CARTA

doi:10.1038/naturaleza14236

Control a nivel humano a través del aprendizaje de refuerzo profundo

Volodymyr Mnih1 *, Koray Kavukcuoglu1 *, david plata 1 *, Andrei A. Rusu1, Joel Veness1, Marc G. Bellemare1, alex tumbas 1, Martín Riedmiller 1, Andreas K. Fidjeland1, Georg Ostrovski 1, Stig Petersen1, charles beattie1, Amir Sadik1, Ioannis Antonoglou1, helen rey1, Dharshan Kumaran1, Daan Wierstra1, Shane Legg1 y Demis Hassabis 1

La teoría del aprendizaje por refuerzo proporciona una explicación normativa y agentes pueden optimizar su control de [1], [2] perspectiva, profundamente arraigada en el comportamiento psicológico de los animales. obtener representaciones eficientes del medio ambiente a partir de entradas sensoriales de alta dimensión, y usarlas para generalizar la experiencia pasada a nuevas situaciones. Sorprendentemente, los humanos y otros animales parecen resolver este problema a través de una combinación armoniosa de aprendizaje reforzado y prosensorial jerárquico.

Los sistemas de [4,5], el primero evidenciado por una gran cantidad de datos neuronales procesamiento que revelan paralelismos notables entre las señales fásicas emitidas por las neuronas dopaminérgicas y los éxitos de los algoritmos de aprendizaje por refuerzo de diferencia temporal [3]. Mientras que los agentes de aprendizaje por refuerzo han logrado dominios en los que las características útiles se pueden [6–8], su aplicabilidad ha sido previamente limitada a espacios de estado de baja dimensión completamente observados.

Aquí usamos avances recientes en el entrenamiento de redes neuronales profundas [9–11] para desarrollar un agente artificial novedoso, denominado red Q profunda, que puede aprender políticas exitosas directamente de entradas sensoriales de alta dimensión utilizando aprendizaje de refuerzo de extremo a extremo. Probamos este agente en el desafiante dominio de los juegos clásicos de Atari 2600. La estrategia de que el agente de la red Q profunda, al recibir solo los píxeles y la puntuación del juego como entradas, fue capaz de superar el rendimiento de todos los algoritmos anteriores y alcanzar un nivel comparable a ese. de un probador profesional de juegos humanos en un conjunto de 49 juegos, utilizando el mismo algoritmo, arquitectura de red e hiperparámetros. Este trabajo cierra la brecha entre las entradas y las acciones sensoriales de alta dimensión, lo que da como resultado el primer agente artificial que es capaz de aprender a sobresalir en una amplia gama de tareas desafiantes.

Nos propusimos crear un algoritmo único que pudiera desarrollar una amplia gama de competencias en una variedad de tareas desafiantes, un objetivo central de la inteligencia artificial general[13] que ha eludido los esfuerzos anteriores [8,14,15]. desarrolló un nuevo agente, una red deepQ (DQN), que es capaz de combinar el aprendizaje por refuerzo con una clase de red neuronal artificial[16] conocida como redes neuronales profundas. En particular, los avances recientes en las redes neuronales profundas en las que se utilizan varias capas de [9–11] para construir progresivamente más abstractas de los datos, han hecho posible que las redes neuronales artificiales aprendan conceptos como categorías de objetos directamente a partir de datos sensoriales sin procesar. Usamos una arquitectura particularmente exitosa, la red convolucional profunda[17] que utiliza capas jerárquicas de filtros convolucionales en mosaico para imitar los efectos de los campos receptivos, inspirados [12] en el trabajo seminal de Hubel y Wiesel sobre el procesamiento feedforward en la corteza visual temprana[18], explotando así las correlaciones espaciales locales presentes en las imágenes y construyendo robustez. a transformaciones naturales como cambios de punto de vista o de escala.

Consideramos tareas en las que el agente interactúa con un entorno a través de una secuencia de observaciones, acciones y recompensas. El objetivo de la agente es seleccionar acciones de una manera que maximice la recompensa futura acumulativa. Más formalmente, usamos una red neuronal convolucional profunda para aproximar la función de acción-valor óptima. rtzcrtz1zc2 rtz2z. . . jst~s, at~a, p ", Q!

$$\mathbb{E}\,\delta s,a\text{Þ}\sim\max_{\pi}$$

que es la suma máxima de recompensas descontada por c en cada paso de tiempo t, alcanzable por una política de comportamiento p 5 P(ajs), después de hacer una observación (s) y tomar una acción (a) (ver Métodos) [19].

Se sabe que el aprendizaje por refuerzo es inestable o incluso diverge cuando se utiliza un aproximador de función no lineal, como una red neuronal, para representar la función de valor de acción (también conocida como Q)[20]. Esta inestabilidad tiene varias causas: las correlaciones presentes en la secuencia de observaciones, el hecho de que pequeñas actualizaciones de Q pueden cambiar significativamente la política y, por lo tanto, cambiar la distribución de datos, y las correlaciones a0 ð Þ. entre los de acción (Q) y los valores objetivo rzc max Q s Abordamos estas inestabilidades [9] con $a0$ variante novedosa de Q-learning, que utiliza dos ideas clave. En primer lugar, utilizamos un mecanismo inspirado en la biología denominado repetición de la experiencia[21–23] que aleatoriza los datos, eliminando así las correlaciones en la secuencia de observación y suavizando los cambios en la distribución de los datos (ver más abajo para obtener más detalles). En segundo lugar, utilizamos una actualización iterativa que ajusta los valores de acción (Q) hacia los valores objetivo que solo se actualizan periódicamente, lo que reduce las correlaciones con el objetivo.

Si bien existen otros métodos estables para entrenar redes neuronales en el entorno de aprendizaje por refuerzo, como la iteración Q ajustada neuronal[24], estos métodos implican el entrenamiento repetido de redes de novo en cientos de iteraciones. En consecuencia, estos métodos, a diferencia de nuestro algoritmo, son demasiado ineficientes para ser utilizados con éxito con grandes redes neuronales. Parametrizamos una función de valor aproximado Q(s,a;hi) utilizando la red neuronal convolucional profunda que se muestra en la Fig. 1, en el que hi son los parámetros (es decir, los pesos) de la red Q en la iteración i. Para realizar la repetición de la experiencia, almacenamos las experiencias del agente et5 (st,at,rt,st 1 1) en cada paso de tiempo t en un conjunto de datos Dt 5 {e1,…,et}. Durante el aprendizaje, aplicamos actualizaciones de Q-learning en muestras (o minilotes) de experiencia (s,a,r,s9), U(D), extraídas uniformemente al azar del grupo de muestras almacenadas. La actualización de Q-learning en la iteración i utiliza la siguiente función de pérdida:

$$\text{Liðhi Þ}\sim \mathbb{E}_{s,a,r,s0}\,\delta\text{Þ}^{*}\text{UðDÞ}\,r\text{zc}\max_{a0}Q(s)0\,,\,a0\,;\,\text{ḥ}\{\,\}\{\text{Qðs,a; hiÞ}\quad\#\ \text{ÿ}2$$

donde c es el factor de descuento que determina el horizonte del agente, hi son los parámetros de la red Q en la iteración i y h{ son los parámetros de red objetivo en la iteración i. Los parámetros de red objetivo h{ solo se actualizan con los parámetros de red Q (hi) cada pasq, y [9] tienen fijos entre actualizaciones individuales (ver Métodos).

Para evaluar nuestro agente DQN, aprovechamos la plataforma Atari 2600, que ofrece una diversa gama de tareas (n 5 49) diseñadas para ser

[1]Google DeepMind, 5 New Street Square, Londres EC4A 3TW, Reino Unido.
*Estos autores contribuyeron igualmente a este trabajo.

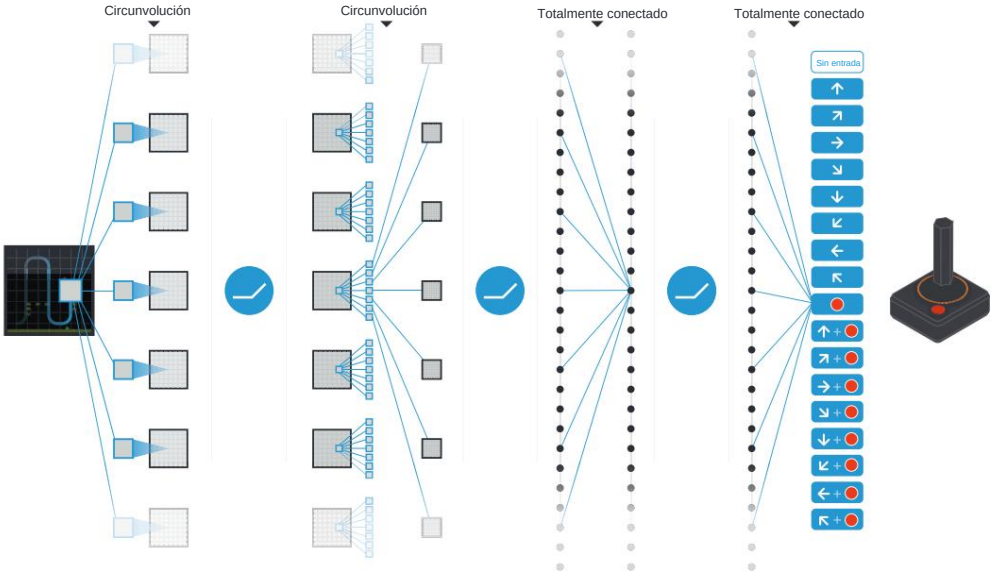


Figura 1 | Ilustración esquemática de la red neuronal convolucional. Los detalles de la arquitectura se explican en los Métodos. La entrada a la red neuronal consta de una imagen 84 3 84 3 4 producida por el mapa de preprocesamiento w, seguida de tres capas convolucionales (nota: línea azul serpenteante simboliza el deslizamiento de cada filtro a través de la imagen de entrada) y dos capas completamente conectadas con una sola salida para cada acción válida. Cada capa oculta va seguida de una no linealidad del rectificador (es decir, maxð0,xÞ).

