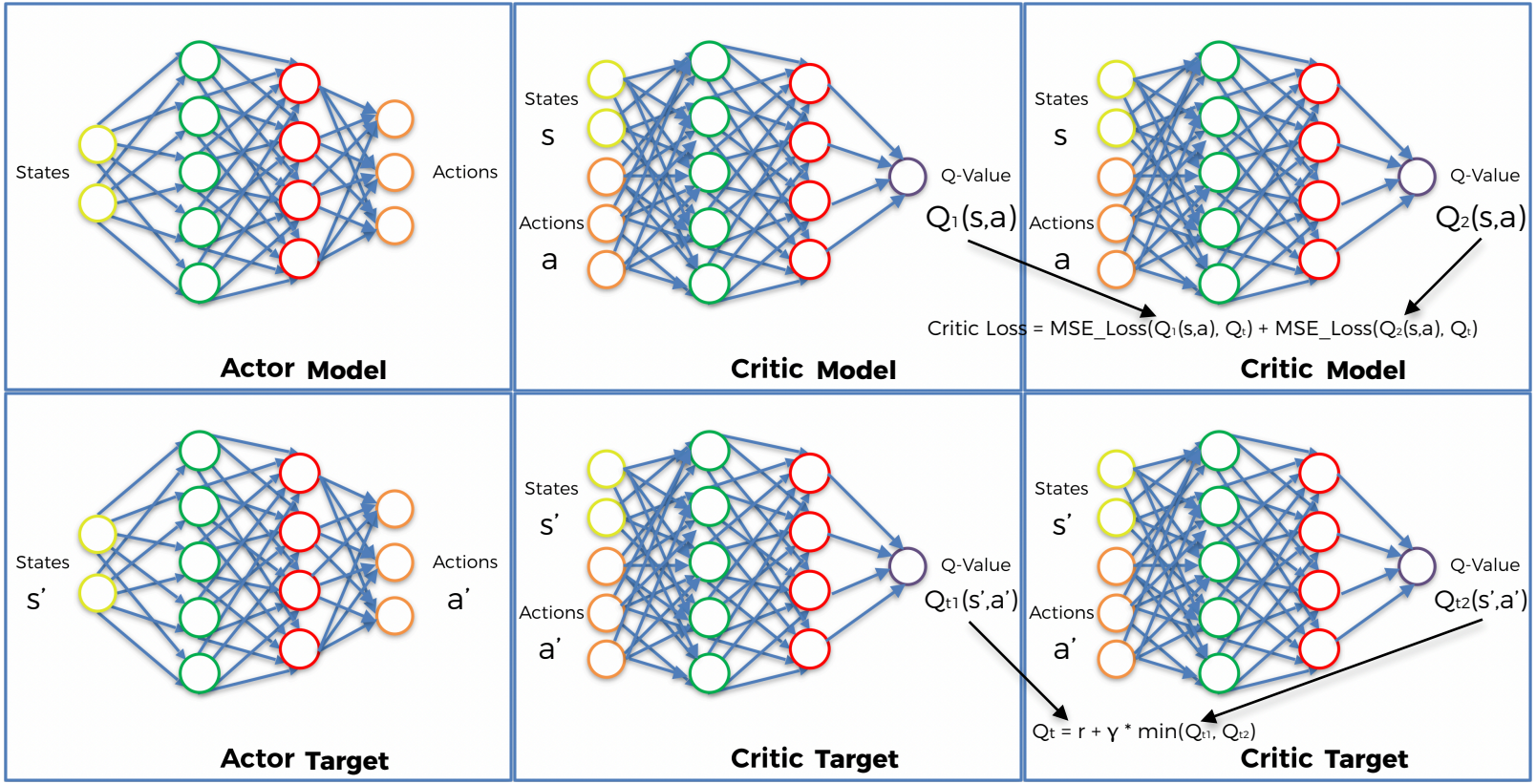
Initialization:

Step 1: We initialize the Experience Replay memory, with a size of 20000. We will populate it with each new transition.
Step 2: We build one neural network for the Actor model and one neural network for the Actor target.
Step 3: We build two neural networks for the two Critic models and two neural networks for the two Critic targets.

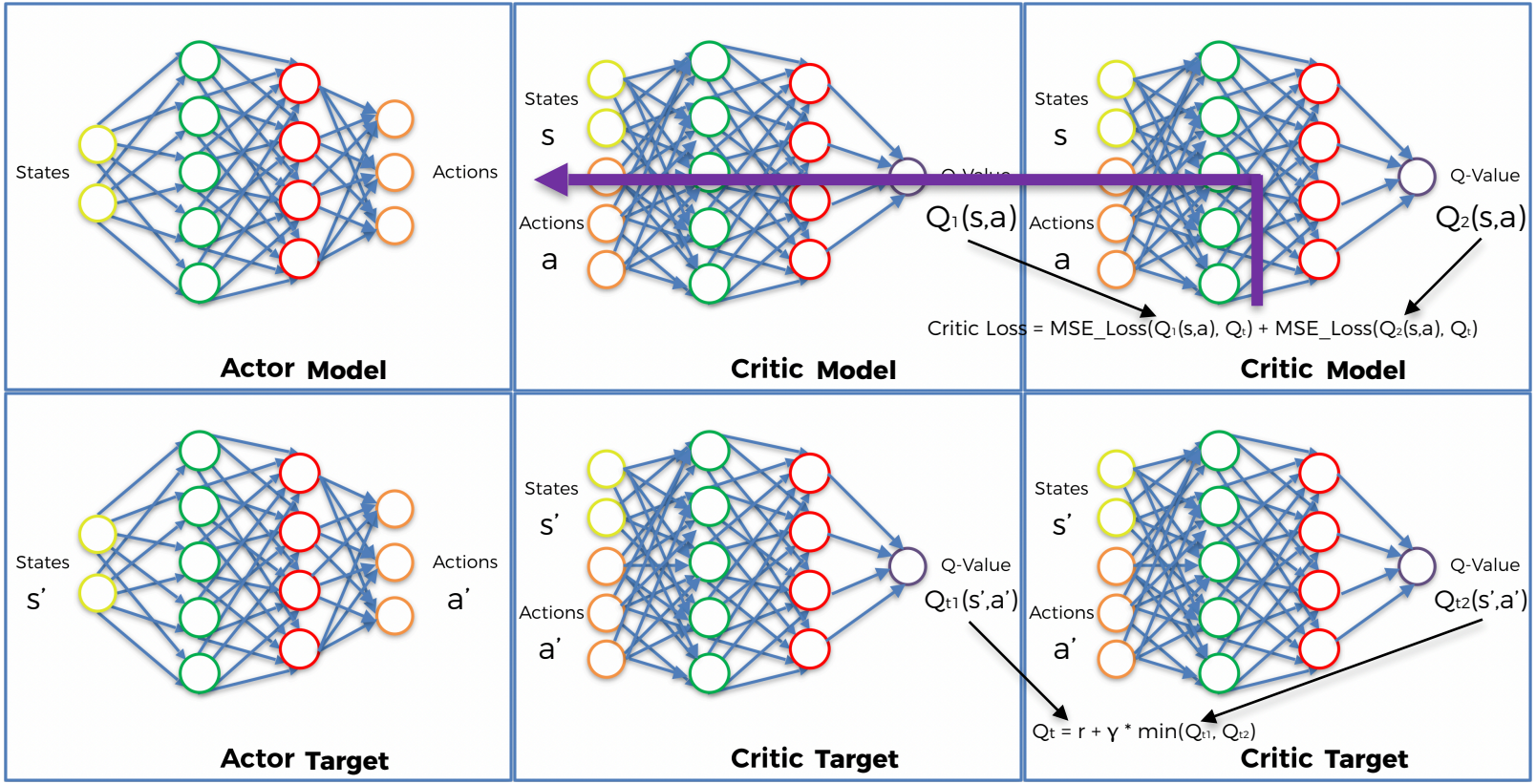
Training Process - We run a full episode with first 10,000 actions played randomly, and then with actions played by the Actor model. Then we repeat the following steps:

Step 4: We sample a batch of transitions (s, s', a, r) from the memory. Then for each element of the batch:
Step 5: From the next state s', the Actor target plays the next action a'.
Step 6: We add Gaussian noise to this next action a' and we clamp it in a range of values supported by the environment.
Step 7: The two Critic targets take each the couple (s', a') as input and return two Q-values $Q_{t1}(s',a')$ and $Q_{t2}(s',a')$ as outputs.
Step 8: We keep the minimum of these two Q-values: $\min(Q_{t1}, Q_{t2})$. It represents the approximated value of the next state.
Step 9: We get the final target of the two Critic models, which is: $Q_t = r + \gamma * \min(Q_{t1}, Q_{t2})$, where γ is the discount factor.
Step 10: The two Critic models take each the couple (s, a) as input and return two Q-values $Q_1(s,a)$ and $Q_2(s,a)$ as outputs.
Step 11: We compute the loss coming from the two Critic models: Critic Loss = MSE_Loss($Q_1(s,a)$, Q_t) + MSE_Loss($Q_2(s,a)$, Q_t)
Step 12: We backpropagate this Critic loss and update the parameters of the two Critic models with a SGD optimizer.

