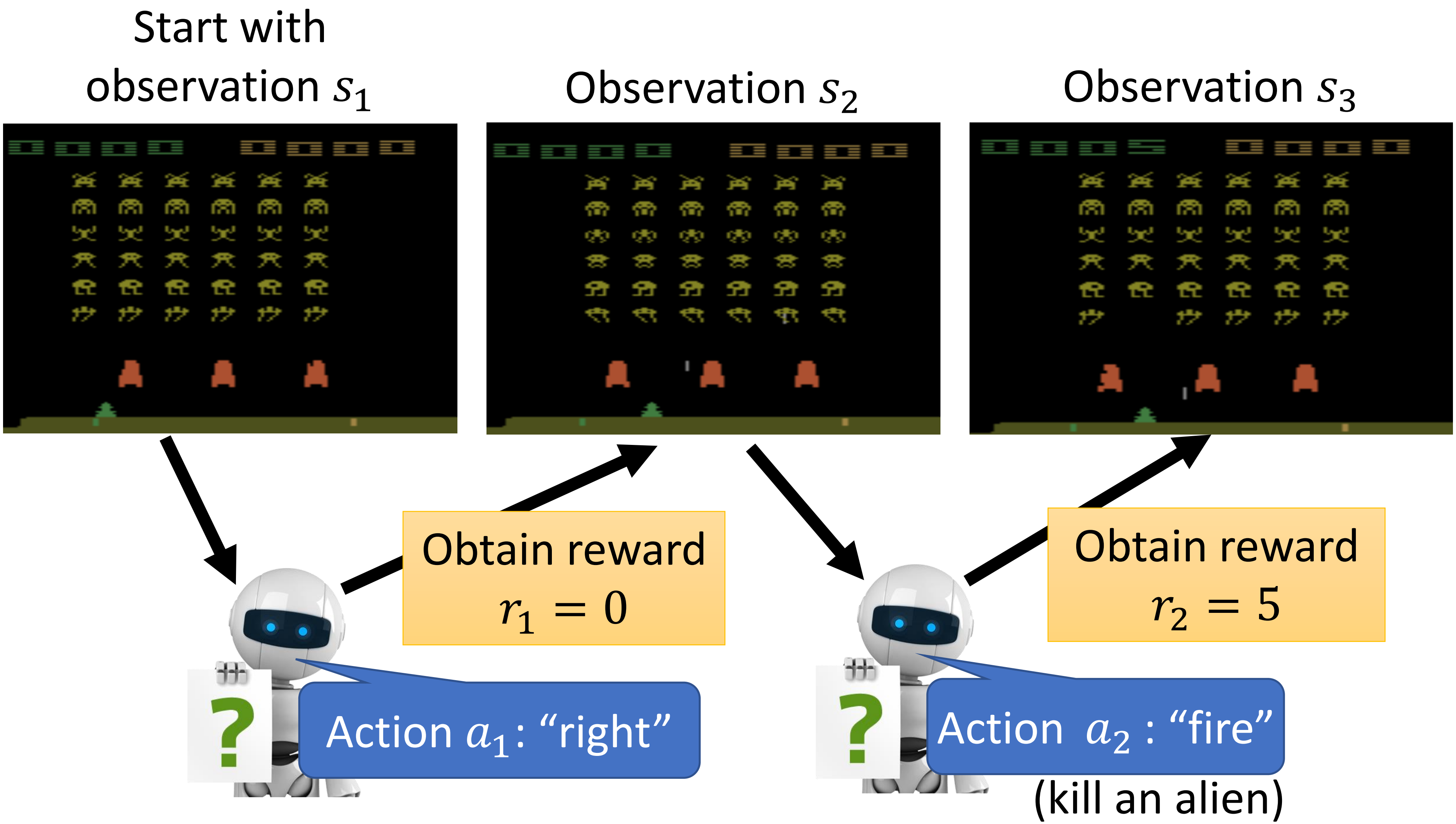
Deep
Reinforcement Learning

Example: Playing Video Game

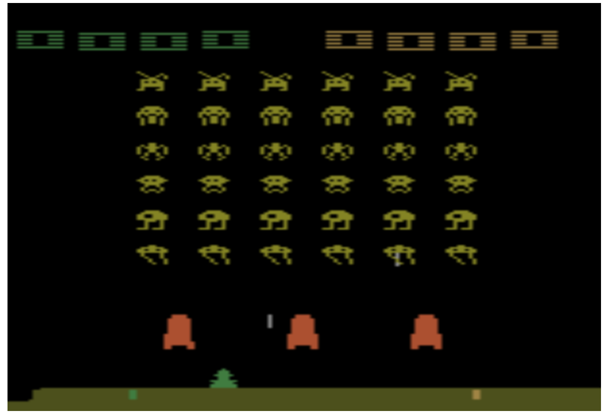


Usually there is some randomness in the environment

Example: Playing Video Game

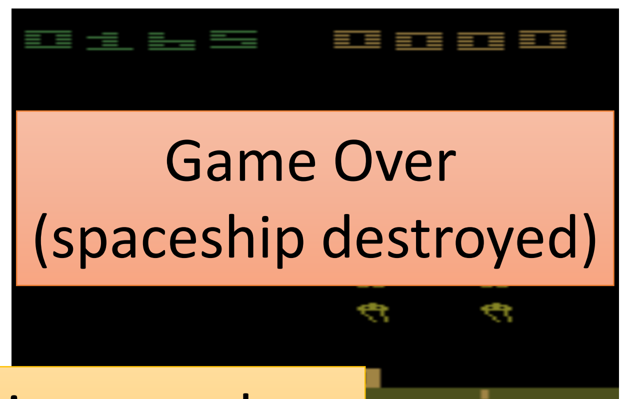
Start with observation s_1

Observation s_2



Observation s_3

After many turns



Obtain reward r_T

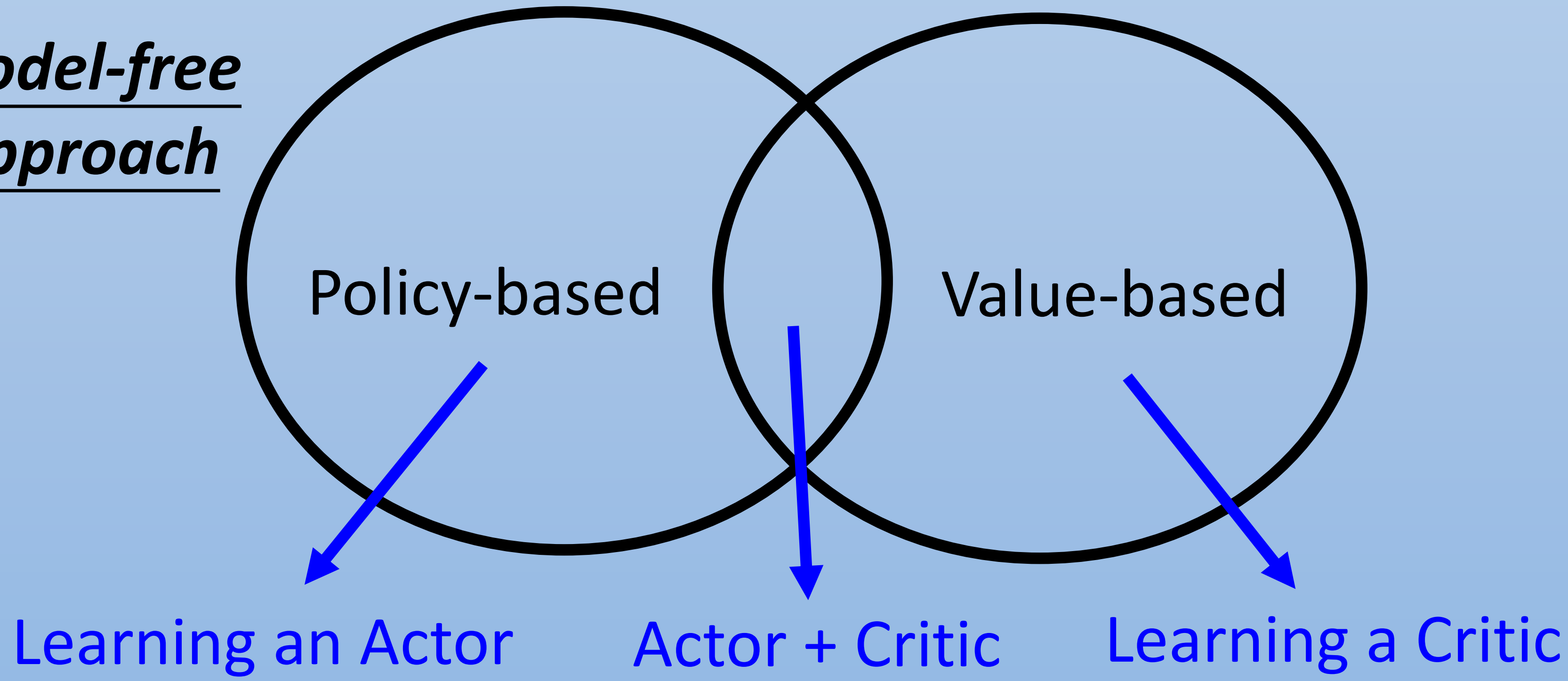
Action a_T

This is an ***episode***.

Learn to maximize the expected cumulative reward per episode

Approaches

Model-free Approach



Model-based Approach

On-policy v.s. Off-policy

- On-policy: The agent learned and the agent interacting with the environment is the same.
- Off-policy: The agent learned and the agent interacting with the environment is different.

