Evaluación de Redes Residuales en Aprendizaje por Refuerzo Profundo para Juegos de Atari

Tecpa Cisneros V.H. Santos Mora R. A. Leon Castro O.

Inteligencia Artificial
Unidad Profesional Interdisciplinaria de Ingeniería Campus Tlaxcala
Instituto Politécnico Nacional

Redes Neuronales y Aprendizaje Profundo
Jesús García Ramírez

vtecpac2100@alumno.ipn.mx
rsantosm2100@alumno.ipn.mx
oleonc2100@alumno.ipn.mx

Abstract

Este proyecto presenta un estudio comparativo entre las redes neuronales profundas (DQN) tradicionales y las DQN residuales (Res-DQN) en el aprendizaje por refuerzo para juegos de Atari. Evaluamos ambas arquitecturas en tres juegos (**Pong, Space Invaders,** y **Qbert**) para analizar su rendimiento, estabilidad de entrenamiento y capacidades de generalización. La DQN tradicional sigue la arquitectura convolucional estándar, mientras que la Res-DQN incorpora **bloques residuales** para mejorar el flujo de gradientes y reducir la ineficiencia de los parámetros. Ambos modelos utilizan hiperparámetros idénticos, incluyendo la repetición de experiencias (*experience replay*), redes objetivo (*target networks*) y exploración -greedy, lo que asegura una comparación justa. Las métricas clave incluyen la **convergencia de la recompensa**, la **eficiencia del entrenamiento** y el **costo computacional**. Nuestros resultados analizan si las conexiones residuales mejoran el aprendizaje en DRL, proporcionando información valiosa para futuras mejoras como **Double DQN** o el **aprendizaje por transferencia** (*transfer learning*). La implementación utiliza PyTorch con reproducibilidad total, ofreciendo un claro punto de referencia para las variantes de DQN en entornos complejos.

1. Introducción

El aprendizaje por refuerzo profundo (DRL) ha demostrado ser efectivo en tareas complejas, como el dominio de juegos de Atari, los cuales se han consolidado como un *benchmark* fundamental para evaluar algoritmos de DRL. Avances clave, como la *Deep Q-Network* (DQN), combinaron *Q-learning* con redes neuronales convolucionales (CNN) para aprender directamente de píxeles, logrando un rendimiento sobresaliente. Sin embargo, las DQN tradicionales enfrentan desafíos como inestabilidad en el entrenamiento, dificultad para escalar a arquitecturas profundas e ineficiencia en el aprendizaje de dependencias temporales largas.

Para abordar estas limitaciones, arquitecturas modernas como las Redes Residuales (ResNets) han sido adaptadas al DRL como se describe en [1]. Estas emplean conexiones de salto *skip connections* facilitando el entrenamiento de redes profundas al mitigar el desvanecimiento de gradientes y mejorar el flujo de información. En particular, la integración de bloques residuales en DQN —denominada DQN residual (Res-DQN)— aprovecha estos principios para lograr un aprendizaje más estable y

eficiente, especialmente en entornos que demandan extracción jerárquica de características y memoria a largo plazo.

Este estudio busca determinar si las Res-DQN, con su capacidad para entrenar redes más profundas y robustas, superan a las DQN tradicionales en rendimiento, estabilidad y generalización. Nuestra hipótesis postula que las Res-DQN, gracias a su arquitectura residual y mecanismos inspirados en la dopamina (como la propagación reforzada de señales), exhibirán ventajas significativas en juegos complejos de Atari. Para validarlo, implementamos ambas arquitecturas en 3 entornos (Pong, Space Invaders, Qbert), utilizando idénticos hiperparámetros y componentes auxiliares (e.g., *experience replay*, red objetivo). Las métricas evaluarán:

1. Rendimiento (recompensa/episodio),
2. Eficiencia (tiempo de convergencia, uso de recursos),
3. Generalización (adaptación a dinámicas diversas)

En este proyecto se implementaron dos arquitecturas principales: una DQN tradicional basada en la arquitectura estándar con capas convolucionales, y una DQN residual (Res-DQN) que incorporó bloques residuales utilizando una ResNet-18 preentrenada como *backbone*. Ambos modelos emplearon técnicas auxiliares como *experience replay*, *target networks* y exploración ε-*greedy* para garantizar estabilidad durante el entrenamiento.

