# Advanced DQN

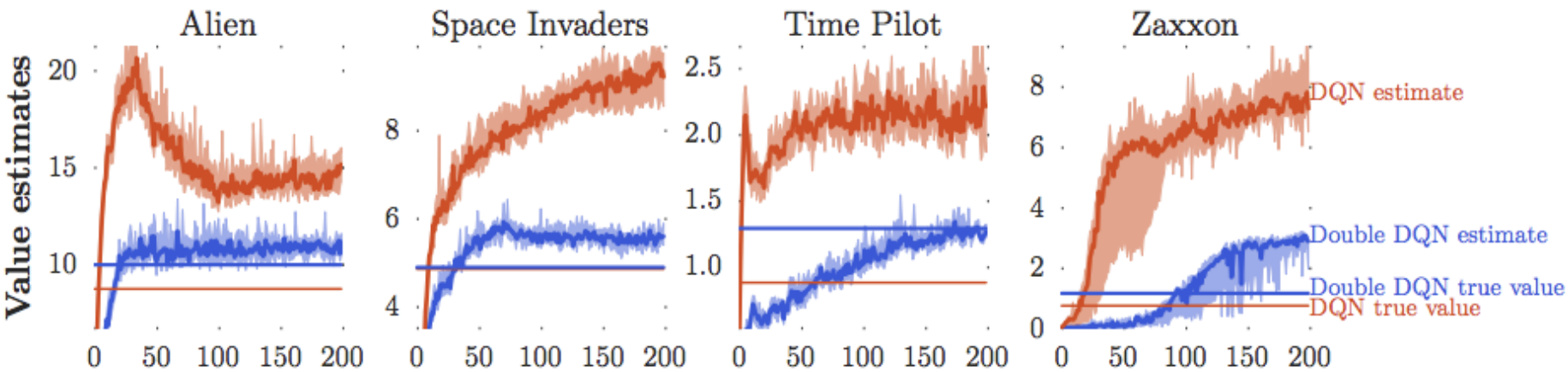
Spring, 2021

# Outline

# Table of Contents

# Double DQN

- Q value is usually over-estimated




Double Q-learning, NIPS 2010

Deep Reinforcement Learning with Double Q-learning, AAAI 2016

# Double DQN

- Q value is usually over estimate

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

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

# Double DQN

- Q value is usually over estimate

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

Tend to select the action that is over-estimated

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

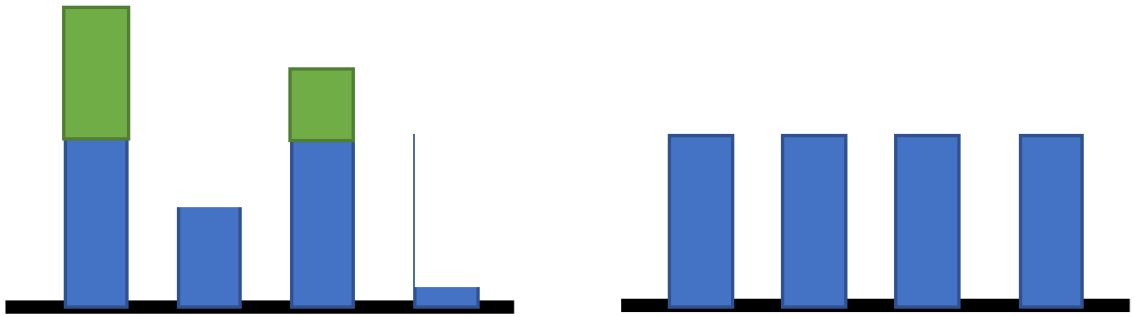
# Double DQN

- Q value is usually over estimate

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

Tend to select the action that is over-estimated

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



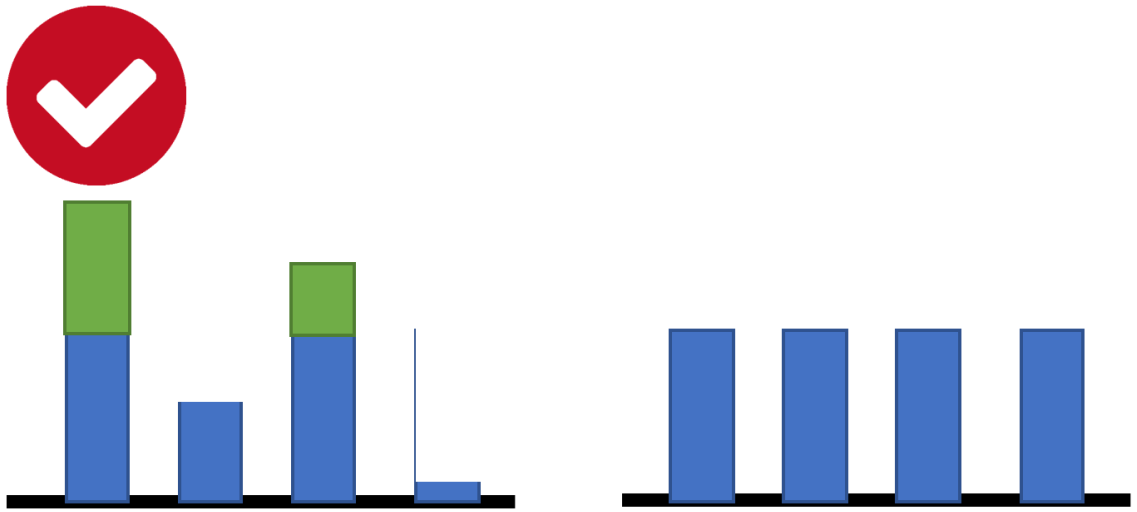
# Double DQN

- Q value is usually over estimate

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

Tend to select the action that is over-estimated

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



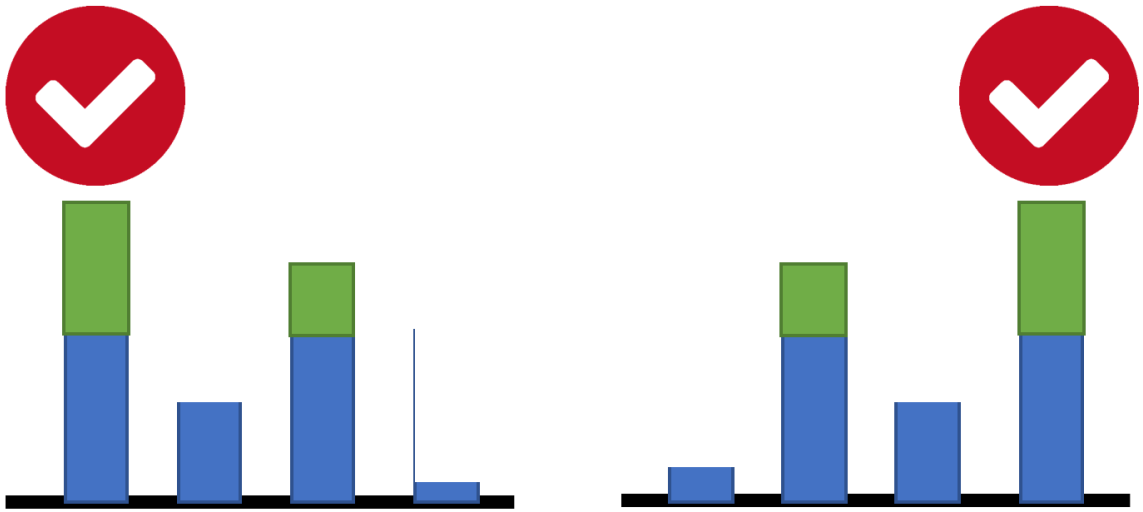
# Double DQN

- Q value is usually over estimate

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

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'

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

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

# 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.

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

# 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 be selected 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