阿光下棋

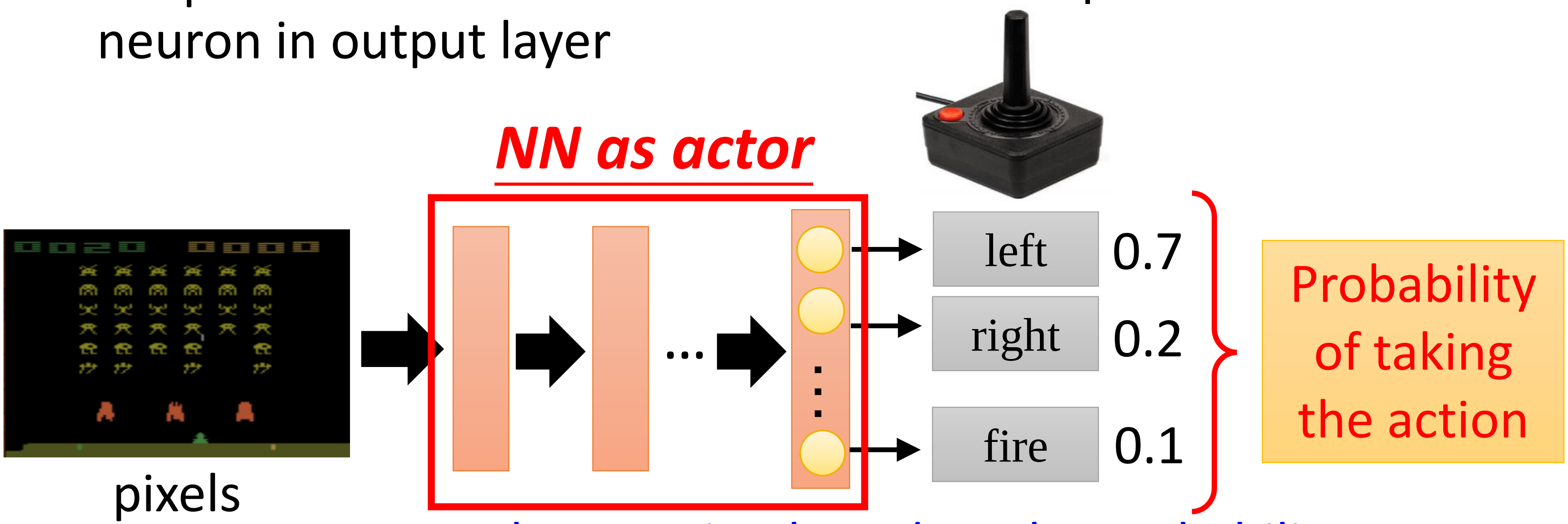
佐為下棋、阿光在旁邊看

Asynchronous Advantage Actor-Critic (A3C)

Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy P. Lillicrap, Tim Harley, David Silver, Koray Kavukcuoglu, “Asynchronous Methods for Deep Reinforcement Learning”, ICML, 2016

Actor is a Neural network

- Input of neural network: the observation of machine represented as a vector or a matrix
- Output neural network : each action corresponds to a neuron in output layer



Sample an action based on the probability or argmax

Actor can also have continuous action.

Actor – Goodness of an Actor

- Given an actor $\pi(s)$ with network parameter θ^{π}
- Use the actor $\pi(s)$ to play the video game
  - Start with observation s_1
  - Machine decides to take a_1
  - Machine obtains reward r_1
  - Machine sees observation s_2
  - Machine decides to take a_2
  - Machine obtains reward r_2
  - Machine sees observation s_3
  - ……
  - Machine decides to take a_T
  - Machine obtains reward r_T END

Total reward: $R = \sum_{t=1}^{T} r_t$

Even with the same actor, R is different each time

Randomness in the actor and the game

We define $\bar{R}_{\theta^{\pi}}$ as the ***expected total reward***

$\bar{R}_{\theta^{\pi}}$ evaluates the goodness of an actor $\pi(s)$

Actor – Policy Gradient

$\theta^{\pi'} \leftarrow \theta^{\pi} + \eta \nabla \bar{R}_{\theta^{\pi}}$ Using θ^{π} to obtain $\{\tau^1, \tau^2, \cdots, \tau^N\}$

$$\nabla \bar{R}_{\theta^{\pi}} \approx \frac{1}{N}\sum_{n=1}^{N} R(\tau^n) \nabla log P(\tau^n|\theta^{\pi}) = \frac{1}{N}\sum_{n=1}^{N} R(\tau^n) \sum_{t=1}^{T_n} \nabla log p(a_t^n|s_t^n, \theta^{\pi})$$

$$= \frac{1}{N}\sum_{n=1}^{N}\sum_{t=1}^{T_n} R(\tau^n) \nabla log p(a_t^n|s_t^n, \theta^{\pi})$$

What if we replace $R(\tau^n)$ with r_t^n ……

If in τ^n machine takes a_t^n when seeing s_t^n

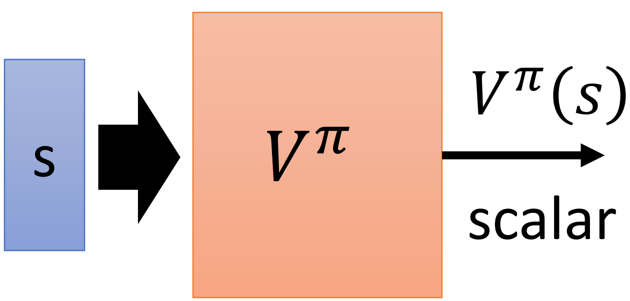
$R(\tau^n)$ is positive → Tuning θ to increase $p(a_t^n|s_t^n)$

$R(\tau^n)$ is negative → Tuning θ to decrease $p(a_t^n|s_t^n)$

It is very important to consider the cumulative reward $R(\tau^n)$ of the whole trajectory τ^n instead of immediate reward r_t^n

Critic

- A critic does not determine the action.
- Given an actor π, it evaluates the how good the actor is
- State value function $V^{\pi}(s)$
  - When using actor π, the *cumulated* reward expects to be obtained after seeing observation (state) s



$V^{\pi}(s)$ is large

$V^{\pi}(s)$ is smaller

Critic

$V^{以前的阿光}(大馬步飛) = \text{bad}$

$V^{變強的阿光}(大馬步飛) = \text{good}$

How to estimate $V^{\pi}(s)$

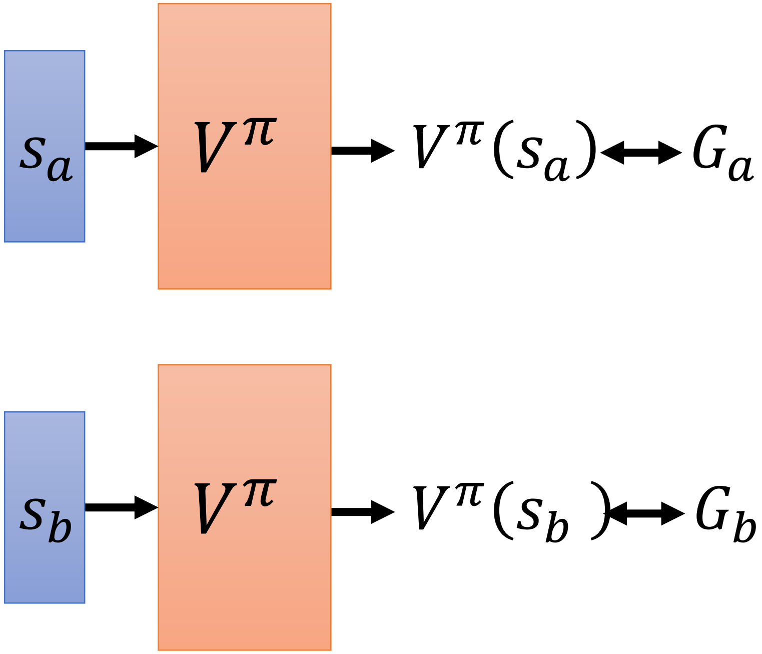
- Monte-Carlo based approach
  - The critic watches π playing the game