difícil y atractivo para los jugadores humanos. Utilizamos la misma arquitectura de red, valores de hiperparámetros (consulte la Tabla 1 de datos ampliados) y el procedimiento de aprendizaje en todo momento (tomando datos de alta dimensión (video en color 210|160 a 60 Hz) como entrada) para demostrar que nuestro enfoque aprende políticas exitosas en una variedad de juegos basado únicamente en entradas sensoriales con un conocimiento previo muy mínimo (es decir, simplemente los datos de entrada eran imágenes visuales y la cantidad de acciones disponibles en cada juego, pero no sus correspondencias; ver Métodos). En particular, nuestro método fue capaz de entrenar grandes redes neuronales utilizando una señal de aprendizaje mental reforzado y un gradiente estocástico descendente de manera estable, ilustrado por la evolución temporal de dos índices de aprendizaje (la puntuación promedio por episodio del agente y los valores Q pronosticados promedio; consulte la Fig. .2 y Discusión Complementaria para más detalles).

Comparamos DQN con los métodos de mejor desempeño de la literatura de aprendizaje por refuerzo en los 49 juegos donde los resultados estaban disponibles12,15 . Además de los agentes aprendidos, también informamos puntajes para un probador de juegos humano profesional que juega bajo condiciones controladas y una política que selecciona acciones uniformemente en aleatorio (Tabla de datos extendida 2 y Fig. 3, indicada por 100% (humano) y 0% (aleatorio) en el eje y; ver Métodos). Nuestro método DQN supera a los mejores métodos de aprendizaje por refuerzo existentes en 43 de los juegos sin incorporar ninguno de los conocimientos previos adicionales sobre los juegos de Atari 2600 utilizados por otros enfoques (por ejemplo, refs 12, 15). Además, nuestro agente de DQN se desempeñó a un nivel comparable al de un probador de juegos humano profesional en el conjunto de 49 juegos, logrando más del 75% de la puntuación humana en más de la mitad de los juegos (29 juegos;

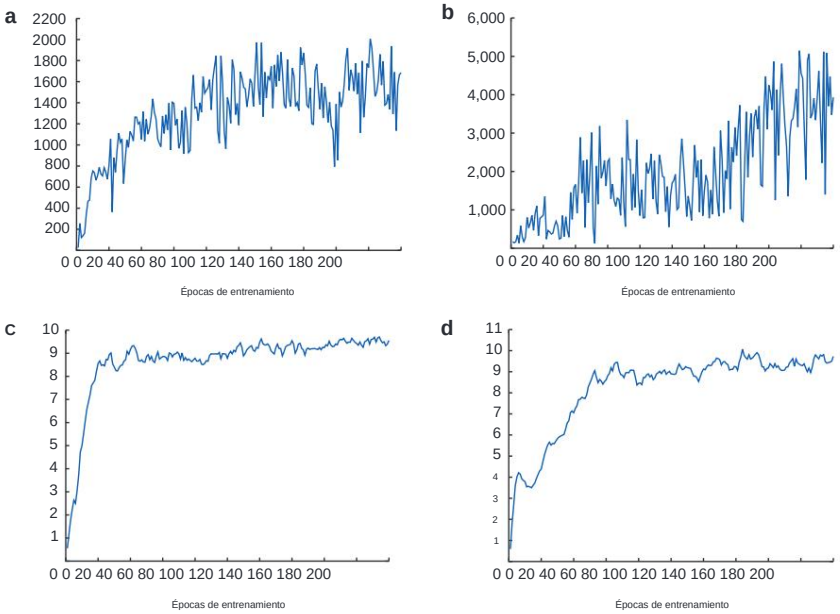


Figura 2 | Curvas de entrenamiento que rastrean el puntaje promedio del agente y el valor de acción pronosticado promedio. a, Cada punto es la puntuación promedio lograda por episodio después de que el agente se ejecuta con la política e-voraz (e 5 0.05) para 520 k cuadros en Space Invaders. b, Puntaje promedio logrado por episodio para Seaquest. c, Valor de acción pronosticado promedio en un conjunto de estados retenidos en Space Invaders. Cada punto en la curva está el promedio del valor de acción Q calculado sobre el conjunto de estados retenidos. Tenga en cuenta que los valores Q se escalan debido al recorte de recompensas (ver Métodos). d, valor de acción pronosticado promedio en Seaquest. Ver Discusión Complementaria para más detalles.

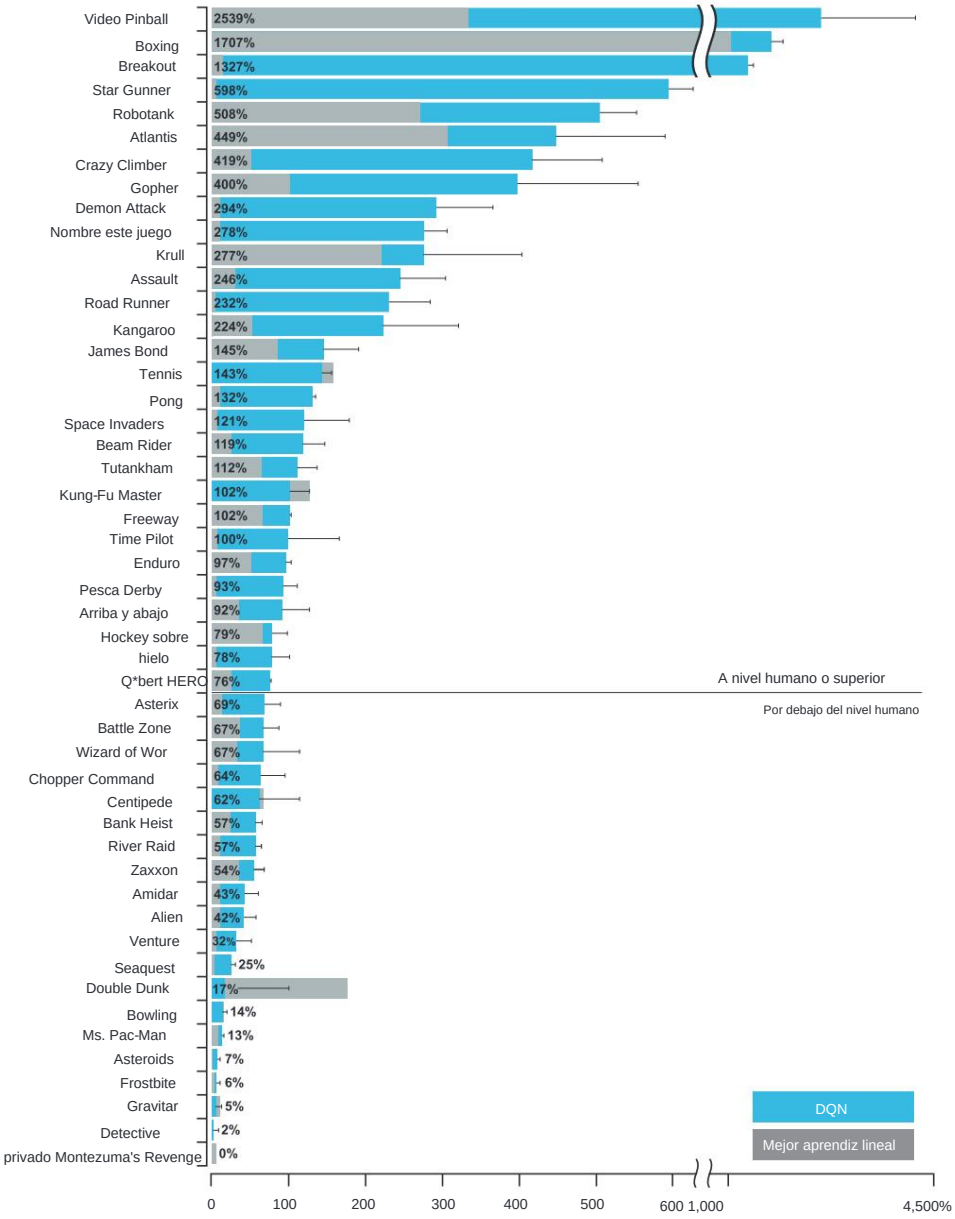


Figura 3 | Comparativa del agente DQN con el mejor refuerzo métodos de aprendizaje[15] en la literatura. El rendimiento de DQN está normalizado con respecto a un probador profesional de juegos humanos (es decir, 100 % de nivel) y juego aleatorio (es decir, 0 % de nivel). Tenga en cuenta que el rendimiento normalizado de DQN, expresado como porcentaje, se calcula como: 100 3 (puntuación DQN 2 puntuación de reproducción aleatoria)/(puntuación humana 2 puntuación de reproducción aleatoria). Se puede ver que DQN supera a los métodos de la competencia (consulte también la Tabla 2 de datos ampliados) en casi todos los juegos y se desempeña a un nivel que es ampliamente comparable o superior al de un evaluador de juegos humano profesional (es decir, operacionalizado como un nivel del 75 % o superior) en la mayoría de los juegos. La salida de audio se deshabilitó tanto para jugadores humanos como para agentes. Las barras de error indican sd en los 30 episodios de evaluación, comenzando con diferentes condiciones iniciales.

