

Double DQN

$$\Delta \mathbf{w} = \alpha \left(r + \gamma \underbrace{\hat{Q}(\arg \max_{a'} \hat{Q}(s', a'; \mathbf{w}^-); \mathbf{w}^-)}_{\text{Action selection: } \mathbf{w}} - \hat{Q}(s, a; \mathbf{w}) \right)$$

Action evaluation: \mathbf{w}^-

Prioritized
order replay

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

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

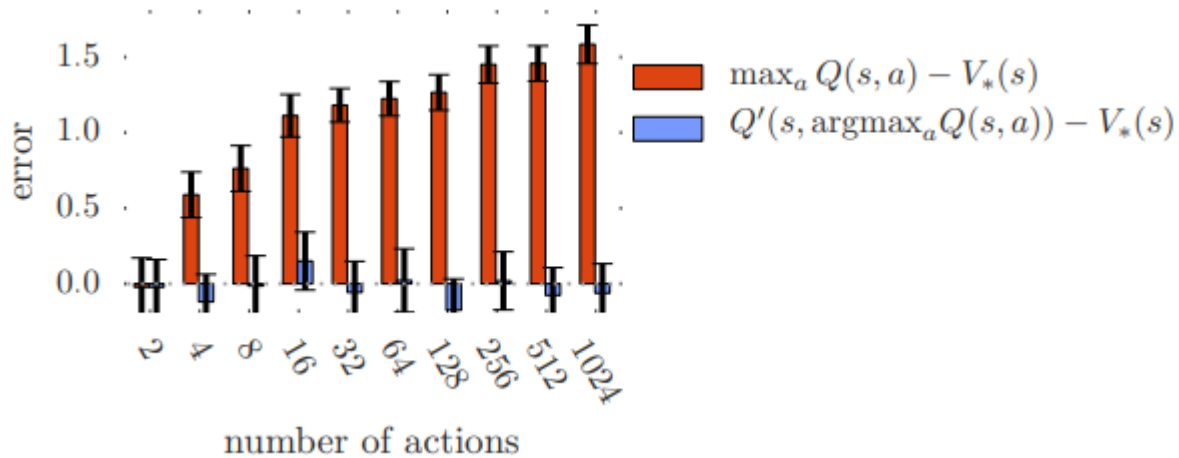
Dueling DQN

$$A^\pi(s, a) = Q^\pi(s, a) - V^\pi(s)$$

Double DQN

$$\Delta \mathbf{w} = \alpha \left(r + \gamma \underbrace{\hat{Q}(\arg \max_{a'} \hat{Q}(s', a'; \mathbf{w}^-); \mathbf{w}^-)}_{\text{Action selection: } \mathbf{w}} - \hat{Q}(s, a; \mathbf{w}) \right)$$

Action evaluation: \mathbf{w}^-



If it was just DQN, as the number of actions increase, the overestimation increases BUT,

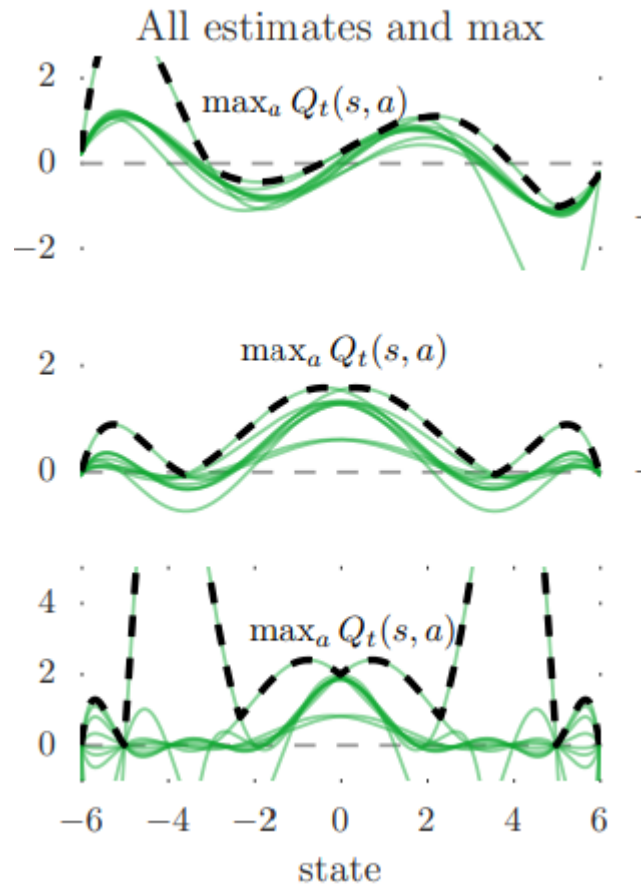
By decoupling Q for action choosing and Q for evaluating, the overestimation can be prevented