Twin-Delayed DDPG (TD3)



Twin-Delayed DDPG (TD3)



Twin-Delayed DDPG (TD3)

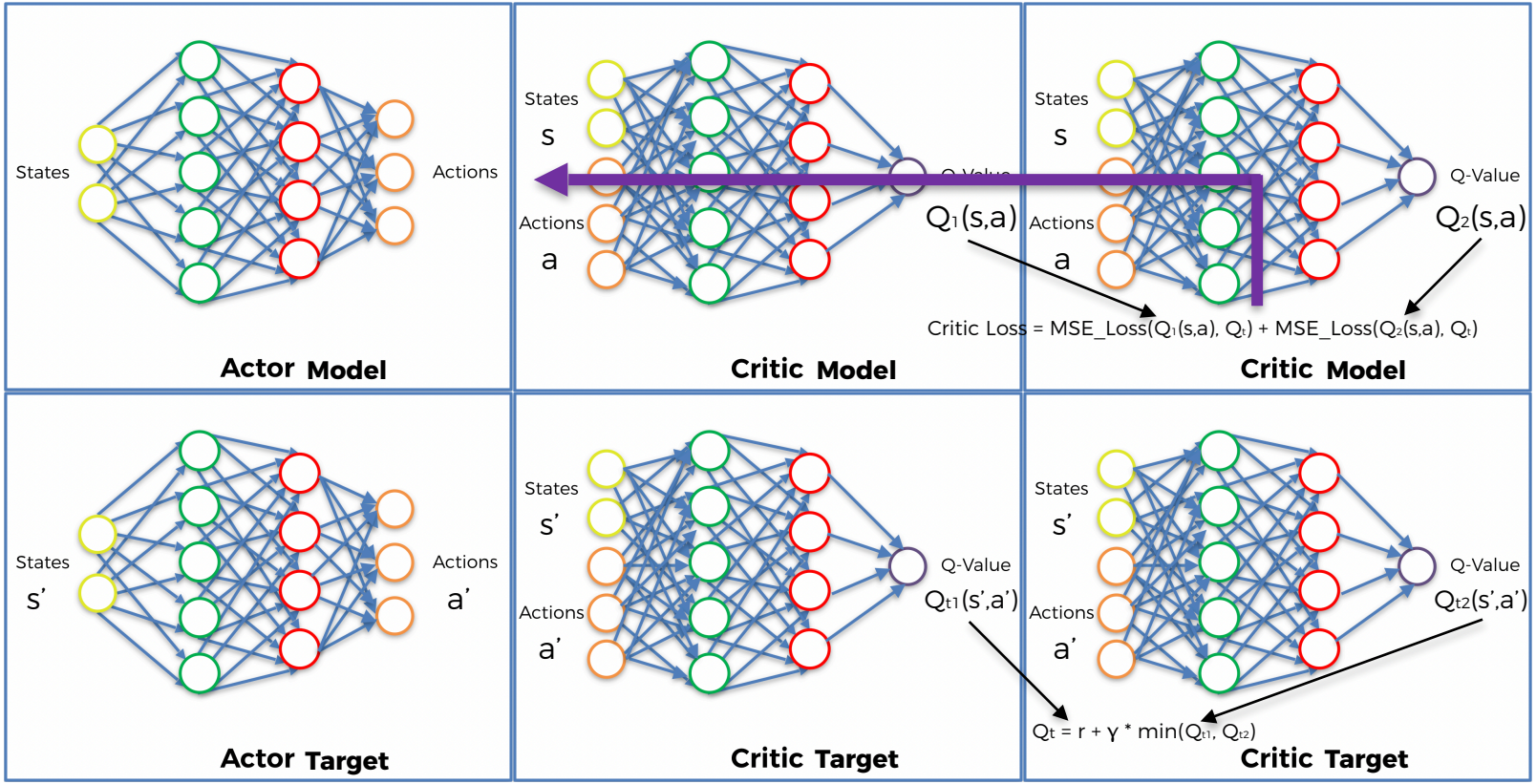
Initialization:

Step 1: We initialize the Experience Replay memory, with a size of 20000. We will populate it with each new transition.
Step 2: We build one neural network for the Actor model and one neural network for the Actor target.
Step 3: We build two neural networks for the two Critic models and two neural networks for the two Critic targets.

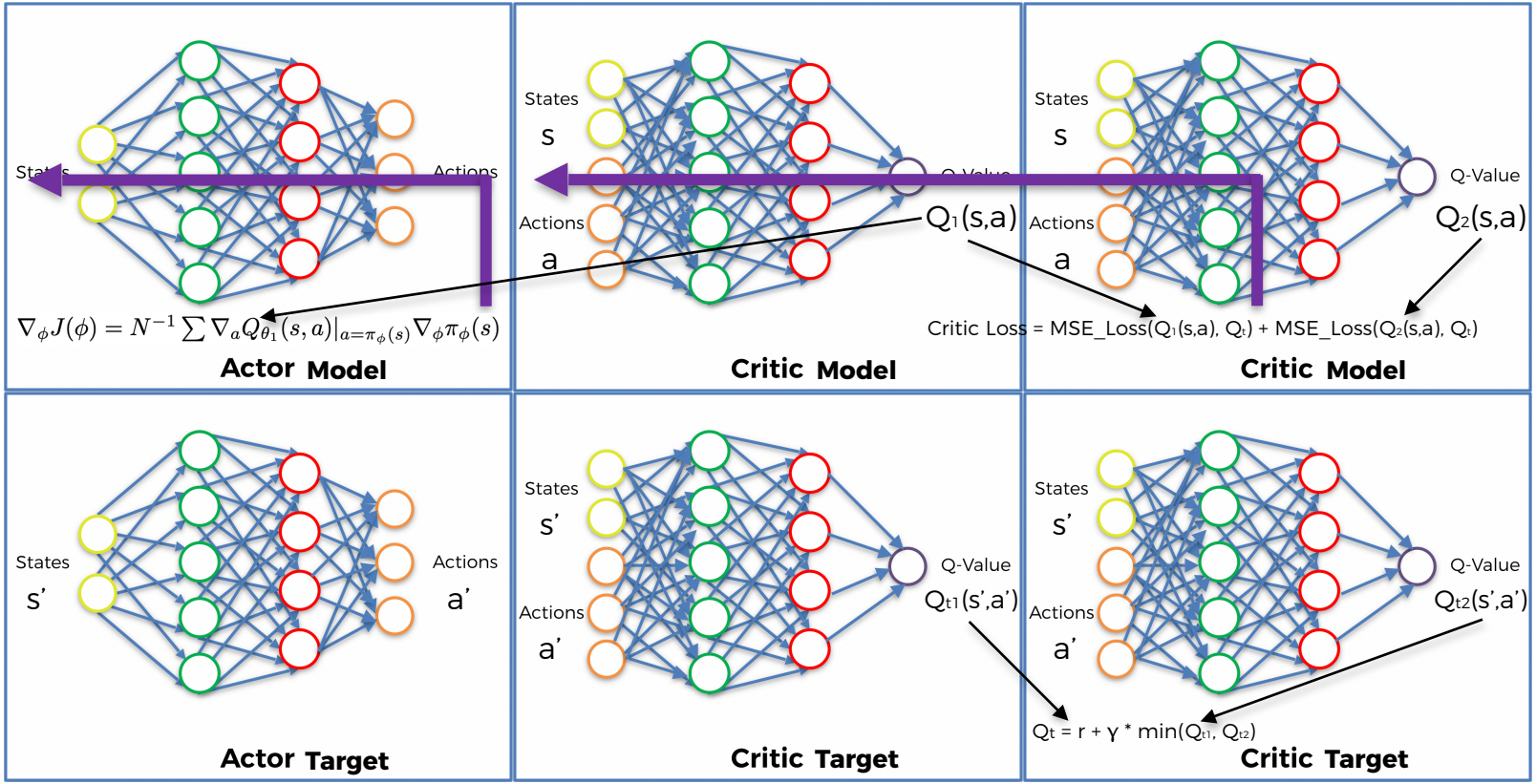
Training Process - We run a full episode with first 10,000 actions played randomly, and then with actions played by the Actor model. Then we repeat the following steps:

Step 4: We sample a batch of transitions (s, s', a, r) from the memory. Then for each element of the batch:
Step 5: From the next state s', the Actor target plays the next action a'.
Step 6: We add Gaussian noise to this next action a' and we clamp it in a range of values supported by the environment.
Step 7: The two Critic targets take each the couple (s', a') as input and return two Q-values $Q_{t1}(s',a')$ and $Q_{t2}(s',a')$ as outputs.
Step 8: We keep the minimum of these two Q-values: $\min(Q_{t1}, Q_{t2})$. It represents the approximated value of the next state.
Step 9: We get the final target of the two Critic models, which is: $Q_t = r + \gamma * \min(Q_{t1}, Q_{t2})$, where γ is the discount factor.
Step 10: The two Critic models take each the couple (s, a) as input and return two Q-values $Q_1(s,a)$ and $Q_2(s,a)$ as outputs.
Step 11: We compute the loss coming from the two Critic models: Critic Loss = MSE_Loss($Q_1(s,a)$, Q_t) + MSE_Loss($Q_2(s,a)$, Q_t)
Step 12: We backpropagate this Critic loss and update the parameters θ_1 of the two Critic models with a SGD optimizer.
Step 13: Once every two iterations, we update our Actor model by performing gradient ascent on the output of the first Critic model: $\nabla_\phi J(\phi) = N^{-1} \sum \nabla_a Q_{\theta_1}(s,a)|_{a=\pi_\phi(s)} \nabla_\phi \pi_\phi(s)$, where ϕ and θ_1 are resp. the weights of the Actor and the Critic.