consulte la Fig. 3, Discusión complementaria y Tabla de datos ampliados 2). En simulaciones adicionales (ver Discusión complementaria y Tablas de datos extendidos 3 y 4), demostramos la importancia de los componentes centrales individuales del DQNagent (la memoria de reproducción, la red Q de destino separada y la arquitectura de red convolucional profunda) desactivándolos y demostrando los efectos perjudiciales sobre el rendimiento

A continuación, examinamos las representaciones aprendidas por DQN que respaldaron el desempeño exitoso del agente en el contexto del juego Space Invaders (consulte el Video complementario 1 para ver una demostración del desempeño de DQN), mediante el uso de una técnica desarrollada para la visualización de datos de alta dimensión. Llamado 't-SNE', el algoritmo t-SNE tiende a mapear la representación DQN de puntos de piedra de estado [25](Figura 4). Como se esperaba, Curiosamente, también encontramos instancias en las que el algoritmo t-SNE generó incrustaciones similares para representaciones DQN de estados que están cerca en términos de recompensa esperada pero perceptivamente diferentes (Fig. 4, abajo a la derecha, arriba a la izquierda y en el medio), consistentes con la noción de que la red es capaz de aprender representaciones que respaldan el comportamiento adaptativo a partir de entradas sensoriales de alta dimensión.

Además, también mostramos que las representaciones aprendidas por DQN pueden generalizarse a datos generados a partir de políticas distintas a las suyas: en simulaciones en las que presentamos como entrada a los estados del juego de red experimentados durante el juego humano y de agentes, registramos las representaciones de la última capa oculta , y visualizó las incrustaciones generadas por el algoritmo t-SNE (Datos extendidos Fig. 1 y Discusión complementaria). Datos extendidos La Fig. 2 proporciona una ilustración adicional de cómo las representaciones aprendidas por DQN le permiten predecir con precisión los valores de estado y acción.

Vale la pena señalar que los juegos en los que DQN sobresale son extremadamente variados en su naturaleza, desde juegos de disparos de desplazamiento lateral (River Raid) hasta juegos de boxeo (Boxing) y juegos de carreras de autos en tres dimensiones (Enduro).

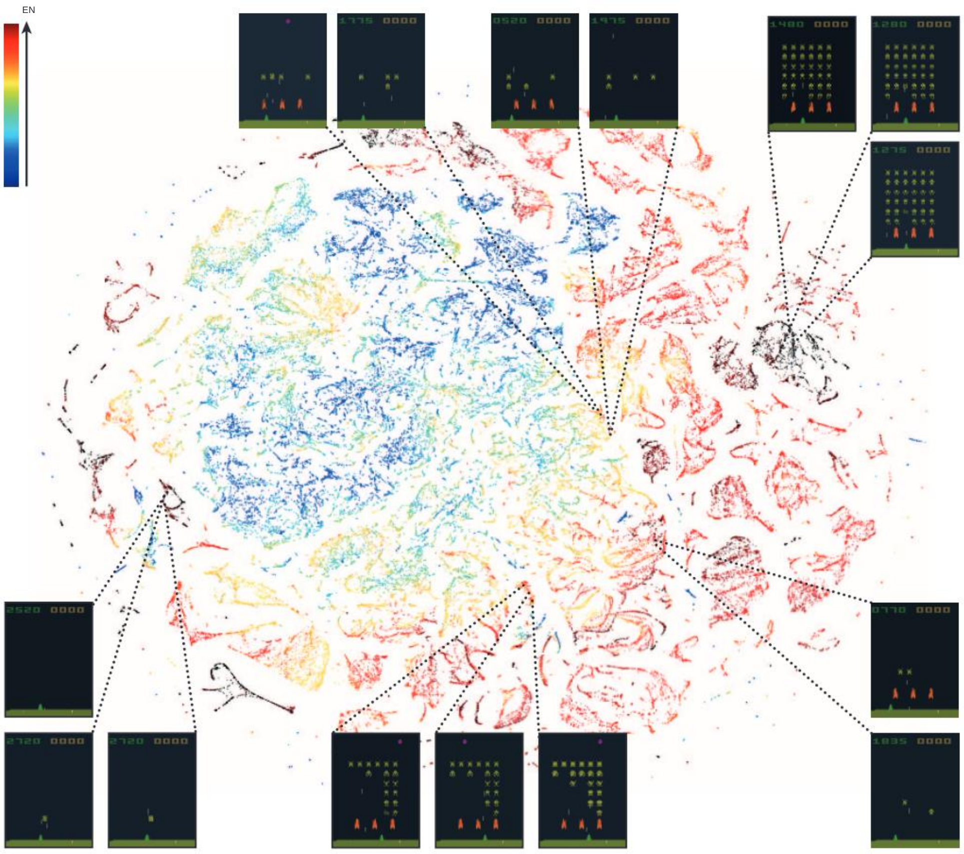


Figura 4 | Incrustación t-SNE bidimensional de las representaciones en la última capa oculta asignada por DQN a los estados del juego experimentados al jugar a Space Invaders. La trama se generó dejando que el agente de DQN jugara durante 2 h de tiempo real de juego y ejecutando el algoritmo t-SNE25 en las últimas representaciones de capa oculta asignadas por DQN a cada estado de juego experimentado. Los puntos están coloreados de acuerdo con los valores de estado (V, recompensa máxima esperada de un estado) predicho por DQN para los estados de juego correspondientes (que van desde el rojo oscuro (V más alto) hasta el azul oscuro (V más bajo)). Se muestran las capturas de pantalla correspondientes a un número seleccionado de puntos. El agente DQN predice valores de estado altos tanto para pantallas completas (capturas de pantalla en la parte superior derecha) como casi completas (capturas de pantalla en la parte inferior izquierda) porque ha aprendido que completar una pantalla conduce a una nueva pantalla llena de naves enemigas. A las pantallas parcialmente completadas (capturas de pantalla inferiores) se les asignan valores de estado más bajos porque hay una recompensa menos inmediata disponible. Las pantallas que se muestran abajo a la derecha, arriba a la izquierda y en el medio son menos similares a la percepción que los otros ejemplos, pero todavía están asignadas a representaciones cercanas y valores similares porque los bunkers naranjas no tienen gran importancia cerca del final de un nivel. Con permiso de Square Enix Limited.

De hecho, en ciertos juegos, DQN es capaz de descubrir una estrategia relativamente a largo plazo (por ejemplo, Breakout: el agente aprende la estrategia óptima, que consiste en cavar primero un túnel alrededor del costado de la pared que permite que la pelota se envíe por la espalda a destruir una gran cantidad de bloques; vea el Video complementario 2 para ver una ilustración del desarrollo del desempeño de DQN durante el transcurso del entrenamiento). .

En este trabajo, demostramos que una sola arquitectura puede aprender con éxito políticas de control en una variedad de entornos diferentes con un conocimiento previo mínimo, recibiendo solo los píxeles y la puntuación del juego como entradas, y usando el mismo algoritmo, arquitectura de red e hiperparámetros en cada juego, privado. solo a las entradas que tendría un jugador humano. En contraste con trabajos previos[24,26], nuestro enfoque incorpora aprendizaje por refuerzo 'de extremo a extremo' que utiliza la recompensa para dar forma continua a las representaciones dentro de la red convolucional hacia las características sobresalientes del entorno que facilitan la estimación del valor. Este principio se basa en evidencia neurobiológica que recompensa las señales durante la percepción. el aprendizaje puede influir en las características de las representaciones dentro de la corteza visual de los primates[27,28] . En particular, la integración exitosa del aprendizaje por refuerzo con las arquitecturas de redes profundas dependía de manera crítica de nuestra incorporación de un algoritmo de reproducción[21–23] que involucraba el almacenamiento y la representación de las transiciones experimentadas recientemente.

realización de un proceso de este tipo en el cerebro de los mamíferos, con la reactivación comprimida en el tiempo de las trayectorias experimentadas recientemente durante los períodos fuera de línea[21,22] (por ejemplo, descanso despierto) proporcionando un mecanismo putativo mediante el cual las funciones de valor pueden actualizarse de manera eficiente a través de las interacciones con los ganglios basales[22] .

En el futuro, será importante explorar el uso potencial de sesgar el contenido de la repetición de la experiencia hacia eventos destacados, un fenómeno que caracteriza la repetición hipocampal[29] observada empíricamente, y se relaciona con la noción de "barrido priorizado"[30] en el aprendizaje por refuerzo. En conjunto, nuestro trabajo ilustra el poder de aprovechar las técnicas de aprendizaje automático de última generación con mecanismos inspirados en la biología para crear agentes que sean capaces de aprender a dominar una amplia gama de tareas desafiantes.

Los métodos de contenido en línea, junto con cualquier elemento adicional de visualización de datos extendidos y datos de origen, están disponibles en la versión en línea del documento; las referencias exclusivas de estas secciones aparecen solo en el documento en línea.

Recibido el 10 de julio de 2014; aceptado el 16 de enero de 2015.

