

---

# VARIANTES DE DQN

# DOUBLE DQN

- ▶ Double DQN fue propuesto en 2016 por Hasselt y otros bajo el título de “Deep Reinforcement Learning with Double Q-Learning” (Hasselt et al., 2016).



Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI-16)

**Deep Reinforcement Learning with Double Q-Learning**

**Hado van Hasselt , Arthur Guez, and David Silver**  
Google DeepMind

**Abstract**

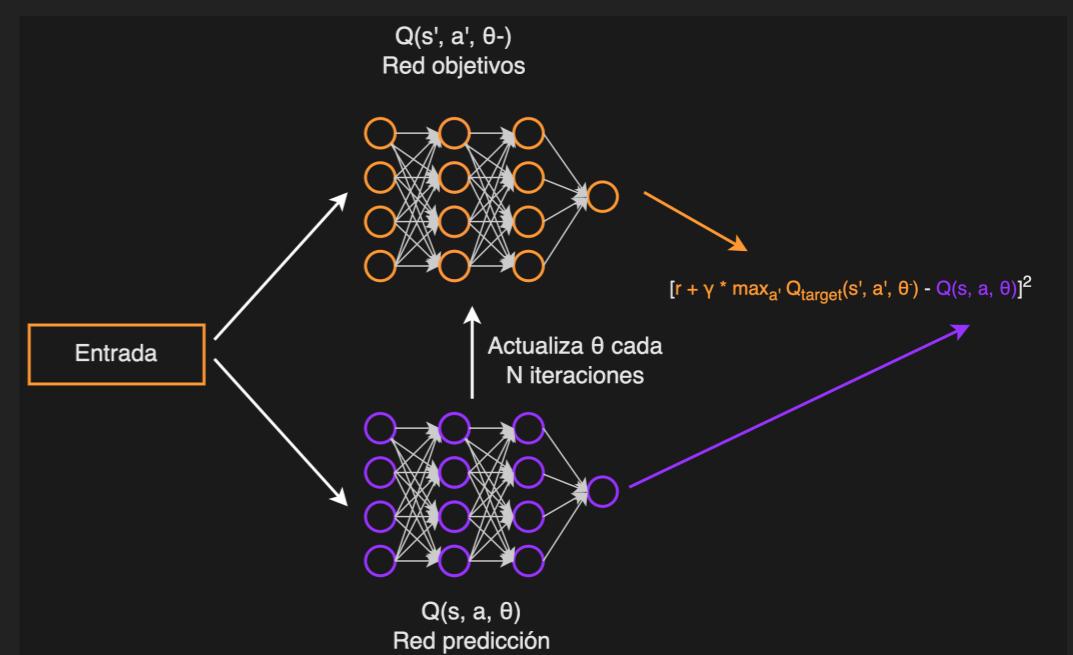
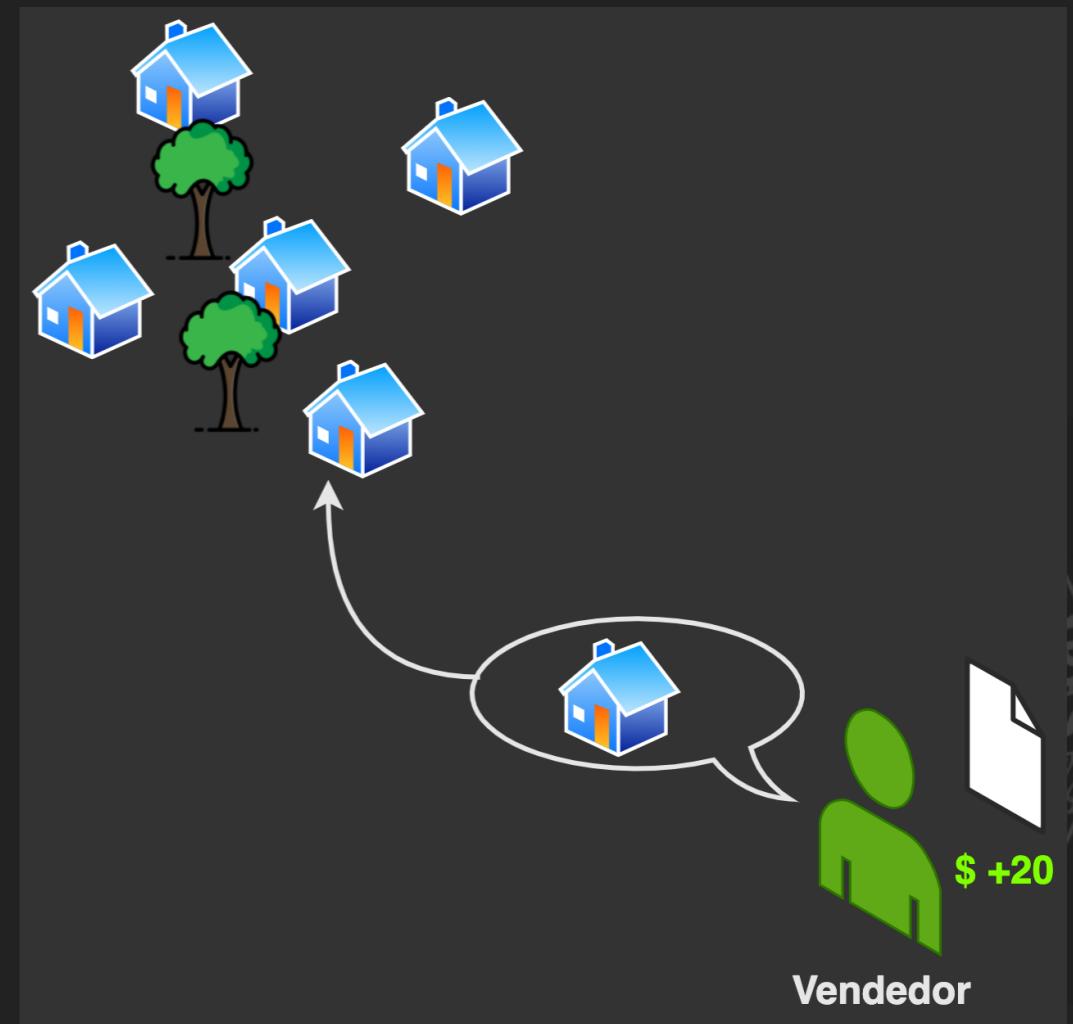
The popular Q-learning algorithm is known to overestimate action values under certain conditions. It was not previously known whether, in practice, such overestimations are common, whether they harm performance, and whether they can generally be prevented. In this paper, we answer all these questions affirmatively. In particular, we first show that the recent DQN algorithm, which combines Q-learning with a deep neural network, suffers from substantial overestimations in some games in the Atari 2600 domain. We then show that the idea behind the Double Q-learning algorithm, which was introduced in a tabular setting, can be generalized to work with large-scale function approximation. We propose a specific adaptation to the DQN algorithm and show that the resulting algorithm not only reduces the observed overestimations, as hypothesized, but that this also leads to much better performance on several games.

exploration technique (Kaelbling et al. 1996). If, however, the overestimations are not uniform and not concentrated at states about which we wish to learn more, then they might negatively affect the quality of the resulting policy. Thrun and Schwartz (1993) give specific examples in which this leads to suboptimal policies, even asymptotically.