Twin-Delayed DDPG (TD3)



Twin-Delayed DDPG (TD3)



Twin-Delayed DDPG (TD3)

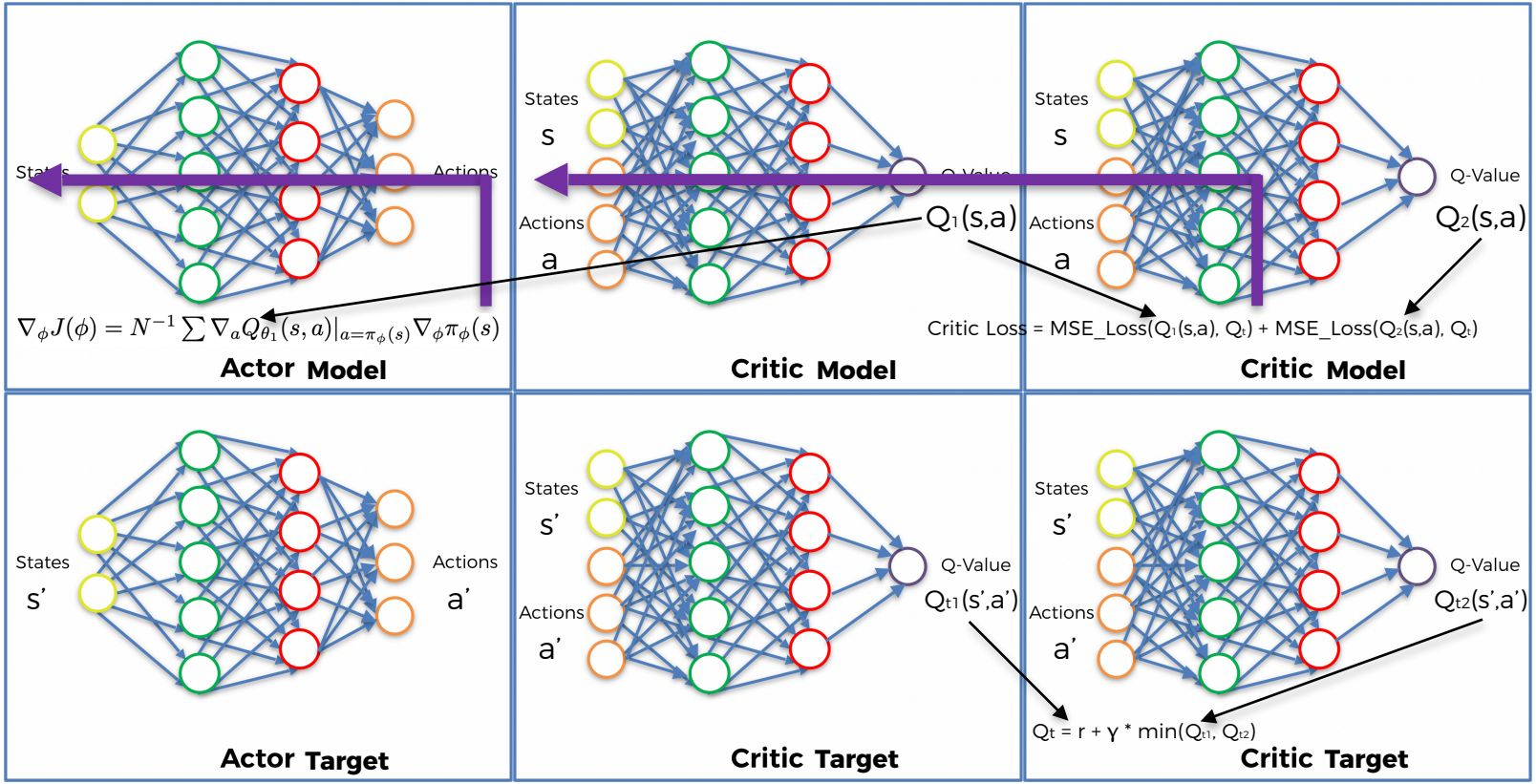
Initialization:

Step 1: We initialize the Experience Replay memory, with a size of 20000. We will populate it with each new transition.
Step 2: We build one neural network for the Actor model and one neural network for the Actor target.
Step 3: We build two neural networks for the two Critic models and two neural networks for the two Critic targets.

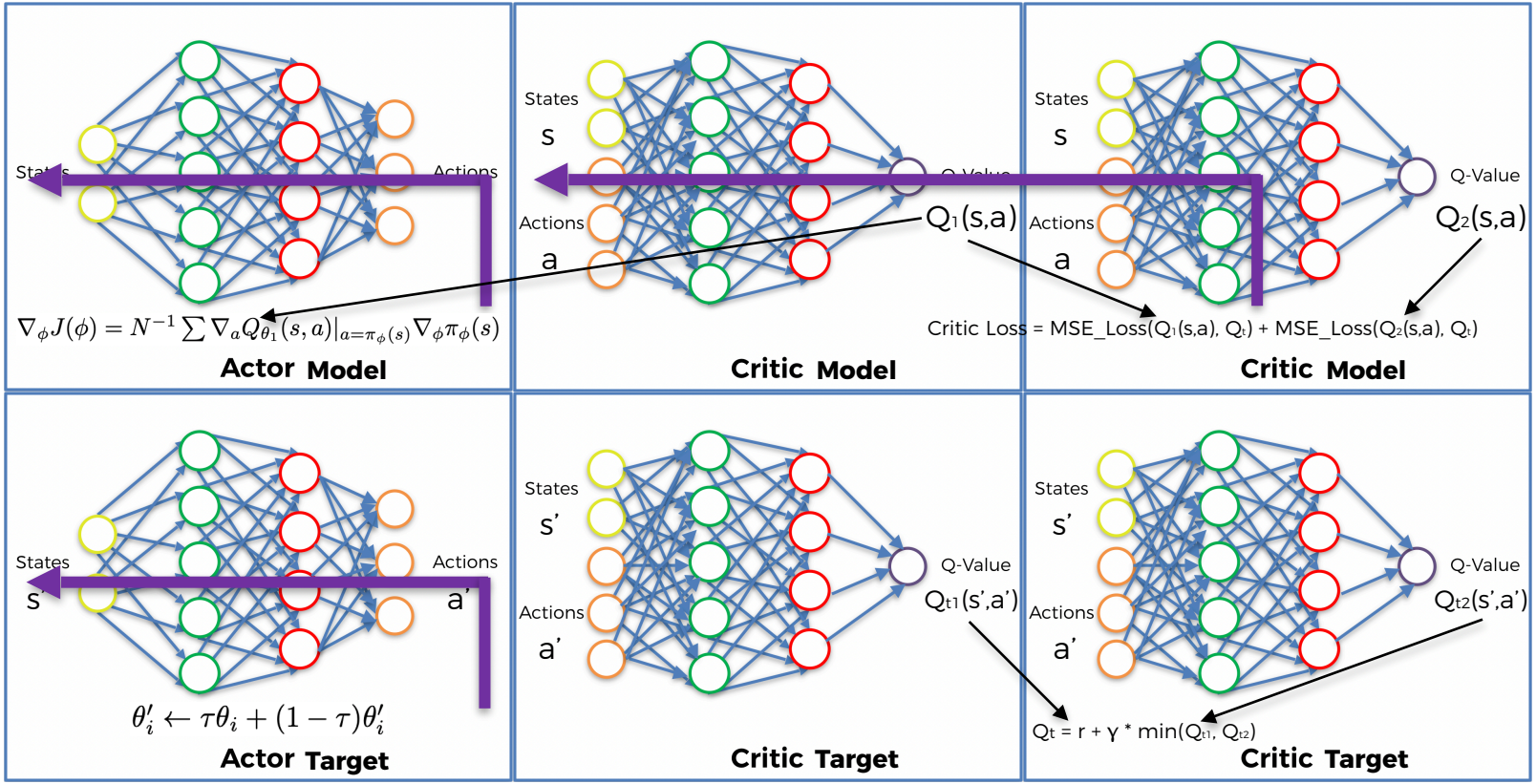
Training Process - We run a full episode with first 10,000 actions played randomly, and then with actions played by the Actor model. Then we repeat the following steps:

Step 4: We sample a batch of transitions (s, s', a, r) from the memory. Then for each element of the batch:
Step 5: From the next state s', the Actor target plays the next action a'.
Step 6: We add Gaussian noise to this next action a' and we clamp it in a range of values supported by the environment.
Step 7: The two Critic targets take each the couple (s', a') as input and return two Q-values $Q_{t1}(s',a')$ and $Q_{t2}(s',a')$ as outputs.
Step 8: We keep the minimum of these two Q-values: $\min(Q_{t1}, Q_{t2})$. It represents the approximated value of the next state.
Step 9: We get the final target of the two Critic models, which is: $Q_t = r + \gamma * \min(Q_{t1}, Q_{t2})$, where γ is the discount factor.
Step 10: The two Critic models take each the couple (s, a) as input and return two Q-values $Q_1(s,a)$ and $Q_2(s,a)$ as outputs.
Step 11: We compute the loss coming from the two Critic models: Critic Loss = MSE_Loss($Q_1(s,a)$, Q_t) + MSE_Loss($Q_2(s,a)$, Q_t)
Step 12: We back-propagate this Critic loss and update the parameters θ_1 of the two Critic models with a SGD optimizer.
Step 13: Once every two iterations, we update our Actor model by performing gradient ascent on the output of the first Critic model: $\nabla_\phi J(\phi) = N^{-1} \sum \nabla_a Q_{\theta_1}(s,a)|_{a=\pi_\phi(s)} \nabla_\phi \pi_\phi(s)$, where ϕ and θ_1 are resp. the weights of the Actor and the Critic.
Step 14: Once every two iterations, we update the weights of the Actor target by polyak averaging: $\theta'_i \leftarrow \tau\theta_i + (1-\tau)\theta'_i$

Twin-Delayed DDPG (TD3)



Twin-Delayed DDPG (TD3)



Twin-Delayed DDPG (TD3)