1. Sutton, R. & Barto, A. Aprendizaje por refuerzo: una introducción (MIT Press, 1998).
2. Thorndike, EL Animal Intelligence: Estudios experimentales (Macmillan, 1911).
3. Schultz, W., Dayan, P. & Montague, PR Un sustrato neuronal de predicción y premio. Ciencia 275, 1593–1599 (1997).
4. Serre, T., Wolf, L. & Poggio, T. Reconocimiento de objetos con características inspiradas en la corteza visual. proc. IEEE. computar Soc. Conf. computar Vis. Patrón. reconocer 994–1000 (2005).
5. Fukushima, K. Neocognitron: un modelo de red neuronal autoorganizado para un mecanismo de reconocimiento de patrones no afectado por el cambio de posición. Biol.Cybern.36, 193–202 (1980).

6. Tesauro, G. Aprendizaje de diferencias temporales y TD-Gammon. común ACM 38, 58–68 (1995).

7. Riedmiller, M., Gabel, T., Hafner, R. & Lange, S. Aprendizaje por refuerzo para robots fútbol. Auton. Robots 27, 55–73 (2009).
8. Diuk, C., Cohen, A. & Littman, ML Una representación orientada a objetos para un aprendizaje reforzado eficiente. proc. En t. Conf. Mach. Aprender. 240–247 (2008).
9. Bengio, Y. Aprendizaje de arquitecturas profundas para IA. Fundamentos y Tendencias en Máquina Aprendizaje 2, 1–127 (2009).
10. Krizhevsky, A., Sutskever, I. & Hinton, G. Clasificación de ImageNet con profundidad redes neuronales convolucionales.Adv.Neural Inf.Process.Syst.25, 1106–1114 (2012).
11. Hinton, GE & Salakhutdinov, RR Reducción de la dimensionalidad de los datos con redes neuronales. Ciencia 313, 504–507 (2006).
12. Bellemare, MG, Naddaf, Y., Veness, J. & Bowling, M. El aprendizaje arcade medio ambiente: Una plataforma de evaluación para agentes generales. J. Artif. Intel. Res. 47, 253–279 (2013).
13. Legg, S. & Hutter, M. Inteligencia universal: una definición de inteligencia artificial. Mentes Mach. 17, 391–444 (2007).
14. Genesereth, M., Love, N. & Pell, B. Juego general: descripción general de la AAAI competencia. IA Mag. 26, 62–72 (2005).
15. Bellemare, MG, Veness, J. & Bowling, M. Investigación de la conciencia de contingencia utilizando juegos Atari 2600. proc. Conf. AAAI. Artefacto Intel. 864–871 (2012).
16. McClelland, JL, Rumelhart, DE & Group, TPR Procesamiento distribuido en paralelo: exploraciones en la microestructura de la cognición (MIT Press, 1986).
17. LeCun, Y., Bottou, L., Bengio, Y. & Haffner, P. Aprendizaje basado en gradiente aplicado al reconocimiento de documentos. proc. IEEE86, 2278–2324 (1998).
18. Hubel, DH & Wiesel, TN Forma y disposición de las columnas en el estriado de gato corteza. J. Physiol. 165, 559–568 (1963).
19. Watkins, CJ y Dayan, P. Q-learning. Mach. Aprender. 8, 279–292 (1992).
20. Tsitsiklis, J. & Roy, BV Un análisis del aprendizaje de diferencias temporales con aproximación de funciones. IEEE Trans. Aparato mecánico. Contr. 42, 674–690 (1997).
21. McClelland, JL, McNaughton, BL y O'Reilly, RC Por qué existen sistemas de aprendizaje complementarios en el hipocampo y el neocórtex: percepciones de los éxitos y fracasos de los modelos conexionistas de aprendizaje y memoria. psicol. Rev. 102, 419–457 (1995).

22. O'Neill, J., Pleydell-Bouverie, B., Dupret, D. & Csicsvari, J. Tócala de nuevo: reactivación de la experiencia de vigilia y la memoria. Tendencias Neurosci. 33, 220–229 (2010).
23. Lin, L.-J. Aprendizaje por refuerzo para robots mediante redes neuronales. Informe Técnico, Documento DTIC (1993).
24. Riedmiller, M. Neural equipado Q iteración: primeras experiencias con un método de aprendizaje de refuerzo neuronal eficiente en datos. Mach. Aprenda.: ECML, 3720, 317–328 (Springer, 2005).
25. Van der Maaten, LJP & Hinton, GE Visualización de datos de alta dimensión usando t-SNE. J. Mach. Aprender. Res. 9, 2579–2605 (2008).
26. Lange, S. & Riedmiller, M. Redes neuronales de autocodificador profundo en el aprendizaje por refuerzo. proc. En t. Jt. Conf. Neural. Neto. 1–8 (2010).
27. Ley, C.-T. & Gold, JI El aprendizaje por refuerzo puede dar cuenta de asociaciones y aprendizaje perceptivo en una tarea de decisión visual. Naturaleza Neurosci. 12, 655 (2009).
28. Sigala, N. & Logothetis, NK Las formas de categorización visual presentan selectividad en el corteza temporal de los primates. Naturaleza 415, 318–320 (2002).
29. Bendor, D. & Wilson, MABiasing the content of hippocampalreplay during sleep. Naturaleza Neurosci. 15, 1439–1444 (2012).
30. Moore, A. & Atkeson, C. Barrido priorizado: aprendizaje por refuerzo con menos datos y menos tiempo real. Mach. Aprender. 13, 103–130 (1993).

La información complementaria está disponible en la versión en línea del documento.

Agradecimientos Agradecemos a G. Hinton, P. Dayan y M. Bowling por las discusiones, a A. Cain y J. Keene por trabajar en las imágenes, a K. Keller y P. Rogers por su ayuda con las imágenes, a G. Wayne por sus comentarios sobre un versión anterior del manuscrito, y al resto del equipo de DeepMind por su apoyo, ideas y aliento.

Autor Contribuciones VM, KK, DS, JV, MGB, MR, AG, DW, SL y DH conceptualizaron el problema y el marco técnico. VM, KK, AAR y DS desarrollaron y probaron los algoritmos. JV, SP, CB, AAR, MGB, IA, AKF, GO y AS crearon la plataforma de prueba. KK, HK, SL y DH gestionaron el proyecto. KK, DK, DH, VM, DS, AG, AAR, JV y MGB escribieron el artículo.

Información del autor La información sobre reimpresiones y permisos está disponible en www.nature.com/reprints. Los autores declaran no tener intereses financieros en competencia. Los lectores pueden comentar sobre la versión en línea del documento. La correspondencia y las solicitudes de materiales deben dirigirse a KK (korayk@google.com) o DH (demishassabis@google.com).

MÉTODOS

Preprocesamiento. Trabajar directamente con marcos crudos de Atari 2600, que son imágenes de 2103 160 píxeles con una paleta de 128 colores, puede ser exigente en términos de requisitos de cómputo y memoria. emulador Primero, para codificar un solo cuadro, tomamos el valor máximo para cada valor de color de píxel sobre el cuadro que se está codificando y el cuadro anterior. Esto fue necesario para eliminar el parpadeo que está presente en los juegos donde algunos objetos aparecen solo en cuadros pares mientras que otros objetos aparecen solo en cuadros impares, un artefacto causado por la cantidad limitada de sprites que Atari 2600 puede mostrar a la vez. En segundo lugar, extraemos el canal Y, también conocido como luminancia, del cuadro RGB y lo cambiamos de escala a 84 3 84. La función w del algoritmo 1 que se describe a continuación aplica este preprocesamiento a los m cuadros más recientes y los apila para producir la entrada a la función Q, en la que m 5 4, aunque el algoritmo es robusto a diferentes valores de m (por ejemplo, 3 o 5).

Disponibilidad de código. Se puede acceder al código fuente en https://sites.google.com/a/deepmind.com/dqn solo para usos no comerciales.

Modelo de arquitectura. Hay varias formas posibles de parametrizar Q utilizando una red neuronal. Debido a que Q asigna pares de historia-acción a estimaciones escalares de su valor Q, la historia y la acción se han utilizado como entradas para la red neuronal. El principal inconveniente de este tipo de arquitectura por algunos enfoques anteriores es [24,26] que se requiere un paso adelante separado para calcular el valor Q de cada acción, lo que da como resultado un costo que escala linealmente con el número de acciones. En su lugar, usamos una arquitectura en la que hay una unidad de salida separada para cada acción posible, y solo la representación del estado es una entrada a la red neuronal. Las salidas corresponden a los valores Q predichos de las acciones individuales para el estado de entrada. La principal ventaja de este tipo de arquitectura es la capacidad de calcular los valores Q para todas las acciones posibles en un estado determinado con un solo paso directo a través de la red.

La arquitectura exacta, que se muestra esquemáticamente en la Fig. 1, es la siguiente. La entrada a la red neuronal consiste en una imagen 843 843 4 producida por el mapa de preprocesamiento w. La primera capa oculta convoluciona 32 filtros de 8 3 8 con paso 4 con la imagen de entrada y aplica una no linealidad rectificadora31,32 . La segunda capa oculta gira 64 filtros de 4 3 4 con zancada 2, seguido nuevamente por una no linealidad rectificadora.

A esto le sigue una tercera capa convolucional que convoluciona 64 filtros de 3 3 3 con paso 1 seguido de un rectificador. La última capa oculta está totalmente conectada y consta de 512 unidades rectificadoras. La capa de salida es una capa lineal completamente conectada con una sola salida para cada acción válida. El número de acciones válidas varió entre 4 y 18 en los juegos considerados.