# 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' Target Network

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

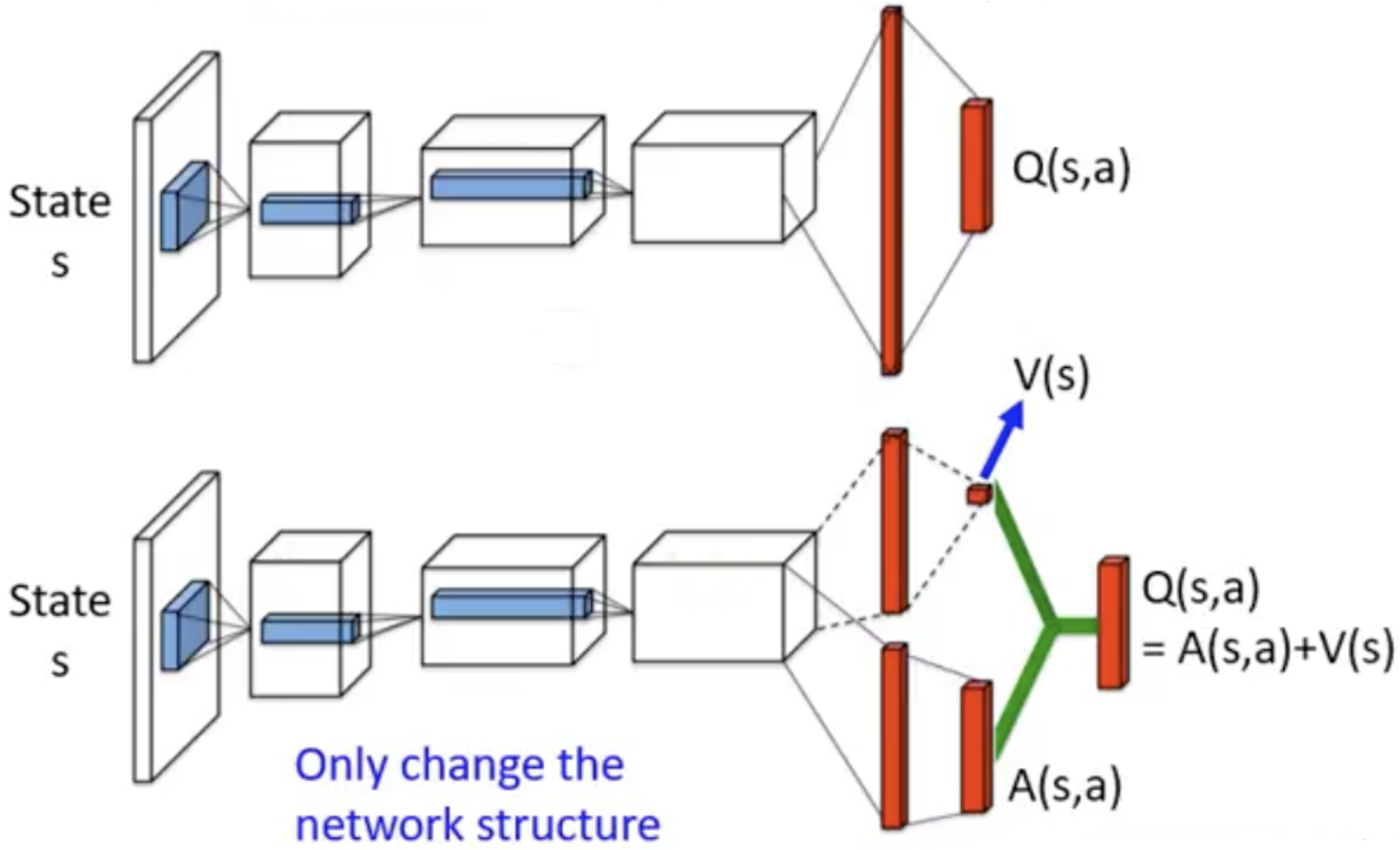
If Q over-estimate a, so it is selected. Q' would give it proper value.

How about Q' overestimate? The action will not be selected 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

# Table of Contents

# Dueling DQN




Dueling Network Architecture for Deep Reinforcement Learning, ICML 2016

# Dueling DQN

如上图所示，在一般的DQN网络模型中，输入层接三个卷积层后，接两个全连接层，输出为每个动作的Q值。而（第二个模型）竞争网络（dueling net）将卷积层提取的抽象特征先分流到两个支路中。一路代表状态值函数 V(s)，表示静态的状态环境本身具有的价值；另一路代表依赖状态的动作优势函数 A(a)（advantage function），表示选择某个Action额外带来的价值。最后这两路再聚合再一起得到每个动作的Q值。这样的话可以更好的适应于不同的环境。

这样可以得到以下的表达式。第一部分仅与状态S有关，与具体要采用的动作A无关，即价值函数部分，第二部分同时与状态S和动作A有关，即优势函数部分。

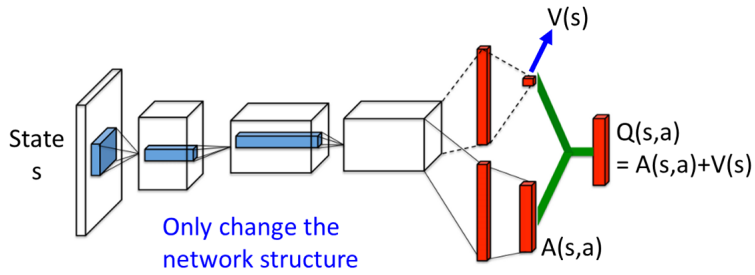
$$Q(s,a;\theta,\alpha,\beta) = V(s;\theta,\beta) + A(s,a;\theta,\alpha).$$

其中，$\theta$是公共部分的网络参数，beta是价值函数独有部分的网络参数，alpha是优势函数独有部分的网络参数。也就是说，Dueling-DQN网络的输出由价值函数网络的输出和优势函数网络的输出线性组合得到。但是上述公式不能辨识最终输出中V和A各自的作用，为了可以体现这种可辨识性(identifiability)，实际使用的组合公式如下。

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

# Dueling DQN

## Dueling DQN




Q(s,a) = V(s) + A(s,a)

| | state | | | |
|---|---|---|---|---|
| action | 3 | 3 | 3 | 1 |
| | 1 | -1 | 6 | 1 |
| | 2 | -2 | 3 | 1 |

=

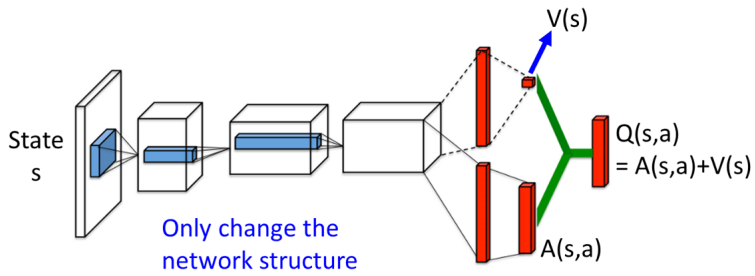
| | | | | |
|---|---|---|---|---|
| | 2 | 0 | 4 | 1 |

+

| | | | |
|---|---|---|---|
| 1 | 3 | -1 | 0 |
| -1 | -1 | 2 | 0 |
| 0 | -2 | -1 | 0 |

# Dueling DQN

## Dueling DQN




Q(s,a) = V(s) + A(s,a)

state

| action | | | | |
|---|---|---|---|---|
| | 3 | ~~3~~ **4** | 3 | 1 |
| | 1 | ~~-1~~ **0** | 6 | 1 |
| | 2 | -2 | 3 | 1 |

=

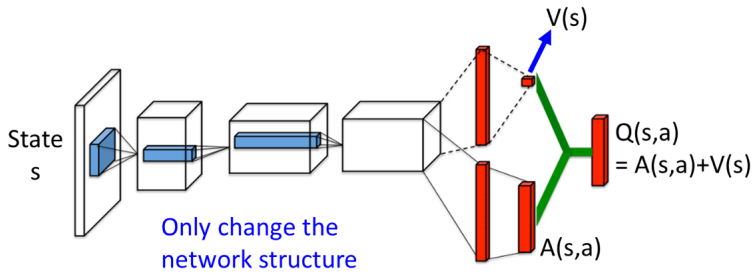
| | | | |
|---|---|---|---|
| 2 | 0 | 4 | 1 |

+

| | | | |
|---|---|---|---|
| 1 | 3 | -1 | 0 |
| -1 | -1 | 2 | 0 |
| 0 | -2 | -1 | 0 |

# Dueling DQN

## Dueling DQN

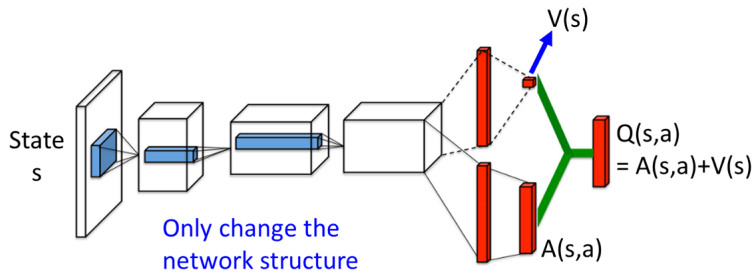



Q(s,a) = V(s) + A(s,a)

| | action | state | | | |
|---|---|---|---|---|---|
| Q(s,a) | | 3 | ~~3~~ 4 | 3 | 1 |
| = | | 1 | ~~-1~~ 0 | 6 | 1 |
| | | 2 | -2 | 3 | 1 |
| = | | | | | |
| V(s) | | 2 | ~~0~~ 1 | 4 | 1 |
| + | | | | | |
| A(s,a) | | 1 | 3 | -1 | 0 |
| | | -1 | -1 | 2 | 0 |
| | | 0 | -2 | -1 | 0 |