To test whether overestimations occur in practice and at scale, we investigate the performance of the recent DQN algorithm (Mnih et al. 2015). DQN combines Q-learning with a flexible deep neural network and was tested on a varied and large set of deterministic Atari 2600 games, reaching human-level performance on many games. In some ways, this setting is a best-case scenario for Q-learning, because the deep neural network provides flexible function approximation with the potential for a low asymptotic approximation error, and the determinism of the environments prevents the harmful effects of noise. Perhaps surprisingly, we show

## COMO FUNCIONA DOUBLE DQN (DDQN)?

- ▶ Analogía de DQN: **Valor de una casa**
- ▶ DQN original tiende a ser demasiado optimista.
- ▶ Sobreestima el valor de ciertas acciones porque usa la misma "calculadora" (la red neuronal) para elegir la mejor acción futura y para evaluar cuán buena es esa acción.
- ▶ Si accidentalmente sobreestima el valor de una acción, es más probable que la elija y refuerce esa sobreestimación.



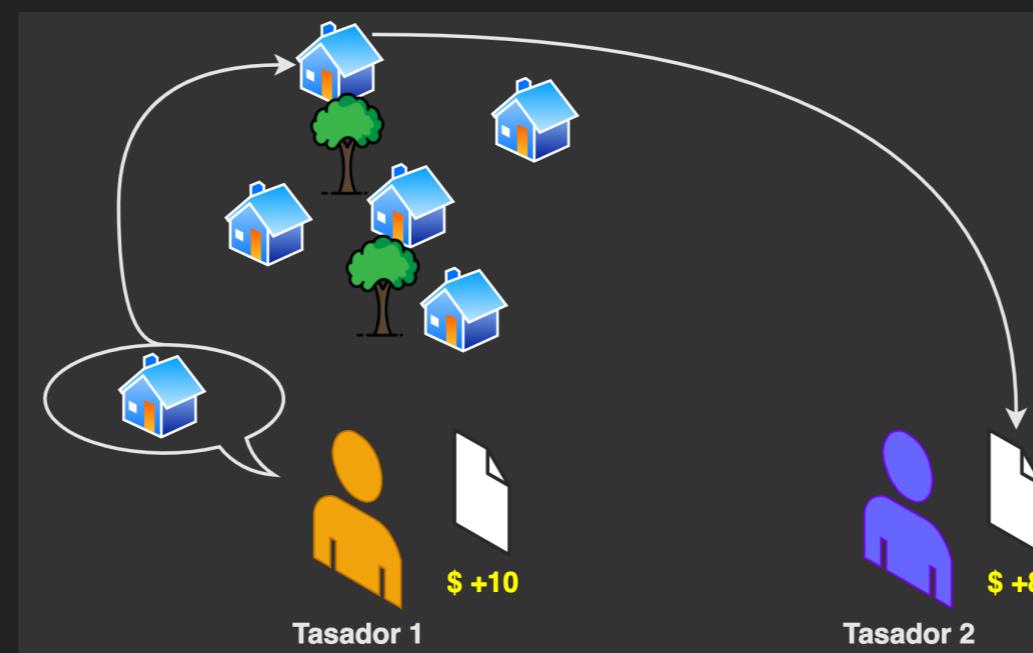
# COMO FUNCIONA DOUBLE DQN (DDQN)?

► Analogía de Double DQN: **Valor de una casa**

► DDQN es como contratar a dos tasadores:

- ✓ **Tasador 1** (Red Online): Mira todas las casas comparables y sugiere cuál parece ser la mejor (**selecciona la acción  $a'$** ).
- ✓ **Tasador 2** (Red Target, más conservadora y actualizada con menos frecuencia): Toma la casa específica que sugirió el Tasador 1 y le da su propia valoración (**evalúa la acción  $a'$  usando los pesos de la red Target**).

► **Resultado:** Al separar quien elige la mejor opción de quién le asigna el valor final, **se reduce el sesgo optimista**. Es como tener una segunda opinión más objetiva para valorar la opción ya elegida, evitando que el optimismo se acumule.



# EN QUÉ AFECTA LA SOBREESTIMACIÓN DE Q EN DQN?

## ‣ Aprendizaje de Políticas Subóptimas (Hasselt et al., 2015):

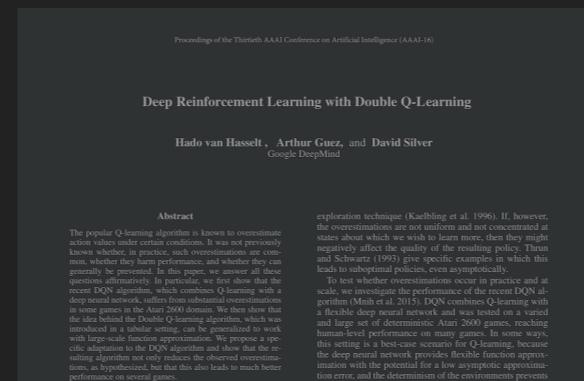
- ✓ DQN elige acciones basadas en el valor máximo estimado  $\max_a Q(s,a)$ , sobreestimados a veces.
- ✓ .. el agente puede aprender y converger hacia políticas que son peores (subóptimas).

## ‣ Inestabilidad en el Aprendizaje:

- ✓ Los picos de sobreestimación en los valores Q coinciden con caídas drásticas en el rendimiento (la puntuación obtenida).
- ✓ .. puede desestabilizar el proceso de aprendizaje, haciendo que el rendimiento fluctúe mucho o incluso empeore a medida que las estimaciones se vuelven irrealmente altas.

## ‣ Propagación de Errores y Estimaciones Engañosas:

- ✓ Al usar bootstrapping, si los valores Q para los estados siguientes ( $s'$ ) ya están sobreestimados, estos valores inflados se usan para calcular el objetivo del estado actual ( $s$ ), propagando y potencialmente amplificando la sobreestimación a través del espacio de estados.
- ✓ .. este "efecto pernicioso de propagar información relativa incorrecta" afecta directamente la calidad de la política aprendida.