Detalles del entrenamiento. Realizamos experimentos en 49 juegos de Atari 2600 donde los resultados estaban disponibles para todos los demás métodos comparables 12,15 . Se entrenó una red diferente en cada juego: se usó la misma arquitectura de red, algoritmo de aprendizaje y configuración de hiperparámetros (consulte la Tabla de datos extendida 1) en todos los juegos, lo que muestra que nuestro enfoque es lo suficientemente sólido para trabajar en una variedad de juegos mientras incorpora solo un conocimiento previo mínimo (ver más abajo). Si bien evaluamos a nuestros agentes en juegos no modificados, hicimos un cambio en la estructura de recompensas de los juegos solo durante el entrenamiento. Como la escala de puntuaciones varía mucho de un juego a otro, recortamos todas las recompensas positivas en 1 y todas las recompensas negativas en 21, dejando 0 recompensas sin cambios.

Recortar las recompensas de esta manera limita la escala de los errores derivados y facilita el uso de la misma tasa de aprendizaje en varios juegos. Al mismo tiempo, podría afectar el rendimiento de nuestro agente, ya que no puede diferenciar entre recompensas de diferente magnitud. Para los juegos en los que hay un contador de vidas, el emulador Atari 2600 también envía el número de vidas que quedan en el juego, que luego se usa para marcar el final de un episodio durante el entrenamiento.

En estos experimentos, usamos el algoritmo RMSProp (ver http://www.cs.toronto.edu/ ,tijmen/csc321/slides/lecture_slides_lec6.pdf) con minilotes de tamaño 32. La política de comportamiento durante el entrenamiento fue e-greedy con e recocido linealmente de 1,0 a 0,1 durante el primer millón de fotogramas y fijado en 0,1 a partir de entonces. Entrenamos para un total de 50 millones de fotogramas (es decir, alrededor de 38 días de experiencia de juego en total) y utilizamos una memoria de reproducción de 1 millón de fotogramas más recientes.

Siguiendo los enfoques anteriores para jugar juegos de Atari 2600, también utilizamos una técnica simple de salto de fotogramas15 . Más precisamente, el agente ve y selecciona acciones en cada cuadro k-ésimo en lugar de en cada cuadro, y su última acción se repite en los cuadros omitidos. Debido a que ejecutar el emulador un paso hacia adelante requiere mucho menos cálculo que hacer que los agentes seleccionen una acción, esta técnica permite que el agente juegue aproximadamente k veces más juegos sin aumentar significativamente el tiempo de ejecución.

Usamos k 5 4 para todos los juegos.

Los valores de todos los hiperparámetros y parámetros de optimización se seleccionaron realizando una búsqueda informal en los juegos Pong, Breakout, Seaquest, Space Invaders y Beam Rider. No realizamos una búsqueda de cuadrícula sistemática debido al alto costo computacional. Luego, estos parámetros se mantuvieron fijos en todos los demás juegos. Los valores y las descripciones de todos los hiperparámetros se proporcionan en Extendido.

Tabla de datos 1.

Nuestra configuración experimental consiste en usar el siguiente conocimiento previo mínimo: que los datos de entrada consistieron en imágenes visuales (lo que motivó nuestro uso de una red profunda convolucional), la puntuación específica del juego (sin modificación), el número de acciones, aunque no sus correspondencias (por ejemplo, especificación del 'botón arriba') y el conteo de vida.

Procedimiento de evaluación. Los agentes capacitados se evaluaron jugando cada juego 30 veces durante un máximo de 5 minutos cada vez con diferentes condiciones aleatorias iniciales ('no op'; consulte la Tabla 1 de datos ampliados) y una política de codicia electrónica con e 5 0,05. Este procedimiento se adopta para minimizar la posibilidad de sobreajuste durante la evaluación. El agente aleatorio sirvió como comparación de referencia y eligió una acción aleatoria a 10 Hz, que es cada sexto cuadro, repitiendo su última acción en los cuadros intermedios. 10 Hz es aproximadamente lo más rápido que un jugador humano puede seleccionar el botón 'disparar', y configurar el agente aleatorio en esta frecuencia evita puntajes de referencia falsos en un puñado de juegos. También evaluamos el rendimiento de un agente aleatorio que seleccionó una acción a 60 Hz (es decir, cada cuadro). Esto tuvo un efecto mínimo: cambió el rendimiento normalizado de DQN en más del 5 % en solo seis juegos (Boxing, Breakout, Crazy Climber, Demon Attack, Krull y Robotank), y en todos estos juegos DQN superó al experto humano por una diferencia considerable. margen.

El probador humano profesional usó el mismo motor emulador que los agentes y jugó bajo condiciones controladas. El probador humano no podía pausar, guardar o recargar juegos. Como en el entorno Atari 2600 original, el emulador se ejecutaba a 60 Hz y la salida de audio estaba desactivada: como tal, la entrada sensorial se equiparaba entre el jugador humano y los agentes. El rendimiento humano es la recompensa media que se consigue a partir de unos 20 episodios de cada juego con una duración máxima de 5 min cada uno, tras unas 2 h de práctica jugando cada juego.

Algoritmo. Consideramos tareas en las que un agente interactúa con un entorno, en este caso el emulador de Atari, en una secuencia de acciones, observaciones y recompensas.

En cada paso de tiempo, el agente selecciona una acción en del conjunto de acciones legales del juego, A~f1, . . . ,Kg. La acción se pasa al emulador y modifica su estado interno y la puntuación del juego. En general, el entorno puede ser estocástico. El agente no observa el estado interno del emulador; en su lugar, el agente observa una imagen xt[Rd del emulador, que es un vector de valores de píxeles que representan la pantalla actual. Además, recibe una recompensa rt que representa el cambio en la puntuación del juego.

Tenga en cuenta que, en general, la puntuación del juego puede depender de toda la secuencia previa de acciones y observaciones; la retroalimentación sobre una acción solo puede recibirse después de que hayan transcurrido muchos miles de pasos de tiempo.

Debido a que el agente solo observa la pantalla actual, la tarea se observa parcialmente33 y muchos estados del emulador tienen un alias perceptivo (es decir, es imposible comprender completamente la situación actual solo desde la pantalla actual xt). Por lo tanto, las secuencias de acciones y observaciones, st~x1,a1,x2,:::,at{1,xt, se ingresan al algoritmo, que luego aprende estrategias de juego dependiendo de estas secuencias. Se supone que todas las secuencias en el emulador terminan en un número finito de pasos de tiempo. Este formalismo da lugar a un proceso de decisión de Markov (MDP) grande pero finito en el que cada secuencia es un estado distinto. Como resultado, podemos aplicar métodos estándar de aprendizaje por refuerzo para MDP, simplemente usando la secuencia completa st como la representación de estado en el tiempo t.

El objetivo del agente es interactuar con el emulador seleccionando acciones que maximicen las recompensas futuras. Hacemos la suposición estándar de que las recompensas futuras se descuentan por un factor de c por paso de tiempo (c se estableció en 0.99 en todo momento), y definimos el rendimiento descontado futuro en tiempo t como Rt~X ct paso de tiempo en el que termina el juego. Definimos la función T 0 {t donde T es el rt0 , acción-valor óptima Q!ðs,aÞ t0~t como el rendimiento máximo esperado que se puede lograr siguiendo cualquier política, después de ver algunas secuencias y luego tomar alguna acción a, Q! ðs,aÞ~ maxp ½RtDst~s,at~a,pÿ en la que p es una política que asigna secuencias a acciones (o distribuciones sobre acciones).

𝔼

La función acción-valor óptima obedece a una identidad importante conocida como la ecuación de Bellman. Esto se basa en la siguiente intuición: si el valor óptimo Q! s 0 ,a0 ð Þ de la secuencia s9 en el siguiente paso de tiempo era conocido para todas las acciones posibles a9, entonces la estrategia óptima es seleccionar la acción a9 maximizando el valor esperado de rzcQ! s 0 ,a0 ð Þ:

$$Q!ðs,aÞ \sim 𝔼 rzc \max_{a0} ð Q! s 0 ,a0 Þ s,a$$

La idea básica detrás de muchos algoritmos de aprendizaje por refuerzo es estimar la función acción-valor mediante el uso de la ecuación de Bellman como una actualización iterativa, ½ ð Qiz1ðs,aÞ~ s0 rzc 𝔼 max a0 Qi s 0 ,a0 ð Þ s,a ÿ. Tales algoritmos de iteración de valor convergen a la función acción-valor óptima, Qi?Q! como i?∞. En la práctica, este enfoque básico no es práctico, porque la función acción-valor se estima por separado para cada secuencia, sin ninguna generalización. En cambio, es común usar un aproximador de función para estimar la función acción-valor, Qðs,a; hÞ<Q! ðs,aÞ. En la comunidad de aprendizaje por refuerzo, este suele ser un aproximador de función lineal, pero

a veces, en su lugar, se utiliza un aproximador de función no lineal, como una red neuronal. Nos referimos a un aproximador de función de red neuronal con pesos h como una red Q. La red AQ se puede entrenar ajustando los parámetros en iteración i para reducir el error cuadrático medio en la ecuación de Bellman, donde los valores objetivo óptimos rzc maxa0 Q ! s 0 ,a0 ð Þ se sustituyen por valores objetivo aproximados ÿ ÿ , usando parámetros Q(s0,a0;h−i) de alguna iteración anterior. Esto conduce a una secuencia de funciones de

i i

LiðhiÞ~ $\mathbb{E}_{s,a,r}$! Es ð 0 ½yDs,aÿ{Qðs,a; holaÞÞ ^{2} "

~ $\mathbb{E}$! s,a,r,s0 ðy{Qðs,a; holaÞÞ ^{2} " zEs,a,r Vs ½ 0 ½yÿÿ:

Tenga en cuenta que los objetivos dependen de los pesos de la red; esto contrasta con los objetivos utilizados para el aprendizaje supervisado, que se fijan antes de que comience el aprendizaje. En cada etapa de la optimización, mantenemos fijos los parámetros de la iteración anterior hi 2 al optimizar la i-ésima función de pérdida Li(hi), lo que da como resultado una secuencia de problemas de optimización bien definidos. El término final es la varianza de los objetivos, que no depende de los parámetros que estamos optimizando actualmente y, por lo tanto, puede ignorarse. Derivando la función de pérdida con respecto a los pesos llegamos al siguiente gradiente:

+hi LðhiÞ ~ $\mathbb{E}$h{ s,a,r,s0 rzc máxQs 0 ,a0 ; ÿ ÿ {Qðs,a; hola ÿ ÿ a0 ÿ +hiQðs,a; hola ÿ :

En lugar de calcular las expectativas completas en el gradiente anterior, a menudo es conveniente desde el punto de vista computacional optimizar la función de pérdida mediante el descenso del gradiente estocástico. El conocido algoritmo Q-learning[19] se puede recuperar en este marco actualizando los pesos después de cada paso de tiempo, reemplazando las expectativas usando muestras individuales y configurando h{ ~hi{1.