# Dueling DQN

## Dueling DQN




Q(s,a) = V(s) + A(s,a)

state

| action | | | | |
|---|---|---|---|---|
| | 3 | ~~3~~ 4 | 3 | 1 |
| | 1 | ~~-1~~ 0 | 6 | 1 |
| | 2 | ~~-2~~ -1 | 3 | 1 |

=

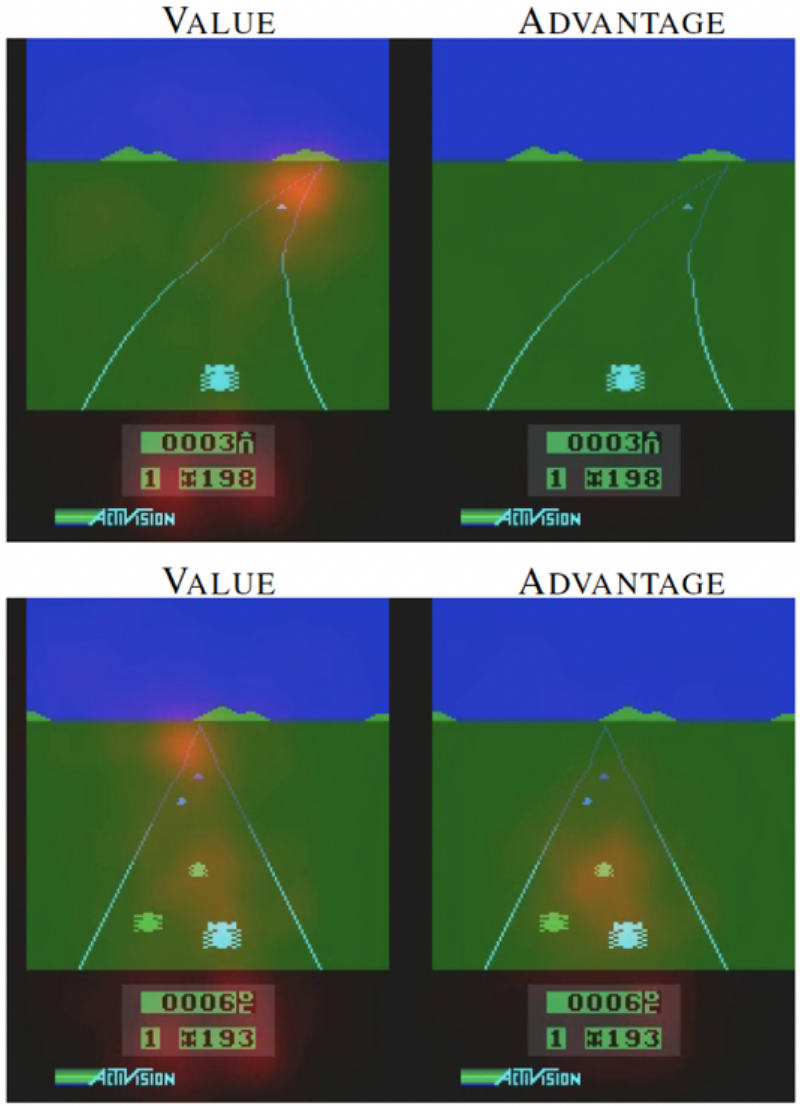
| | | | | |
|---|---|---|---|---|
| | 2 | ~~0~~ 1 | 4 | 1 |

+

| | | | | |
|---|---|---|---|---|
| | 1 | 3 | -1 | 0 |
| | -1 | -1 | 2 | 0 |
| | 0 | -2 | -1 | 0 |

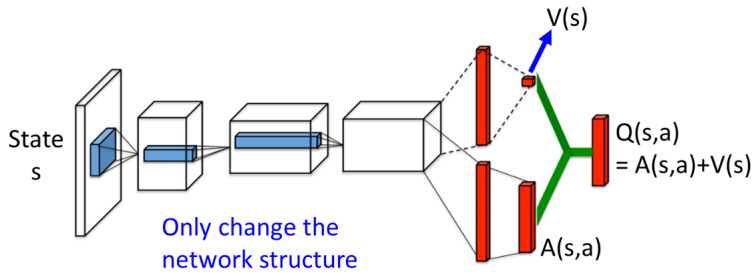
# Dueling DQN

- **For many states, it is unnecessary to estimate the value of each action choice.**
- **The value stream learns to pay attention to the road. The advantage stream learns to pay attention only when there are cars immediately in front, so as to avoid collisions.**



# Dueling DQN

## Dueling DQN

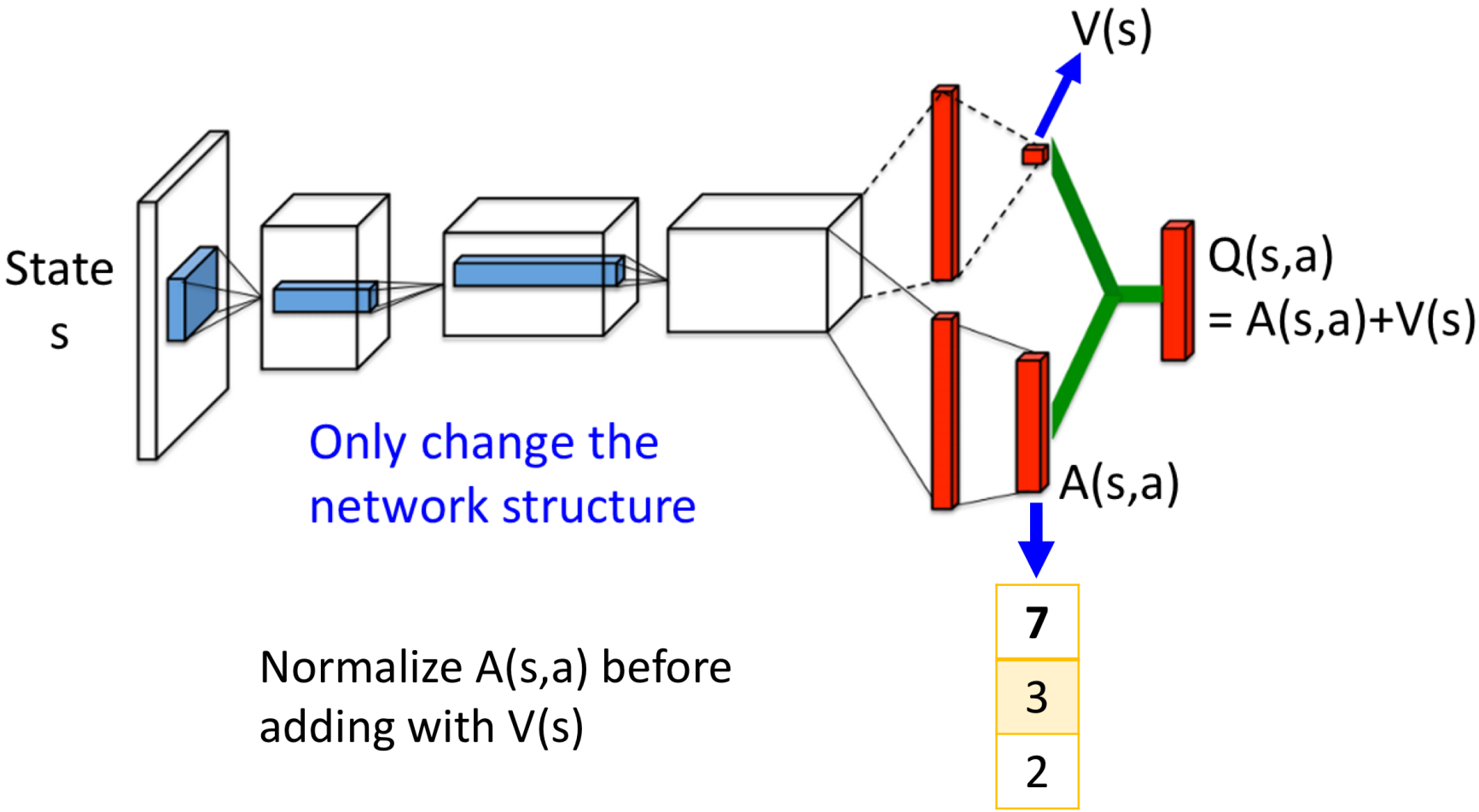



Q(s,a) = V(s) + A(s,a)