Double DQN

$$\Delta \mathbf{w} = \alpha \left(r + \gamma \underbrace{\hat{Q}(\arg \max_{a'} \hat{Q}(s', a'; \mathbf{w}^-); \mathbf{w}^-)}_{\text{Action selection: } \mathbf{w}} - \hat{Q}(s, a; \mathbf{w}) \right)$$

Action evaluation: \mathbf{w}^-

Action selection: \mathbf{w}



There could have been problem of function approximation not flexible enough...

But the bottom approximation has enough flexibility but still has overestimation,

While DDQN doesn't

== just take in mind sometimes the function approx. may need to be more flexible = deeper and larger networks

Double DQN

$$\Delta \mathbf{w} = \alpha \left(r + \gamma \underbrace{\hat{Q}(\arg \max_{a'} \hat{Q}(s', a'; \mathbf{w}^-))}_{\text{Action selection: } \mathbf{w}} - \hat{Q}(s, a; \mathbf{w}) \right)$$

Action evaluation: \mathbf{w}^-

Q7. Difference between fixed Q-target.

A7. Just the period of update is diff..
that is for deepmind version of
efficient implementation of DDQN

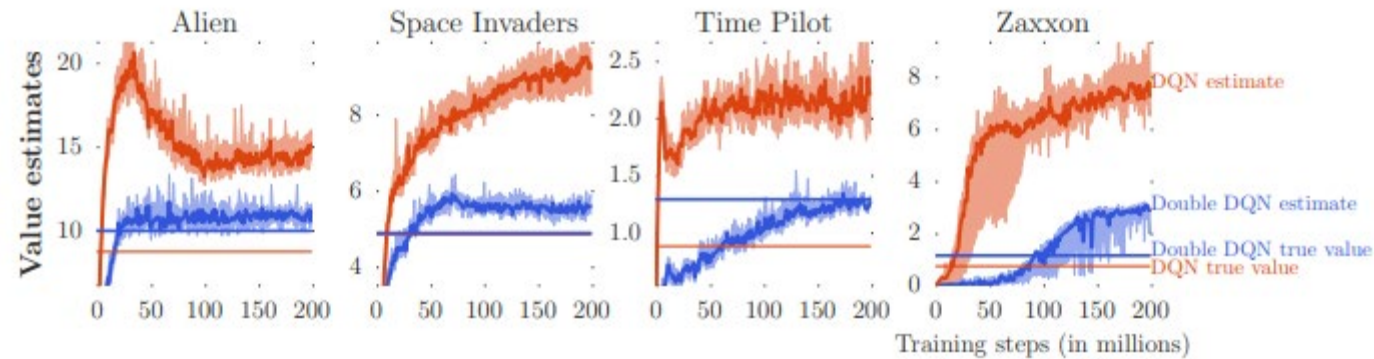
"the update to the target network
stays unchanged from DQN.
Remaining as a periodic copy of the
online network"

"minimal possible change to DQN
towards Double Q-learning."

Double DQN

$$\Delta \mathbf{w} = \alpha \left(r + \gamma \underbrace{\hat{Q}(\arg \max_{a'} \hat{Q}(s', a'; \mathbf{w}^-); \mathbf{w}^-)}_{\text{Action selection: } \mathbf{w}} - \hat{Q}(s, a; \mathbf{w}) \right)$$