Tenga en cuenta que este algoritmo no tiene modelo: resuelve la tarea de aprendizaje por refuerzo directamente usando muestras del emulador, sin estimar explícitamente la recompensa y la dinámica de transición P r,s0 Ds,a . También está fuera de política: aprende sobre la política codiciosa a~argmaxa0Q(s,a0; h), siguiendo una distribución de comportamiento que asegura una exploración adecuada del espacio de estados. En la práctica, la distribución del comportamiento a menudo se selecciona mediante una política codiciosa que sigue la política codiciosa con probabilidad 1 2 e y selecciona una acción aleatoria con probabilidad e.

Algoritmo de entrenamiento para redes Q profundas. El algoritmo completo para entrenar redes Q profundas se presenta en el Algoritmo 1. El agente selecciona y ejecuta acciones de acuerdo con una política de avidez electrónica basada en Q. Debido a que usar historiales de longitud arbitraria como entradas a una red neuronal puede ser difícil, nuestra función Q en cambio, trabaja en una representación de longitud fija de las historias producidas por la función w descrita anteriormente. El algoritmo modifica el Q-learning en línea estándar de dos maneras para que sea adecuado para entrenar grandes redes neuronales sin divergir.

Primero, usamos una técnica conocida como repetición de experiencia[23] en la que almacenamos las experiencias del agente en cada paso de tiempo, et5 (st, at,rt,st 1 1), en un conjunto de datos Dt5 {e1,…,et}, agrupados sobre muchos episodios (donde el final de un episodio ocurre cuando se alcanza un estado terminal) en una memoria de repetición. Durante el ciclo interno del algoritmo, aplicamos actualizaciones de Q-learning, o actualizaciones de minilotes, a muestras de experiencia, (s, a,r,s9), U(D), extraídas al azar del grupo de muestras almacenadas. Este enfoque tiene varias ventajas sobre el Q-learning en línea estándar. Primero, cada paso de la experiencia se usa potencialmente en muchas actualizaciones de peso, lo que permite una mayor eficiencia de los datos. Segundo, aprender directamente de muestras consecutivas es ineficiente, debido a las fuertes correlaciones entre las muestras; la aleatorización de las muestras rompe estas correlaciones y, por lo tanto, reduce la varianza de las actualizaciones. En tercer lugar, al aprender sobre políticas, los parámetros actuales determinan la siguiente muestra de datos en la que se entrenan los parámetros. Por ejemplo, si la acción de maximización es moverse hacia la izquierda, las muestras de entrenamiento estarán dominadas por muestras del lado izquierdo; si la acción de maximización luego cambia a la derecha, entonces la distribución de entrenamiento también cambiará. Es fácil ver cómo pueden surgir bucles de retroalimentación no deseados y los parámetros pueden estancarse en un mínimo local deficiente, o incluso divergir catastróficamente[20] .

La distribución del comportamiento se promedia sobre muchos de sus estados anteriores, suavizando el aprendizaje y evitando oscilaciones o divergencias en los parámetros. Tenga en cuenta que cuando se aprende mediante la repetición de la experiencia, es necesario aprender fuera de la política (porque nuestros parámetros actuales son diferentes a los utilizados para generar la muestra), lo que motiva la elección de Q-learning.

En la práctica, nuestro algoritmo solo almacena las últimas N tuplas de experiencia en la memoria de reproducción y toma muestras de forma aleatoria de D al realizar actualizaciones. al tamaño finito de la memoria N. De manera similar, el muestreo uniforme otorga la misma importancia a todas las transiciones en la memoria de reproducción. Una estrategia de muestreo más sofisticada podría enfatizar las transiciones de las que podemos aprender más, similar al barrido priorizado[30] .

La segunda modificación de Q-learning en línea destinada a mejorar aún más la estabilidad de nuestro método con redes neuronales es usar una red separada para generar los objetivos yj en la actualización de Q-learning. Más precisamente, cada actualización de C clonamos la red Q para obtener una red objetivo Q^ y usamos Q^ para generar los objetivos de Q-learning yj para las siguientes actualizaciones de C a Q. Esta modificación hace que el algoritmo sea más estable en comparación con el Q-learning en línea estándar, donde una actualización que aumenta Q(st,at) a menudo también aumenta Q(st 1 1,a) para todo a y, por lo tanto, también aumenta el objetivo yj, lo que posiblemente provoque oscilaciones o divergencias en la política. La generación de objetivos utilizando un conjunto de parámetros más antiguo agrega un retraso entre el momento en que se realiza una actualización de Q y el momento en que la actualización afecta a los objetivos yj, lo que hace que la divergencia o las oscilaciones sean mucho más improbables.

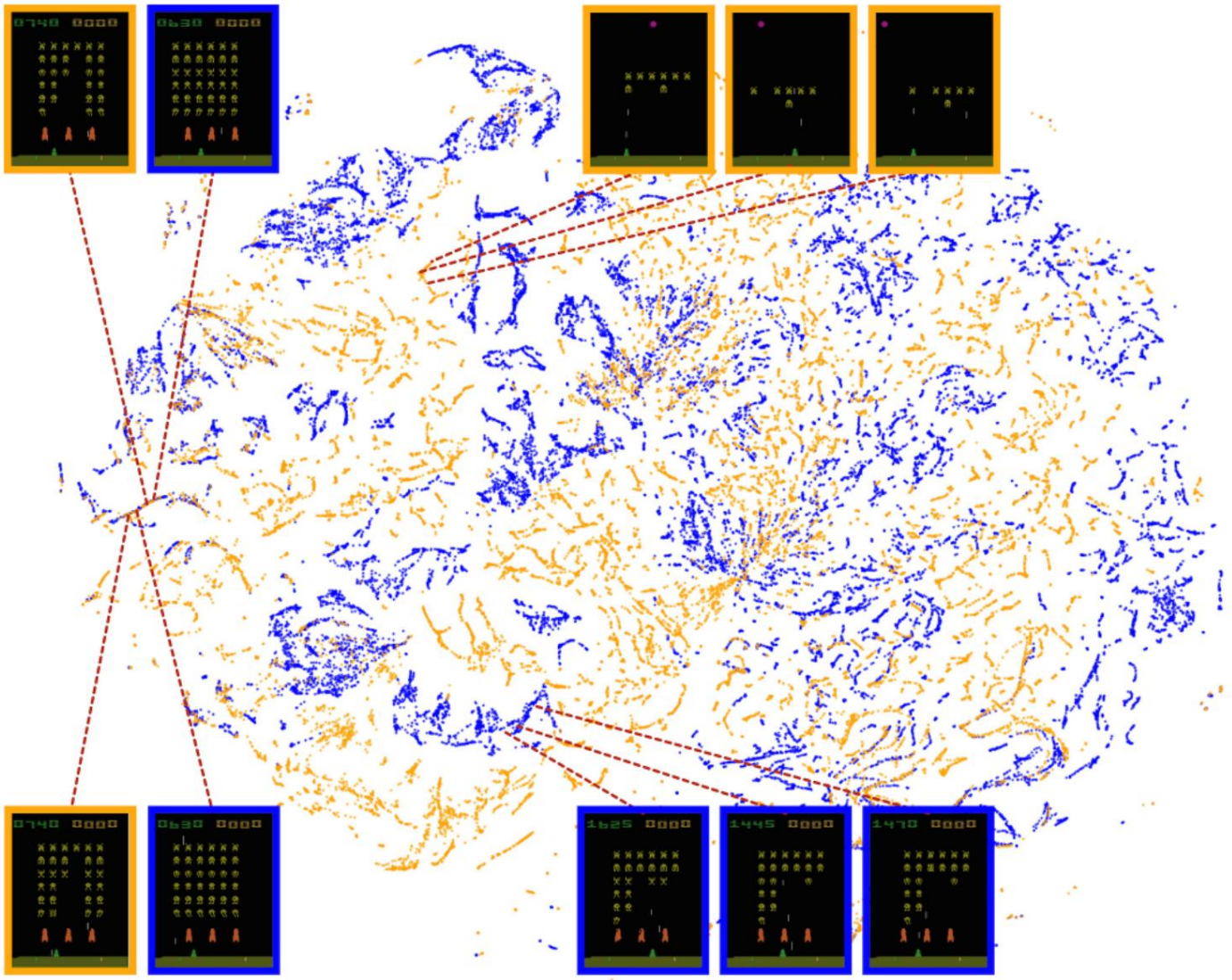
También nos resultó útil recortar el término de error de la actualización rzc maxa0 Q ÿ ÿ {Qðs,a; hiÞ s0 ,a0 ; h{ debe estar entre 21 y 1. Debido a que la función de pérdida de valor absoluto jxj tiene una derivada de 21 para todos los valores negativos de x y una derivada de 1 para todos los valores positivos de x, recortar el error cuadrático para que esté entre 21 y 1 corresponde a usar una función de pérdida de valor absoluto para errores fuera del intervalo (21,1). Esta forma de recorte de errores mejoró aún más la estabilidad del algoritmo.