After seeing s_a,

Until the end of the episode, the cumulated reward is G_a

After seeing s_b,

Until the end of the episode, the cumulated reward is G_b

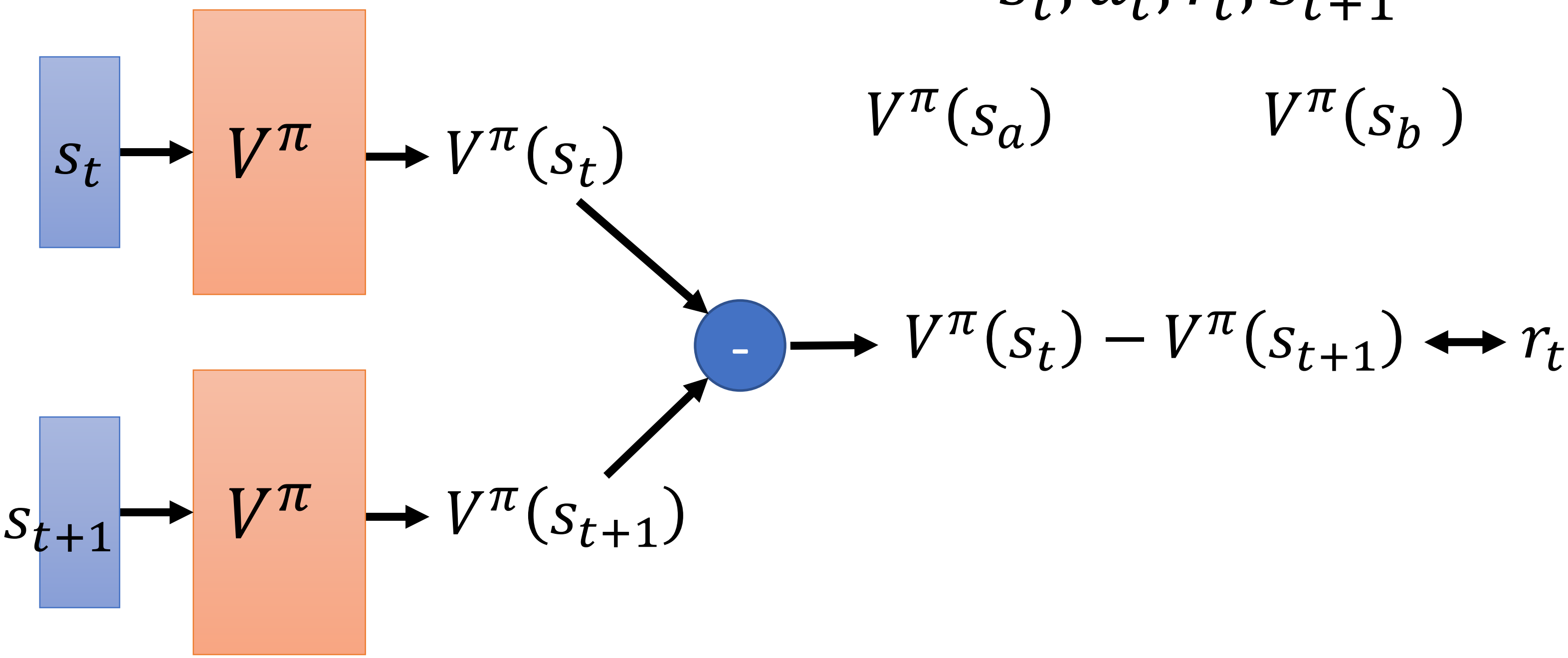


How to estimate $V^\pi(s)$

- Temporal-difference approach

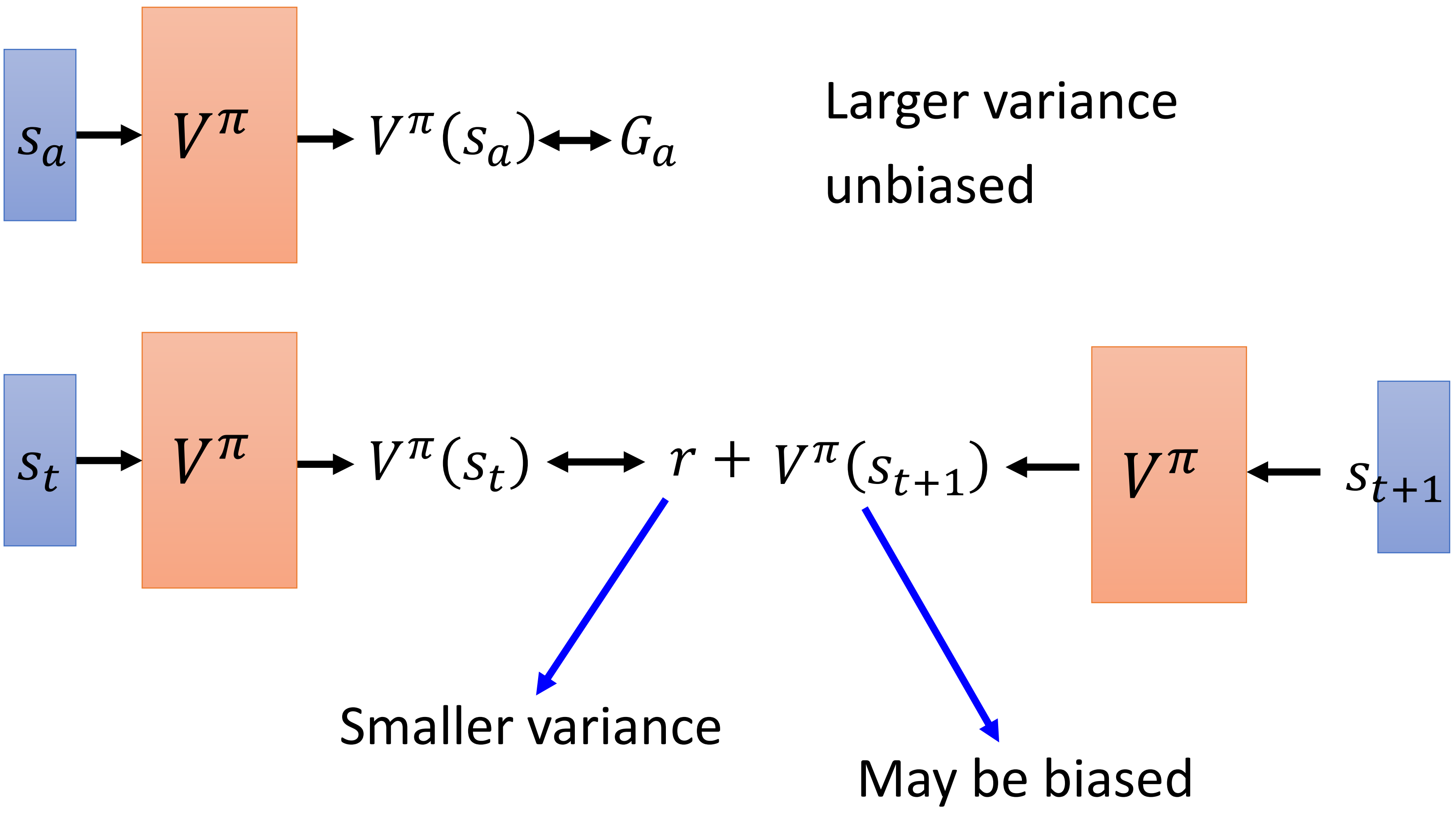
$$\cdots s_t, a_t, r_t, s_{t+1} \cdots$$

$V^\pi(s_a)$ $\quad V^\pi(s_b)$



Some applications have very long episodes, so that delaying all learning until an episode's end is too slow.

MC v.s. TD



s_a
V^π
$V^\pi(s_a) \leftrightarrow G_a$
Larger variance
unbiased
s_t
V^π
$V^\pi(s_t) \leftrightarrow r + V^\pi(s_{t+1})$
V^π
s_{t+1}
Smaller variance
May be biased

MC v.s. TD

[Sutton, v2, Example 6.4]

- The critic has the following 8 episodes
  - $s_a, r = 0, s_b, r = 0$, END
  - $s_b, r = 1$, END
  - $s_b, r = 1$, END
  - $s_b, r = 1$, END
  - $s_b, r = 1$, END
  - $s_b, r = 1$, END
  - $s_b, r = 1$, END
  - $s_b, r = 0$, END

$$V^\pi(s_b) = 3/4$$

$V^\pi(s_a) =?$ 0? 3/4?

Monte-Carlo: $V^\pi(s_a) = 0$

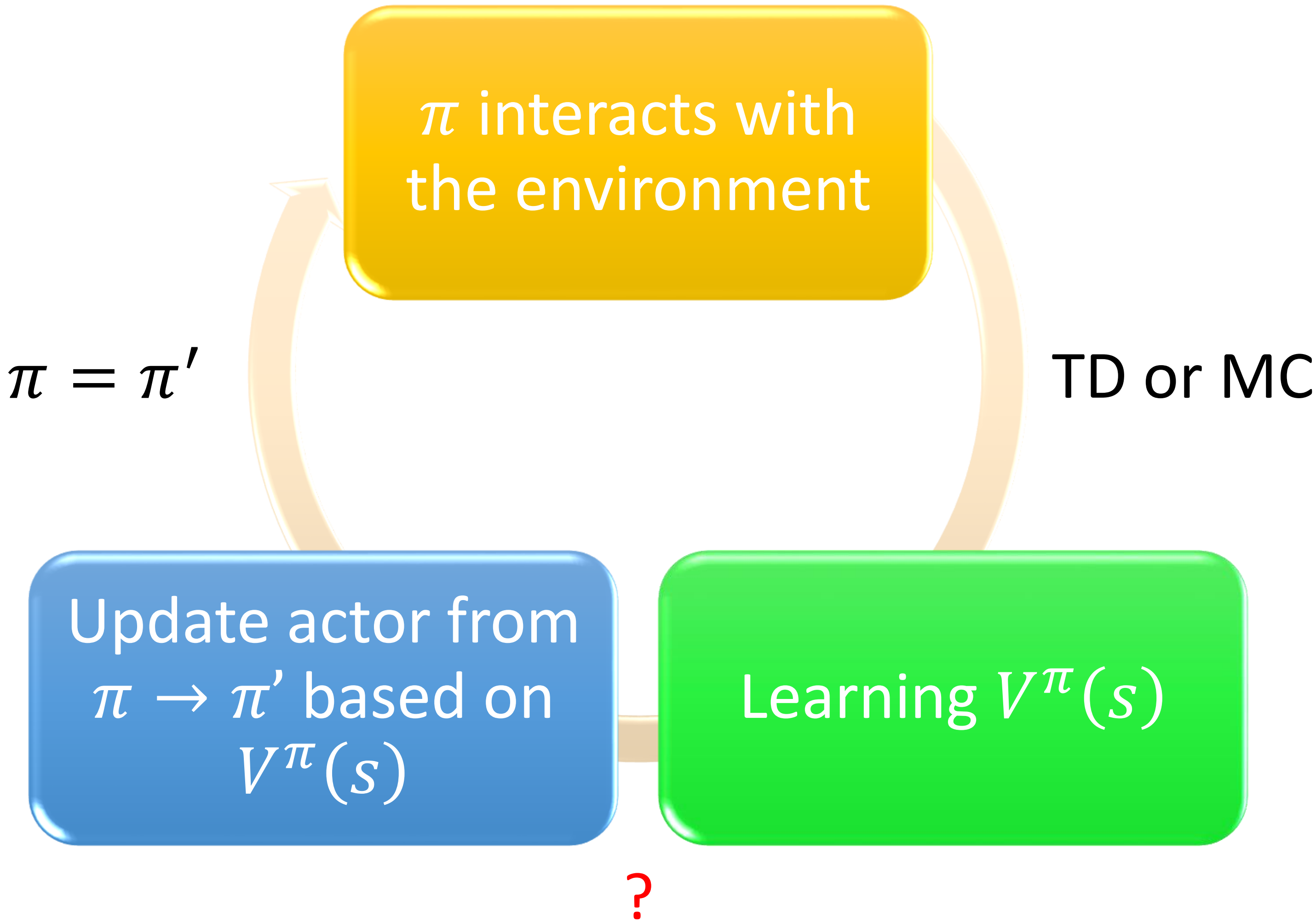
Temporal-difference:

$$V^\pi(s_a) + r = V^\pi(s_b)$$

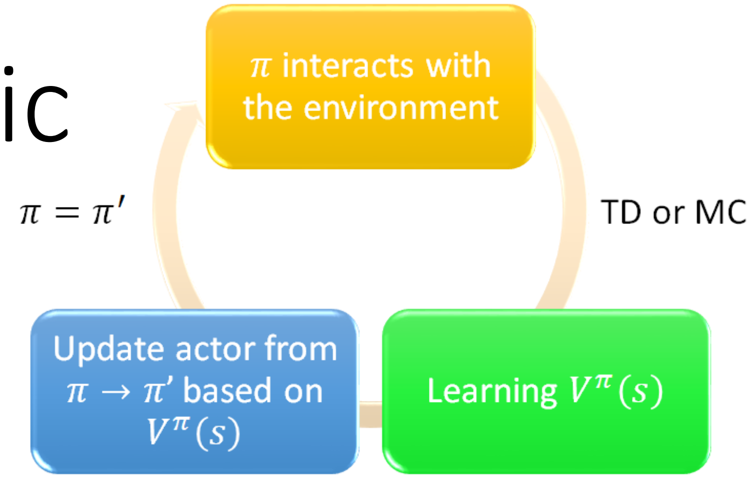
3/4 0 3/4

(The actions are ignored here.)