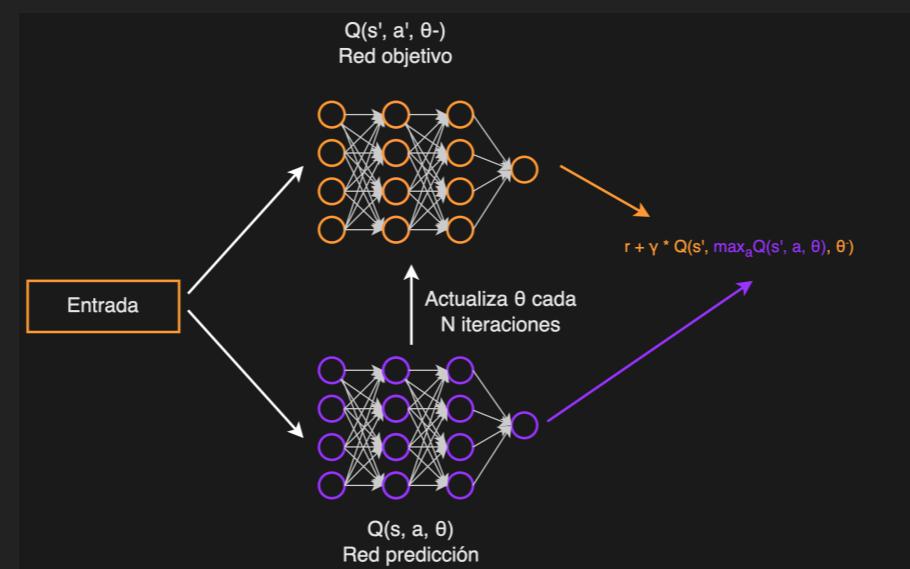


## QUE PROPONEN LOS AUTORES PARA SOLUCIONARLO?

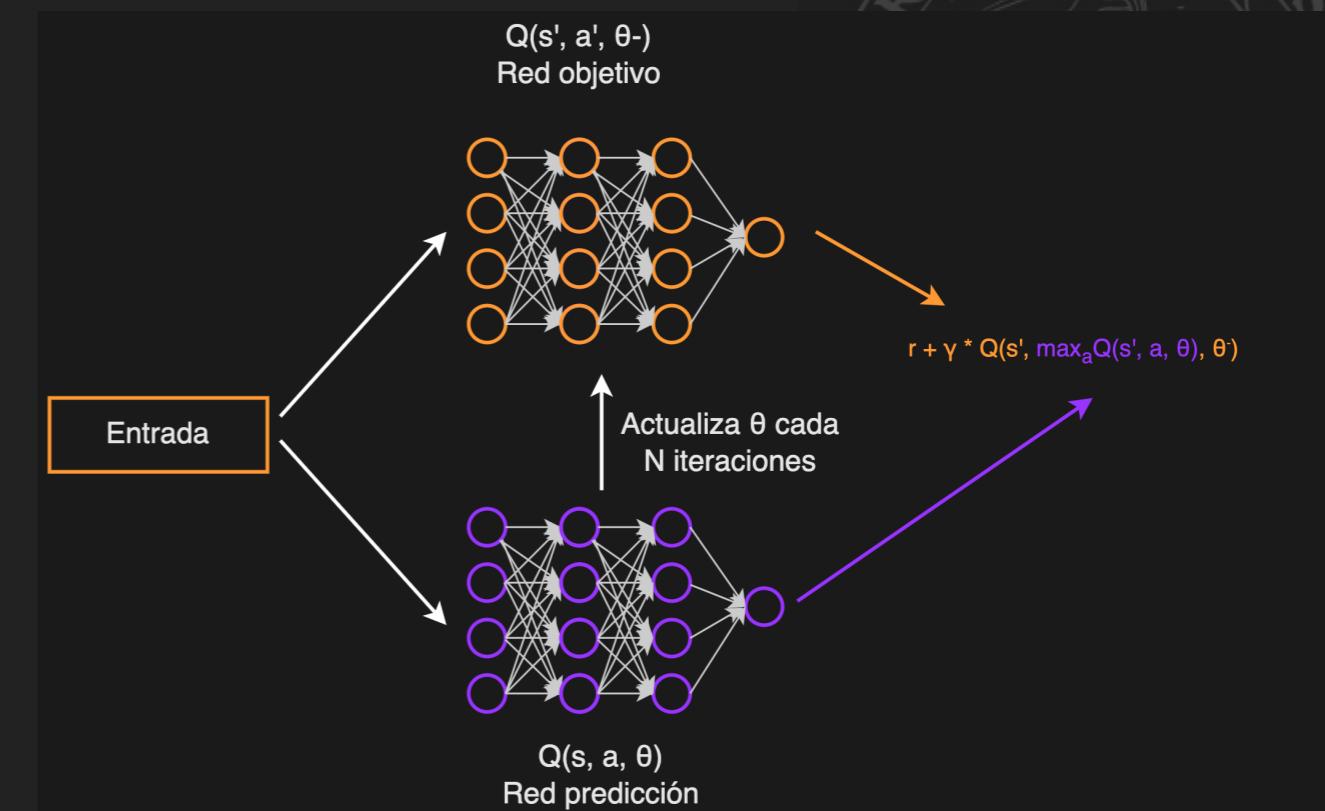
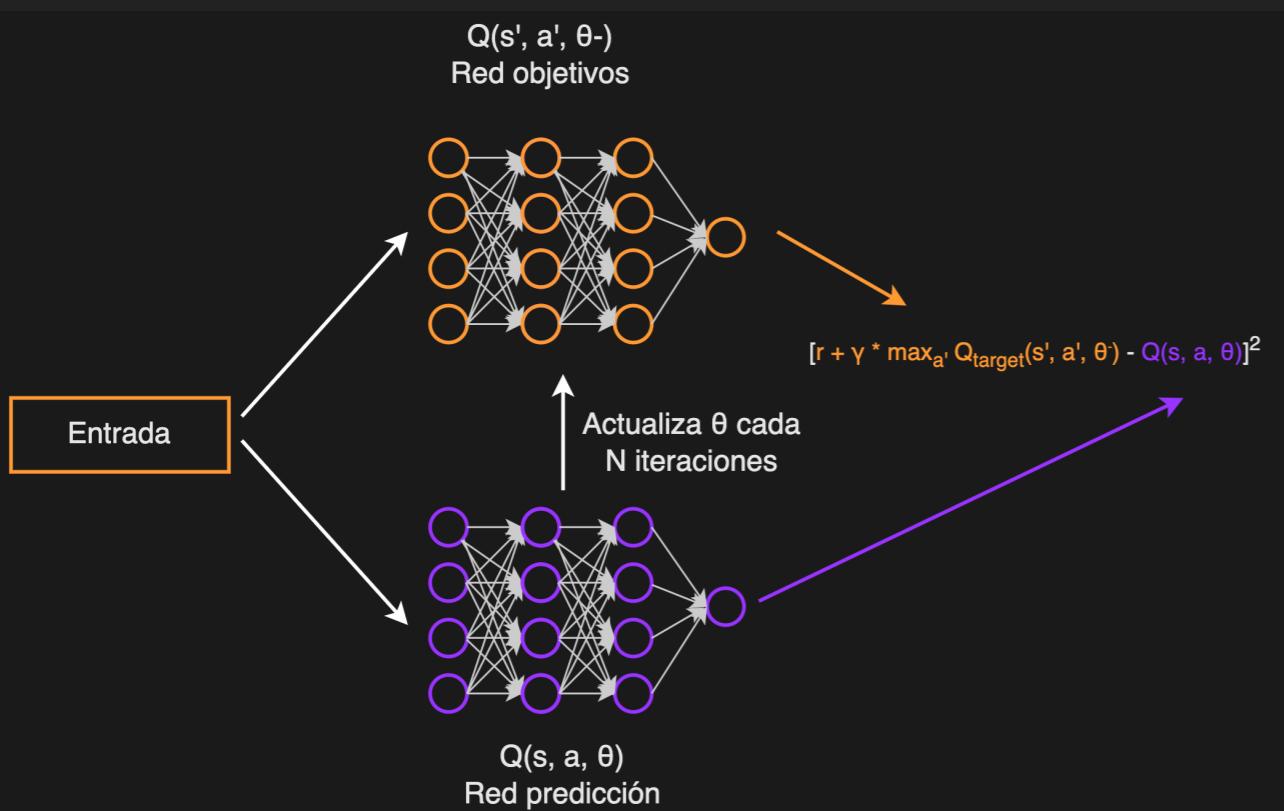
- ▶ Los autores proponen implementar la política greedy con la red en línea, pero utilizar la red objetivo para estimar el valor de la acción  $a'$ .
- ▶ La actualización es la misma que para DQN, pero reemplazando el  $Y_t^{\text{DQN}}$  objetivo por:

$$Y_t^{\text{DoubleDQN}} \equiv R_{t+1} + \gamma Q(S_{t+1}, \operatorname{argmax}_a Q(S_{t+1}, a; \theta_t), \theta_t^-)$$

- ▶ La actualización de la red objetivo se mantiene sin cambios con respecto a DQN y se mantiene como una copia periódica de la red en línea.



# QUE PROPONEN LOS AUTORES PARA SOLUCIONARLO?



## QUE HACE DQN Y DDQN

- ▶ En **DQN** clásico la target o valor-objetivo se calcula como:

$$y = r + \gamma \max_{a'} Q_{\text{target}}(s', a'; \theta^-)$$

- ▶ El  $\max_{a'}$  hace dos cosas a la vez:

- ✓ Selecciona la acción  $a'$  que tenga el mayor valor Q estimado para el estado  $s'$ .
  - ✓ Evalúa (usa) directamente  $Q_{\text{target}}(s', a'; \theta^-)$  para calcular el valor  $y$ .

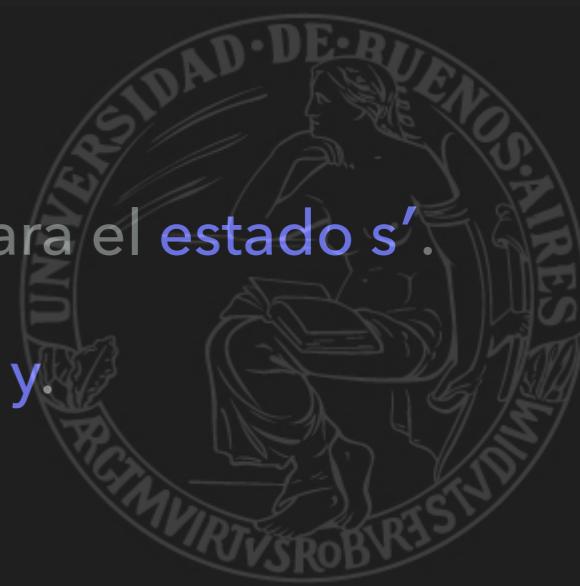
- ▶ En **Double DQN** separa la **selección** y la **evaluación** en dos pasos:

- ✓ Con la red online  $\theta$ , selecciona la mejor acción

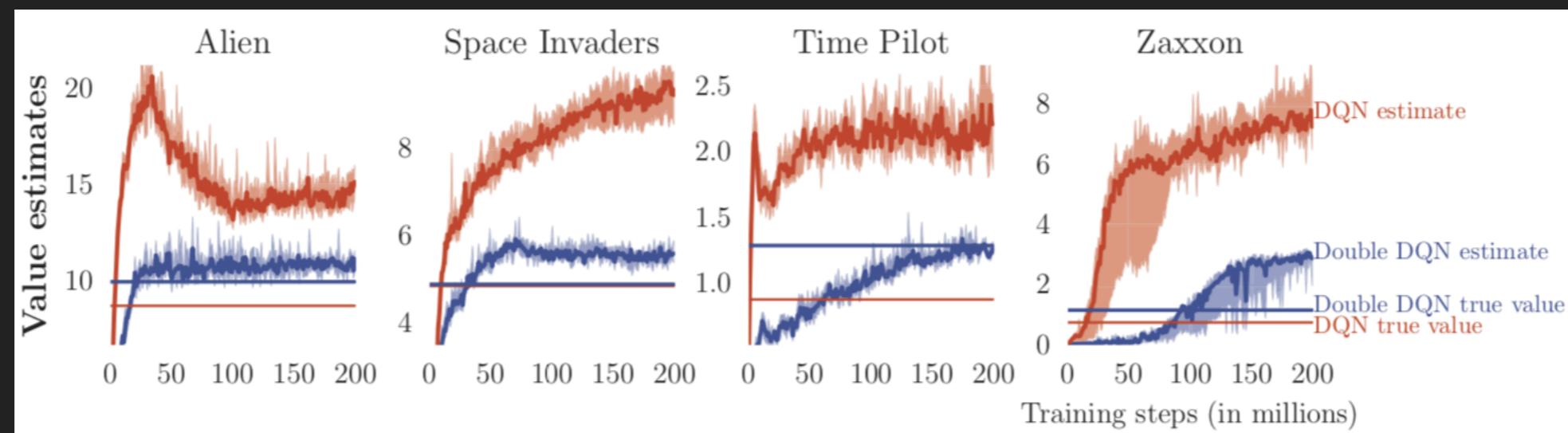
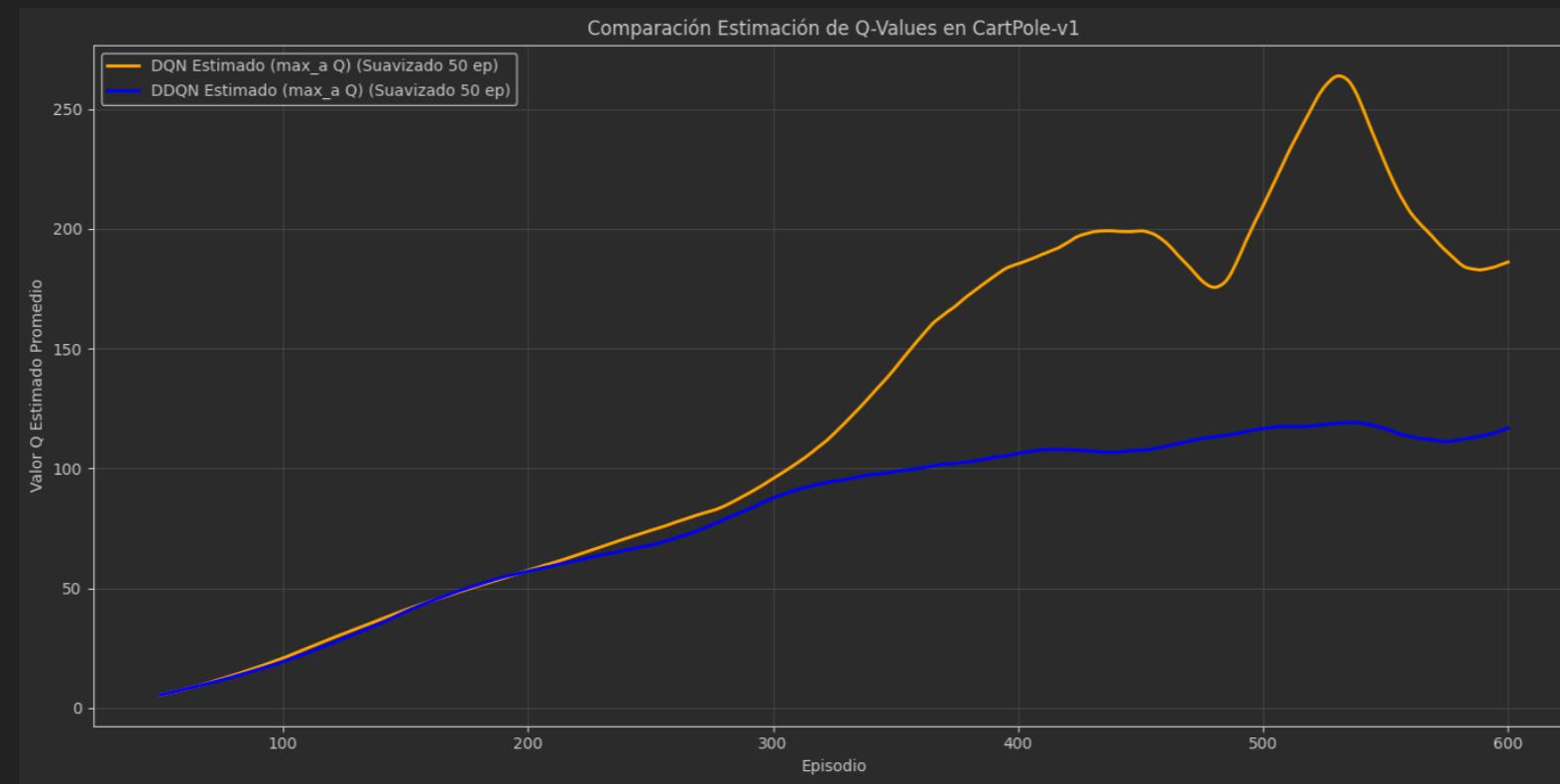
$$a = \arg \max_a Q(s', a; \theta)$$

- ✓ Con la red target  $\theta^-$ , evalúa esa acción

$$y = r + \gamma Q(s', a; \theta^-)$$



# COMPARATIVA ENTRE DQN Y DDQN



# DUELING DQN

- ▶ **Dueling DQN** fue propuesto en 2016 por Wang y otros bajo el título de “Dueling Network Architectures for Deep Reinforcement Learning” (Wang et al., 2016).

**Dueling Network Architectures for Deep Reinforcement Learning**

Ziyu Wang  
Tom Schaul  
Matteo Hessel  
Hado van Hasselt  
Marc Lanctot  
Nando de Freitas  
Google DeepMind, London, UK

ZIYU@GOOGLE.COM  
SCHAUL@GOOGLE.COM  
MTTHSS@GOOGLE.COM  
HADO@GOOGLE.COM  
LANCTOT@GOOGLE.COM  
NANDODEFREITAS@GOOGLE.COM

**Abstract**

In recent years there have been many successes of using deep representations in reinforcement learning. Still, many of these applications use conventional architectures, such as convolutional networks, LSTMs, or auto-encoders. In this paper, we present a new neural network architecture for model-free reinforcement learning. Our dueling network represents two separate estimators: one for the state value function and one for the state-dependent action advantage function. The main benefit of this factoring is to generalize learning across actions without imposing any change to the underlying reinforcement learning algorithm. Our results show that this architecture leads to better policy evaluation in the presence of many similar-valued actions. Moreover, the dueling architecture enables our RL agent to outperform the state-of-the-art on the Atari 2600 domain.

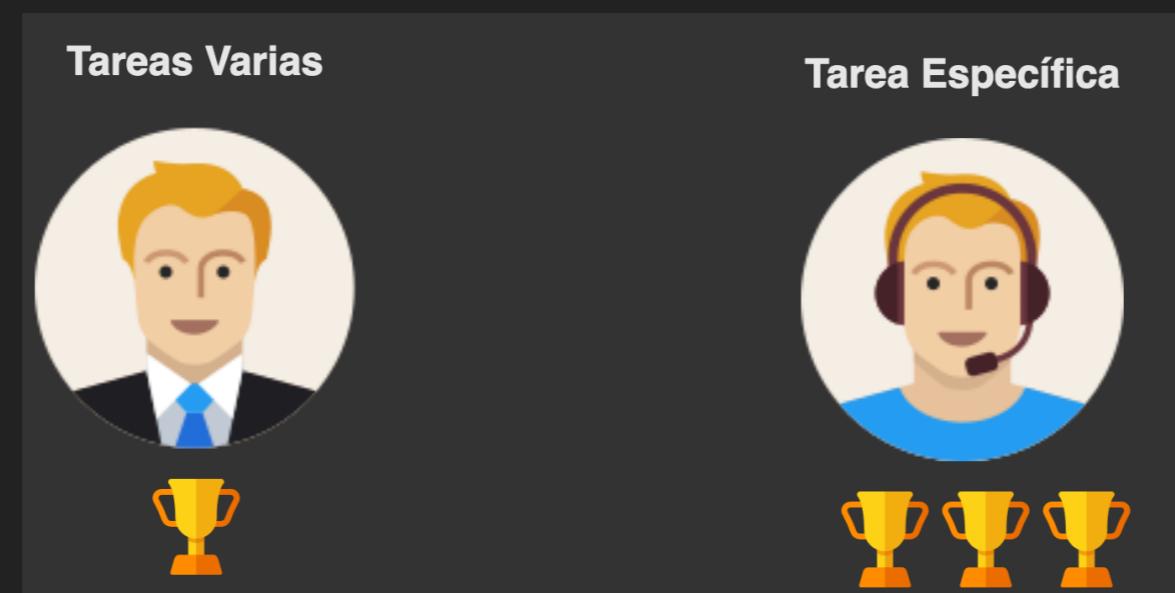
which produced policies matching those of Monte Carlo tree search programs, and squarely beaten a professional player when combined with search (Silver et al., 2016).  
In spite of this, most of the approaches for RL use standard neural networks, such as convolutional networks, MLPs, LSTMs and autoencoders. The focus in these recent advances has been on designing improved control and RL algorithms, or simply on incorporating existing neural network architectures into RL methods. Here, we take an *alternative but complementary approach* of focusing primarily on innovating a neural network architecture that is better suited for model-free RL. This approach has the benefit that the new network can be easily combined with existing and future algorithms for RL. That is, this paper advances a new network (Figure 1), but uses already published algorithms.