Algoritmo 1: Q-learning profundo con repetición de experiencia.
Inicialice la memoria de reproducción D a capacidad
N Inicialice la función de valor de acción Q con ponderaciones aleatorias
h Inicialice la función de valor de acción objetivo Q^ con ponderaciones h2 5
h Para el episodio 5 1, M do
Inicialice la secuencia s1~fx1g y la secuencia preprocesada w1~wðs1Þ
Para t 5 1,T hacer
Con probabilidad e seleccione una acción aleatoria en
caso contrario seleccione at~argmaxaQðwðstÞ,a; hÞ
Ejecute la acción en el emulador y observe la recompensa rt y la imagen xt 1 1
Establecer stz1~st,at,xtz1 y preprocesar wtz1~wðstz1Þ
Almacene la transición wt,at,rt,wtz1 ÿ ÿ en D
Ejemplo de minilote aleatorio de transiciones wj,aj,rj,wjz1 ÿ ÿ de D
Establecer yj~ rj si el episodio termina en el paso jz1
(ÿ rjzc maxa0 Q^ wjz1,a0 ; h{ ÿ de lo contrario
Realice un paso de descenso de gradiente en yj{Q wj,aj; h ÿ ÿÿ2 con respecto a la parámetros de red h
Cada paso de C reinicia Q^~Q
Terminar por
Terminar por

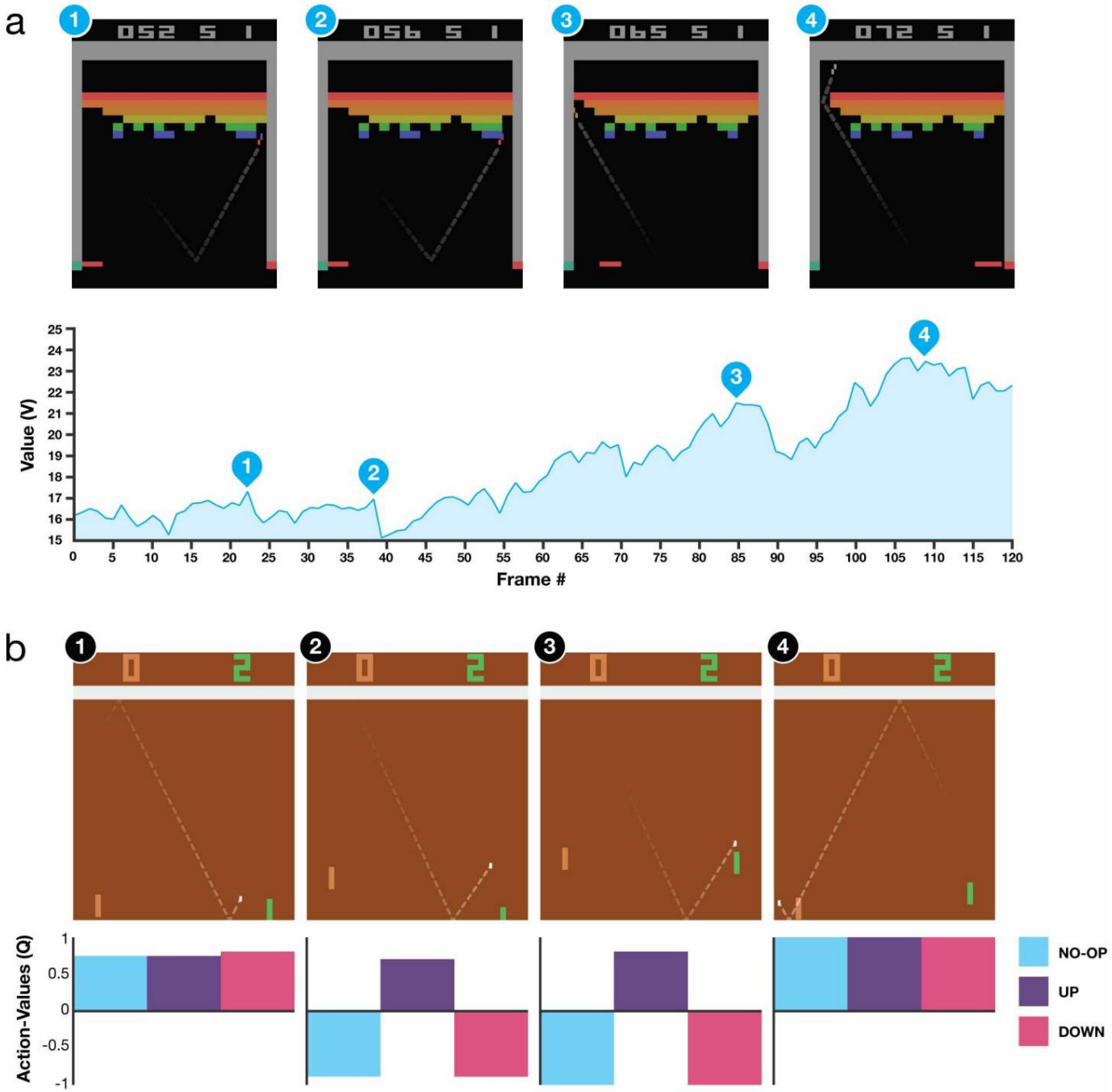
31. Jarrett,K.,Kavukcuoglu,K.,Ranzato, MA& LeCun, Y.¿Cuál es la mejor arquitectura multietapa para el reconocimiento de objetos? Proc.IEEE. Int.Conf.Comput.Vis.2146–2153 (2009).

32. Nair, V. & Hinton, GE Las unidades lineales rectificadas mejoran las máquinas Boltzmann restringidas. proc. En t. Conf. Mach. Aprender. 807–814 (2010).

33. Kaelbling, LP, Littman, ML y Cassandra, AR Planificación y actuación en dominios estocásticos parcialmente observables. Inteligencia artificial 101, 99–134 (1994).



Datos extendidos Figura 1 | Incorporación bidimensional de t-SNE de las representaciones en la última capa oculta asignada por DQN a los estados del juego experimentados durante una combinación de juego humano y de agente en Space Invaders. La trama se generó ejecutando el algoritmo t-SNE25 en la última representación de capa oculta asignada por DQN a los estados del juego experimentados durante una combinación de juego humano (30 min) y agente (2 h). El hecho de que haya una estructura similar en las incrustaciones bidimensionales correspondientes a la representación DQN de estados experimentados durante el juego humano (naranja puntos) y el juego DQN (puntos azules) sugiere que las representaciones aprendidas por DQN sí se generalizan a los datos generados a partir de políticas distintas a las suyas. La presencia en la incrustación de t-SNE de grupos superpuestos de puntos correspondientes a la representación de la red de estados experimentados durante el juego humano y del agente muestra que el agente DQN también sigue secuencias de estados similares a las que se encuentran en el juego humano. Se muestran capturas de pantalla correspondientes a los estados seleccionados (humano: borde naranja; DQN: borde azul).



Datos extendidos Figura 2 | Visualización de funciones de valor aprendido en dos juegos, Breakout y Pong. a, Una visualización de la función de valor aprendido en el juego Breakout. En los puntos de tiempo 1 y 2, se prevé que el valor de estado sea ,17 y el agente está limpiando los ladrillos en el nivel más bajo. Cada uno de los picos en la curva de función de valor corresponde a una recompensa obtenida al despejar un ladrillo.

En el punto de tiempo 3, el agente está a punto de atravesar el nivel superior de ladrillos y el valor aumenta a ,21 en previsión de romper y limpiar un gran conjunto de ladrillos. En el punto 4, el valor está por encima de 23 y el agente se ha abierto paso. Pasado este punto, la pelota rebotará en la parte superior de los ladrillos, despejando muchos de ellos por sí sola. b, Una visualización de la función de valor de acción aprendida en el juego Pong. En el punto de tiempo 1, la pelota se mueve hacia la paleta controlada por el agente en el lado derecho de la pantalla y los valores de todas las acciones están alrededor de 0,7, lo que refleja el valor esperado de este estado basado en la experiencia previa. En el punto de tiempo 2, el agente comienza a mover la paleta hacia la pelota y el valor de la acción "arriba" se mantiene alto mientras que el valor de la acción "abajo" cae a 20,9. Esto refleja el hecho de que presionar 'abajo' haría que el agente perdiera la pelota e incurriera en una recompensa de 21. En el punto de tiempo 3, el agente golpea la pelota presionando 'arriba' y la recompensa esperada sigue aumentando hasta el punto de tiempo 4, cuando la pelota llega a la izquierda. borde de la pantalla y el valor de todas las acciones refleja que el agente está a punto de recibir una recompensa de 1. Tenga en cuenta que la línea discontinua muestra la trayectoria pasada de la pelota únicamente con fines ilustrativos (es decir, no se muestra durante el juego). Con permiso de Atari Interactive, Inc.