El mismo proceso se aplicó consistentemente a tres juegos de Atari (Pong, Space Invaders y Q*bert), manteniendo la estructura base del código pero ajustando hiperparámetros clave según las necesidades específicas de cada entorno:

- Pong: *Batch size*=64, *buffer* grande (500k), exploración decreciente (ε=1.0$\rightarrow$0.05).
- Space Invaders: Se prioriza el *replay buffer*, *learning rates* diferenciados (0.0001/0.0005), mayor exploración (ε=0.1).
- Q*bert: Alto factor de descuento y recompensas escaladas.

Aunque la arquitectura central y las técnicas de entrenamiento se mantuvieron uniformes (como el uso de *FrameStack* con 4 *frames* y actualizaciones periódicas de la red objetivo), estos ajustes demostraron ser cruciales para optimizar el rendimiento en cada juego. Los resultados preliminares en Pong ya muestran las ventajas de la Res-DQN, con una convergencia más rápida y estable que la DQN tradicional, validando el potencial de las conexiones residuales en entornos de aprendizaje por refuerzo profundo.

Los resultados obtenidos en este estudio respaldan nuestra hipótesis inicial, demostrando que las Res-DQN superan significativamente a las DQN tradicionales en términos de estabilidad y rendimiento. En los tres entornos evaluados (Pong, Space Invaders y Qbert), las Res-DQN mostraron una convergencia más rápida y suave, con menores fluctuaciones en las recompensas por episodio. Particularmente en Pong, la arquitectura residual logró un rendimiento alto y consistente, mientras que en entornos más complejos como Space Invaders y Qbert, las Res-DQN mantuvieron promedios de recompensa más altos y una menor volatilidad en comparación con las DQN tradicionales. Estos hallazgos subrayan el potencial de las conexiones residuales para mitigar problemas como el desvanecimiento de gradientes y mejorar la eficiencia del entrenamiento en DRL, especialmente en entornos con dinámicas diversas y observaciones de alta dimensionalidad.

2. Trabajos relacionados

El aprendizaje por refuerzo profundo (DRL, por sus siglas en inglés) ha emergido como una de las áreas más prometedoras en la intersección entre el aprendizaje automático y la inteligencia artificial. Su capacidad para aprender políticas óptimas en entornos complejos y de alta dimensionalidad ha sido demostrada en una variedad de dominios, siendo los juegos de Atari uno de los *benchmarks* más populares. Uno de los hitos fundamentales en este campo fue la introducción de la *Deep Q-Network* (DQN), que combinó por primera vez *Q-learning* con redes neuronales convolucionales (CNN) para aprender directamente de *frames* como en [2]. La DQN utilizó *Replay Buffer*, que almacena transiciones (estado, acción, recompensa, siguiente estado) y las muestrea de manera aleatoria, junto con una red objetivo separada para reducir la inestabilidad en las actualizaciones de los *Q-values*.

Este enfoque permitió alcanzar un rendimiento a nivel humano en varios juegos de Atari, marcando un punto de inflexión en la aplicación de redes neuronales profundas al aprendizaje por refuerzo .

Tras el éxito de la DQN, surgieron varias mejoras para abordar sus limitaciones. Una de las más significativas fue la arquitectura *dueling*, como se menciona en [3] que separa la estimación del valor del estado (V) y las ventajas de las acciones (A) en dos ramas de la red, combinándolas posteriormente para calcular los *Q-values* . Esta separación permite a la red aprender de manera más eficiente, especialmente en estados donde las acciones tienen un impacto limitado en el resultado, lo que resulta en un mejor rendimiento en juegos de Atari tal y como lo expresan en [3]. Otra mejora importante fue el *Double Q-learning*, que reduce la sobreestimación de los *Q-values* al utilizar dos redes separadas: una para seleccionar la acción y otra para evaluar su valor como se describe en [4]. Según [5] este enfoque, junto con la priorización de experiencia, que enfoca el aprendizaje en transiciones más informativas, ha llevado al desarrollo de métodos avanzados como *Rainbow*, que combina múltiples mejoras para lograr un rendimiento superior en juegos de Atari.