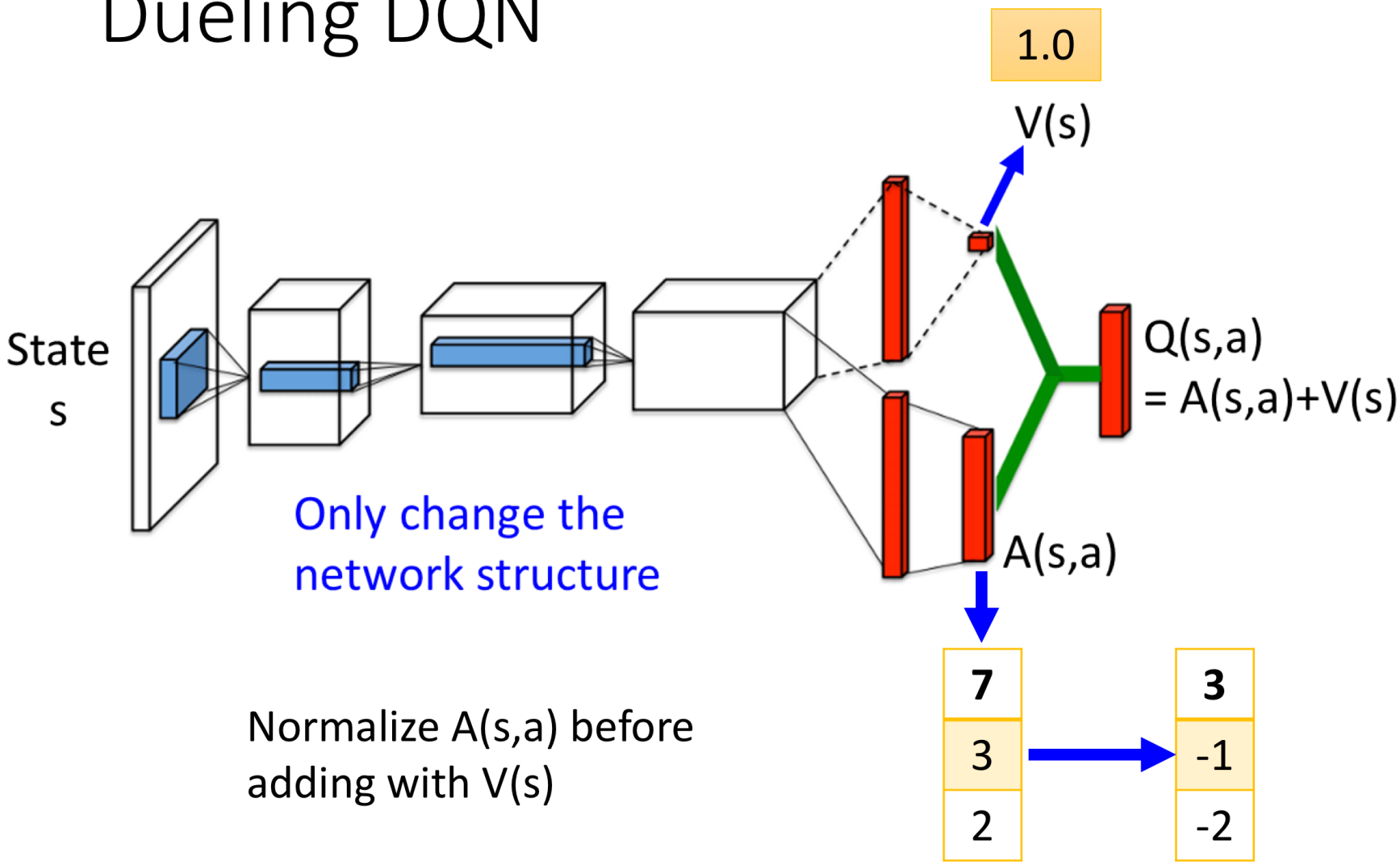
| | action | state | | | |
|---|---|---|---|---|---|
| Q(s,a) | | 3 | ~~3~~ 4 | 3 | 1 |
| = | | 1 | ~~-1~~ 0 | 6 | 1 |
| | | 2 | ~~-2~~ -1 | 3 | 1 |
| | | = | | | |
| V(s) Average of column | | 2 | ~~0~~ 1 | 4 | 1 |
| + | | + | | | |
| A(s,a) sum of column = 0 | | 1 | 3 | -1 | 0 |
| | | -1 | -1 | 2 | 0 |
| | | 0 | -2 | -1 | 0 |

# Dueling DQN

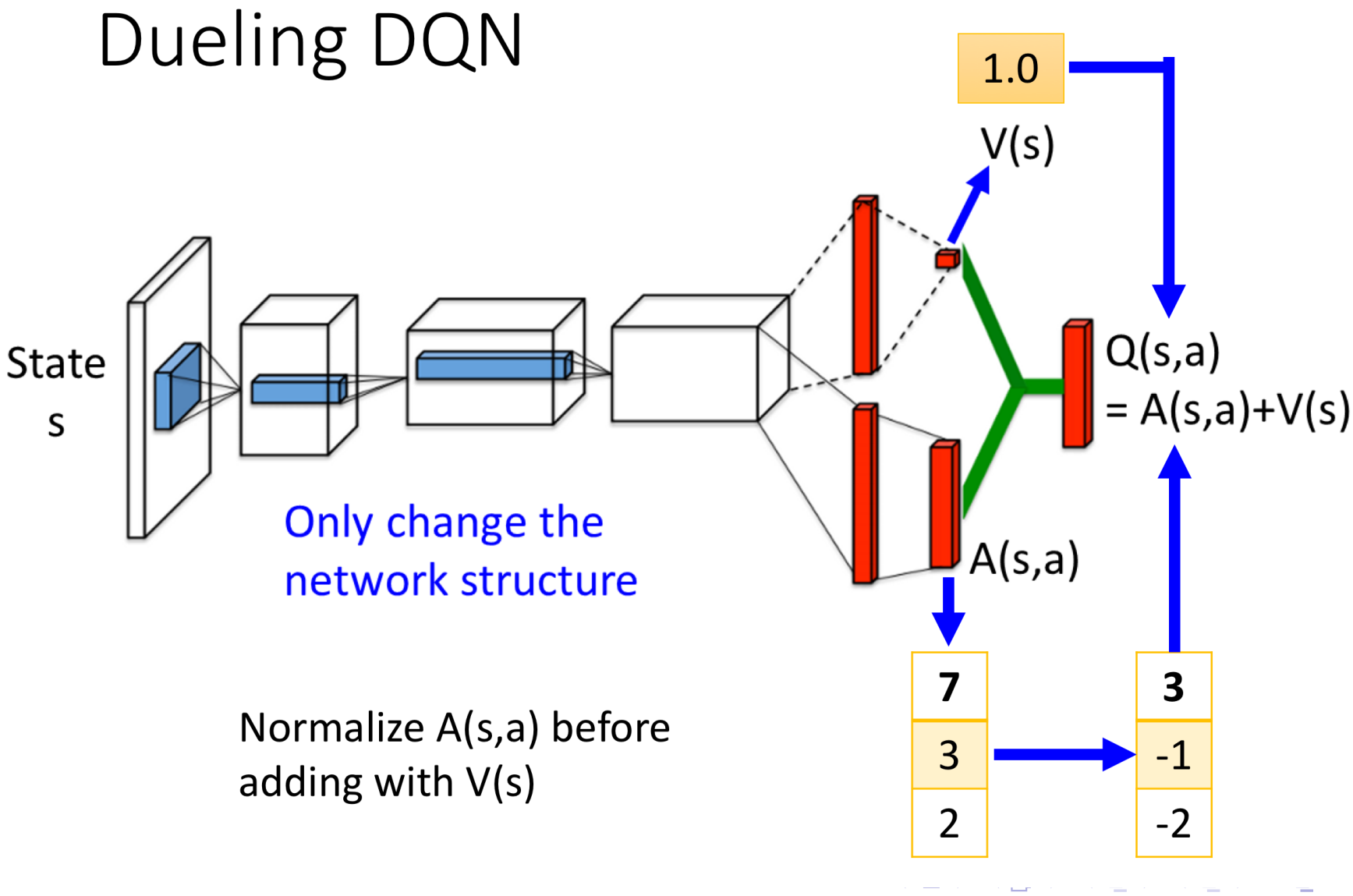
## Dueling DQN



# Dueling DQN


Dueling DQN
1.0
V(s)
State
s
Q(s,a)
= A(s,a)+V(s)
Only change the
network structure
A(s,a)
7
3
2
3
-1
-2
Normalize A(s,a) before
adding with V(s)