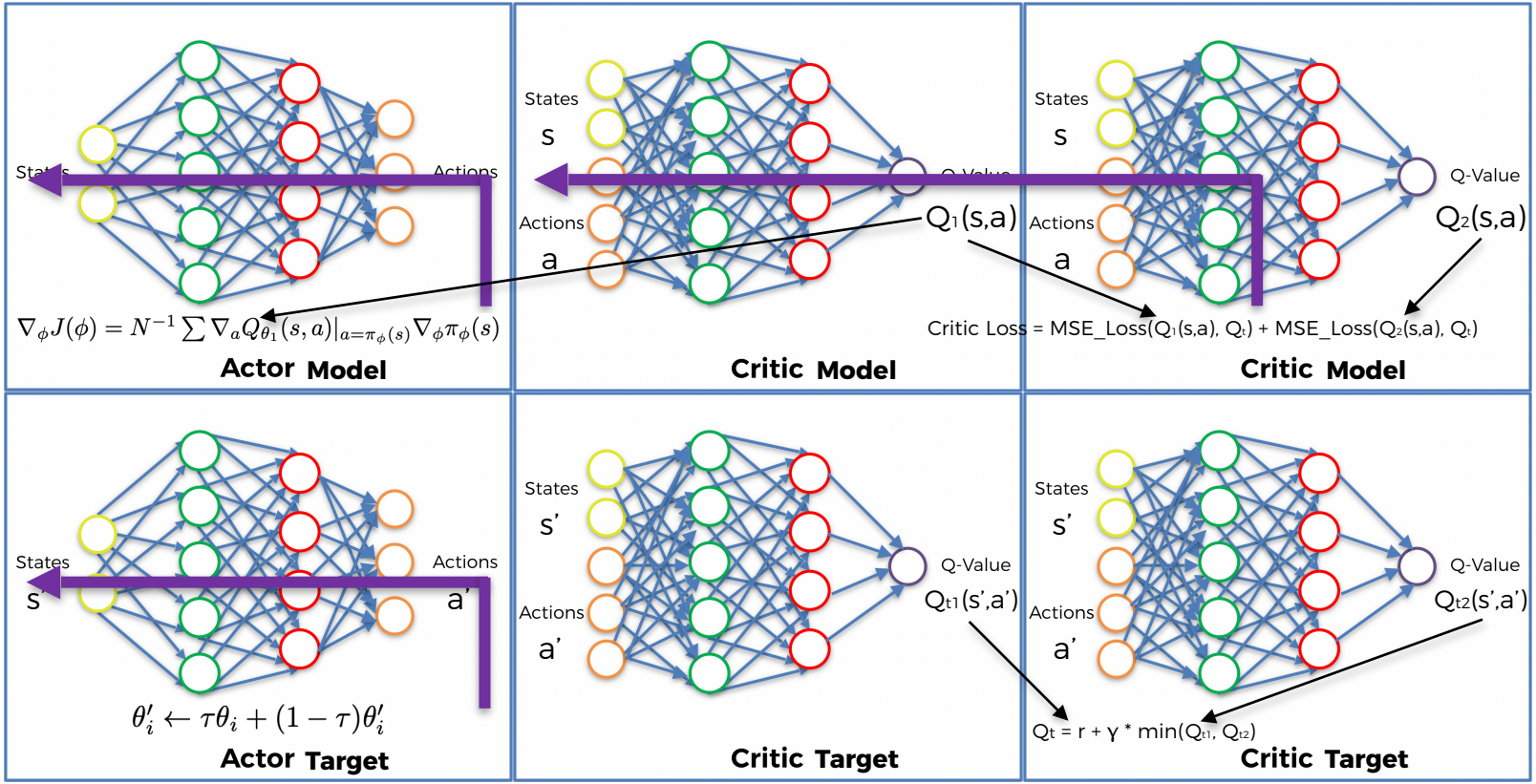
Initialization:

Step 1: We initialize the Experience Replay memory, with a size of 20000. We will populate it with each new transition.
Step 2: We build one neural network for the Actor model and one neural network for the Actor target.
Step 3: We build two neural networks for the two Critic models and two neural networks for the two Critic targets.

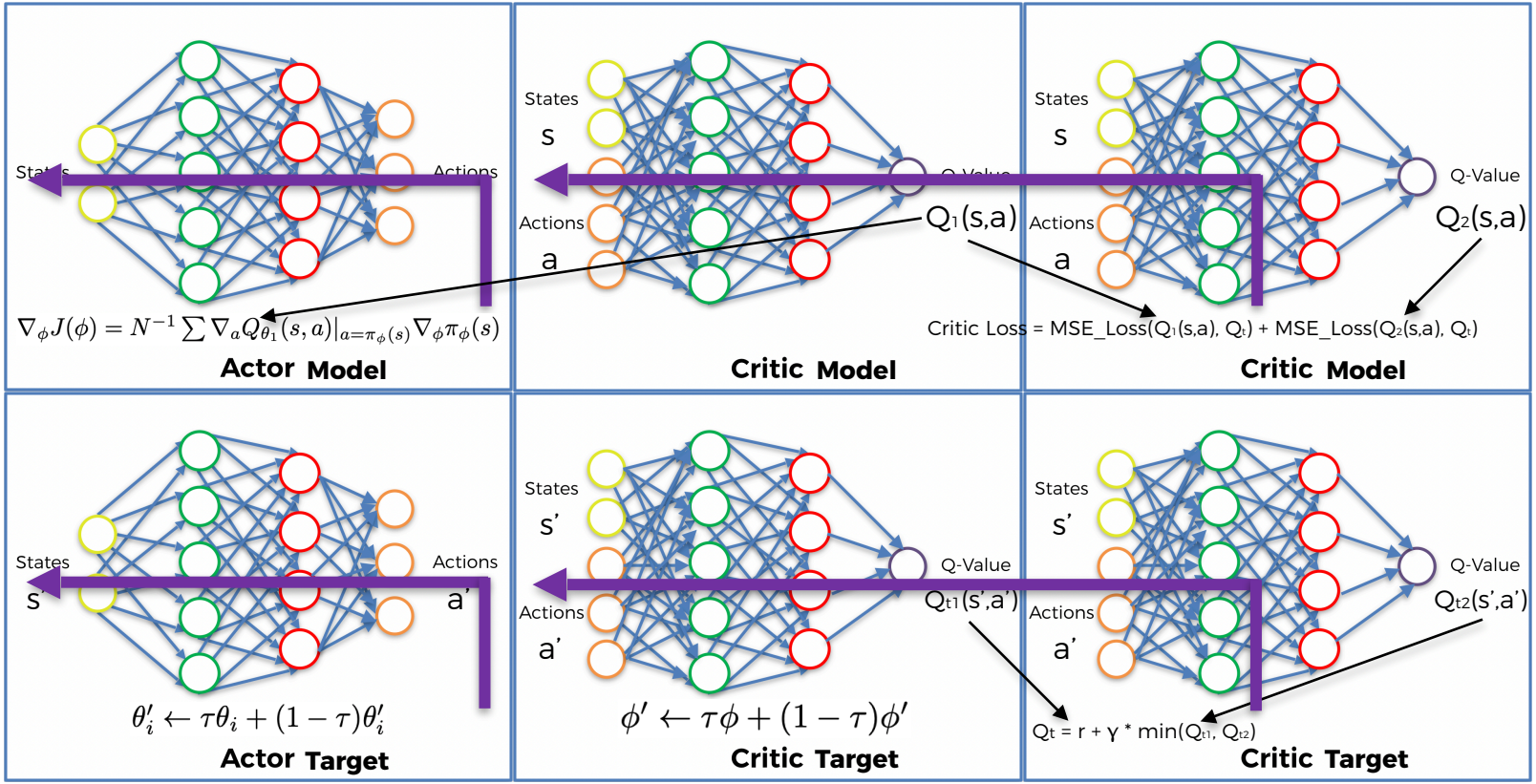
Training Process - We run a full episode with first 10,000 actions played randomly, and then with actions played by the Actor model. Then we repeat the following steps:

Step 4: We sample a batch of transitions (s, s', a, r) from the memory. Then for each element of the batch:
Step 5: From the next state s', the Actor target plays the next action a'.
Step 6: We add Gaussian noise to this next action a' and we clamp it in a range of values supported by the environment.
Step 7: The two Critic targets take each the couple (s', a') as input and return two Q-values $Q_{t1}(s',a')$ and $Q_{t2}(s',a')$ as outputs.
Step 8: We keep the minimum of these two Q-values: $\min(Q_{t1}, Q_{t2})$. It represents the approximated value of the next state.
Step 9: We get the final target of the two Critic models, which is: $Q_t = r + \gamma * \min(Q_{t1}, Q_{t2})$, where γ is the discount factor.
Step 10: The two Critic models take each the couple (s, a) as input and return two Q-values $Q_1(s,a)$ and $Q_2(s,a)$ as outputs.
Step 11: We compute the loss coming from the two Critic models: Critic Loss = MSE_Loss($Q_1(s,a)$, Q_t) + MSE_Loss($Q_2(s,a)$, Q_t)
Step 12: We backpropagate this Critic loss and update the parameters θ_1 of the two Critic models with a SGD optimizer.
Step 13: Once every two iterations, we update our Actor model by performing gradient ascent on the output of the first Critic model: $\nabla_\phi J(\phi) = N^{-1} \sum \nabla_a Q_{\theta_1}(s,a)|_{a=\pi_\phi(s)} \nabla_\phi \pi_\phi(s)$, where ϕ and θ_1 are resp. the weights of the Actor and the first Critic.
Step 14: Once every two iterations, we update the weights of the Actor target by polyak averaging: $\theta_i' \leftarrow \tau\theta_i + (1-\tau)\theta_i'$ $\phi' \leftarrow \tau\phi + (1-\tau)\phi'$