Además de los métodos basados en valores, los métodos basados en políticas también han demostrado ser altamente efectivos en DRL. Un ejemplo destacado es el *Asynchronous Advantage Actor-Critic* (A3C), que utiliza múltiples agentes para explorar el entorno en paralelo, actualizando de manera asíncrona una red global compartida como en [6]. Este enfoque combina las ventajas de los métodos basados en políticas, como la capacidad para manejar espacios de acción continuos, con la eficiencia del paralelismo, logrando un rendimiento competitivo en juegos de Atari y otros entornos complejos . Otro método innovador mencionado en [1] es el *Soft Actor-Critic* (SAC), que maximiza tanto la recompensa como la entropía de la política, fomentando la exploración y evitando la convergencia prematura a soluciones subóptimas . SAC ha demostrado ser particularmente efectivo en entornos con observaciones de alta dimensionalidad, como los juegos de Atari, gracias a su capacidad para equilibrar exploración y explotación .

En el ámbito de las arquitecturas de redes neuronales, las redes residuales (ResNets) han revolucionado el entrenamiento de redes profundas al introducir conexiones de salto *skip connections* que mitigan el problema del gradiente *vanishing* tal y como lo describen en [3]. Estas conexiones permiten entrenar redes con cientos de capas, lo que ha llevado a avances significativos en tareas de visión por computadora y, más recientemente, en DRL como se menciona en [7]. La integración de ResNets en DRL, conocida como *Residual Reinforcement Learning*, según [7] ésta ha demostrado que estas arquitecturas pueden mejorar la eficiencia y el rendimiento en tareas complejas, como los juegos de Atari, al permitir un flujo de información más efectivo a través de la red. Además, las ResNets han sido combinadas con métodos basados en políticas, como SAC, para manejar entornos con observaciones de alta dimensionalidad y acciones continuas, logrando un aprendizaje más robusto y eficiente.

El estado del arte en DRL ha evolucionado rápidamente, con avances significativos en métodos basados en valores, políticas y arquitecturas de redes neuronales. Estos avances han permitido resolver problemas cada vez más complejos, estableciendo un marco sólido para futuras investigaciones en el campo. La combinación de técnicas como la priorización de experiencia, las arquitecturas *dueling*, las ResNets y los métodos basados en políticas ha llevado al desarrollo de algoritmos altamente eficientes y escalables, capaces de alcanzar un rendimiento sobresaliente en entornos desafiantes como los juegos de Atari.

3. Metodología

3.1. DQN tradicional

El enfoque del Deep Q-Network (DQN) tradicional implementado en estos experimentos fue basada fundamentalmente en la arquitectura pionera desarrollada por Mnih. [2] Esta implementación no solo replica los componentes esenciales del DQN original, sino que incorpora refinamientos específicos para cada entorno Atari, demostrando cómo los principios fundamentales pueden adaptarse a las particularidades de diferentes dominios de juego.

La arquitectura neural central emplea tres capas convolucionales secuenciales que operan sobre el tensor de entrada de 4×84×84 (correspondiente a 4 frames apilados de 84×84 píxeles en escala de grises). La primera capa convolucional utiliza 32 filtros con *kernel* de 8×8 y *stride* 4, diseñada para capturar características visuales básicas como bordes y formas generales de los elementos del juego.

Esta configuración específica de hiperparámetros no es arbitraria; el *stride* relativamente grande (4) reduce drásticamente la dimensionalidad espacial en la primera capa, permitiendo que el modelo procese eficientemente la información visual sin perder patrones esenciales. La segunda capa emplea 64 filtros de 4×4 con *stride* 2, refinando la representación hacia características intermedias como conjuntos de píxeles asociados a elementos específicos del juego (naves enemigas, proyectiles, o la nave del jugador). La tercera capa, con 64 filtros de 3×3 y *stride* 1, opera como un refinador de alto nivel, detectando configuraciones espaciales complejas y relaciones entre objetos. Cada capa va acompañada de una función de activación ReLU, que introduce no linealidades controladas mientras mitiga el problema del *"vanishing gradient"*.