# Dueling DQN

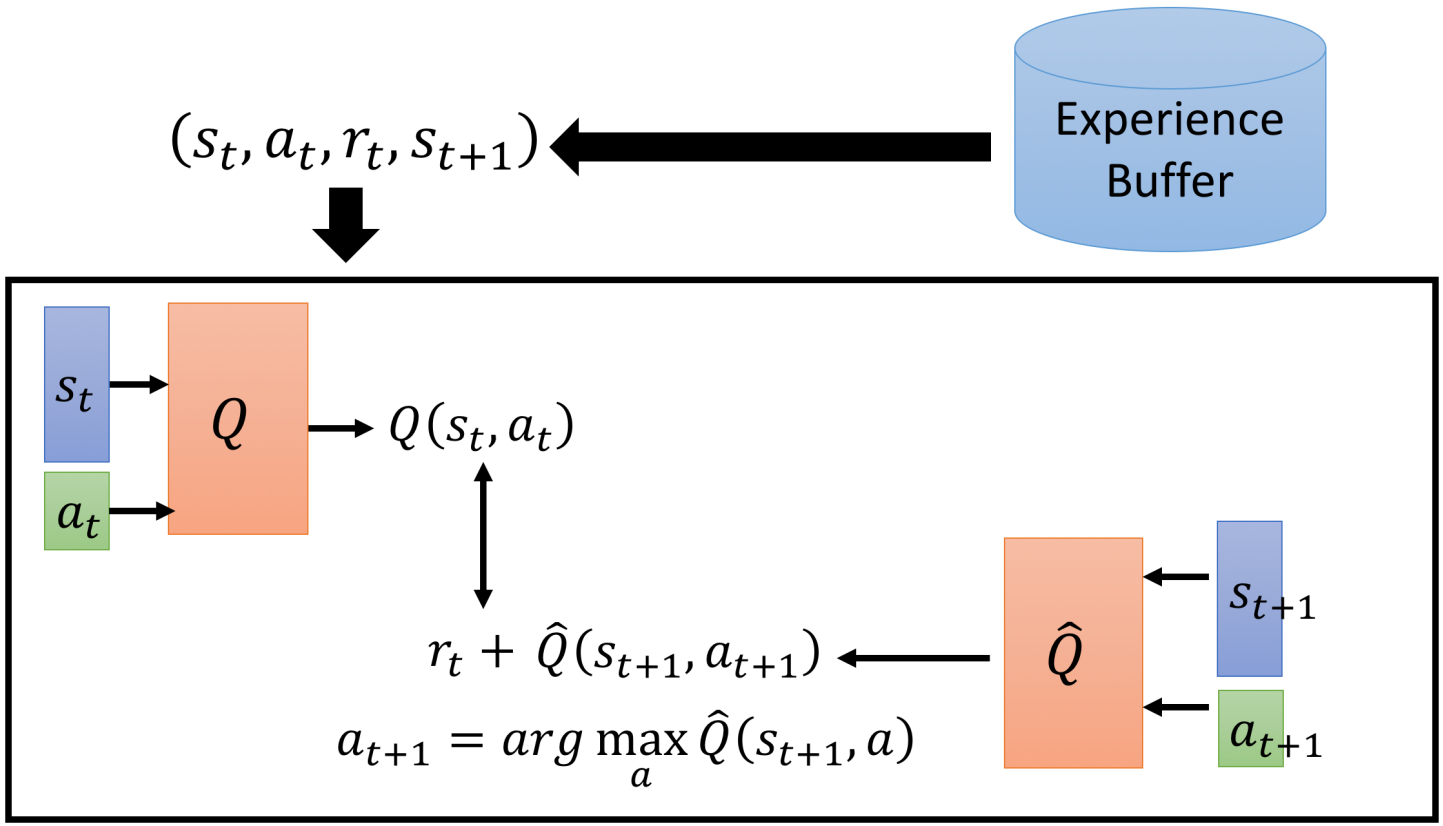

Dueling DQN
1.0
V(s)
State
s
Q(s,a)
= A(s,a)+V(s)
Only change the
network structure
A(s,a)
7
3
2
3
-1
-2
Normalize A(s,a) before
adding with V(s)

# Table of Contents

# Prioritized Experience Replay

https://arxiv.org/abs/1511.05952?context=cs

## Prioritized Reply

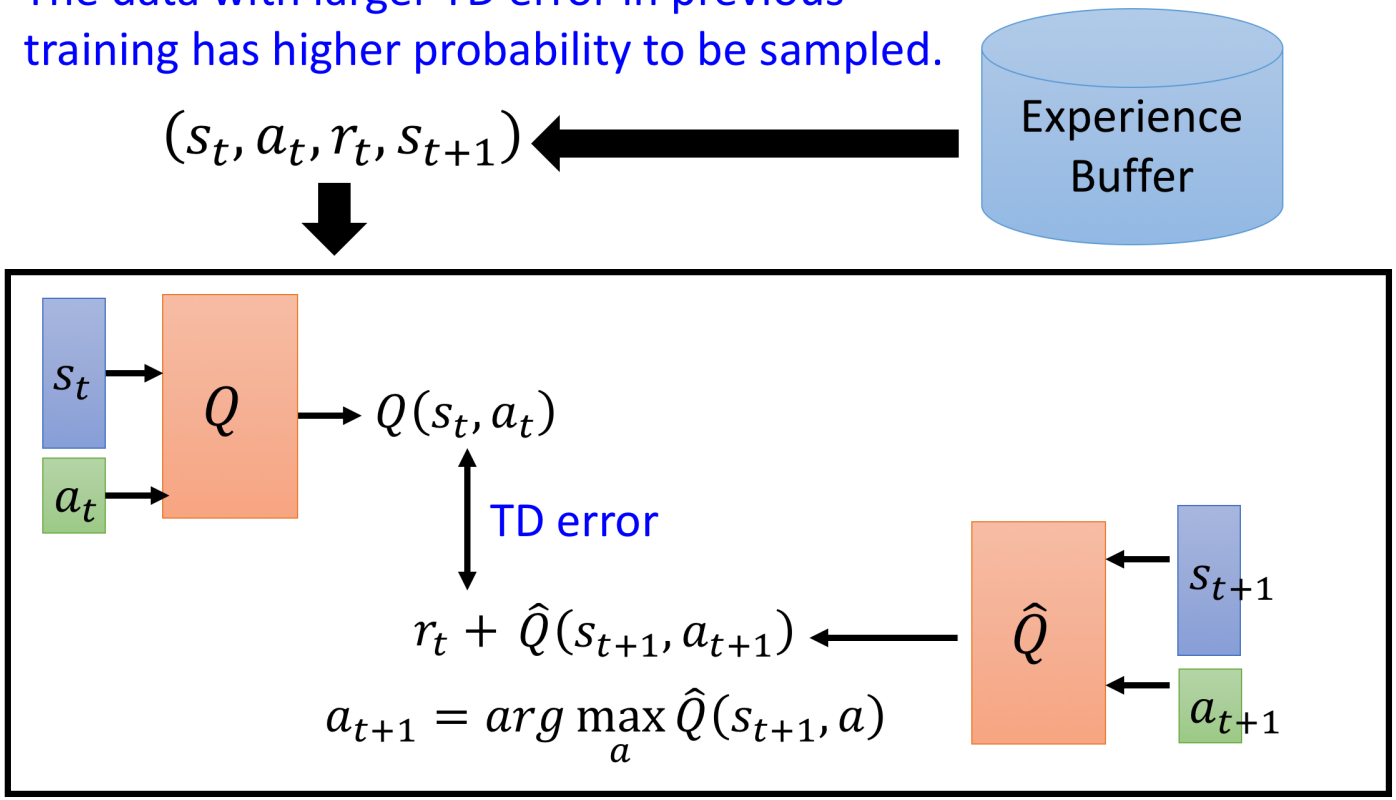


# Prioritized Experience Replay

https://arxiv.org/abs/1511.05952?context=cs

## Prioritized Reply

The data with larger TD error in previous training has higher probability to be sampled.