Twin-Delayed DDPG (TD3)



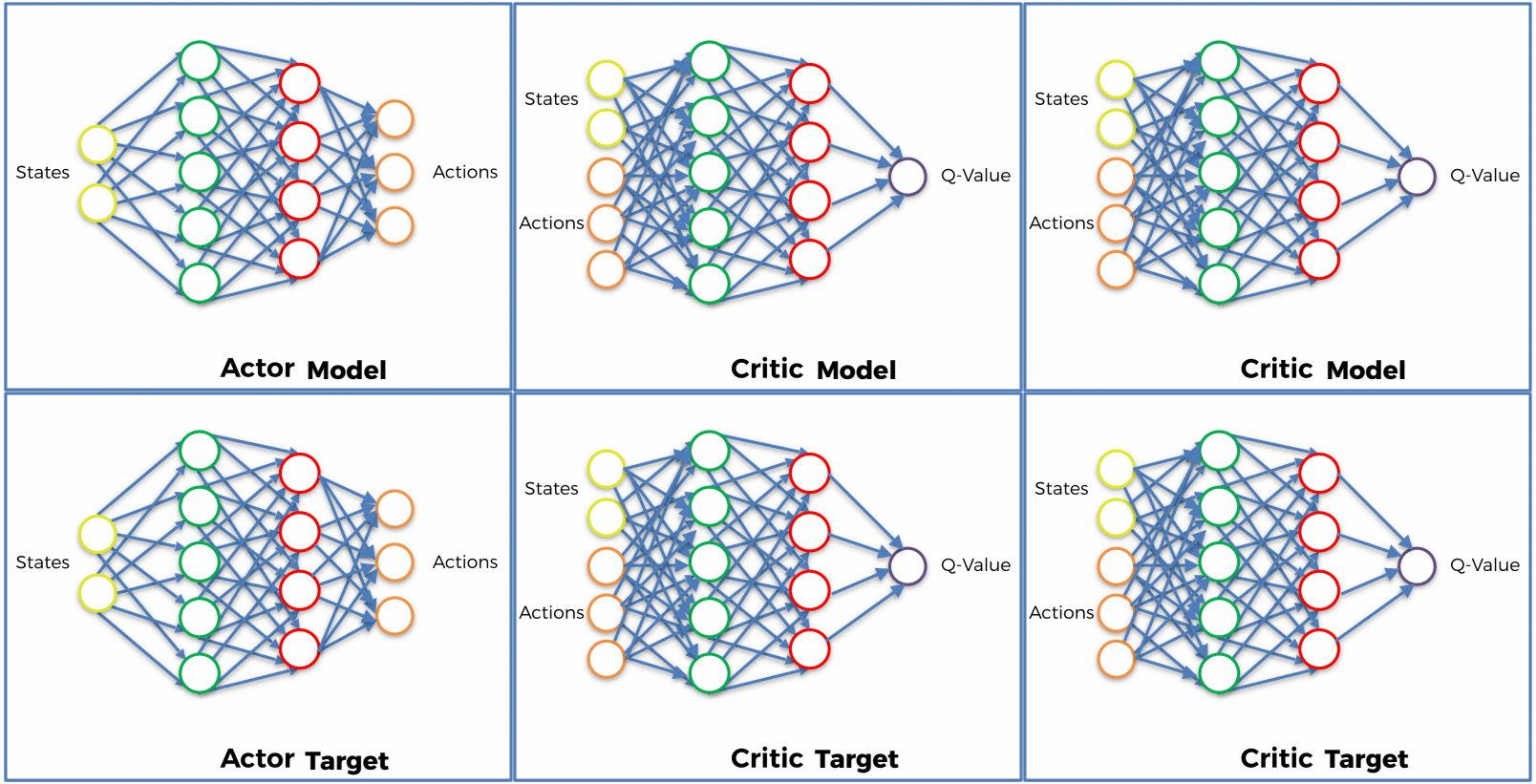
Twin-Delayed DDPG (TD3)



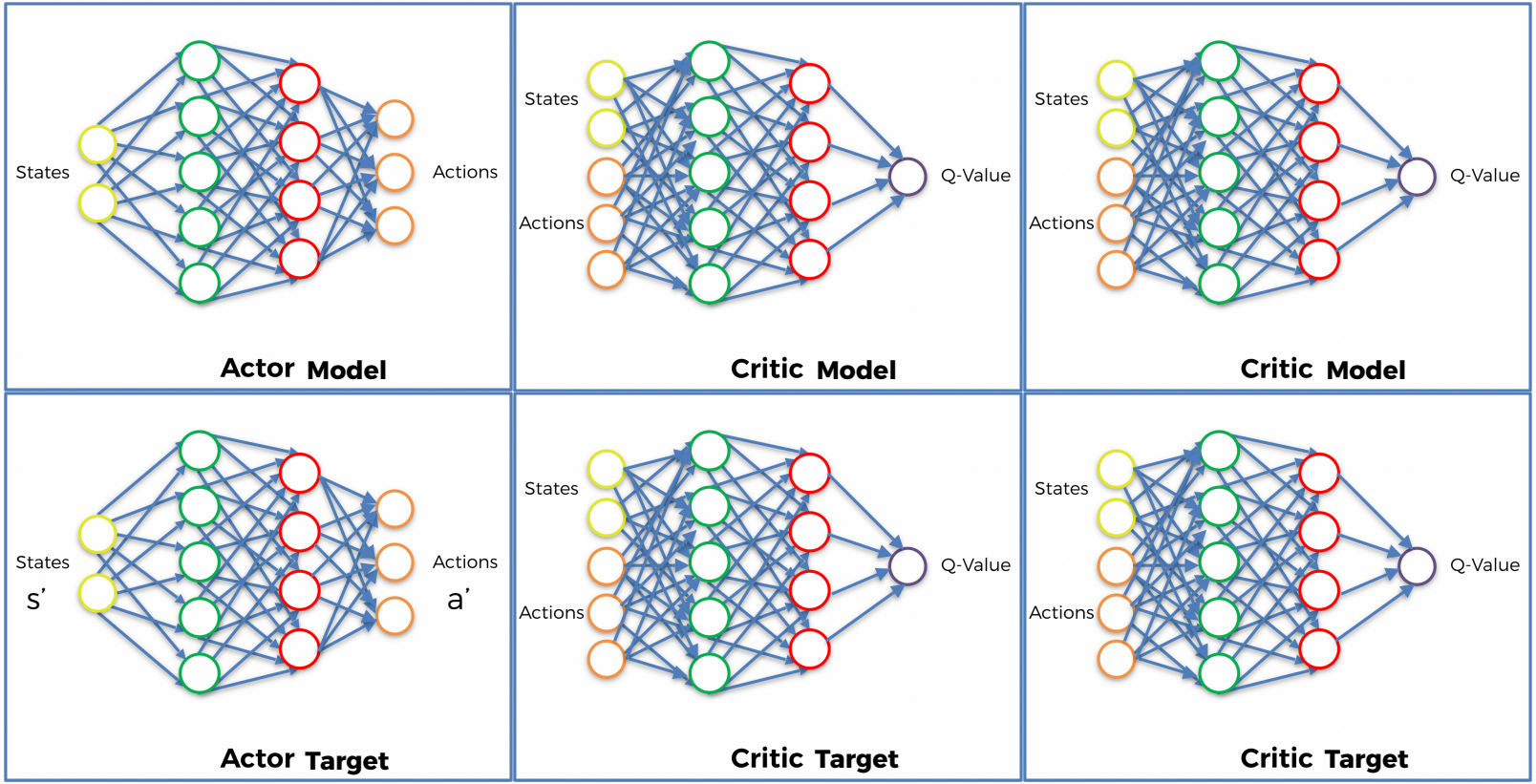
Twin-Delayed DDPG (TD3)

Once again?

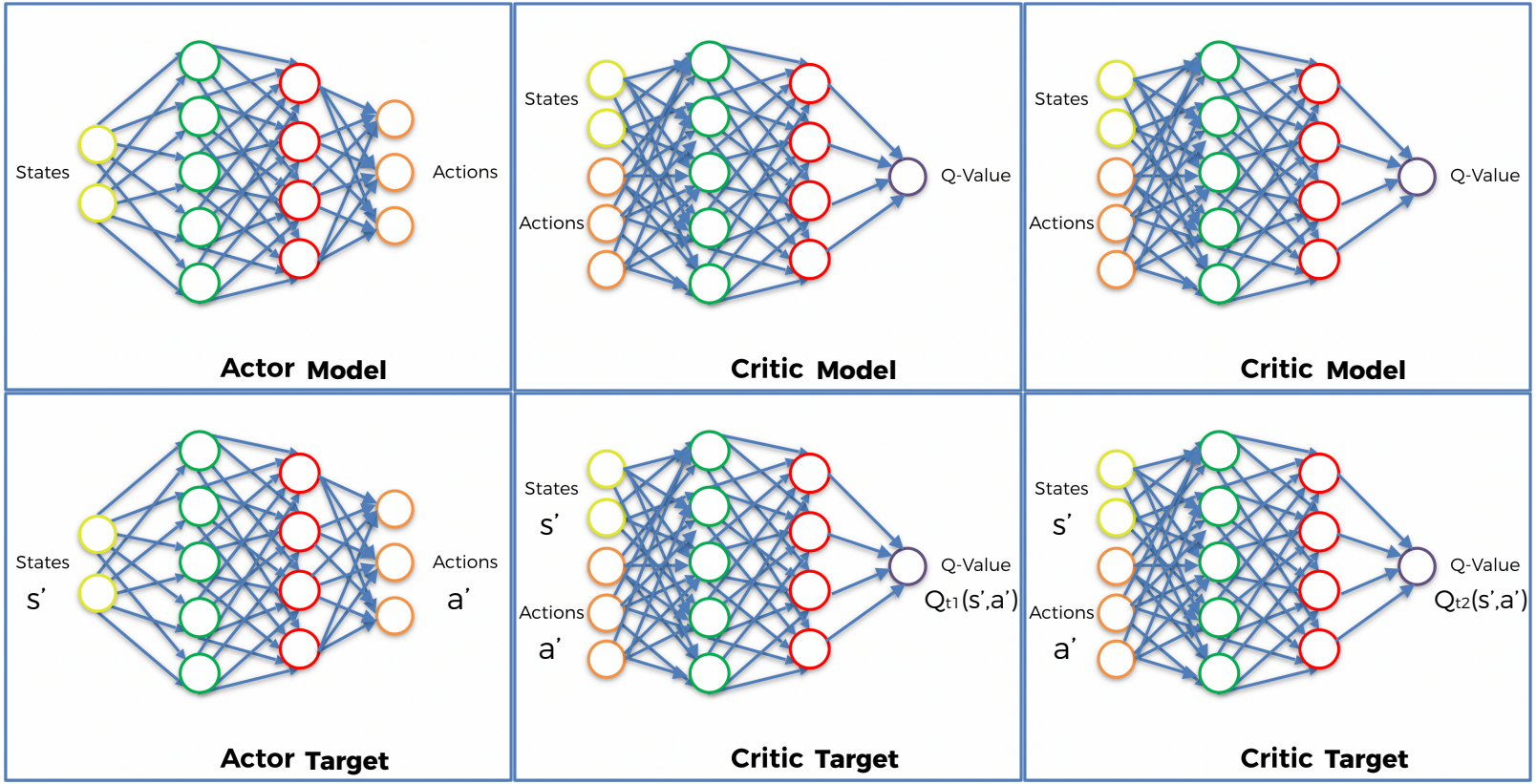
Twin-Delayed DDPG (TD3)