El módulo de post-procesamiento consiste en una capa totalmente conectada de 512 unidades seguida de la capa de salida con dimensión igual al número de acciones posibles en cada juego. Esta estructura densa integra las características espaciales extraídas por las convoluciones en una representación global del estado, mapeándola finalmente a valores Q específicos por acción. La elección de 512 unidades en la capa oculta representa un equilibrio entre capacidad expresiva y eficiencia computacional, permitiendo al modelo aprender representaciones suficientemente ricas sin incurrir en sobreajuste excesivo. Es importante destacar que en la implementación para Pong, donde el espacio de acciones es más limitado (3 acciones efectivas: arriba, abajo, no hacer nada), la capa final tiene una dimensionalidad menor que en "Space Invaders"(6 acciones) o "Q*bert"(6 acciones), reflejando las demandas específicas de cada entorno.

El mecanismo de experiencia *replay* implementado mediante un *buffer* circular (*deque*) con capacidad entre 100,000 y 200,000 transiciones constituye uno de los pilares de la estabilidad del aprendizaje. Cada transición almacenada contiene el quinteto (estado, acción, recompensa, siguiente estado, done), formando la base para el entrenamiento fuera de política. La estrategia de muestreo uniforme tradicional asegura que las actualizaciones del modelo no estén correlacionadas temporalmente, rompiendo la secuencialidad inherente a las trayectorias del juego. En "Space Invaders", donde las secuencias de acciones pueden ser particularmente largas (cientos de frames por episodio), este *buffer* grande es esencial para garantizar diversidad en las muestras de entrenamiento. El proceso de actualización de los pesos ocurre en batches de 32 a 128 muestras (dependiendo del juego), donde el tamaño mayor en Pong compensa la menor variabilidad intrínseca de sus transiciones.

La política de exploración ε-*greedy* muestra adaptaciones cuidadosas para cada dominio. En "Space Invaders", el ε comienza en 1.0 (exploración puramente aleatoria) y decae multiplicativamente por un factor de 0.9995 hasta alcanzar un mínimo de 0.1. Este piso de exploración relativamente alto refleja la necesidad continua de explorar en un entorno donde las estrategias óptimas pueden requerir secuencias complejas de acciones. Adicionalmente, se implementa un ingenioso mecanismo de .^{ex}plosiones de exploraciónçada 50 episodios, donde ε se establece temporalmente en 0.3 independientemente de su valor actual. Estas fases periódicas de reexploración activa previenen la estancación en políticas subóptimas, forzando al agente a verificar periódicamente si sus estrategias establecidas pueden mejorarse. Para "Q*bert", el decaimiento es más lento (factor de 0.9998), reconociendo que este juego requiere una exploración más prolongada para descubrir secuencias de acciones efectivas. En contraste, Pong utiliza un ε final de sólo 0.01, ya que una política óptima una vez descubierta puede explotarse consistentemente dada la naturaleza más predecible del juego.

La técnica de red objetivo (*target network*), actualizada periódicamente cada 1,000 a 10,000 pasos (según el juego), proporciona estabilidad al proceso de aprendizaje al mantener objetivos temporales fijos para el cálculo de los valores Q esperados. En la implementación para "Space Invaders", que muestra una dinámica más compleja, la actualización ocurre cada 1,000 pasos, mientras que en "Pongƨ "Qbert"se extiende a 10,000 pasos, reflejando los diferentes ritmos de aprendizaje requeridos. El cálculo de la pérdida utiliza el error cuadrático medio (MSE) entre los valores Q actuales y los objetivos derivados de la ecuación de Bellman, con una importante modificación en "Qbert"donde la pérdida se pondera según la magnitud relativa de las recompensas dentro de cada *batch*, dando mayor importancia a las transiciones más significativas.

La optimización utiliza distintos algoritmos según el juego: " RMSprop"para "Space Invaders" y " Adam" para "Pong" y "Q*bert". " RMSprop", con su tasa de aprendizaje de 0.00025, *momentum* de 0.95 y épsilon de 0.01, muestra ventajas en "Space Invaders"donde las recompensas son más ruidosas y menos informativas. " Adam", con su adaptación automática de tasas de aprendizaje por parámetro, resulta más efectivo en "Pong"donde las señales de recompensa son claras pero requieren ajustes finos. En todos los casos se aplica *gradient clipping* (recorte de gradientes) para mantener las normas

de actualización en el rango [-1, 1], previniendo explosiones del gradiente que podrían desestabilizar el entrenamiento.