Action evaluation: \mathbf{w}^-



** straight orange and blue line

Shows that DDQN does not only prevent overestimation does more accurate values

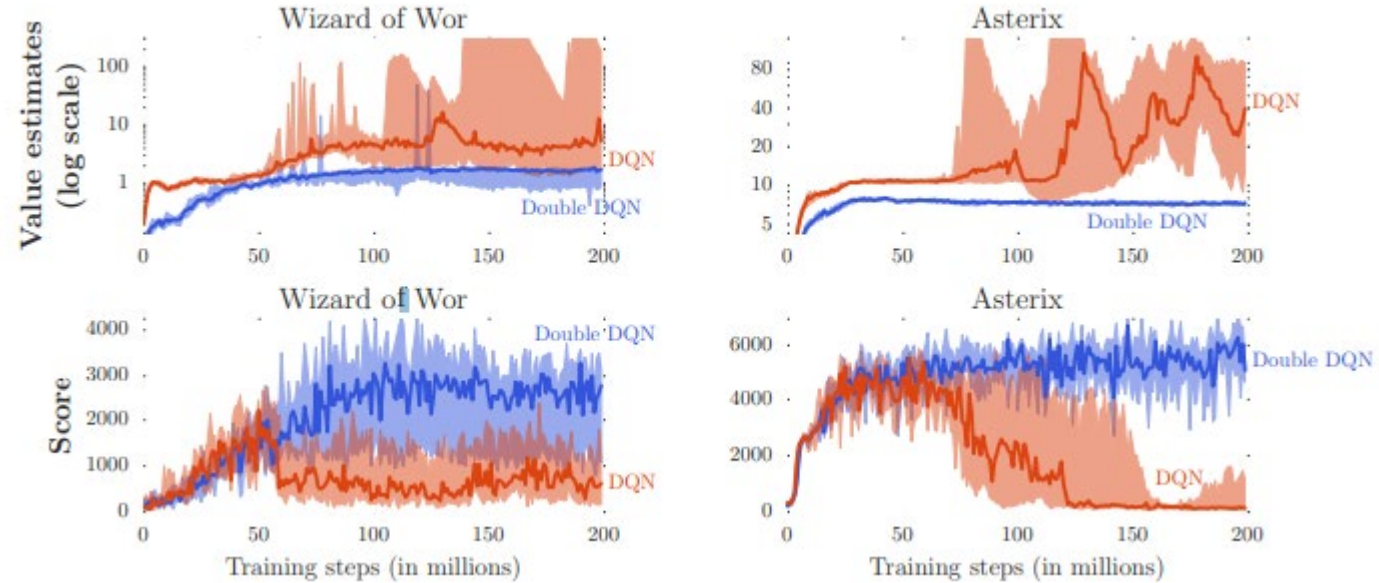
But also leads to better policies

Why? Not indicated in paper T.T

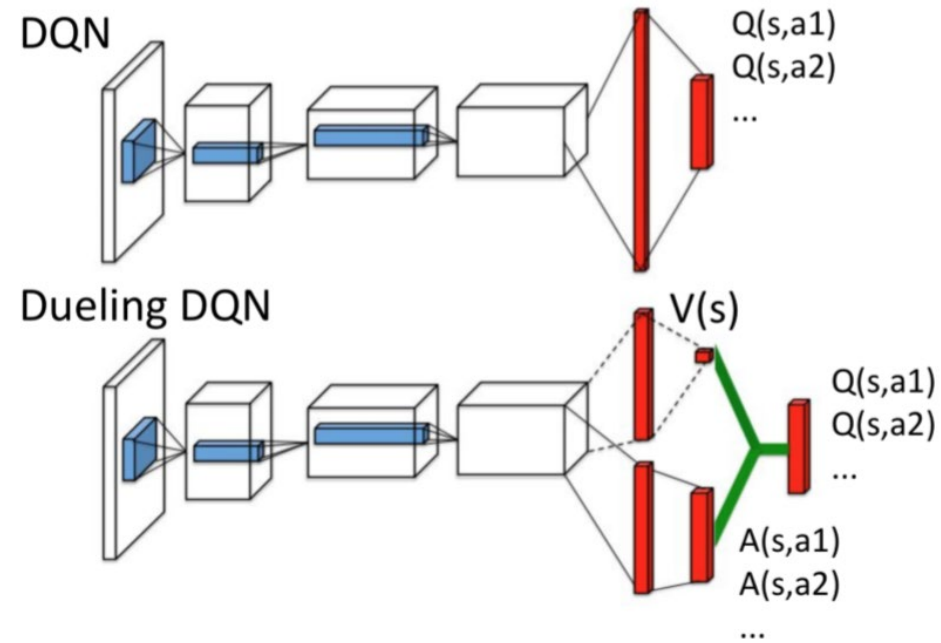
Double DQN

$$\Delta \mathbf{w} = \alpha \left(r + \gamma \underbrace{\hat{Q}(\arg \max_{a'} \hat{Q}(s', a'; \mathbf{w}^-); \mathbf{w}^-)}_{\text{Action selection: } \mathbf{w}} - \hat{Q}(s, a; \mathbf{w}) \right)$$