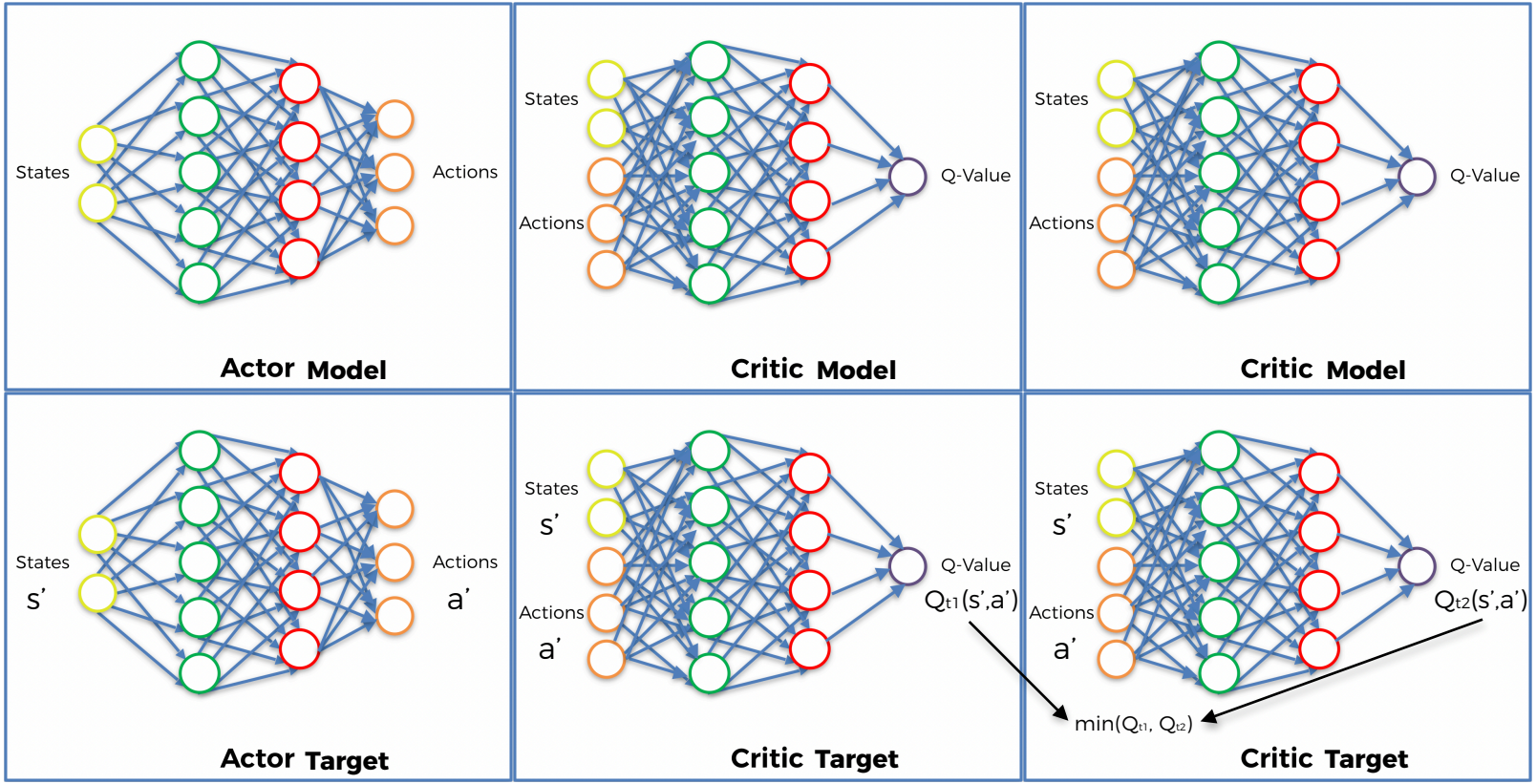
Twin-Delayed DDPG (TD3)



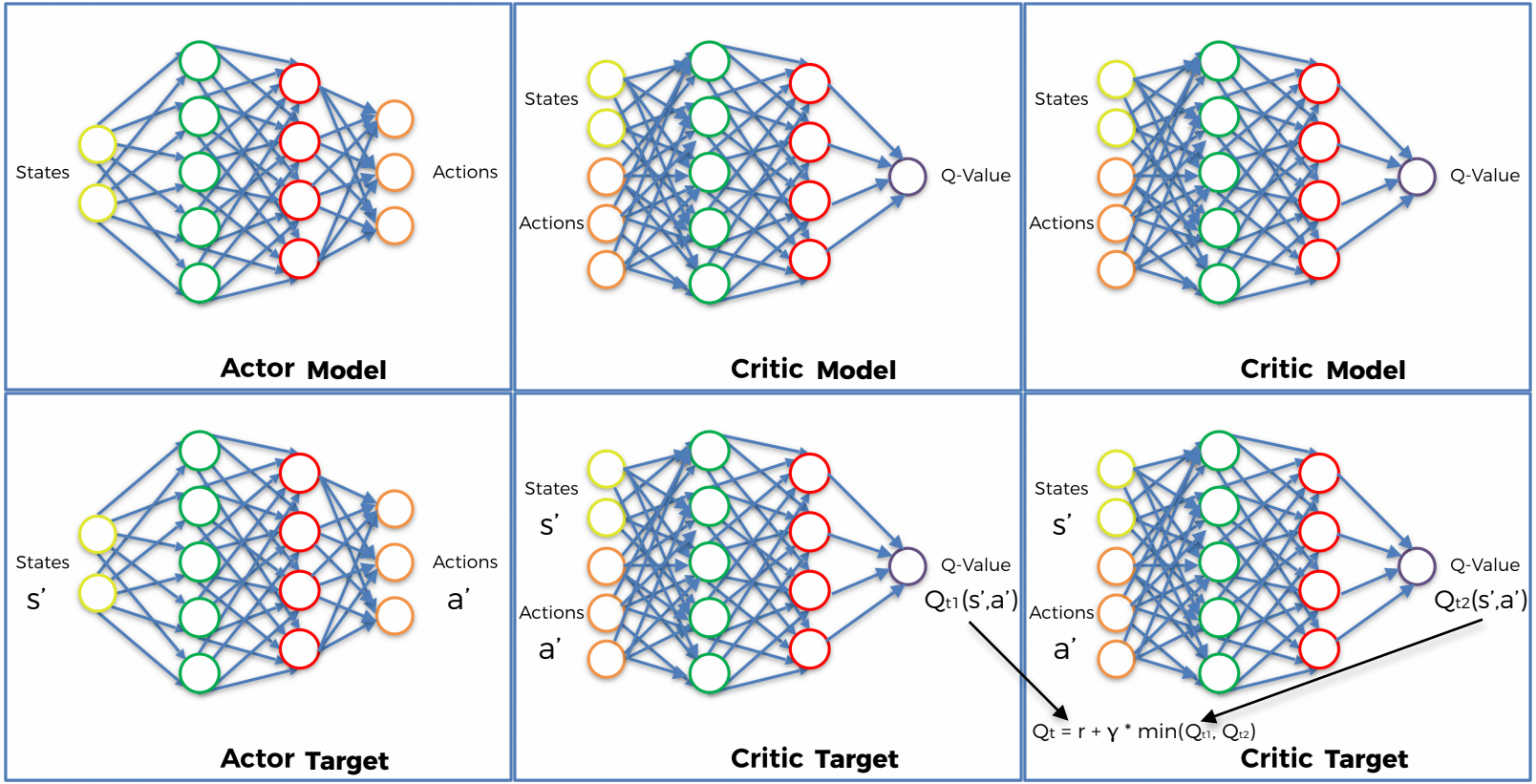
Twin-Delayed DDPG (TD3)