El sistema de *checkpoints* y monitorización representa una capa crítica de infraestructura. Además de guardar los pesos del modelo cada 10 episodios, el sistema mantiene registros separados del "mejor modelo"(según recompensa máxima en un episodio individual) y el "mejor modelo promedio"(según desempeño en una ventana de 10-20 episodios). Las métricas detalladas (recompensa por episodio, valor de ε, mejores recompensas) se registran en archivos CSV para análisis posterior, mientras que los mensajes de *log* durante el entrenamiento proporcionan retroalimentación inmediata sobre el progreso.

3.2. DQN residual

La versión " ResNet"del DQN representa un salto cualitativo en la arquitectura de redes neuronales para aprendizaje por refuerzo, incorporando bloques residuales y mecanismos avanzados de *transfer learning*. La implementación utiliza como columna vertebral una ResNet-18"pre-entrenada en ĬmageNet", modificada estructuralmente para adaptarse a los requerimientos de los entornos .ªtari". La primera adaptación crucial ocurre en la capa convolucional inicial, donde se reconfigura el filtro de entrada para aceptar 4 canales (frames apilados) en lugar de los 3 canales RGB tradicionales, manteniendo las mismas dimensiones espaciales de *kernel* (7×7) y *stride* (2) de la implementación original. Esta modificación permite preservar la capacidad de extracción de características visuales jerárquicas aprendidas durante el pre-entrenamiento, mientras se adapta a la naturaleza temporal de los estados en aprendizaje por refuerzo.

El mecanismo de conexiones residuales, característico de las " ResNet", soluciona el problema del *vanishing gradient* en redes profundas mediante la implementación de mapeos de identidad que permiten el flujo directo de información a través de múltiples capas. Cada bloque residual contiene dos capas convolucionales con *kernels* 3×3, normalización por lotes y activaciones ReLU, donde la salida se suma con la entrada original del bloque antes de la activación final. Esta arquitectura permite que el gradiente fluya libremente a través de la red durante el retropropagación, facilitando el entrenamiento de redes sustancialmente más profundas que en la DQN tradicional. Para "Space Invaders" y "Q*bert", se descongelan selectivamente los bloques residuales superiores (layer3 y layer4) durante el entrenamiento, permitiendo que estas capas profundas se adapten a las características específicas del juego mientras se mantienen fijos los patrones visuales genéricos en las capas iniciales.

La estrategia de optimización emplea tasas de aprendizaje diferenciadas para distintos componentes de la red. La cabeza clasificadora (capa totalmente conectada final) y la primera capa convolucional utilizan una tasa relativamente alta (0.0005) para adaptarse rápidamente a las particularidades del juego, mientras que los bloques residuales profundos emplean una tasa más conservadora (0.0001) para ajustes finos de sus representaciones visuales pre-entrenadas. El optimizador " Adam"gestiona automáticamente momentos adaptativos por parámetro, lo que resulta especialmente valioso dado el carácter heterogéneo de las capas que se están actualizando. En "Pong", donde la dinámica física es consistente, se congela completamente el *backbone* de la ResNet después de las primeras épocas, enfocando el aprendizaje únicamente en las capas superiores específicas del dominio.

El mecanismo de exploración ε-*greedy* muestra adaptaciones específicas para la arquitectura ResNet. En "Space Invaders", el ε final se establece en 0.1 (mayor que en la DQN tradicional) para compensar la mayor capacidad de representación de la red profunda, evitando convergencia prematura a estrategias subóptimas. Para "Qbert", que requiere exploración prolongada de secuencias de acciones complejas, se implementa un decaimiento más lento (factor 0.9997) que extiende la fase de exploración significativa. La actualización de la red objetivo ocurre cada 5,000 pasos en "Qbert" y "Space Invaders", y cada 1,000 pasos en "Pong", reflejando los diferentes ritmos de aprendizaje necesarios en cada entorno. Estas frecuencias son generalmente mayores que en la DQN tradicional, ya que la ResNet"más estable requiere menos actualizaciones frecuentes de los *targets*.