Action evaluation: \mathbf{w}^-



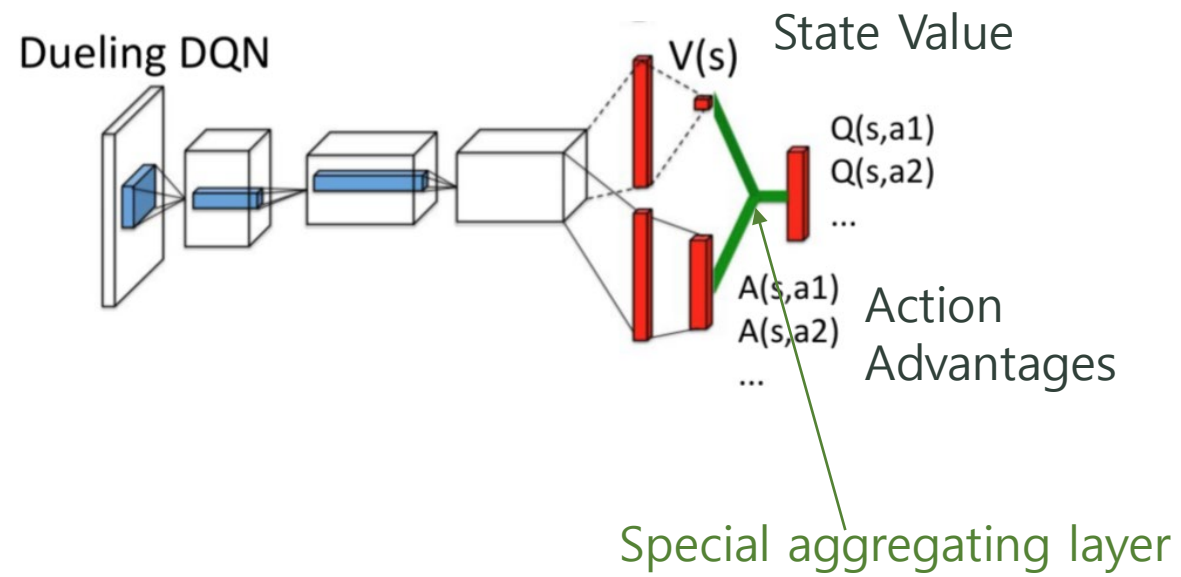
From preventing overestimation, becomes more stable == less noisy



Advantage : How much better or worse taking a particular action versus following the current policy

Identifiability : Whether there exists a unique Q for A and V given π (policy)

Dueling DQN $A^\pi(s, a) = Q^\pi(s, a) - V^\pi(s)$ This is misleading!



Dueling DQN

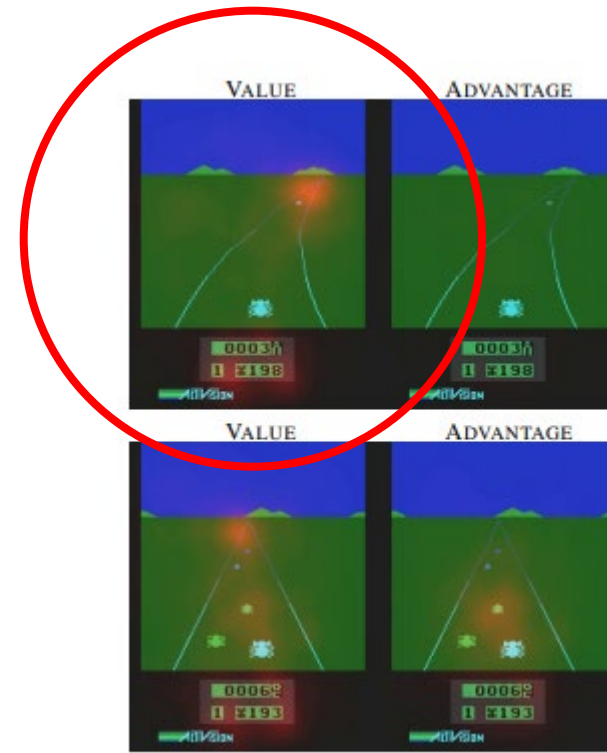
$$Q(s, a; \theta, \alpha, \beta) = V(s; \theta, \beta) + A(s, a; \theta, \alpha), \quad (7)$$

그래서 왜 이리 하는건데?