The proposed network architecture, which we name the *dueling architecture*, explicitly separates the representation of



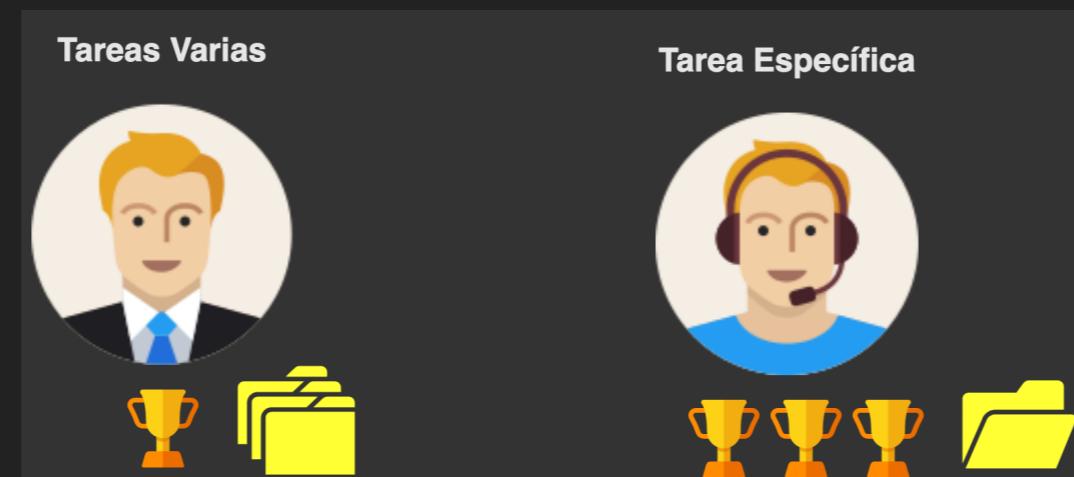
# COMO FUNCIONA DUELING DQN?

- ▶ Analogía: Valoración de un empleado
- ▶ Una empresa piensa en evaluar a un empleado para una tarea específica. Su **valor total (Q)** depende de dos cosas:
  1. Su **valor general** para la empresa (**Valor del Estado, V(s)**): ¿Es un empleado valioso, experimentado, confiable, independientemente de la tarea actual?)
  2. Qué tan adecuado es para esta **tarea específica** en comparación con otras (**Ventaja de la Acción, A(s,a)**): ¿Tiene las habilidades exactas para esta tarea? ¿Es mucho mejor en esto que en sus tareas promedio?)



# COMO FUNCIONA DUELING DQN?

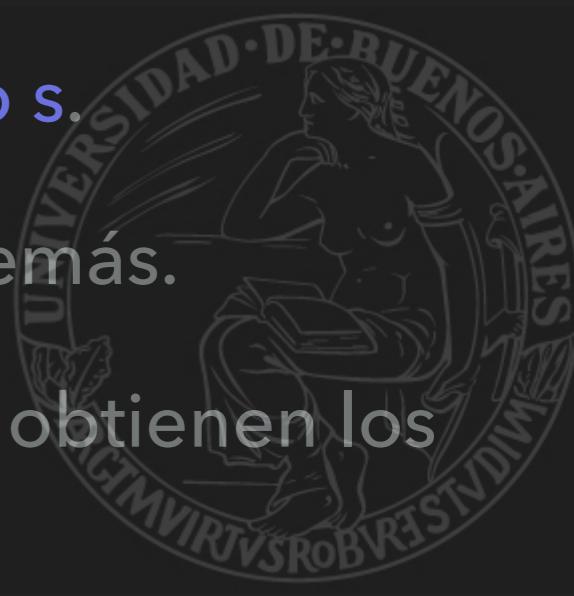
- ▶ Cómo lo resuelve Dueling DQN: Usa una red neuronal que se divide internamente en 2 "departamentos":
  - ✓ Departamento 1: Estima el **valor general** del estado  $V(s)$ . El valor base del empleado.
  - ✓ Departamento 2: Estima la **ventaja (Advantage)** de cada acción posible en ese estado  $A(s,a)$ . Es decir, qué tan bueno es el empleado para cada tarea específica comparado con su promedio.
- ▶ **Resultado:** Dueling DQN combina estas 2 estimaciones para obtener el **Q-value final**. Esto permite que **la red aprenda el valor de los estados de forma más eficiente**, especialmente cuando **muchas acciones tienen un valor similar** (la red puede centrarse en aprender  $V(s)$  sin preocuparse demasiado por las pequeñas diferencias en  $A(s,a)$  para acciones irrelevantes).



