UNIVERSIDAD NACIONAL DE INGENIERÍA
Facultad de Ingeniería Industrial y de Sistemas
Unidad de Posgrado
Maestría en Inteligencia Artificial

Trabajo Final

Artículo de Investigación

Aprendizaje por refuerzo profundo para CarRacing-v3

Curso: Aprendizaje por Refuerzo

Grupo 4:

Koc Góngora, Luis Enrique
Mancilla Antaya, Alex Felipe
Meléndez García, Herbert Antonio
Paitán Cano, Dennis Jack

Docente: María Tejada Begazo

14 de enero de 2026

Aprendizaje por refuerzo profundo para CarRacing-v3 con DQN y variantes

Luis Koc, Alex Mancilla, Herbert Meléndez, Dennis Jack Paitán
Unidad de Posgrado, Universidad Nacional de Ingeniería (UNI)
luis.koc@gmail.com amancillaa@uni.pe
hamg.94@gmail.com dennis.paitan.c@uni.pe

***Resumen*—Se implementa y evalúa un agente de aprendizaje por refuerzo profundo para el entorno visual *CarRacing-v3* de Gymnasium, donde el agente aprende a conducir usando únicamente observaciones RGB de 96×96 píxeles. Se compara DQN [1] con dos extensiones ampliamente adoptadas para estabilizar el aprendizaje y mejorar el desempeño: Double DQN [2] (para reducir sobreestimación) y Dueling DQN [3] (para descomponer el valor del estado y la ventaja de la acción).**

Para adaptar CarRacing (acciones continuas) a DQN (acciones discretas), se discretiza el control (steer, gas, brake) en 12 acciones fijas, siguiendo la especificación del entorno [4]. El estado se preprocesa con conversión a escala de grises, normalización y apilamiento de 4 frames (frame stacking) para capturar dinámica temporal, siguiendo el esquema de DQN con entradas visuales [1].

Los resultados se reportan mediante evaluación sin exploración ($\epsilon = 0$) en 5 episodios, y se presentan tablas y gráficos de entrenamiento/evaluación generados automáticamente por el proyecto. En las corridas evaluadas, Double DQN alcanzó el mayor retorno promedio a 1000 episodios, mientras que a 200 episodios los retornos fueron más variables, evidenciando sensibilidad a la etapa de entrenamiento y a la varianza del entorno.

***Index Terms*—Aprendizaje por refuerzo, Deep Q-Network (DQN), Double DQN, Dueling DQN, CarRacing-v3, Gymnasium, PyTorch.**

I. Introducción

El aprendizaje por refuerzo (RL) estudia agentes que aprenden políticas a partir de interacción con un entorno, maximizando una recompensa acumulada [5]. En problemas de control a partir de percepción visual, el uso de redes convolucionales para aproximar funciones de valor permitió escalar RL a entradas de alta dimensión [1].

En este trabajo se aborda el entorno *CarRacing-v3* [4], un problema de control continuo con observaciones visuales y dinámica física basada en Box2D [6]. La motivación es evaluar, en un entorno visual no trivial, el impacto de mejoras conocidas sobre DQN: Double DQN [2] y Dueling DQN [3], comparando también con el DQN base [1]. Además, se busca asegurar reproducibilidad a partir del repositorio, el *README* y el cuaderno final de resultados.

Este trabajo se desarrolla como trabajo final del curso de *Aprendizaje por Refuerzo*, como pare de la maestría en Inteligencia Artifical de la Universidad Nacional de Ingeniería (Perú).

Objetivos: (i) entrenar agentes DQN y variantes en CarRacing-v3; (ii) registrar métricas y gráficos de entrenamiento; (iii) evaluar los modelos sin exploración y comparar retornos con tablas/figuras reproducibles.

II. Problema y motivación

CarRacing-v3 requiere aprender una política de conducción que mantenga el vehículo en la pista y progrese sobre la ruta usando únicamente visión [4]. La alta dimensionalidad de la observación, la dinámica continua y la necesidad de exploración hacen que el problema sea un caso representativo de RL profundo [5], [1].