# Table of Contents

# NoisyNet DQN

## Noisy Net

https://arxiv.org/abs/1706.01905
https://arxiv.org/abs/1706.10295

- Noise on Action (Epsilon Greedy)

$$a = \begin{cases} arg\max_a Q(s,a), & with\ probability\ 1-\varepsilon \\ random, & otherwise \end{cases}$$

- Noise on Parameters

$$a = arg\max_a \tilde{Q}(s,a)$$

# NoisyNet DQN

## Noisy Net

https://arxiv.org/abs/1706.01905
https://arxiv.org/abs/1706.10295

- Noise on Action (Epsilon Greedy)

$$a = \begin{cases} arg\max_a Q(s,a), & with\ probability\ 1-\varepsilon \\ random, & otherwise \end{cases}$$

- Noise on Parameters

$$a = arg\max_a \tilde{Q}(s,a)$$

Inject noise into the parameters of Q-function **at the beginning of each episode**

$$Q(s,a) \xrightarrow[\text{Add noise}]{} \tilde{Q}(s,a)$$

# NoisyNet DQN

## Noisy Net

https://arxiv.org/abs/1706.01905
https://arxiv.org/abs/1706.10295

- Noise on Action (Epsilon Greedy)

$$a = \begin{cases} arg\max_a Q(s,a), & with\ probability\ 1-\varepsilon \\ random, & otherwise \end{cases}$$

- Noise on Parameters

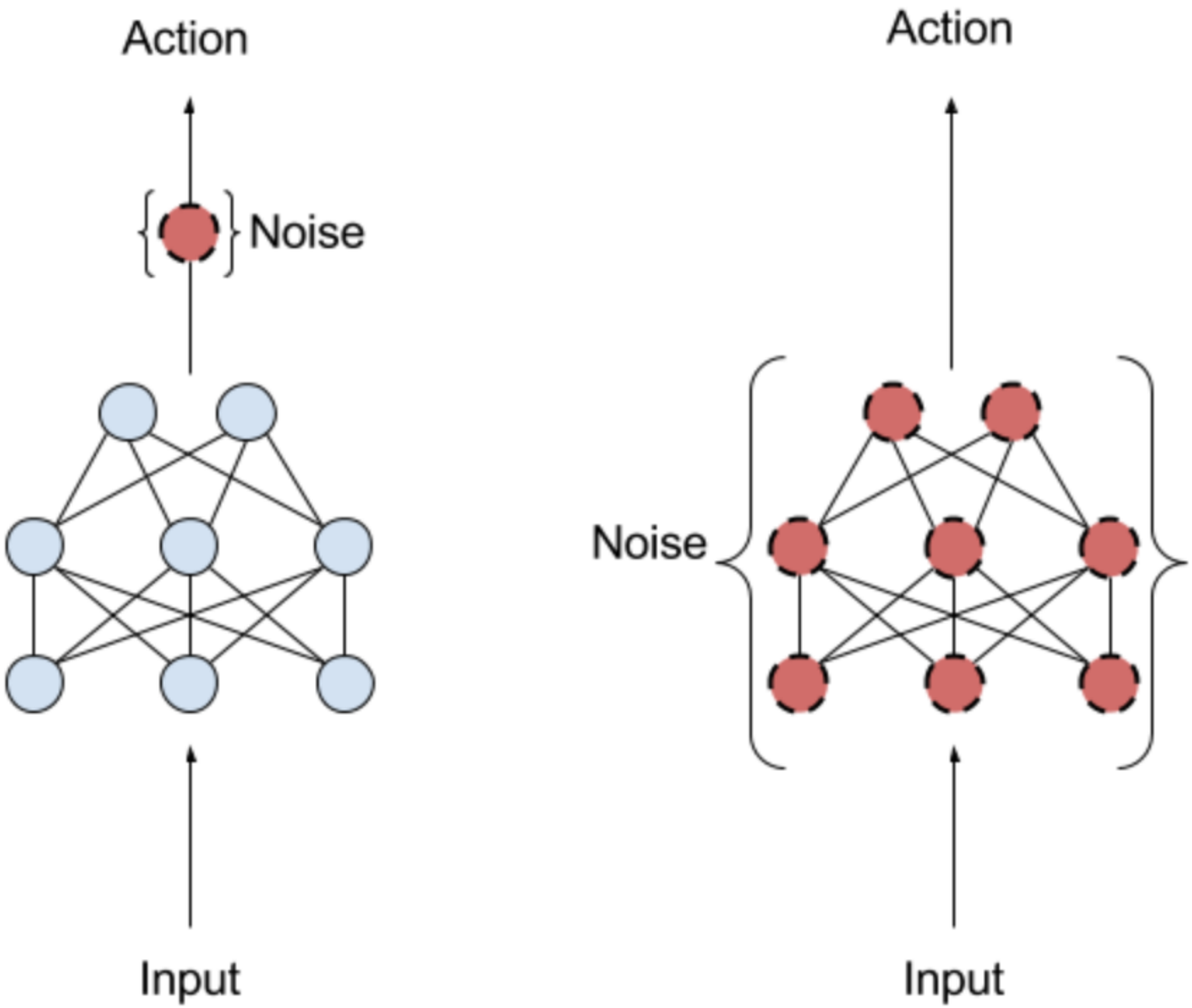
$$a = arg\max_a \tilde{Q}(s,a)$$

Inject noise into the parameters of Q-function **at the beginning of each episode**

$$Q(s,a) \xrightarrow{\text{Add noise}} \tilde{Q}(s,a)$$

The noise would **NOT** change in an episode.

# NoisyNet DQN



# NoisyNet DQN

## Noisy Net

- Noise on Action
  - Given the same state, the agent may takes different actions.
  - No real policy works in this way

# NoisyNet DQN

## Noisy Net

- Noise on Action
  - Given the same state, the agent may takes different actions.
  - No real policy works in this way
- Noise on Parameters
  - Given the same (similar) state, the agent takes the same action.

# NoisyNet DQN

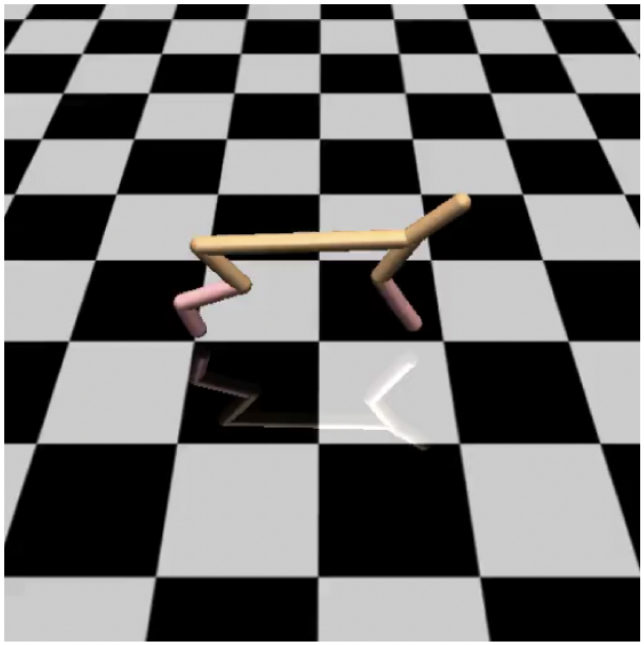
## Noisy Net

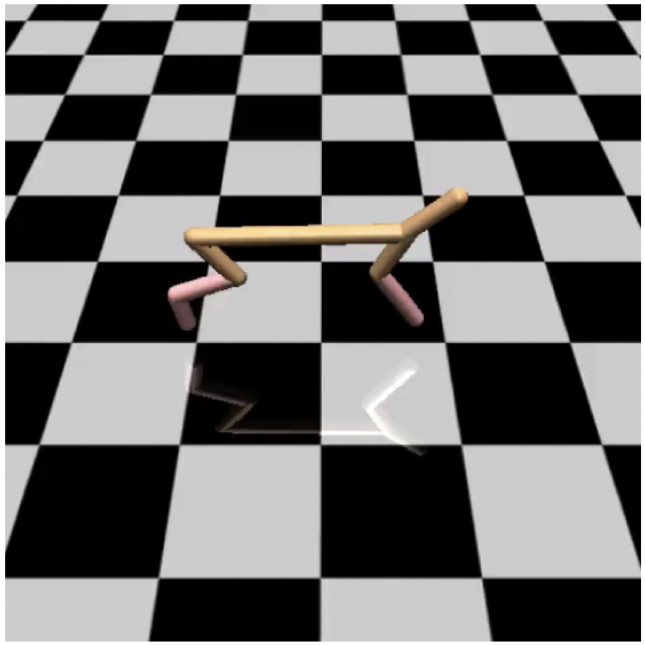
- Noise on Action
  - Given the same state, the agent may takes different actions.
  - No real policy works in this way
- Noise on Parameters
  - Given the same (similar) state, the agent takes the same action.
    - $\longrightarrow$ State-dependent Exploration
  - Explore in a *consistent* way