Actor-Critic



Advantage Actor-Critic



$$\theta^{\pi'} \leftarrow \theta^{\pi} + \eta \nabla \bar{R}_{\theta^{\pi}}$$

$$\nabla \bar{R}_{\theta^{\pi}} = \frac{1}{N} \sum_{n=1}^{N} \sum_{t=1}^{T_n} \boxed{R(\tau^n)} \nabla logp(a_t^n | s_t^n, \theta^{\pi})$$

Evaluated by critic

Advantage Function: $\underbrace{r_t^n}_{} - \underbrace{\left(V^{\pi}(s_t^n) - V^{\pi}(s_{t+1}^n)\right)}_{}$

Baseline is added

The reward r_t^n we truly obtain when taking action a_t^n

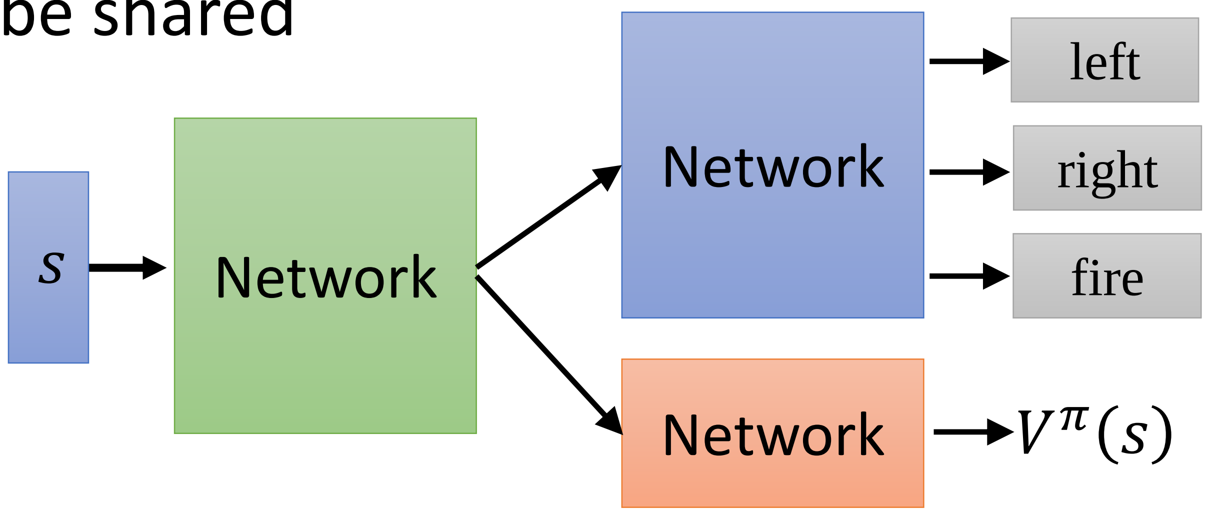
Expected reward r_t^n we obtain if we use actor π

Positive advantage function $\Leftrightarrow$ Increasing the prob. of action a_t^n

Negative advantage function $\Leftrightarrow$ decreasing the prob. of action a_t^n

Advantage Actor-Critic

- Tips
  - The parameters of actor $\pi(s)$ and critic $V^{\pi}(s)$ can be shared

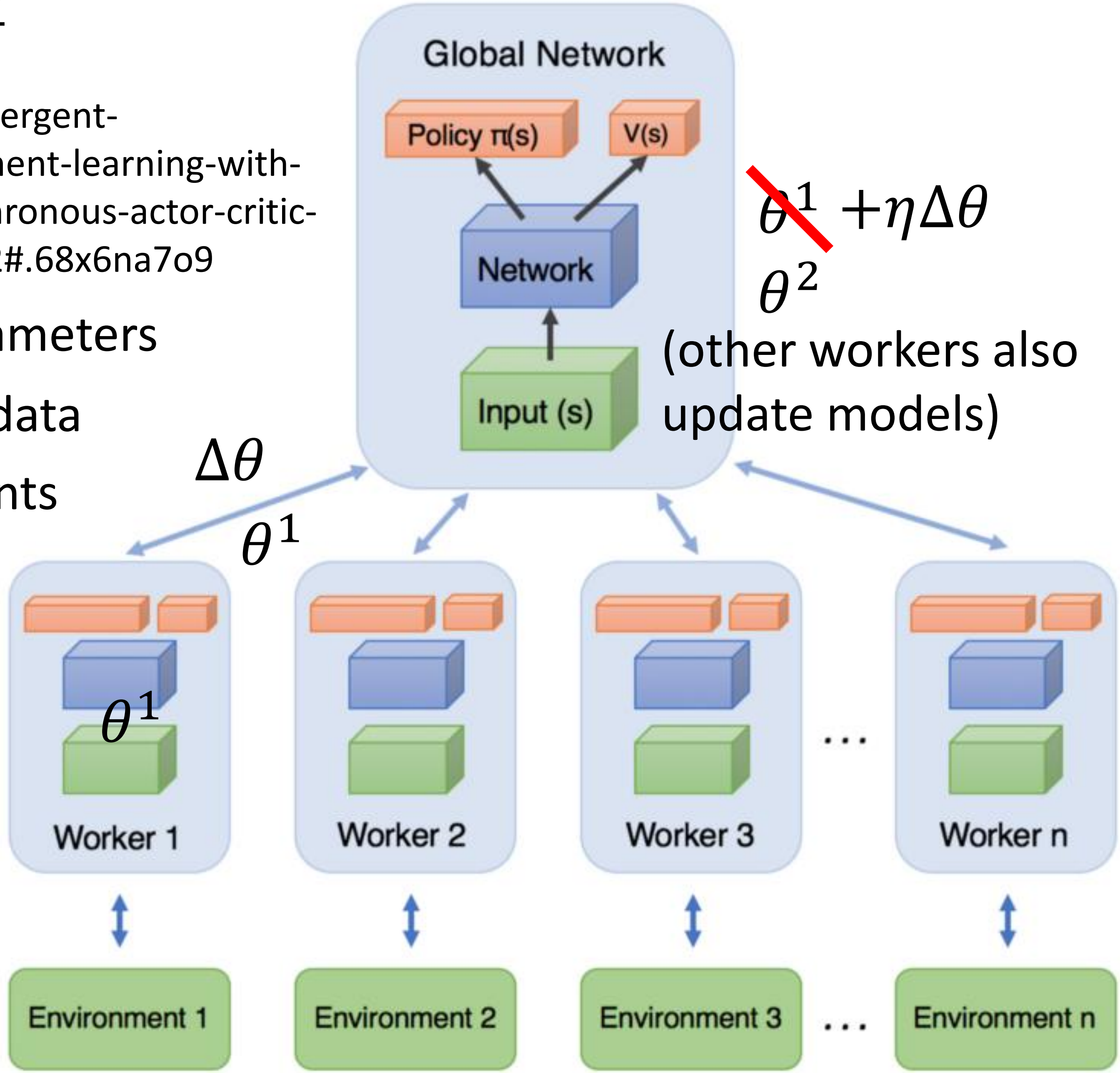


  - Use output entropy as regularization for $\pi(s)$
    - Larger entropy is preferred → exploration

Asynchronous

Source of image: https://medium.com/emergent-future/simple-reinforcement-learning-with-tensorflow-part-8-asynchronous-actor-critic-agents-a3c-c88f72a5e9f2#.68x6na7o9

1. Copy global parameters
2. Sampling some data
3. Compute gradients
4. Update global models



$\cancel{\theta^1} + \eta\Delta\theta$

θ^2

(other workers also update models)

$\Delta\theta$

θ^1

θ^1

$\Delta\theta$

Pathwise Derivative Policy Gradient

David Silver, Guy Lever, Nicolas Heess, Thomas Degris, Daan Wierstra, Martin Riedmiller, “Deterministic Policy Gradient Algorithms”, ICML, 2014

Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, Daan Wierstra, “CONTINUOUS CONTROL WITH DEEP REINFORCEMENT LEARNING”, ICLR, 2016

Actor

Critic

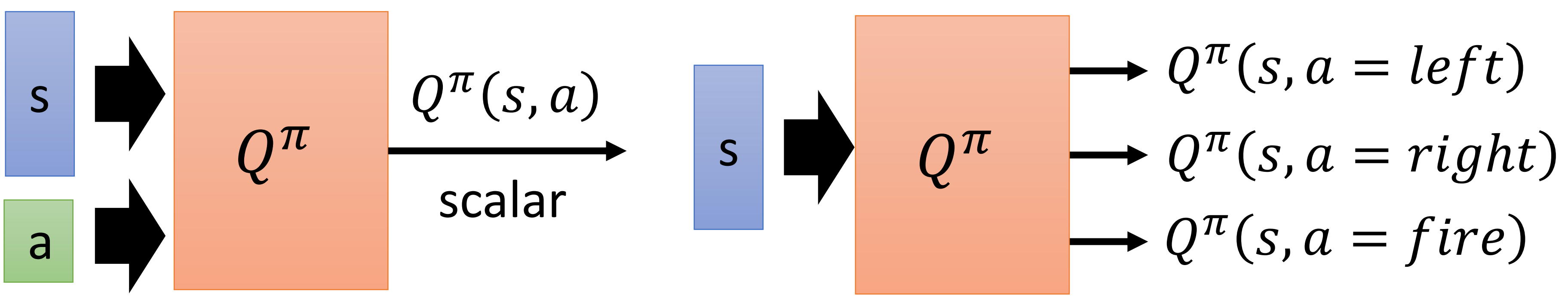
Pathwise derivative policy gradient

Original Actor-critic

http://www.cartomad.com/comic/109000081104011.html

Another Critic

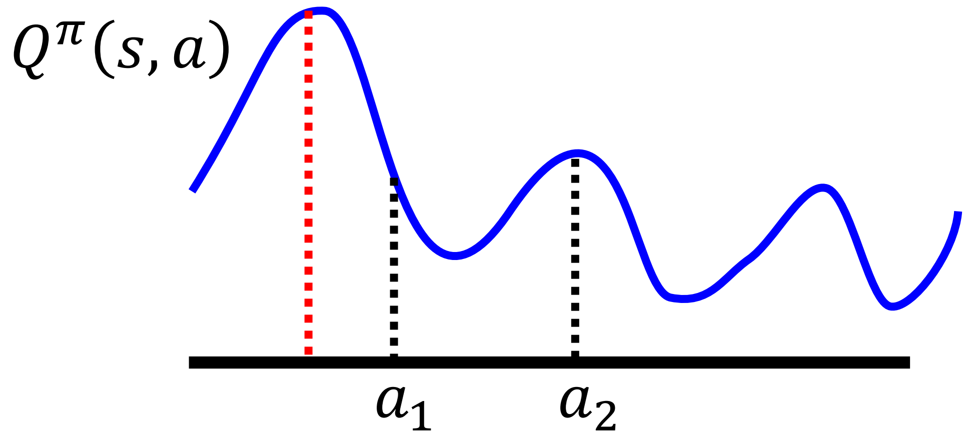
- State-action value function $Q^{\pi}(s, a)$
  - When using actor π, the *cumulated* reward expects to be obtained after seeing observation s and taking a