# COMO FUNCIONA DUELING DQN?

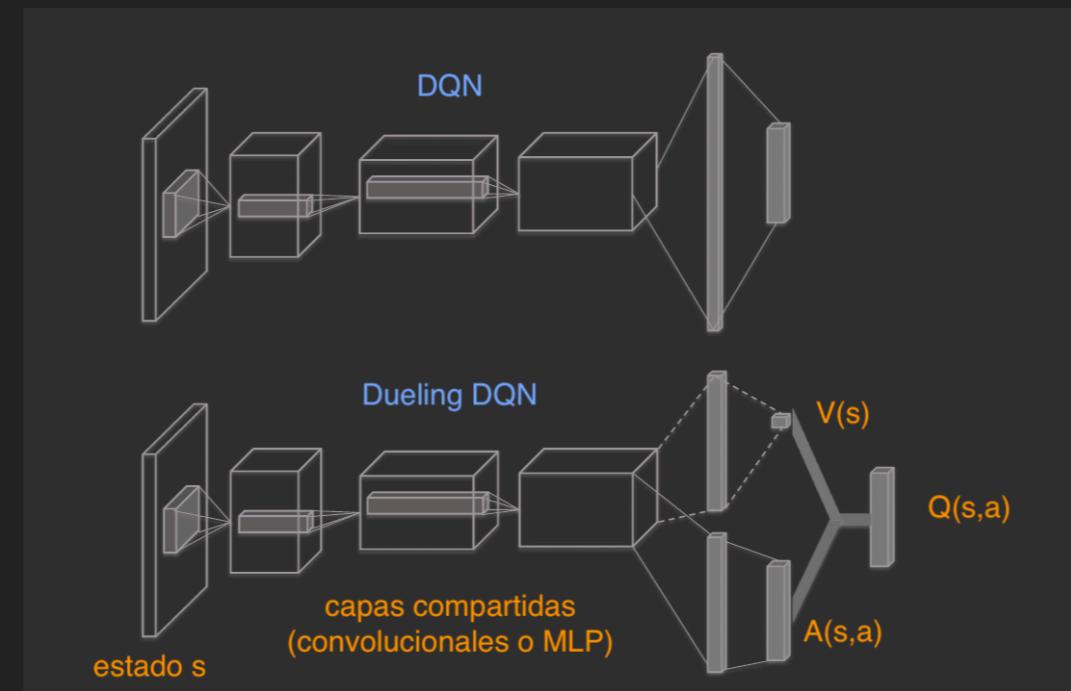
- ▶ Dueling DQN divide los Q-values en 2 partes, la función de valor (value function)  $V(s)$  y la función de ventaja (advantage function)  $A(s, a)$ .
- ▶  $V(s)$  dice cuánta recompensa se obtendrá desde el estado  $s$ .
- ▶  $A(s, a)$  indica cuánto mejor es una acción respecto a las demás.
- ▶ Al combinar el valor  $V$  con la ventaja  $A$  de cada acción, se obtienen los valores-Q:

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



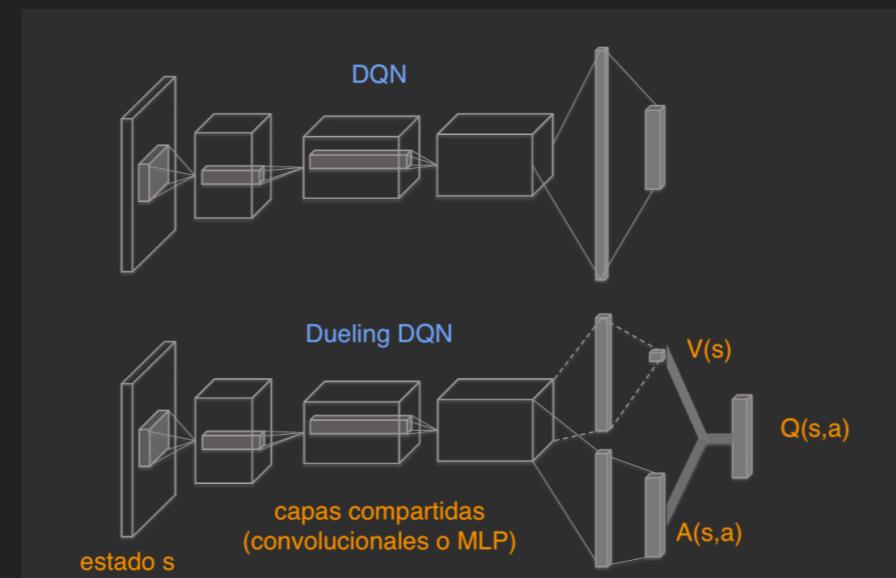
# COMO FUNCIONA DUELING DQN?

- ▶ Dueling DQN propone que la red neuronal divida su capa final en 2 partes (streams):
  - ✓ Una para estimar el valor del estado  $s$ ,  $V(s)$
  - ✓ Otra para estimar la ventaja de cada acción  $a$ ,  $A(s, a)$
- ▶ y luego unir ambas partes en una sola que estimará los **valores-Q**.
- ▶ Este planteo es útil en algunos casos, dado a veces **no es necesario saber exactamente al valor de cada acción**, es decir, que aprender el valor del estado puede ser suficiente.



# INIDENTIFICABILIDAD

- ▶ Entrenar la red neuronal de esta forma, es decir, sumando el valor **V** y la ventaja **A** en la capa final, no es posible.
- ▶ Si se tiene  $Q=V+A$ , a partir de la función  $Q$ , no se puede determinar **V** y **A**, es decir, que el problema es “**inidentifiable**”.
- ▶ Ejemplo 1: si  $Q = 20$ , y se busca saber qué dos números suman  $20$  ( $20 = V + A$ ) => infinitas posibilidades.



# INIDENTIFICABILIDAD

## ► Ejemplo 2:

✓ Expresión 1: "El árbol del vecino llena de hojas la vereda"

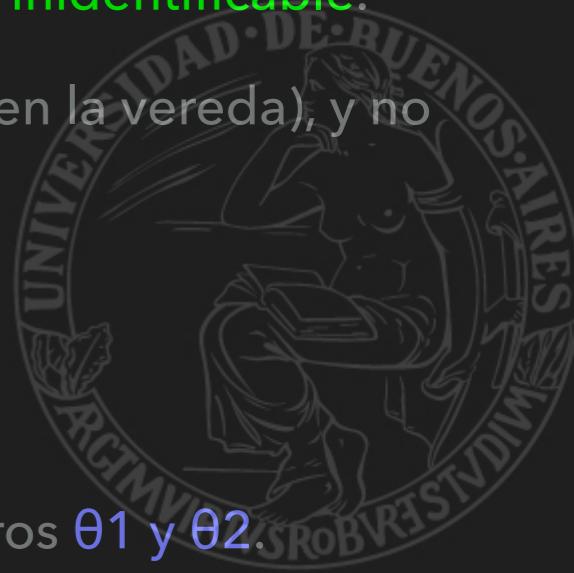
✓ Expresión 2: "Un árbol llena de hojas la vereda"

## ► Ambas describen el mismo efecto observable: la vereda llena de hojas (datos x).



# INIDENTIFICABILIDAD

- ▶ Causa (parámetro):
  - ✓ En Expresión 1: se identifica cuál árbol fue → parámetro bien definido → modelo **identifiable**.
  - ✓ En Expresión 2: no se sabe cuál árbol fue → parámetro ambiguo → modelo **inidentifiable**.
- ▶ Si dos causas diferentes  $\theta$  (árboles) generan el mismo efecto observable "x" (hojas en la vereda), y no se puede distinguir cuál fue, entonces el "modelo" es **inidentifiable**.
- ▶ En estadística un modelo es **inidentifiable** si existen  $\theta_1 \neq \theta_2$  tales que:
$$p(x | \theta_1) = p(x | \theta_2)$$
- ▶ Eso significa que, viendo solo los **datos x**, no podemos distinguir entre los parámetros  $\theta_1$  y  $\theta_2$ .