Learn which states are valuable and which states are not = there can be some states that the action does not affect the env.

Saliency map on right

The state probably depends on if new cars come out from horizon and the score

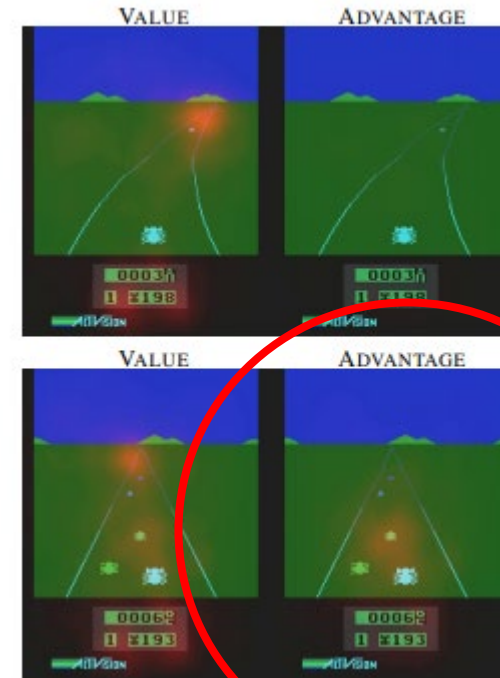


Dueling DQN
$$Q(s, a; \theta, \alpha, \beta) = V(s; \theta, \beta) + A(s, a; \theta, \alpha), \quad (7)$$

그래서 왜 이리 하는건데?

Note when there are no cars, there is no attention for the advantage stream

But when there are cars nearby, the point of interest is on the nearby areas where actions are important in avoiding the incoming cars



Dueling DQN $Q(s, a; \theta, \alpha, \beta) = V(s; \theta, \beta) + A(s, a; \theta, \alpha), \quad (7)$

So how to define A?

Need to be able to solve the problem that:

Given Q, cannot recover V and A uniquely

Two solutions

$$Q(s, a; \theta, \alpha, \beta) = V(s; \theta, \beta) + \left(A(s, a; \theta, \alpha) - \max_{a' \in |\mathcal{A}|} A(s, a'; \theta, \alpha) \right). \quad (8)$$

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

Dueling DQN $Q(s, a; \theta, \alpha, \beta) = V(s; \theta, \beta) + A(s, a; \theta, \alpha), \quad (7)$

So how to define A?

$$Q(s, a; \theta, \alpha, \beta) = V(s; \theta, \beta) + \left(A(s, a; \theta, \alpha) - \max_{a' \in |\mathcal{A}|} A(s, a'; \theta, \alpha) \right). \quad (8)$$

How would you calc A and V unique values from just Q value?

Q value for unique a – Q value avg?

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

Dueling DQN $Q(s, a; \theta, \alpha, \beta) = V(s; \theta, \beta) + A(s, a; \theta, \alpha), \quad (7)$

So how to define A?

$$Q(s, a; \theta, \alpha, \beta) = V(s; \theta, \beta) + \left(A(s, a; \theta, \alpha) - \max_{a' \in |\mathcal{A}|} A(s, a'; \theta, \alpha) \right). \quad (8)$$