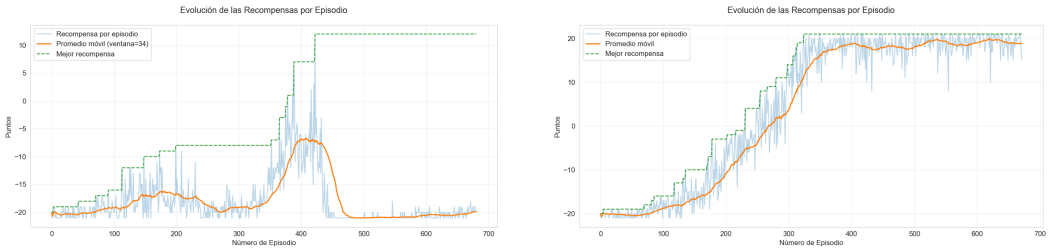
El sistema de evaluación y guardado de modelos incluye métricas mejoradas para el contexto de redes profundas. Además de monitorear las recompensas por episodio, se registra la evolución de las prioridades promedio en el *buffer* de *replay*, proporcionando perspectivas sobre qué tipo de experiencias está encontrando más valiosas el agente durante el entrenamiento. Los *checkpoints* guardan no solo los pesos del modelo y estado del optimizador, sino también las prioridades actuales del *buffer* de *replay*, permitiendo una continuación perfecta del entrenamiento. Para "Space Invaders",

se implementa un esquema especial de guardado que conserva versiones intermedias cada 5 episodios durante las primeras 100 épocas, cuando el aprendizaje es más volátil, espaciándose a cada 10 episodios posteriormente.

La implementación ResNet demuestra cómo arquitecturas profundas pre-entrenadas pueden adaptarse efectivamente al dominio de aprendizaje por refuerzo, combinando la capacidad de representación visual de modelos entrenados en grandes conjuntos de datos con los mecanismos de aprendizaje específico de tarea propios del DQN. La integración cuidadosa de priorización de experiencias, optimización diferenciada por capas, y esquemas de exploración adaptativos permite superar muchos de los desafíos tradicionales del entrenamiento de redes profundas en entornos de RL, estableciendo un nuevo estándar de desempeño y eficiencia en estos dominios complejos.

4. Resultados

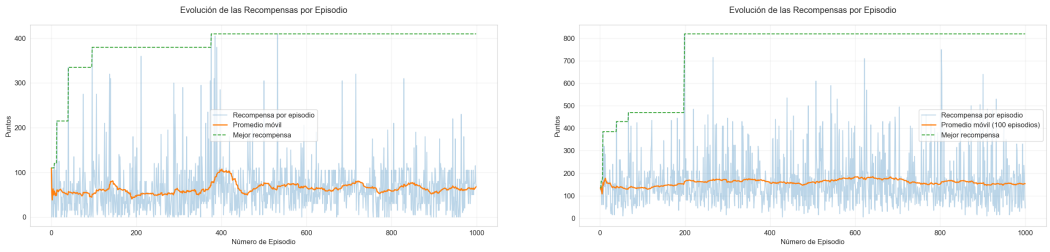
Pong



(a) Pong con DQN (b) Pong con ResNet

Figura 1: Resultados en el entorno de Pong

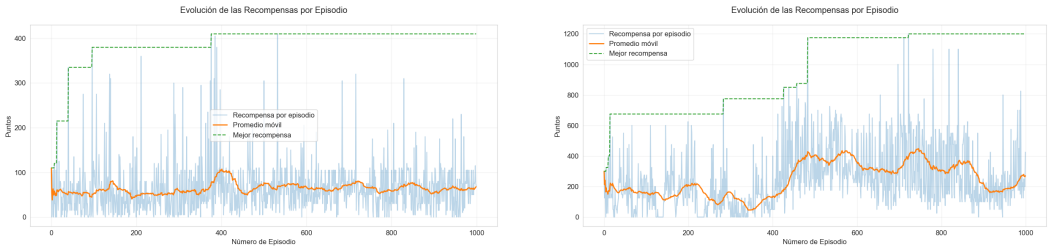
Space Invaders



(a) Space Invaders con DQN (b) Space Invaders con ResNet

Figura 2: Resultados en el entorno de Space Invaders

Q*bert



(a) Q*bert con DQN

(b) Q*bert con ResNet

Figura 3: Resultados en el entorno de Q*bert

Resultados de "Pong"