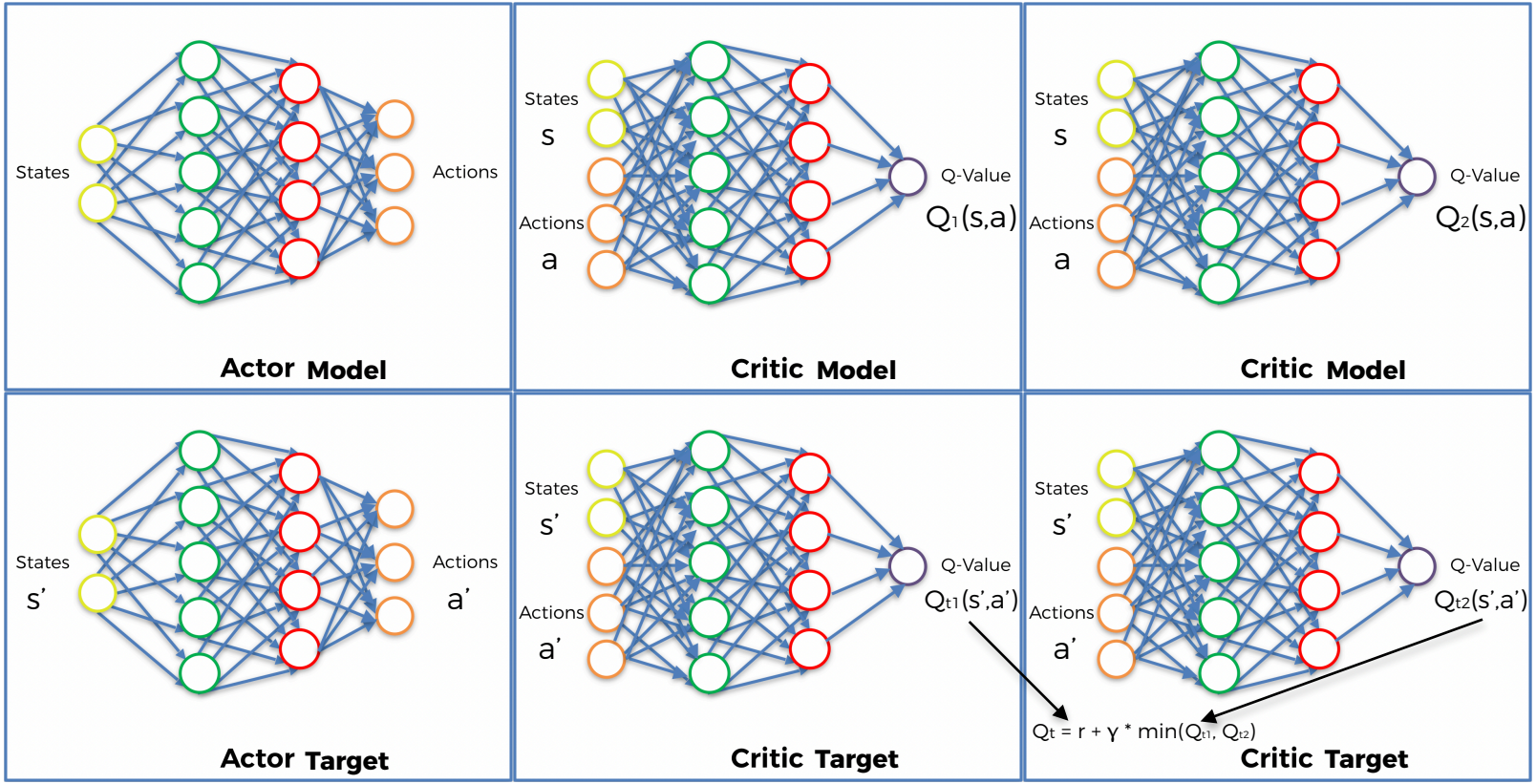
Twin-Delayed DDPG (TD3)



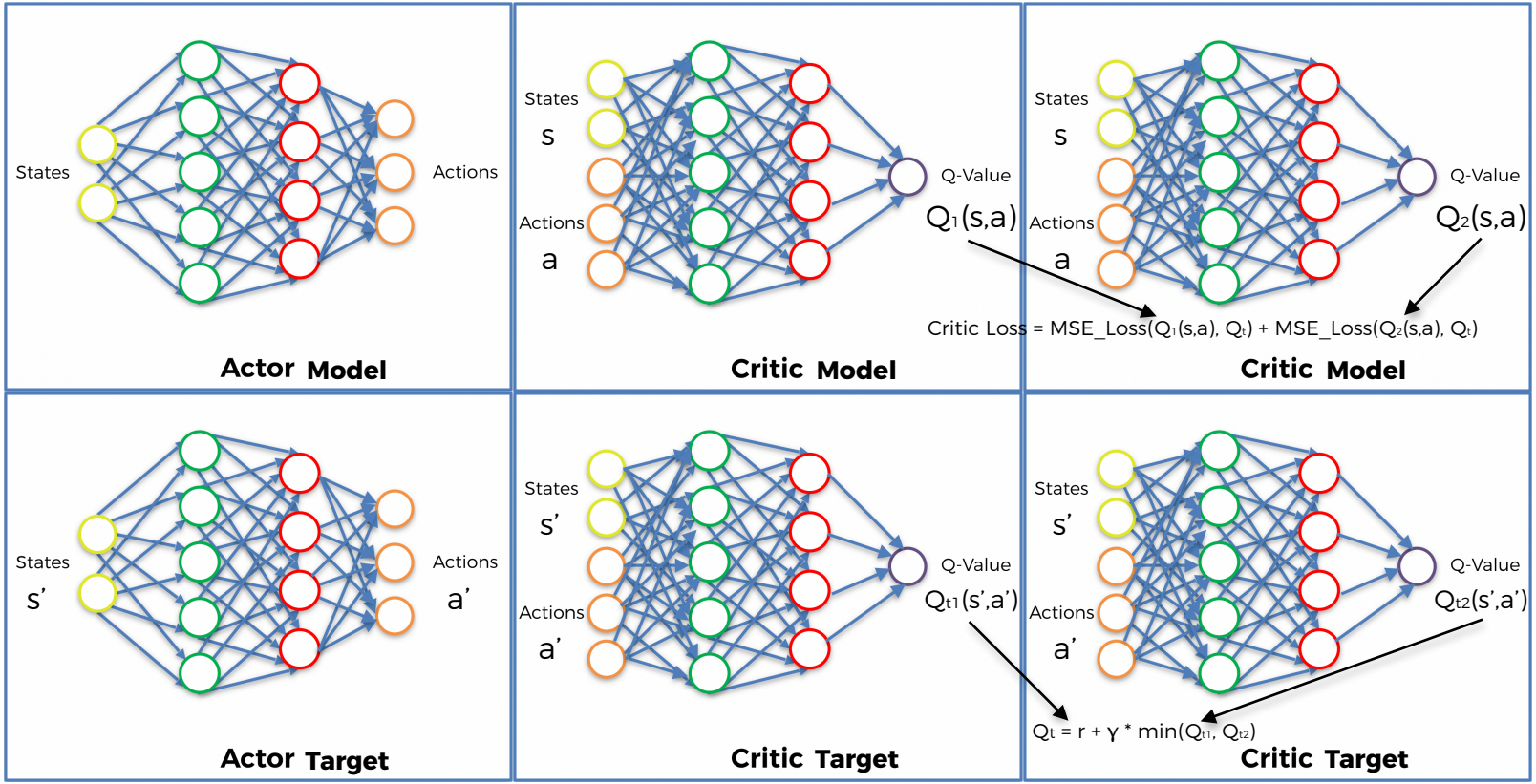
Twin-Delayed DDPG (TD3)



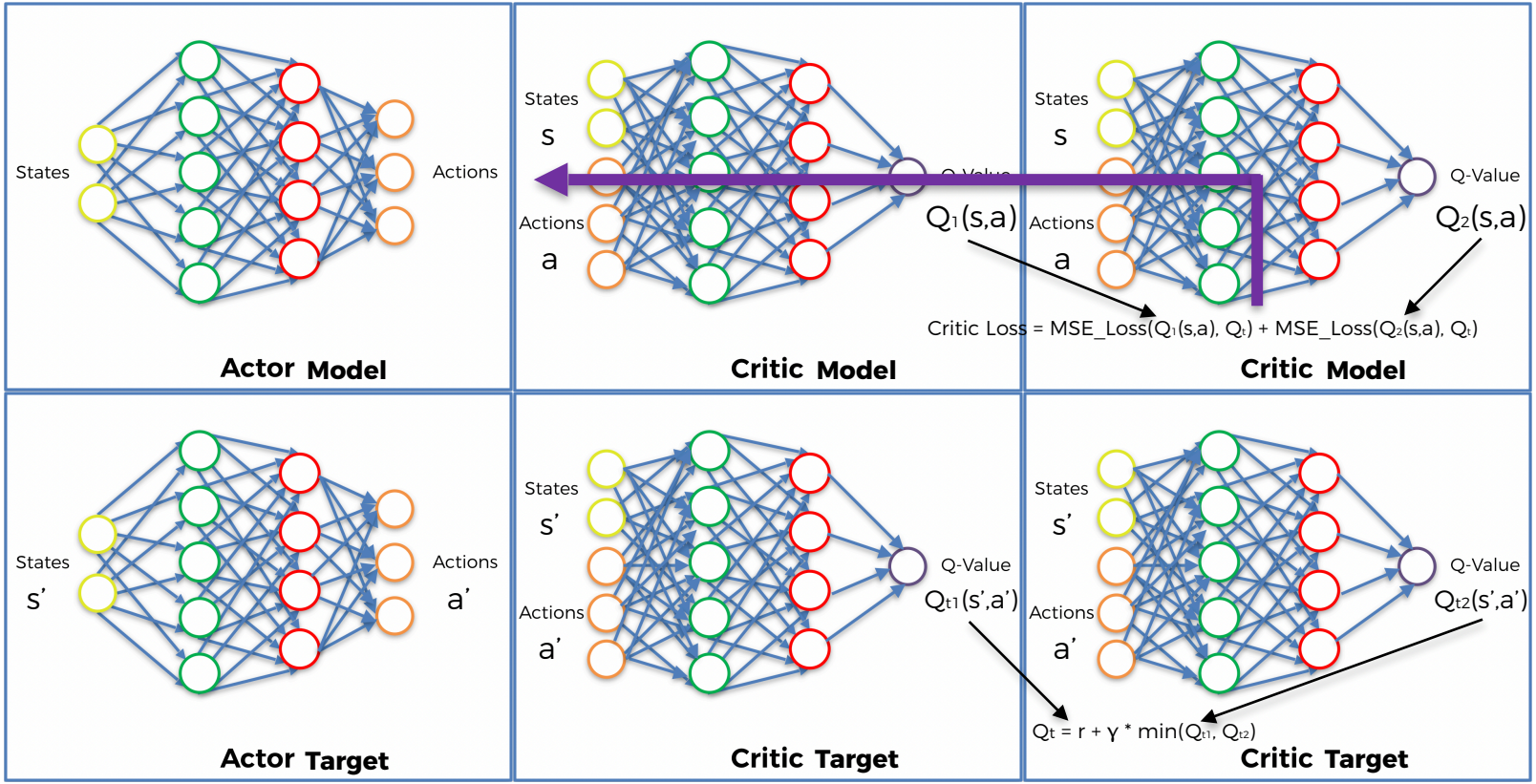
Twin-Delayed DDPG (TD3)