어쨌든 결국 애를 쓴다
DDQN 이랑 하면 성능
이 더 좋다

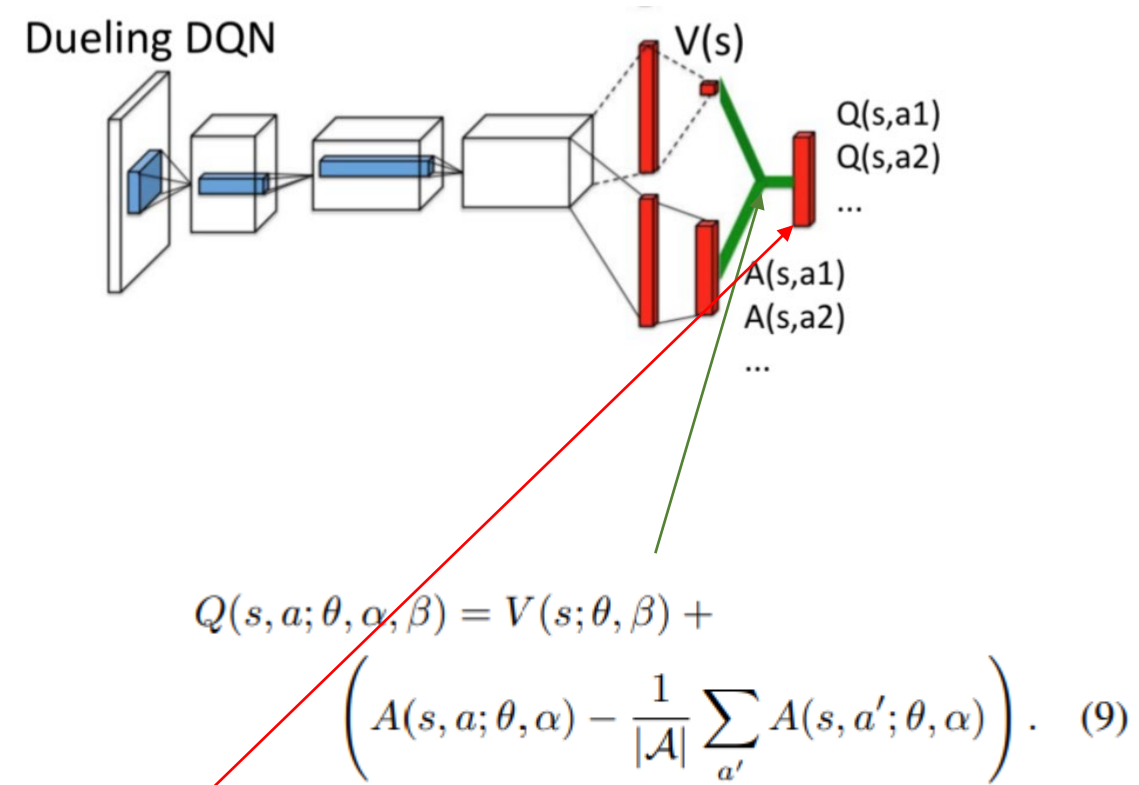
해볼만한거 DDDQN 에
서 둘이 비교해보기

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

Dueling DQN $Q(s, a; \theta, \alpha, \beta) = V(s; \theta, \beta) + A(s, a; \theta, \alpha), \quad (7)$

network

Our network architecture has the same low-level convolutional structure of DQN (Mnih et al., 2015; van Hasselt et al., 2015). There are 3 convolutional layers followed by 2 fully-connected layers. The first convolutional layer has $32\ 8 \times 8$ filters with stride 4, the second $64\ 4 \times 4$ filters with stride 2, and the third and final convolutional layer consists $64\ 3 \times 3$ filters with stride 1. As shown in Figure 1, the dueling network splits into two streams of fully connected layers. The value and advantage streams both have a fully-connected layer with 512 units. The final hidden layers of the value and advantage streams are both fully-connected with the value stream having one output and the advantage



Then... this is not a layer? Just simple maths?

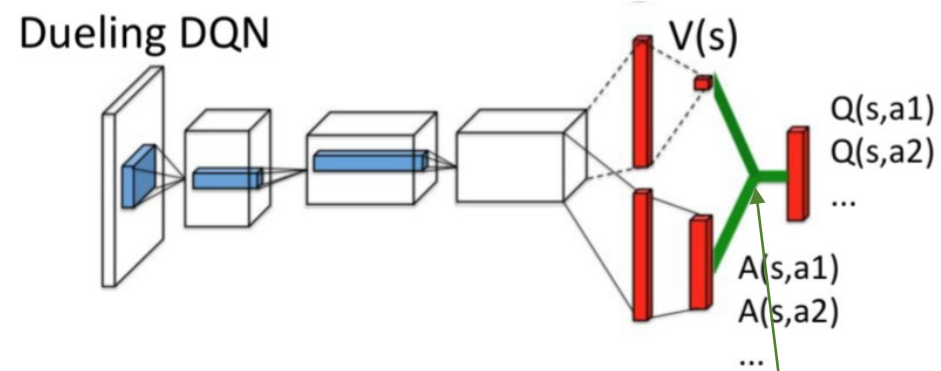
Dueling DQN

Part of network but not necessarily a layer?

doesn't this mean that there is another final layer...?