for discrete action only

Another Way to use Critic

Original Actor-critic

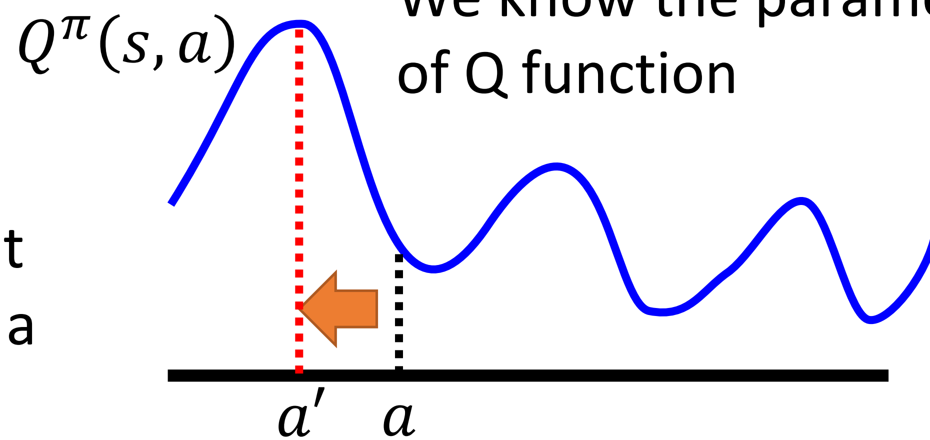


decrease increase

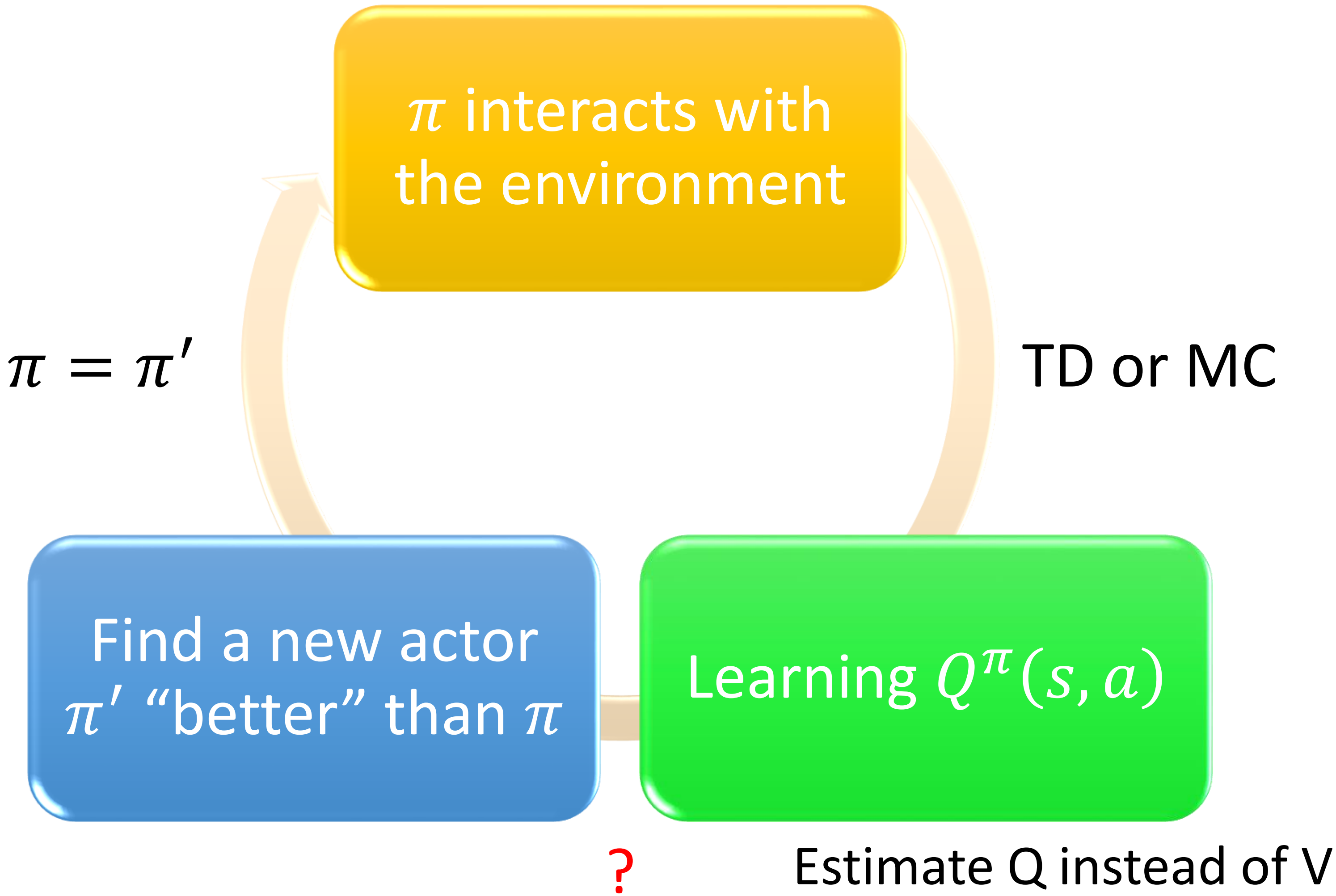
Q-Learning

We know the parameters of Q function

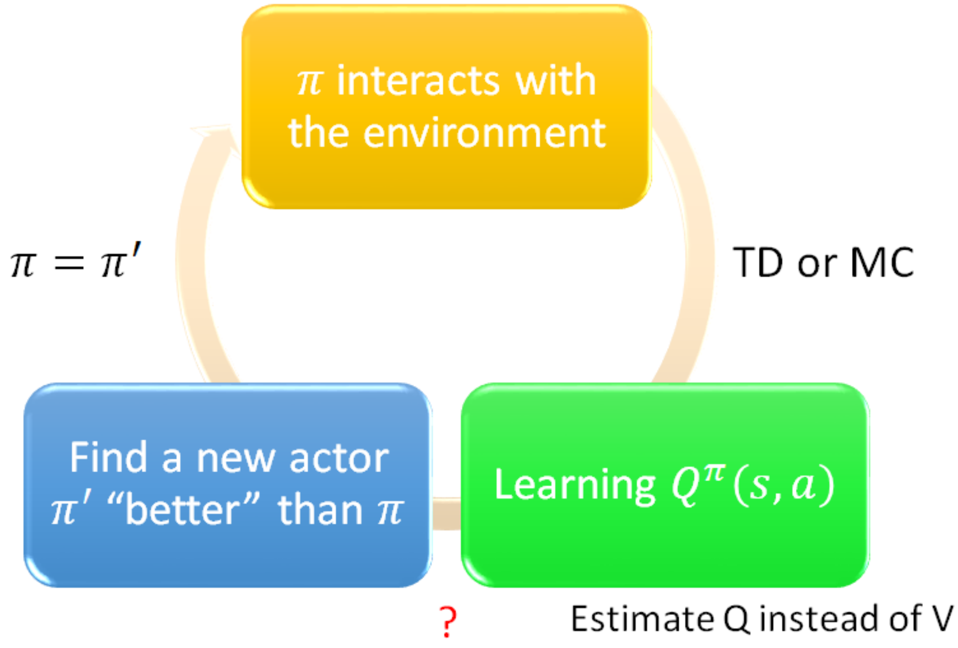
From Q function we know that taking a' at state s is better than a



Q-Learning


π interacts with the environment
TD or MC
Learning $Q^{\pi}(s,a)$
Estimate Q instead of V
?
Find a new actor π' "better" than π
$\pi = \pi'$

Q-Learning



- Given $Q^{\pi}(s,a)$, find a new actor π' "better" than π
  - "Better": $V^{\pi'}(s) \geq V^{\pi}(s)$, for all state s

$$\pi'(s) = arg\max_{a} Q^{\pi}(s,a)$$

➢ π' does not have extra parameters. It depends on Q

➢ Not suitable for continuous action a (solve it later)