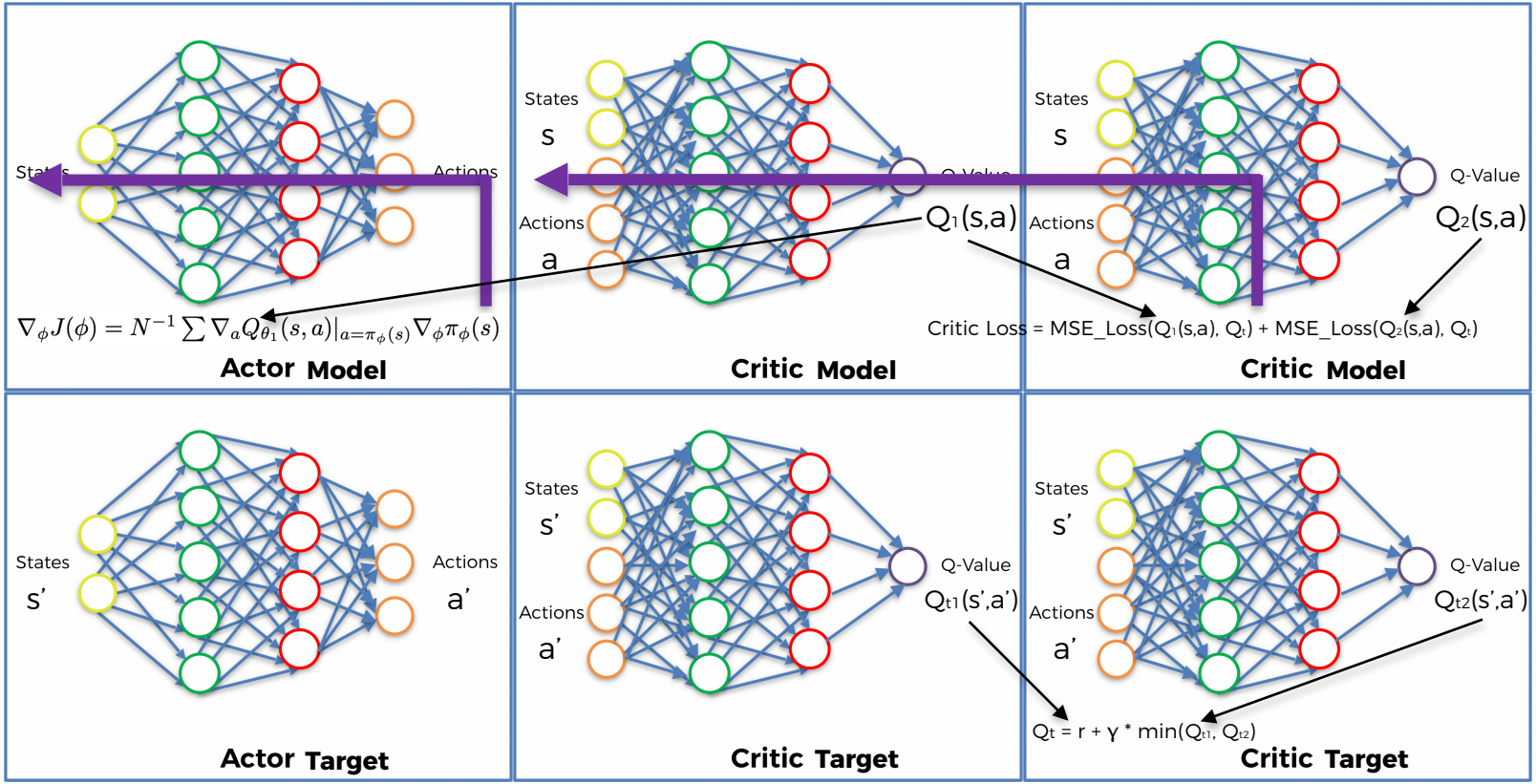
Twin-Delayed DDPG (TD3)



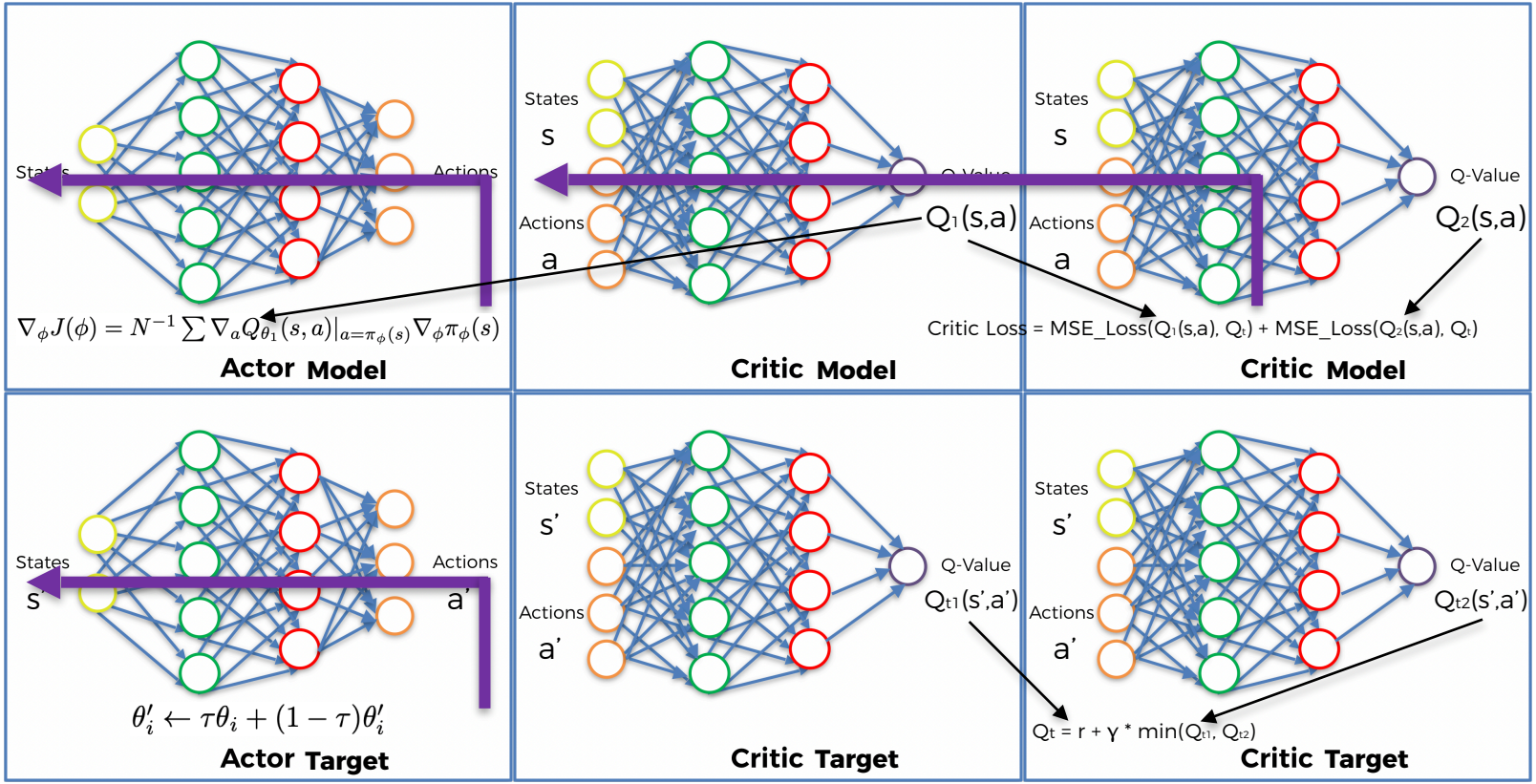
Twin-Delayed DDPG (TD3)



Twin-Delayed DDPG (TD3)



Twin-Delayed DDPG (TD3)



Twin-Delayed DDPG (TD3)