It is important to note that equation (9) is viewed and implemented as part of the network and not as a separate algorithmic step. Training of the dueling architectures, as with standard Q networks (e.g. the deep Q -network of Mnih et al. (2015)), requires only back-propagation. The estimates $V(s; \theta, \beta)$ and $A(s, a; \theta, \alpha)$ are computed automatically without any extra supervision or algorithmic modifications.

network



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

Dueling DQN

$$Q(s, a; \theta, \alpha, \beta) = V(s; \theta, \beta) + A(s, a; \theta, \alpha), \quad (7)$$

TD Error - Priority of a tuple is
proportional to DQN error

Prioritized
order replay

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

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

TD Error - Required to be replayed When transition has no replays, it has highest priority to calc TD error

Prioritized
order replay

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

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

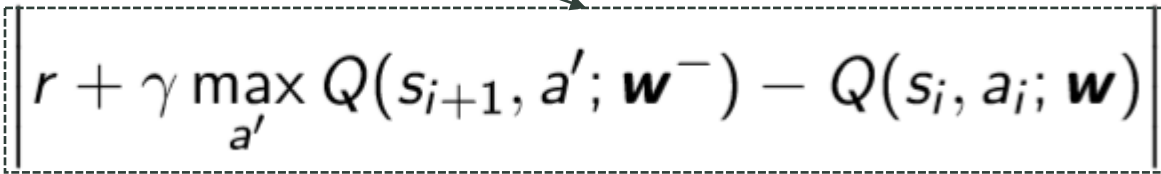
Problem: Greedy prioritization can make initially high error transition to be replayed frequently == lack of diversity for learning

Solution: Interpolate between greedy prioritization and uniform random sampling == ensure non-zero prob for lowest-priority transition

TD Error - Required to be replayed When transition has no replays, it has highest priority to calc TD error

This is especially beneficial because recent transitions tend to have large error : much to learn from :)

Prioritized
order replay


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

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

Alpha determines how much prioritization is used

1: greedy prioritization

0: uniform random sampling

Prioritized
order replay

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

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

proportional prioritization where $p_i = |\delta_i| + \epsilon$,
rank-based prioritization where $p_i = \frac{1}{\text{rank}(i)}$,

Another problem: PER introduce bias since it changes the distribution in an uncontrolled fashion

Solution : Importance Sampling

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

Prioritized
order replay

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

proportional prioritization where $p_i = |\delta_i| + \epsilon$,
rank-based prioritization where $p_i = \frac{1}{\text{rank}(i)}$,

Rambling about importance sampling

Another problem: PER introduce bias since it changes the distribution in an uncontrolled fashion

Solution : Importance Sampling

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

Compensate for bias: where some transitions can be replayed with higher probability while others cannot

that fully compensates for the non-uniform probabilities $P(i)$ if $\beta = 1$. These weights can be folded into the Q-learning update by using $w_i \delta_i$ instead of δ_i (this is thus *weighted* IS, not ordinary IS, see e.g. Mahmood et al., 2014). For stability reasons, we always normalize weights by $1/\max_i w_i$ so that they only scale the update downwards.

Beta determines how much to compensate for non-uniform properties.

If beta = 1, fully compensates for non-uniform properties

Q. Does this mean that it becomes same as uniform sampling?

Possible technique is where beta increases as learning progresses, until beta = 1

Rambling about importance sampling

Another problem: PER introduce bias since it changes the distribution in an uncontrolled fashion

Solution : Importance Sampling

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

Gradient is reliable only locally so for global step-size it has to be modified smaller

IS does the job

"prioritization makes sure high-error transitions are seen many times, while the IS corrections reduces the gradient magnitued

PER + IS

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

proportional prioritization where $p_i = |\delta_i| + \epsilon$,

+

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

Deepmind found the sweet spot as alpha = 0.7 and beta = 0.5

코드 여기 있어용

[https://github.com/laphisboy/RL_fall/tree/master/fall week 3](https://github.com/laphisboy/RL_fall/tree/master/fall_week_3)