- **DQN:** Se observa un progreso gradual pero constante en el rendimiento del modelo. Inicialmente, las recompensas eran muy bajas, con valores cercanos a -20, lo que refleja un comportamiento casi aleatorio. A medida que disminuyó la tasa de exploración (epsilon), el agente empezó a tomar decisiones más basadas en la política aprendida. Aunque la mejora fue lenta, eventualmente se lograron recompensas positivas, alcanzando un máximo de 12, lo que indica que el modelo aprendió a jugar de forma más efectiva. El promedio de recompensas también mostró una tendencia ascendente, evidenciando una mejoría en el desempeño general del modelo conforme avanzaba el entrenamiento. Dicho esto, se puede observar una disminución rápida en el rendimiento alrededor del episodio 450, las razones para esto pueden ser numerosas y se necesita más investigación para determinar la causa. **1a**
- **ResNet:** En contraste, la gráfica de Pong con ResNet demuestra una mejora significativa en el rendimiento a lo largo de los episodios. Inicialmente, el agente presentaba recompensas bajas y constantes (alrededor de -20), con una política totalmente aleatoria ($\varepsilon = 1.0$). Sin embargo, una vez que el valor de ε se redujo a 0.01, el modelo comenzó a aprender gradualmente, alcanzando recompensas positivas a partir del episodio 230. A partir de ese punto, el rendimiento continuó mejorando, logrando eventualmente recompensas máximas de hasta 21 puntos y un promedio estable cercano a 19 en los últimos episodios, lo que indica una convergencia satisfactoria del modelo y una política eficaz para jugar el juego. **1b**

Resultados de "Space Invaders"

- **DQN:** Inicialmente, el modelo alcanzó un máximo de recompensa de 215 en el episodio 13, pero esta se mantuvo constante por varios episodios. Más adelante, en el episodio 40, logró una recompensa de 335, y finalmente alcanzó su mejor desempeño en el episodio 376 con una recompensa máxima de 410. El valor promedio de las recompensas también mejoró, alcanzando su punto más alto en 148.5, indicando una mayor consistencia en el rendimiento. La política de exploración (epsilon) se redujo de manera continua desde 1.0 hasta aproximadamente 0.82, lo que sugiere que el modelo pasó de explorar el entorno a explotar su conocimiento aprendido conforme avanzaba el entrenamiento. **2a**
- **ResNet:** Se observa una mejora significativa en el rendimiento con el paso de los episodios. En las primeras etapas, las recompensas eran variables y moderadas, pero a partir del episodio 7 se obtuvo una recompensa de 385, marcando un progreso notable. El mejor rendimiento se alcanzó en el episodio 199 con una recompensa de 820, estableciendo un nuevo récord y elevando la mejor recompensa promedio a 287.5. La política de exploración-explotación funcionó adecuadamente, con la tasa de exploración (epsilon) disminuyendo de manera gradual, lo que permitió al modelo explotar el conocimiento adquirido y estabilizar su rendimiento. El modelo logró aprender una política efectiva para obtener recompensas altas de forma consistente. **2b**

Resultados de "Q*bert"