# NoisyNet DQN

## Demo

https://blog.openai.com/better-exploration-with-parameter-noise/





# Table of Contents

# Distributional DQN

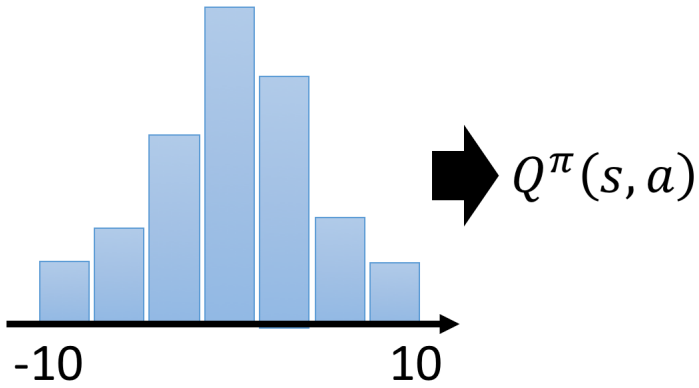
## Distributional Q-function

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

# Distributional DQN

## Distributional Q-function

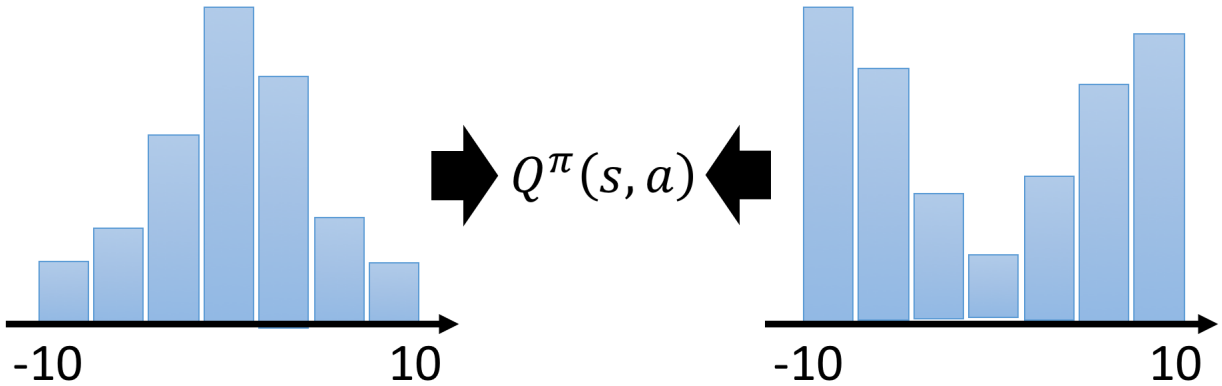
- State-action value function $Q^{\pi}(s, a)$
  - When using actor $\pi$, the *cumulated* reward expects to be obtained after seeing observation s and taking a



# Distributional DQN

## Distributional Q-function

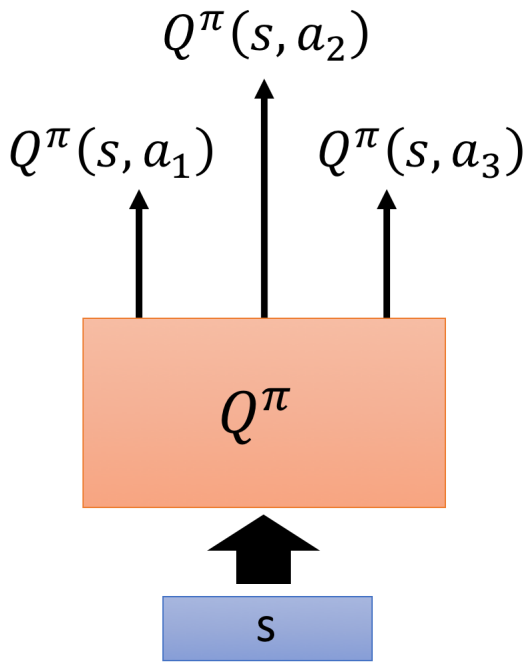
- State-action value function $Q^{\pi}(s, a)$
  - When using actor $\pi$, the *cumulated* reward expects to be obtained after seeing observation s and taking a




Different distributions can have the same values.

# Distributional DQN

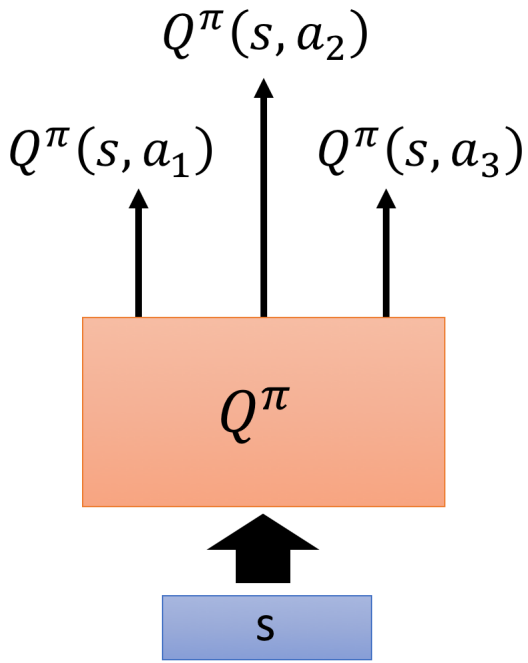
## Distributional Q-function



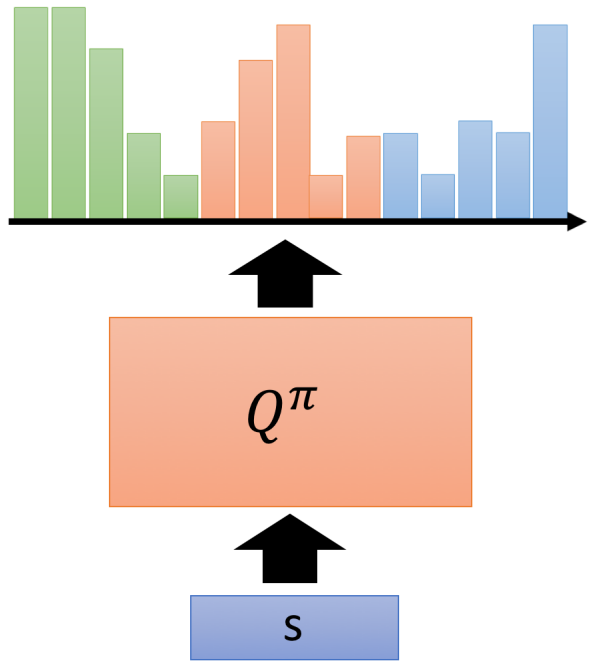

A network with 3 outputs

# Distributional DQN

## Distributional Q-function



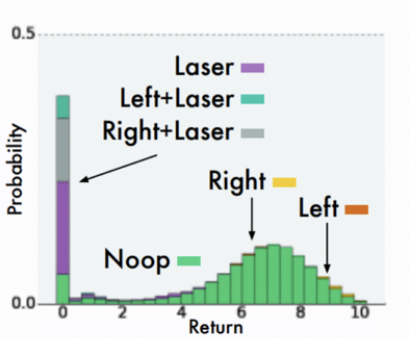

A network with 3 outputs

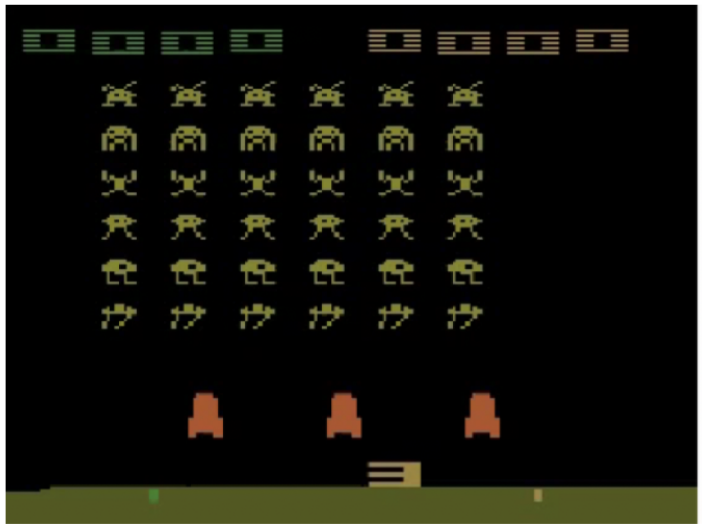


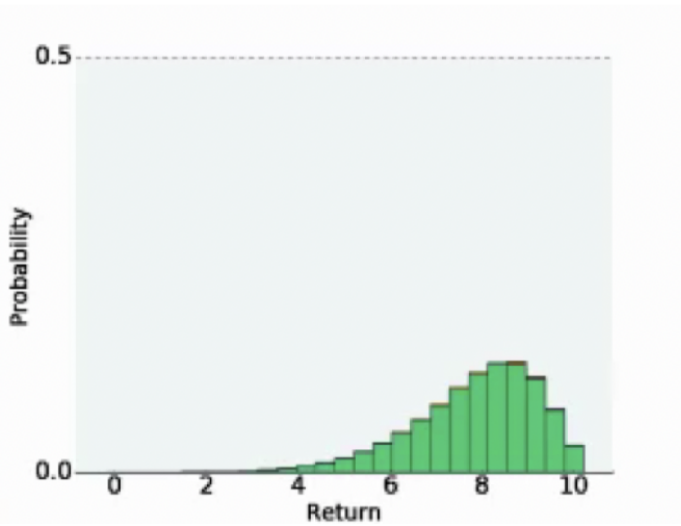

A network with 15 outputs
(each action has 5 bins)