# SOLUCIÓN PARA LA INIDENTIFICABILIDAD

- Para solucionar la “inidentificabilidad”, los autores proponen un artificio matemático que consiste en forzar al Q-value más alto a ser igual al valor  $V$ , de modo que el valor más alto en la función de ventaja  $A$  sea cero, y los demás valores sean negativos.

$$\max_{a'} A(s, a') = 0$$

- Para la mejor acción se tiene:

$$Q(s, a) < V(s)$$

- Para las demás acciones se tendrá:

$$Q(s, a^*) = V(s)$$

- Esto dirá exactamente cuál es el valor de  $V$ , y permitirá calcular los valores de la función de ventaja  $A$  desde ahí, solucionando el problema quedando:

$$Q(s, a) = V(s) + \left( A(s, a) - \max_{a'} A(s, a') \right)$$



# EJEMPLO

- ▶ Supongamos que se tienen 3 acciones:  $a_1, a_2, a_3$  y que en un cierto **estado  $s$** , la red produce lo siguiente:
- ▶ **Caso 2:** La red calcula:

$$V(s) = 5$$

$$A(s, a_1) = 2$$

$$A(s, a_2) = 0$$

$$A(s, a_3) = -1$$

- ▶ Entonces:

$$Q(s, a_1) = 5 + 2 = 7,$$

$$Q(s, a_2) = 5 + 0 = 5,$$

$$Q(s, a_3) = 5 - 1 = 4$$



# EJEMPLO

- ▶ Caso 2: La red calcula:

$$V(s) = 6$$

$$A(s, a_1) = 1$$

$$A(s, a_2) = -1$$

$$A(s, a_3) = -2$$

- ▶ Tambien produce:

$$Q(s, a_1) = 5 + 2 = 7,$$

$$Q(s, a_2) = 5 + 0 = 5,$$

$$Q(s, a_3) = 5 - 1 = 4$$

- ▶ Mismo  $Q(s, a)$ , pero con distintos  $V(s)$  y  $A(s, a)$  → es Identifiable



## EJEMPLO

- ▶ Con la corrección forzando a cero tenemos que:

$$\max_a A(s,a) = 0$$

- ▶ Tomemos:

$$A(s,a_1) = 0$$

$$A(s,a_2) = -2$$

$$A(s,a_3) = -3$$

- ▶ y suponiendo que:

$$V(s) = 7$$

- ▶ Entonces:

$$Q(s,a_1) = 7 + (0) = 7, Q(s,a_2) = 7 + (-2) = 5, Q(s,a_3) = 7 + (-3) = 4$$

- ▶ Mismos Q-values que antes, pero ahora la descomposición es única



# SOLUCIÓN PARA LA INIDENTIFICABILIDAD

- ▶ Los autores sugieren un cambio mínimo a esta última ecuación, esto es, en lugar de usar el máximo, usar la media de las ventajas para:

- ✓ Suavidad y estabilidad del gradiente
- ✓ Evita el sesgo hacia una sola acción
- ✓ Mejor rendimiento empírico

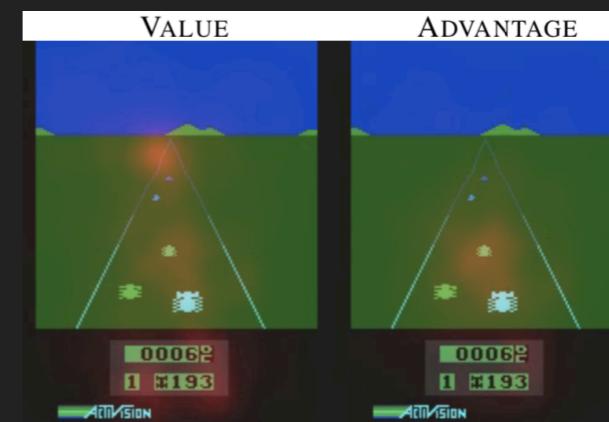
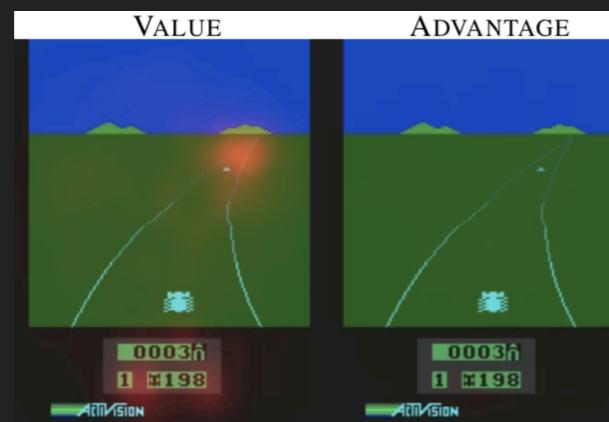
- ▶ Quedando:

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



## DUELING DQN EN ENDURO (ATARI)

- ▶ En el artículo de (Wang et al., 2016) se plantea 2 escenarios de juego Enduro (Atari).
  - ▶ Esc1: el auto solo tiene a otro auto por delante pero a lo lejos.
  - ▶ Esc2: el auto tiene otro auto que lo sobrepasa por la izquierda y otro que se le acerca por delante obstaculizando su carril.
- ▶ En el gráfico izquierdo de ambos escenarios se muestra en qué se fija (punto anaranjado) la función  $V(s)$ , mientras que en el gráfico derecho se observa en qué se fija la función  $A(s,a)$ .
- ▶ La función  $V(s)$  se fija más en los autos que se encuentran lejos que en los cercanos, donde empiezan a verse, en cualquiera de los escenarios y también se fija en la puntuación.
- ▶ La función  $A(s,a)$  en el primer escenario no señala nada debido a que no hay ningún auto delante y cercano que motive para forzar una acción; esto es, cualquier acción en ese estado será totalmente irrelevante.
- ▶ En el segundo escenario sí que señala el auto que está obstaculizando el carril, indicando que hay una acción a tomar para evitar la colisión.
- ▶ En este ejemplo, los valores Q del estado del primer escenario no serían calculados y la dueling DQN pasaría directamente a evaluar el siguiente estado.



## REFERENCIAS BIBLIOGRÁFICAS Y WEB (I)

- ▶ Van Hasselt, H., Guez, A., & Silver, D. (2016, March). Deep reinforcement learning with double q-learning. In Proceedings of the AAAI conference on artificial intelligence (Vol. 30, No. 1).
- ▶ Wang, Z., Schaul, T., Hessel, M., Hasselt, H., Lanctot, M., & Freitas, N. (2016, June). Dueling network architectures for deep reinforcement learning. In International conference on machine learning (pp. 1995-2003). PMLR.