Q-Learning

$$\pi'(s) = arg\max_a Q^\pi(s,a)$$

$V^{\pi'}(s) \geq V^\pi(s)$, for all state s

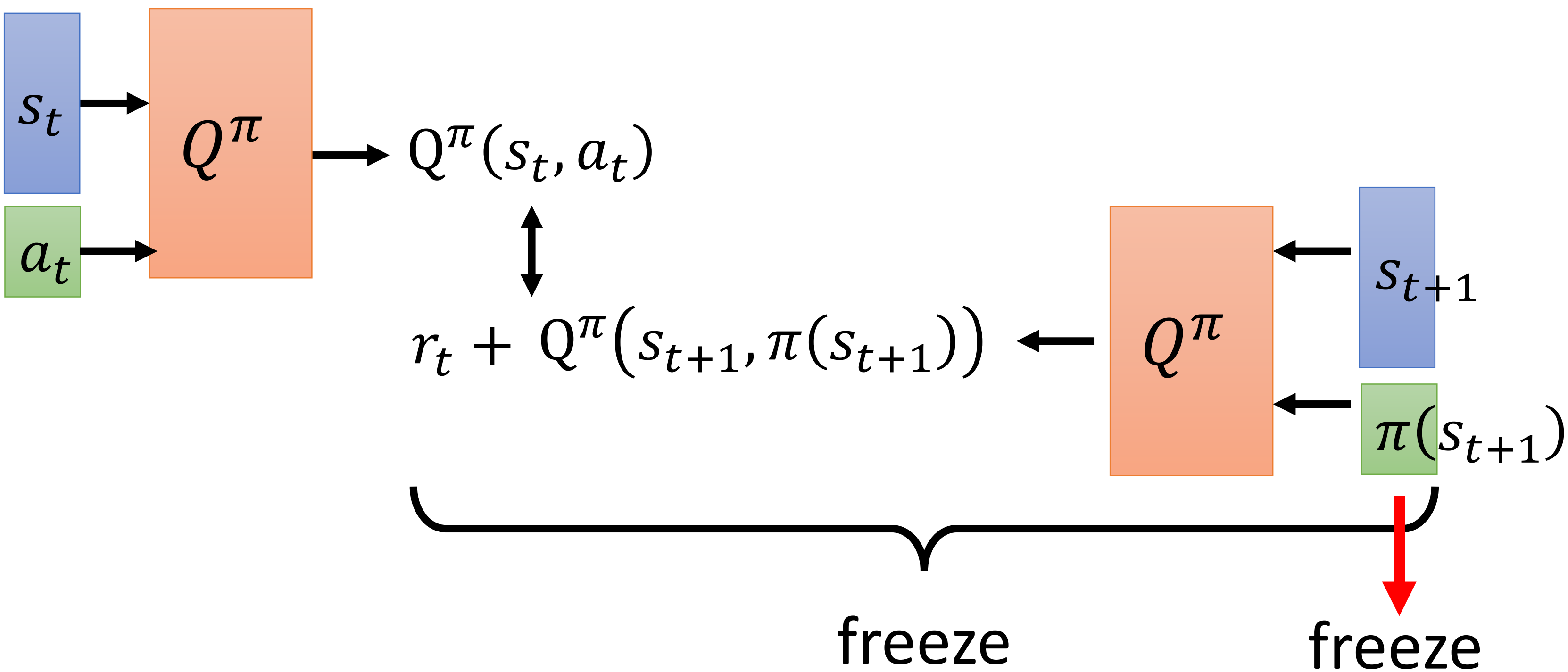
$$V^\pi(s) = Q^\pi\big(s, \pi(s)\big)$$
$$\leq \max_a Q^\pi(s,a) = Q^\pi\big(s, \pi'(s)\big) = V^{\pi'}(s)$$

$$V^\pi(s) \leq Q^\pi\big(s, \pi'(s)\big)$$
$$= E_{\pi'}[r_{t+1} + V^\pi(s_{t+1})|s_t = s]$$
$$\leq E_{\pi'}[r_{t+1} + Q^\pi\big(s_{t+1}, \pi'(s_{t+1})\big)|s_t = s]$$
$$= E_{\pi'}[r_{t+1} + r_{t+2} + V^\pi(s_{t+2})|s_t = s]$$
$$\leq E_{\pi'}[r_{t+1} + r_{t+2} + Q^\pi\big(s_{t+2}, \pi'(s_{t+2})\big)|s_t = s]$$
$$\dots\dots \leq V^{\pi'}(s)$$

Estimate $Q^\pi(s,a)$ by TD

$$\cdots s_t, a_t, r_t, s_{t+1} \cdots$$

$$Q^\pi(s_t, a_t) \qquad Q^\pi\big(s_{t+1}, \pi(s_{t+1})\big)$$



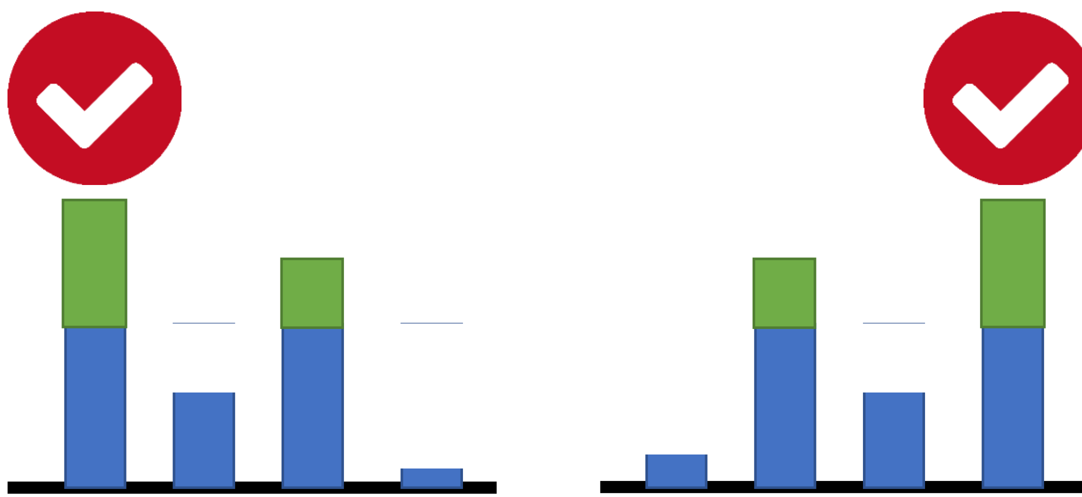
Double DQN

- Q value is usually over estimate

$$Q(s_t, a_t) \longleftrightarrow r_t + \max_a Q(s_{t+1}, a)$$

Tend to select the action that is over-estimated

$Q(s_{t+1}, a)$



Double DQN

- Q value is usually over estimate

$$Q(s_t, a_t) \longleftrightarrow r_t + \max_a Q(s_{t+1}, a)$$

- Double DQN: two functions Q and Q'

$$Q(s_t, a_t) \longleftrightarrow r_t + Q'\left(s_{t+1}, arg \max_a Q(s_{t+1}, a)\right)$$

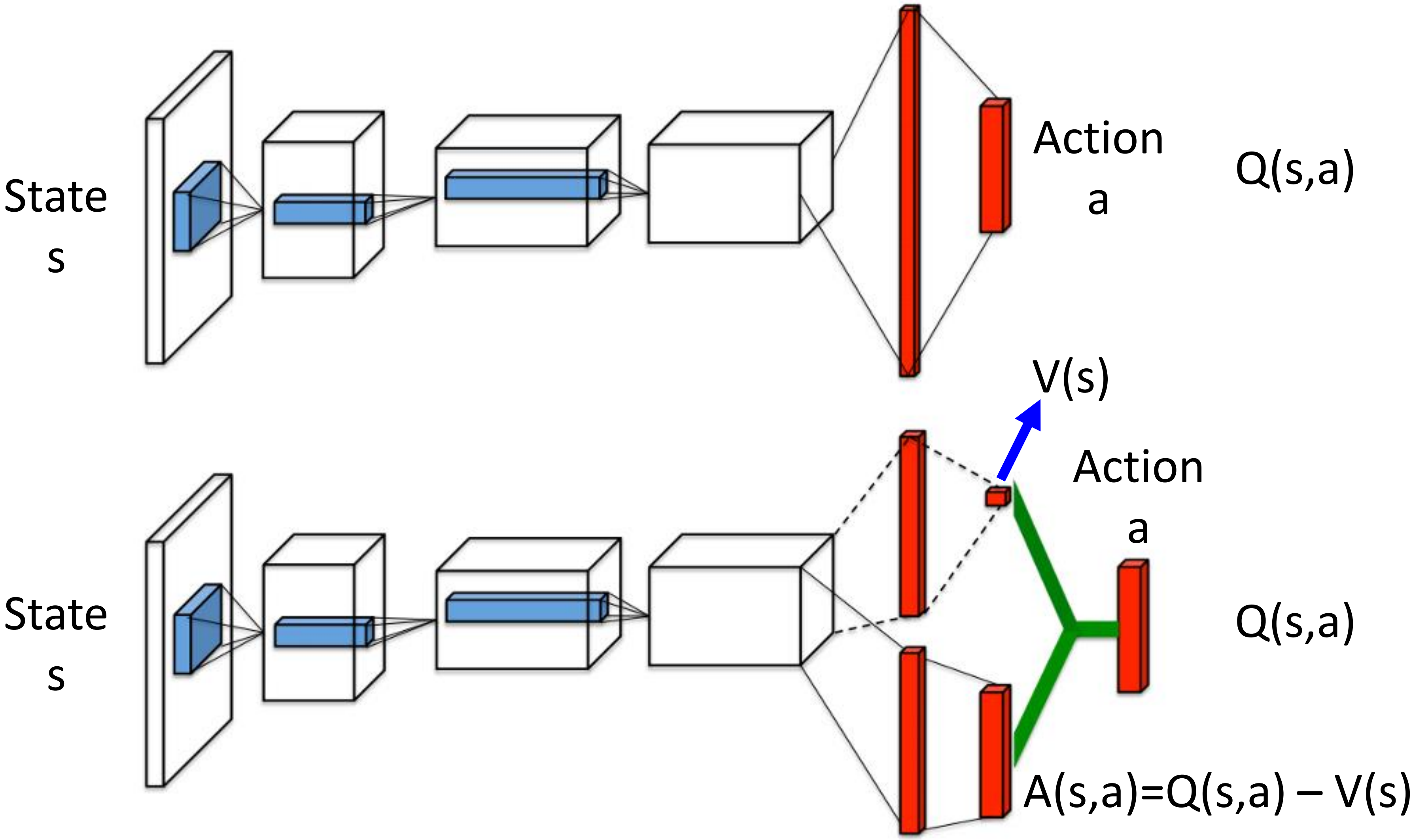
If Q over-estimate a, so it is selected. Q' would give it proper value.

How about Q' overestimate? The action will not select by Q.