Tabla de datos extendida 1 | Lista de hiperparámetros y sus valores

Hyperparameter	Value	Description
minibatch size	32	Number of training cases over which each stochastic gradient descent (SGD) update is computed.
replay memory size	1000000	SGD updates are sampled from this number of most recent frames.
agent history length	4	The number of most recent frames experienced by the agent that are given as input to the Q network.
target network update frequency	10000	The frequency (measured in the number of parameter updates) with which the target network is updated (this corresponds to the parameter *C* from Algorithm 1).
discount factor	0.99	Discount factor gamma used in the Q-learning update.
action repeat	4	Repeat each action selected by the agent this many times. Using a value of 4 results in the agent seeing only every 4th input frame.
update frequency	4	The number of actions selected by the agent between successive SGD updates. Using a value of 4 results in the agent selecting 4 actions between each pair of successive updates.
learning rate	0.00025	The learning rate used by RMSProp.
gradient momentum	0.95	Gradient momentum used by RMSProp.
squared gradient momentum	0.95	Squared gradient (denominator) momentum used by RMSProp.
min squared gradient	0.01	Constant added to the squared gradient in the denominator of the RMSProp update.
initial exploration	1	Initial value of ε in ε-greedy exploration.
final exploration	0.1	Final value of ε in ε-greedy exploration.
final exploration frame	1000000	The number of frames over which the initial value of ε is linearly annealed to its final value.
replay start size	50000	A uniform random policy is run for this number of frames before learning starts and the resulting experience is used to populate the replay memory.
no-op max	30	Maximum number of “do nothing” actions to be performed by the agent at the start of an episode.

Los valores de todos los hiperparámetros se seleccionaron realizando una búsqueda informal en los juegos Pong, Breakout, Seaquest, Space Invaders y Beam Rider. No realizamos una búsqueda de cuadrícula sistemática debido al alto costo computacional, aunque es concebible que se puedan obtener resultados aún mejores ajustando sistemáticamente los valores de los hiperparámetros.

Tabla de datos extendida 2 | Comparación de puntajes de juegos obtenidos por agentes DQN con métodos de la literatura12,15 y un probador de juegos humano profesional

Game	Random Play	Best Linear Learner	Contingency (SARSA)	Human	DQN (± std)	Normalized DQN (% Human)
Alien	227.8	939.2	103.2	6875	3069 (±1093)	42.7%
Amidar	5.8	103.4	183.6	1676	739.5 (±3024)	43.9%
Assault	222.4	628	537	1496	3359(±775)	246.2%
Asterix	210	987.3	1332	8503	6012 (±1744)	70.0%
Asteroids	719.1	907.3	89	13157	1629 (±542)	7.3%
Atlantis	12850	62687	852.9	29028	85641(±17600)	449.9%
Bank Heist	14.2	190.8	67.4	734.4	429.7 (±650)	57.7%
Battle Zone	2360	15820	16.2	37800	26300 (±7725)	67.6%
Beam Rider	363.9	929.4	1743	5775	6846 (±1619)	119.8%
Bowling	23.1	43.9	36.4	154.8	42.4 (±88)	14.7%
Boxing	0.1	44	9.8	4.3	71.8 (±8.4)	1707.9%
Breakout	1.7	5.2	6.1	31.8	401.2 (±26.9)	1327.2%
Centipede	2091	8803	4647	11963	8309(±5237)	63.0%
Chopper Command	811	1582	16.9	9882	6687 (±2916)	64.8%
Crazy Climber	10781	23411	149.8	35411	114103 (±22797)	419.5%
Demon Attack	152.1	520.5	0	3401	9711 (±2406)	294.2%
Double Dunk	-18.6	-13.1	-16	-15.5	-18.1 (±2.6)	17.1%
Enduro	0	129.1	159.4	309.6	301.8 (±24.6)	97.5%
Fishing Derby	-91.7	-89.5	-85.1	5.5	-0.8 (±19.0)	93.5%
Freeway	0	19.1	19.7	29.6	30.3 (±0.7)	102.4%
Frostbite	65.2	216.9	180.9	4335	328.3 (±250.5)	6.2%
Gopher	257.6	1288	2368	2321	8520 (±3279)	400.4%
Gravitar	173	387.7	429	2672	306.7 (±223.9)	5.3%
H.E.R.O.	1027	6459	7295	25763	19950 (±158)	76.5%
Ice Hockey	-11.2	-9.5	-3.2	0.9	-1.6 (±2.5)	79.3%
James Bond	29	202.8	354.1	406.7	576.7 (±175.5)	145.0%
Kangaroo	52	1622	8.8	3035	6740 (±2959)	224.2%
Krull	1598	3372	3341	2395	3805 (±1033)	277.0%
Kung-Fu Master	258.5	19544	29151	22736	23270 (±5955)	102.4%
Montezuma's Revenge	0	10.7	259	4367	0 (±0)	0.0%
Ms. Pacman	307.3	1692	1227	15693	2311(±525)	13.0%
Name This Game	2292	2500	2247	4076	7257 (±547)	278.3%
Pong	-20.7	-19	-17.4	9.3	18.9 (±1.3)	132.0%
Private Eye	24.9	684.3	86	69571	1788 (±5473)	2.5%
Q*Bert	163.9	613.5	960.3	13455	10596 (±3294)	78.5%
River Raid	1339	1904	2650	13513	8316 (±1049)	57.3%
Road Runner	11.5	67.7	89.1	7845	18257 (±4268)	232.9%
Robotank	2.2	28.7	12.4	11.9	51.6 (±4.7)	509.0%
Seaquest	68.4	664.8	675.5	20182	5286(±1310)	25.9%
Space Invaders	148	250.1	267.9	1652	1976 (±893)	121.5%
Star Gunner	664	1070	9.4	10250	57997 (±3152)	598.1%
Tennis	-23.8	-0.1	0	-8.9	-2.5 (±1.9)	143.2%
Time Pilot	3568	3741	24.9	5925	5947 (±1600)	100.9%
Tutankham	11.4	114.3	98.2	167.6	186.7 (±41.9)	112.2%
Up and Down	533.4	3533	2449	9082	8456 (±3162)	92.7%
Venture	0	66	0.6	1188	380.0 (±238.6)	32.0%
Video Pinball	16257	16871	19761	17298	42684 (±16287)	2539.4%
Wizard of Wor	563.5	1981	36.9	4757	3393 (±2019)	67.5%
Zaxxon	32.5	3365	21.4	9173	4977 (±1235)	54.1%

Best Linear Learner es el mejor resultado obtenido por un aproximador de funciones lineales en diferentes tipos de características diseñadas a mano12 . Las cifras de agentes de contingencia (SARSA) son los resultados obtenidos en la ref. 15. Tenga en cuenta que las cifras de la última columna indican el rendimiento de DQN en relación con el probador de juegos humanos, expresado como un porcentaje, es decir, 100 3 (puntaje DQN 2 puntaje de juego aleatorio)/(puntaje humano 2 puntaje de juego aleatorio).

Tabla de datos extendida 3 | Los efectos de la reproducción y la separación de la red Q de destino

Game	With replay, with target Q	With replay, without target Q	Without replay, with target Q	Without replay, without target Q
Breakout	316.8	240.7	10.2	3.2
Enduro	1006.3	831.4	141.9	29.1
River Raid	7446.6	4102.8	2867.7	1453.0
Seaquest	2894.4	822.6	1003.0	275.8
Space Invaders	1088.9	826.3	373.2	302.0

Se capacitó a los agentes de DQN para 10 millones de fotogramas utilizando hiperparámetros estándar para todas las combinaciones posibles de activación o desactivación de la reproducción, uso o no uso de una red Q de destino separada y tres tasas de aprendizaje diferentes. Cada agente se evaluó cada 250 000 cuadros de entrenamiento para 135 000 cuadros de validación y se informa el puntaje promedio más alto del episodio. Tenga en cuenta que estos episodios de evaluación no se truncaron a los 5 minutos, lo que generó puntajes más altos en Enduro que los informados en la Tabla 2 de datos ampliados. Tenga en cuenta también que el número de cuadros de entrenamiento fue más corto (10 millones de cuadros) en comparación con los resultados principales presentados en Tabla de datos extendida 2 (50 millones de fotogramas).

Tabla de datos extendida 4 | Comparación del rendimiento de DQN con el aproximador de función lineal

Game	DQN	Linear
Breakout	316.8	3.00
Enduro	1006.3	62.0
River Raid	7446.6	2346.9
Seaquest	2894.4	656.9
Space Invaders	1088.9	301.3

El rendimiento del agente DQN se compara con el rendimiento de un aproximador de función lineal en los 5 juegos de validación (es decir, donde se usó una sola capa lineal en lugar de la red convolucional, en combinación con reproducción y red objetivo separada). Se capacitó a los agentes para 10 millones de fotogramas utilizando hiperparámetros estándar y tres tasas de aprendizaje diferentes. Cada agente se evaluó cada 250 000 cuadros de entrenamiento para 135 000 cuadros de validación y se informa el puntaje promedio más alto del episodio. Tenga en cuenta que estos episodios de evaluación no se truncaron a los 5 minutos, lo que generó puntajes más altos en Enduro que los informados en la Tabla 2 de datos ampliados. Tenga en cuenta también que el número de cuadros de entrenamiento fue más corto (10 millones de cuadros) en comparación con los resultados principales presentados en Tabla de datos extendida 2 (50 millones de cuadros).