# Distributional DQN

Demo

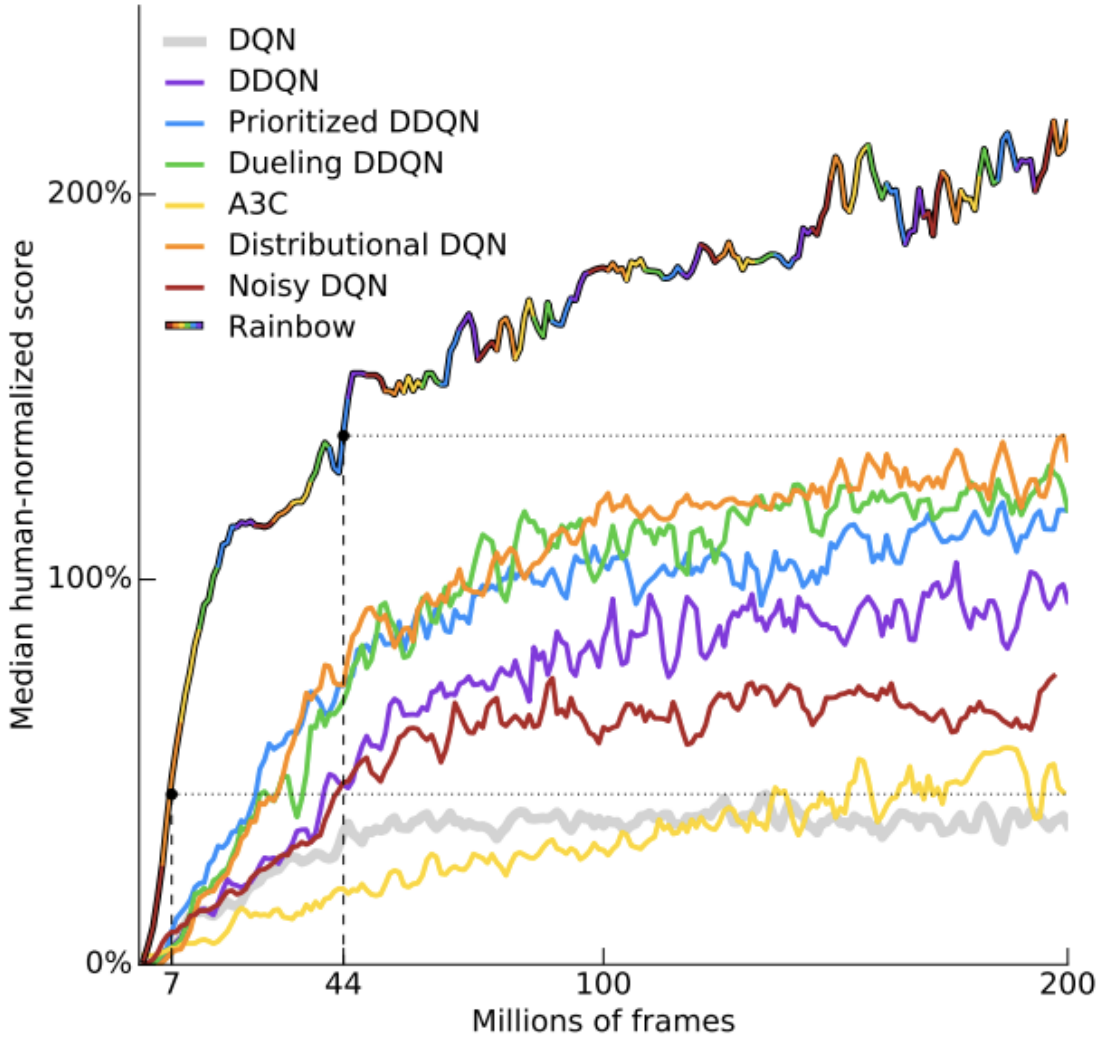




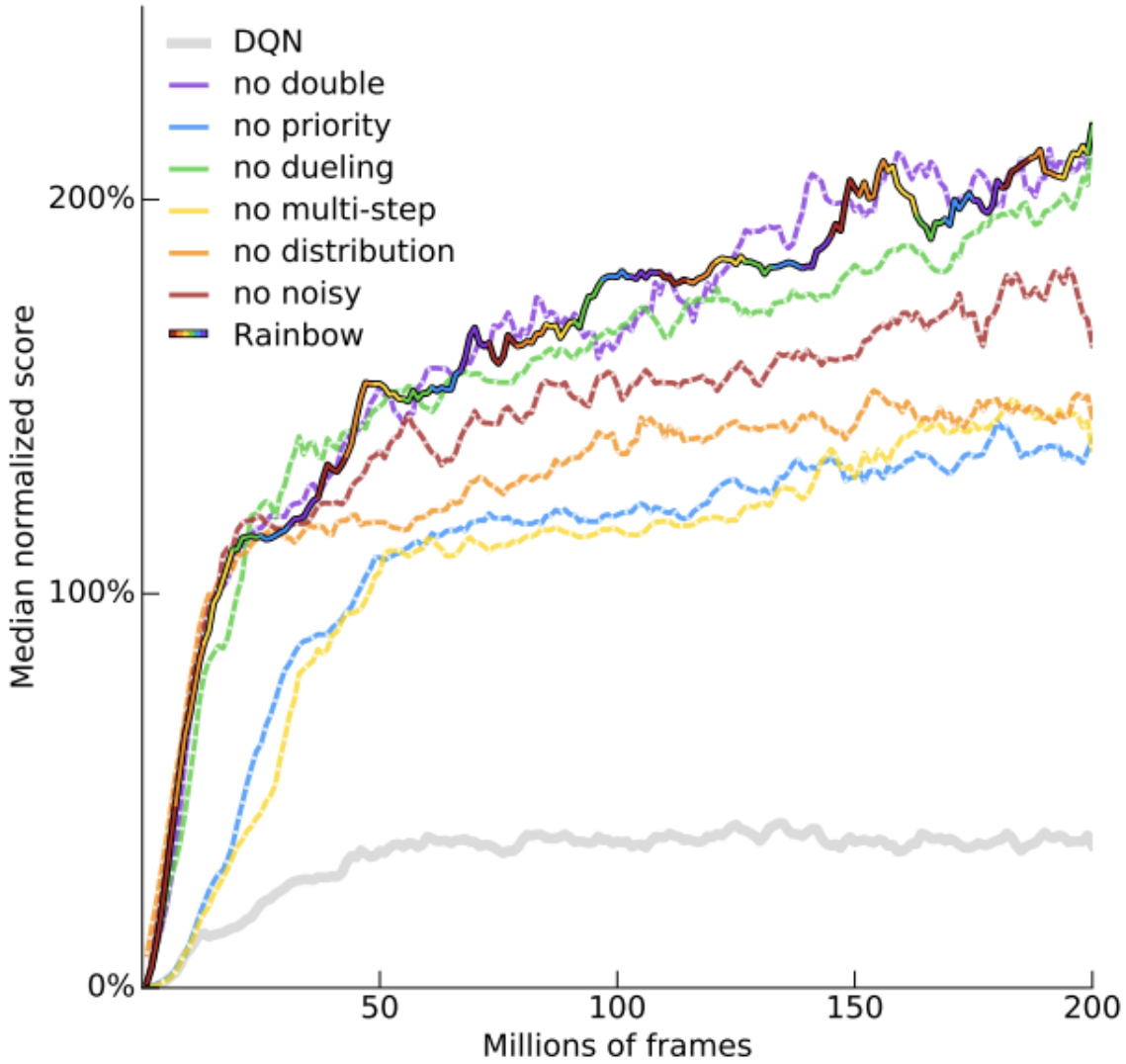

https://youtu.be/yFBwyPuO2Vg

# Rainbow




Rainbow: Combining Improvements in Deep Reinforcement earning, AAAI 2018

# Rainbow


DQN
no double
no priority
no dueling
no multi-step
no distribution
no noisy
Rainbow
200%
100%
0%
Median normalized score
50
100
150
200
Millions of frames