Hado V. Hasselt, "Double Q-learning", NIPS 2010
Hado van Hasselt, Arthur Guez, David Silver, "Deep Reinforcement Learning with Double Q-learning", AAAI 2016

Dueling DQN

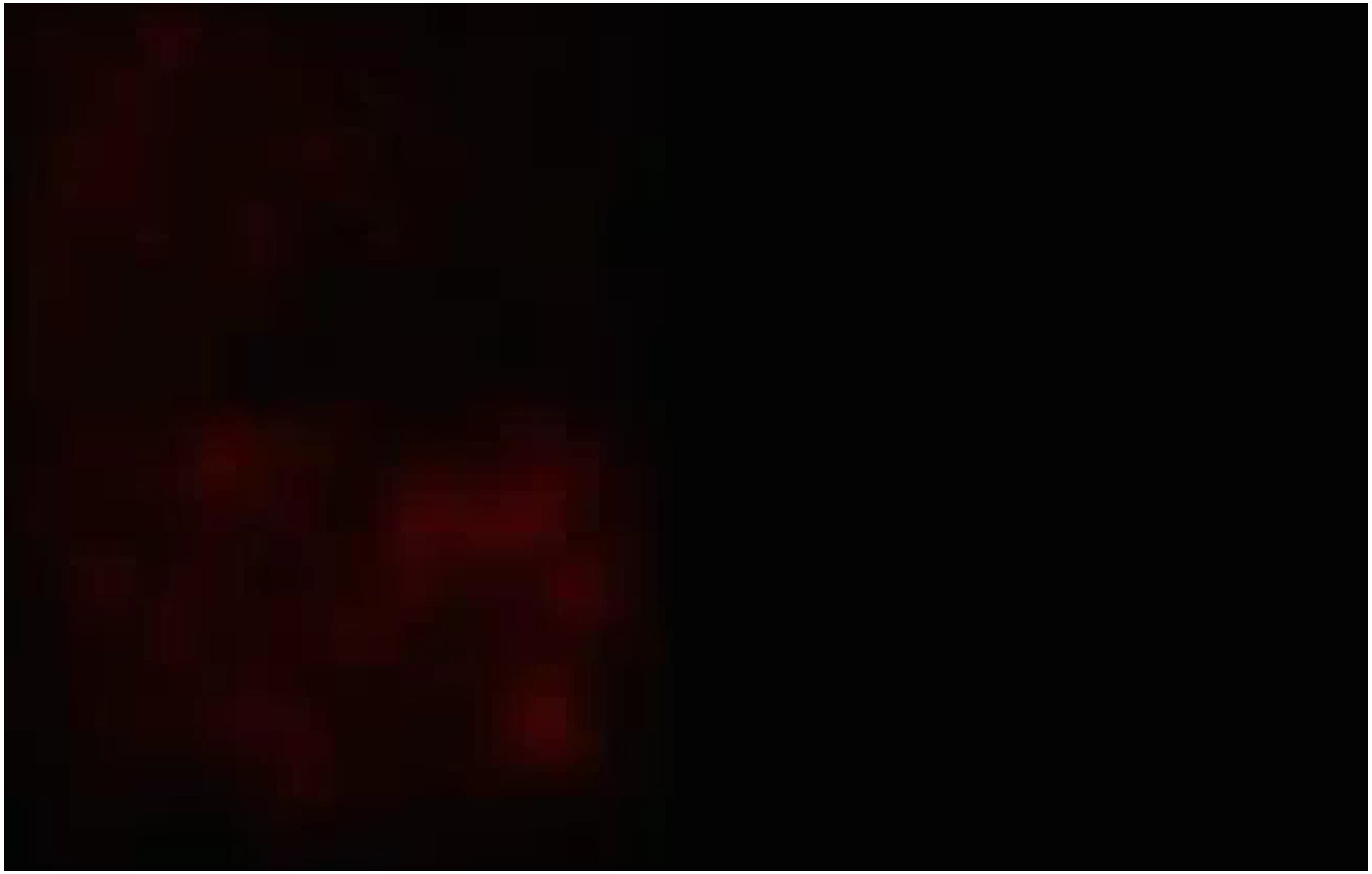
Ziyu Wang, Tom Schaul, Matteo Hessel, Hado van Hasselt, Marc Lanctot, Nando de Freitas, “Dueling Network Architectures for Deep Reinforcement Learning”, arXiv preprint, 2015



Dueling DQN - Visualization

(from the link in the original paper)

Dueling DQN - Visualization



(from the link in the original paper)

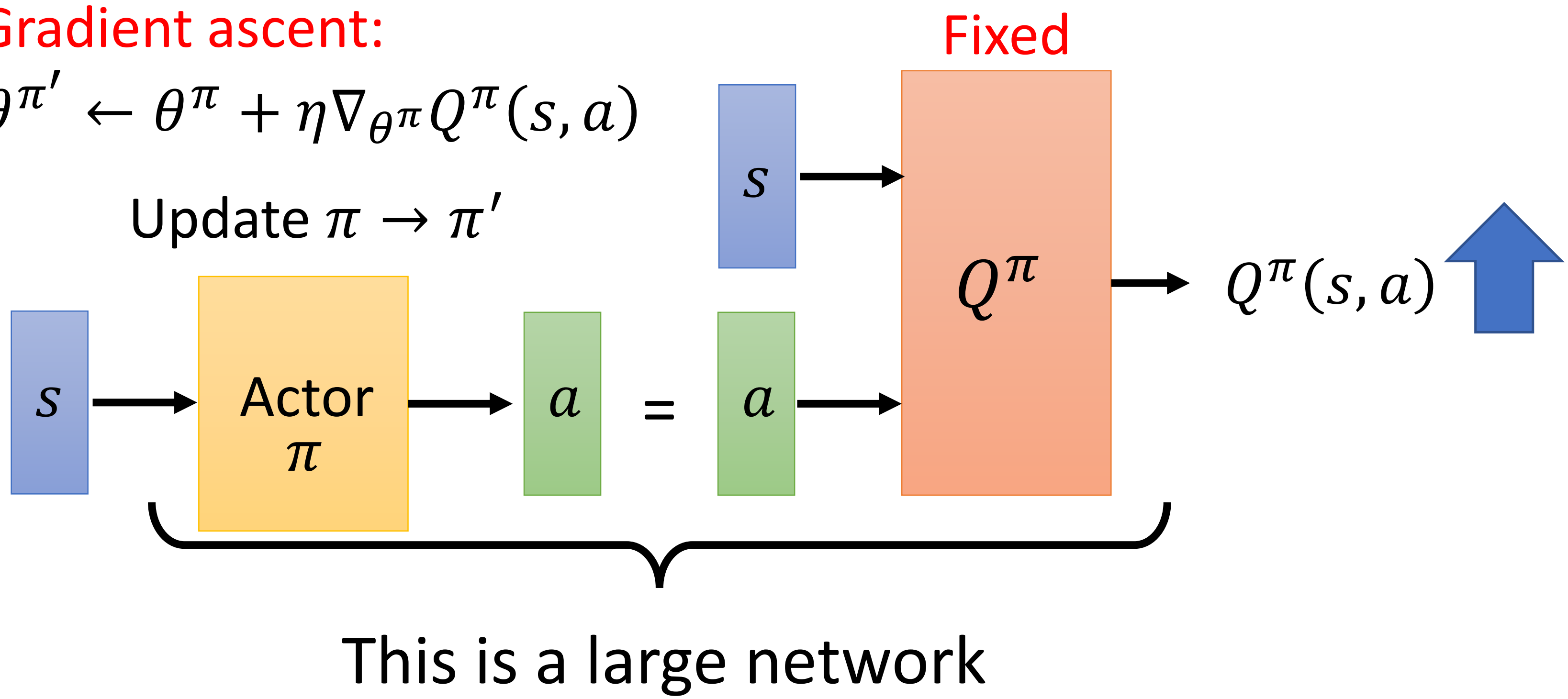
Pathwise Derivative Policy Gradient

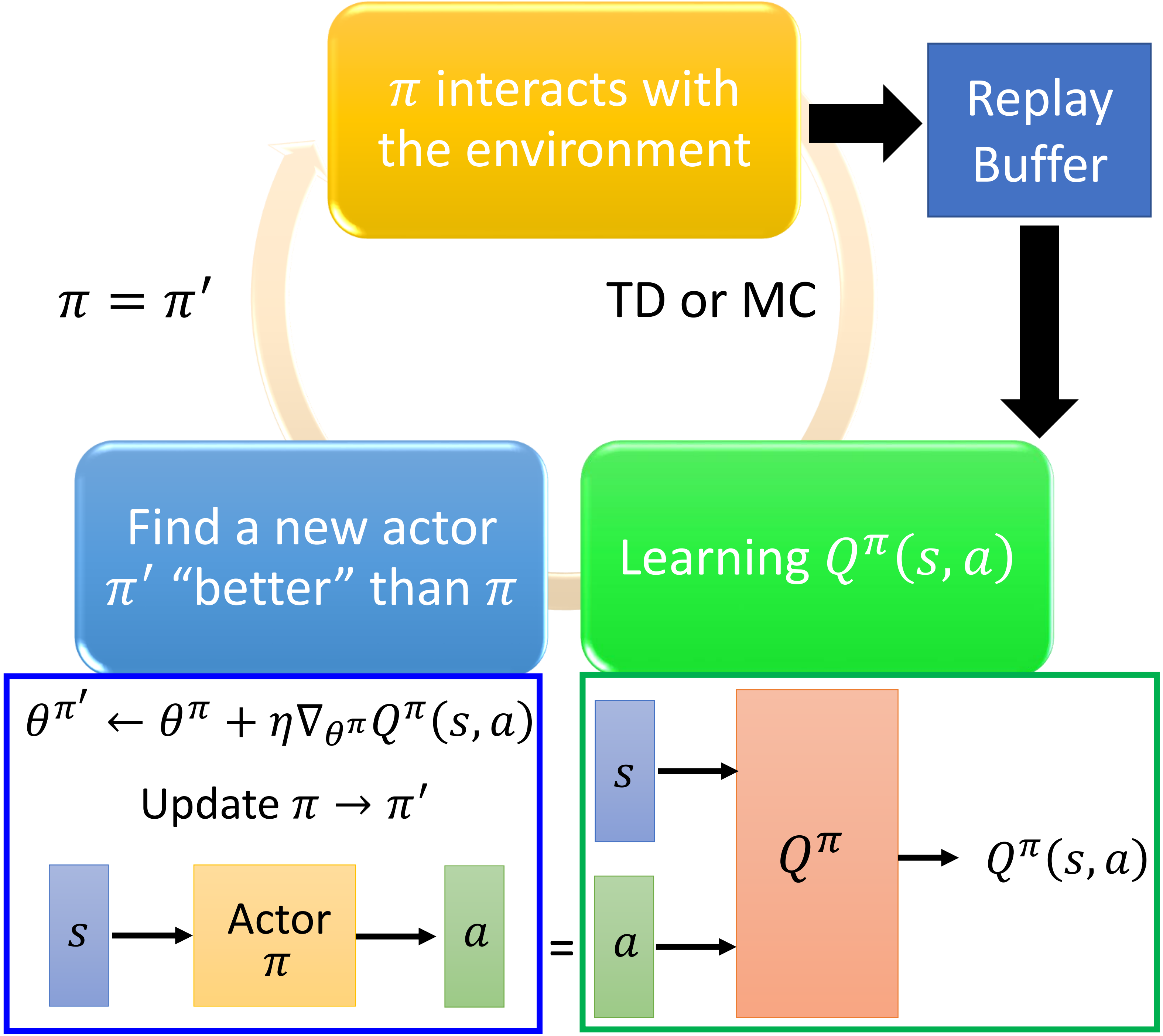
$\pi'(s) = arg\max_{a} Q^{\pi}(s,a)$ ⬅ a is the output of an actor