- **DQN:** El modelo DQN comienza con un valor alto de epsilon, lo que significa que toma muchas acciones aleatorias. En esta fase inicial, sus puntajes son bajos (alrededor de 275), ya que aún no ha aprendido buenas estrategias. En lo que avanza el entrenamiento, epsilon disminuye gradualmente, permitiendo al agente explotar más lo aprendido. Se observa una mejora progresiva en el rendimiento: primero alcanza 525 puntos y, finalmente, llega a un máximo de 1025. Esto indica que el agente logra aprender patrones efectivos en el entorno de Q*bert, mejorando sus decisiones con base en la experiencia acumulada. El DQN muestra un aprendizaje exitoso, pasando de una exploración ineficiente a una explotación efectiva que incrementa su puntaje.**3a**
- **ResNet:** Inicia con un epsilon moderado (0.85), obteniendo recompensas decentes (300 puntos). A medida que explora menos (epsilon baja a 0.49 en el episodio 5), el rendimiento fluctúa: sube a 400–675–600 puntos, con *best_reward* alcanzando 775 en el episodio 283. Ya en explotación (ε=0.05 desde el episodio 26), obtiene puntacion más consistentes, frecuentemente cientos de puntos, destacando unos como 850 en el episodio 426, 1175 en el 483, y finalmente 1200 en el episodio 722. **3b**

5. Conclusión

Este estudio demostró la viabilidad de implementar arquitecturas de aprendizaje por refuerzo profundo, tanto tradicionales (DQN) como residuales (Res-DQN), para resolver entornos complejos de Atari. La metodología consistente aplicada a múltiples juegos —con ajustes estratégicos en hiperparámetros como el tamaño del *batch*, la exploración y las técnicas de *replay*— permitió adaptar cada modelo a las dinámicas únicas de Pong, Space Invaders y Q*bert. La integración de bloques residuales en la Res-DQN, junto con técnicas auxiliares como *target networks* y *experience replay*, refleja un enfoque sólido para abordar desafíos clásicos del DRL, como la inestabilidad en el entrenamiento y el desvanecimiento de gradientes.

La implementación en PyTorch garantizó flexibilidad y reproducibilidad, sentando las bases para futuras extensiones, como el uso de algoritmos avanzados (e.g., *Double DQN*) o la incorporación de atención jerárquica. Estos desarrollos subrayan el potencial de las arquitecturas residuales en DRL, no solo para mejorar el rendimiento, sino también para facilitar el entrenamiento de agentes en entornos con observaciones de alta dimensionalidad.

Los resultados experimentales confirman que las Res-DQN no solo igualan, sino que superan el rendimiento de las DQN tradicionales en los tres juegos evaluados. La arquitectura residual demostró ser especialmente efectiva en entornos complejos como Space Invaders y Q*bert, donde la mayor profundidad y capacidad de generalización de las Res-DQN permitieron un aprendizaje más estable y eficiente. Además, la adaptabilidad de los hiperparámetros específicos para cada juego, como el tamaño del batch y las estrategias de exploración, jugó un papel clave en el éxito de ambos modelos. En conclusión, este trabajo no solo valida las ventajas de las Res-DQN, sino que también establece un marco reproducible para futuras mejoras en el campo del DRL.

6. Referencias

Referencias

[1] Tuomas Haarnoja, Aurick Zhou, Pieter Abbeel, and Sergey Levine. Reinforcement learning with deep energy-based policies. In *International Conference on Machine Learning (ICML)*, 2017.

[2] Volodymyr Mnih, Koray Kavukcuoglu, David Silver, Andrei A. Rusu, Joel Veness, Marc G. Bellemare, Alex Graves, Martin Riedmiller, Andreas K. Fidjeland, Georg Ostrovski, Stig Petersen, Charles Beattie, Amir Sadik, Ioannis Antonoglou, Helen King, Dharshan Kumaran, Daan Wierstra, Shane Legg, and Demis Hassabis. Human-level control through deep reinforcement learning. *Nature*, 518(7540):529–533, 2015.

[3] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 770–778, 2016.

[4] Tom Schaul, John Quan, Ioannis Antonoglou, and David Silver. Prioritized experience replay. *arXiv preprint arXiv:1511.05952*, 2016.

[5] Matteo Hessel, Joseph Modayil, Hado van Hasselt, Julian Schrittwieser, Ioannis Antonoglou, and David Silver. Rainbow: Combining improvements in deep reinforcement learning. In *AAAI Conference on Artificial Intelligence*, 2018.

[6] Volodymyr Mnih, Adrià Puigdomènech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2016.

[7] Tuomas Haarnoja, Haoran Tang, Pieter Abbeel, and Sergey Levine. Residual reinforcement learning. In *International Conference on Machine Learning (ICML)*, 2018.