Gradient ascent:

$\theta^{\pi'} \leftarrow \theta^{\pi} + \eta \nabla_{\theta^{\pi}} Q^{\pi}(s,a)$

Update $\pi \rightarrow \pi'$




π interacts with the environment
Replay Buffer
$\pi = \pi'$
TD or MC
Find a new actor π' "better" than π
Learning $Q^\pi(s,a)$
$\theta^{\pi'} \leftarrow \theta^{\pi} + \eta \nabla_{\theta^{\pi}} Q^{\pi}(s,a)$
Update $\pi \rightarrow \pi'$
s
Actor π
a
=
s
a
Q^π
$Q^\pi(s,a)$

DDPG Algorithm

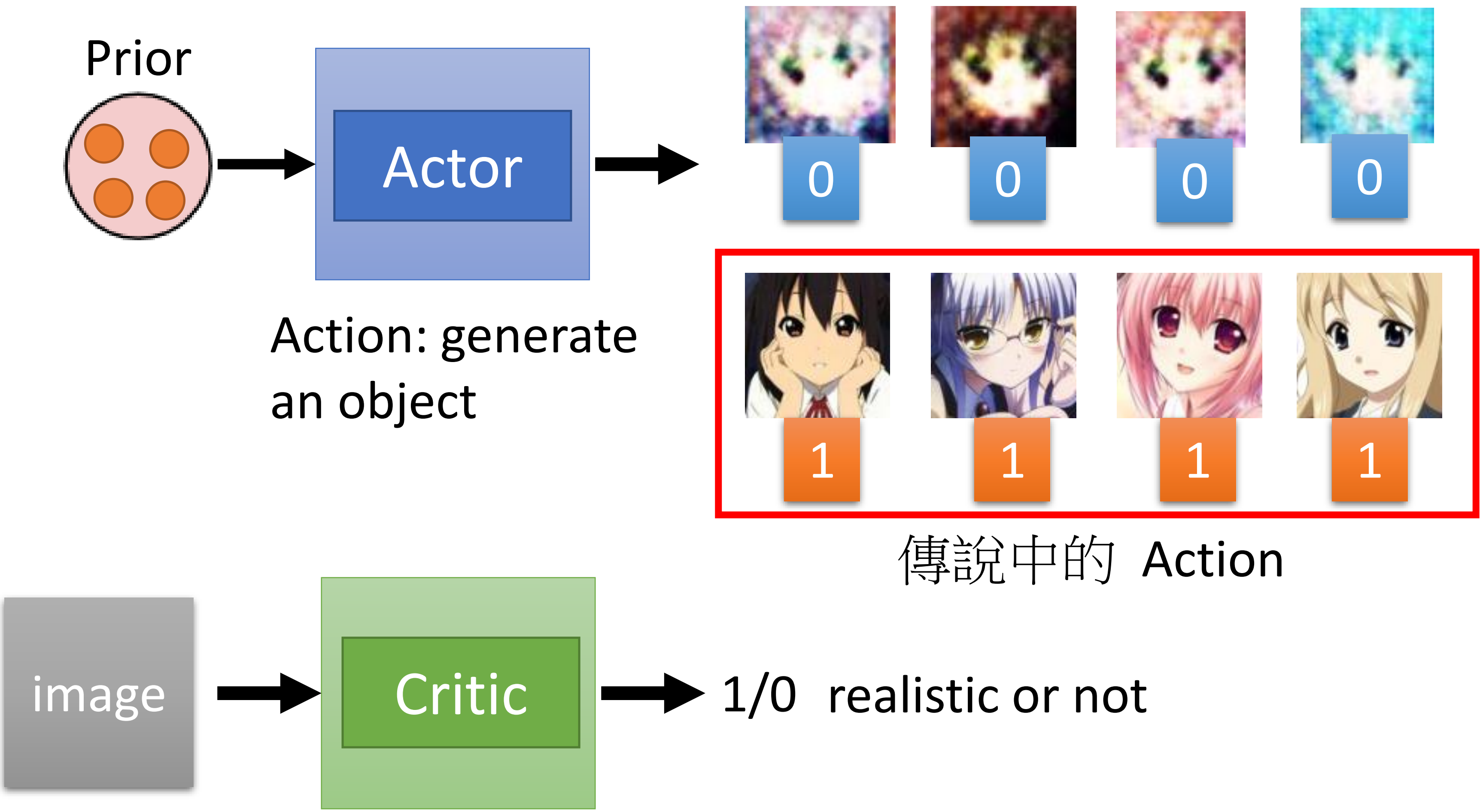
- Initialize critic network θ^Q and actor network θ^π
- Initialize target critic network $\theta^{Q'} = \theta^Q$ and target actor network $\theta^{\pi'} = \theta^\pi$
- Initialize replay buffer R
- In each iteration
  - Use $\pi(s) + noise$ to interact with the environment, collect a set of $\{s_t, a_t, r_t, s_{t+1}\}$, put them in R
  - Sample N examples $\{s_n, a_n, r_n, s_{n+1}\}$ from R
  - Update critic Q to minimize: $L = \sum_n \left(\hat{y}_n - Q(s_n, a_n)\right)^2$
    - $\hat{y}_n = r_n + Q'\left(s_{n+1}, \pi'(s_{n+1})\right)$ Using target networks
  - Update actor π to maximize: $J = \sum_n Q\left(s_n, \pi(s_n)\right)$
  - Update target networks:

$$\theta^{\pi'} \leftarrow m\theta^\pi + (1-m)\theta^{\pi'}$$
$$\theta^{Q'} \leftarrow m\theta^Q + (1-m)\theta^{Q'}$$

The target networks update slower

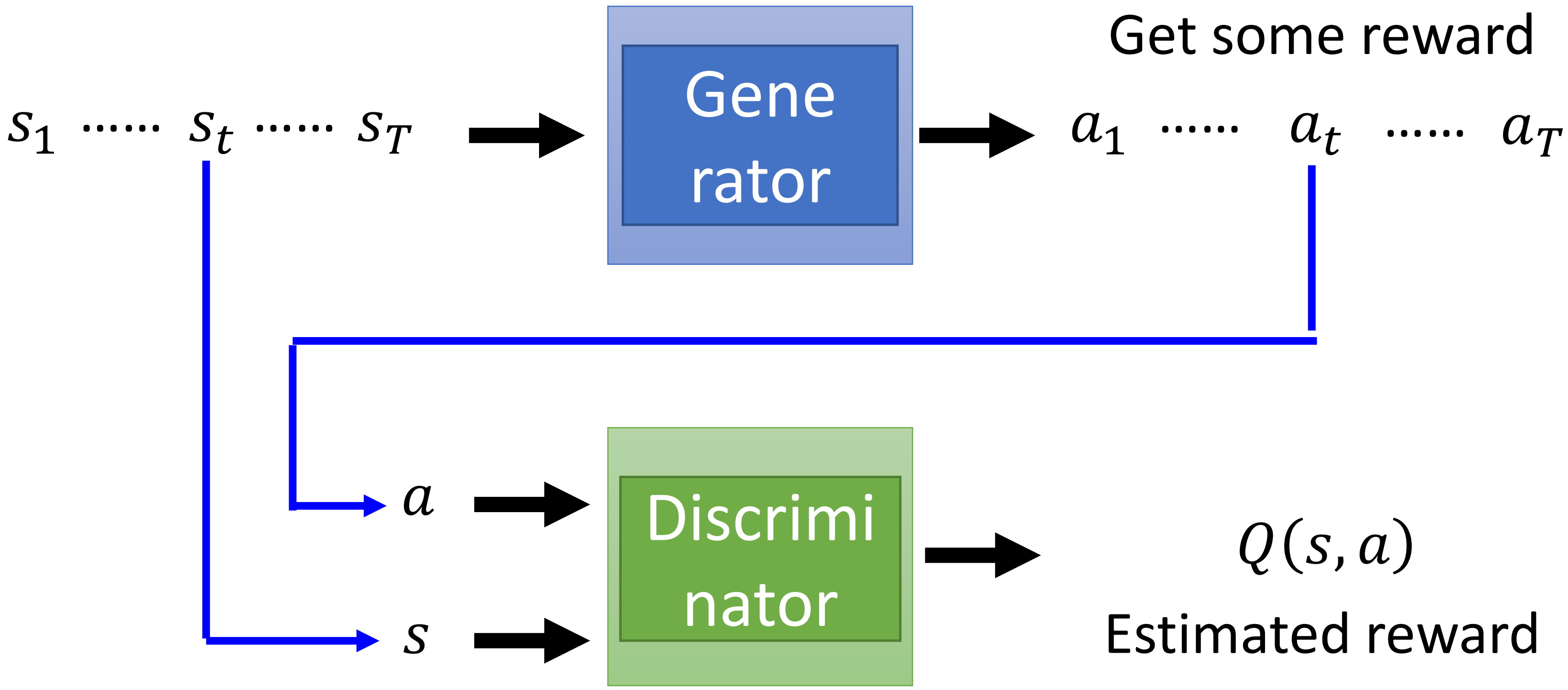
Connection with GAN

GAN as Actor-critic



Actor-critic as GAN

Generator sometimes generates good stuff



Method	GANs	AC
Freezing learning	yes	yes
Label smoothing	yes	no
Historical averaging	yes	no
Minibatch discrimination	yes	no
Batch normalization	yes	yes
Target networks	n/a	yes
Replay buffers	no	yes
Entropy regularization	no	yes
Compatibility	no	yes

David Pfau, Oriol Vinyals, "Connecting Generative Adversarial Networks and Actor-Critic Methods", arXiv preprint, 2016