La motivación práctica es estudiar la estabilidad del entrenamiento con recursos limitados (CPU y entrenamiento sin renderizado) y comprender qué variante de DQN ofrece mejor compromiso entre complejidad y desempeño, manteniendo una implementación controlada.

III. Metodología

III-A. Entorno

Se utiliza Gymnasium [7] con el entorno *CarRacing-v3* [4]. La observación es una imagen RGB 96×96, y la acción original es continua. El motor físico es Box2D [6].

III-B. Discretización del espacio de acciones

DQN asume un espacio de acciones discreto [1]. Por ello, se discretiza el control continuo (steer, gas, brake) en 12 acciones predefinidas (combinaciones de giro $\in \{-1, 0, 1\}$, aceleración $\in \{0, 1\}$ y freno $\in \{0, 0{,}2\}$), alineadas con el diseño documentado del entorno [4].

III-C. Preprocesamiento y estado

Cada frame se convierte a escala de grises y se normaliza a $[0, 1]$ usando OpenCV [8]. Se apilan 4 frames consecutivos (frame stacking) para capturar velocidad/movimiento de forma implícita, práctica estándar en DQN con entradas visuales [1]. El estado final tiene forma $(4, 96, 96)$.

III-D. Modelos: DQN, Double DQN y Dueling DQN

Se implementa un aproximador $Q_\theta(s, a)$ con una CNN sencilla (2 capas convolucionales + MLP). El entrenamiento usa *experience replay* [9] y política ϵ-greedy [5].

DQN: el objetivo temporal (target) usa la red objetivo Q_{θ^-} [1].

Double DQN: selecciona la acción con la red online y

Cuadro I
Evaluación a 200 episodios (5 episodios, $\epsilon = 0$)

Algoritmo	Mean	Std	Min	Max	Full
DQN	238.85	207.06	67.24	646.67	4
Double DQN	108.76	81.52	30.74	265.52	5
Dueling Double DQN	262.85	174.93	46.63	463.14	4

Cuadro II
Evaluación a 1000 episodios (5 episodios, $\epsilon = 0$)

Algoritmo	Mean	Std	Min	Max	Full
DQN	418.58	209.95	252.73	825.09	5
Double DQN	567.72	257.19	278.29	883.33	5
Dueling Double DQN	462.46	222.57	189.76	793.47	4

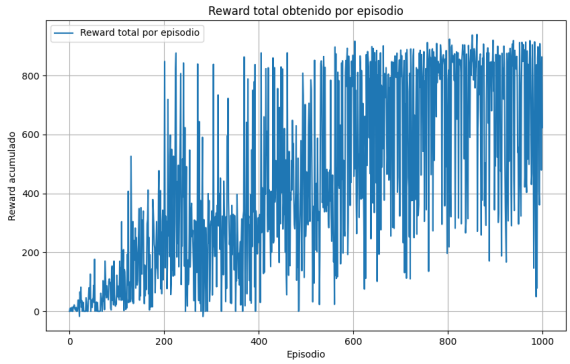

Figura 1. Entrenamiento: reward por episodio (Double DQN).

la evalúa con la red objetivo para reducir sobreestimación [2].

Dueling DQN: descompone $Q(s, a) = V(s)+A(s, a)$ para aprender valor del estado y ventaja de acción [3].

Las variantes se activan con banderas de configuración (Double/Dueling) y pueden combinarse (Dueling Double DQN).

III-E. Entrenamiento

El optimizador usado es Adam [10]. Se entrena por episodios, con red objetivo actualizada periódicamente (cada 5 episodios) y guardado de checkpoints. Para acelerar entrenamiento, se aplica *frame skipping* [1] con `SKIP_FRAMES=2`.

IV. Diseño experimental

IV-A. Configuración

Se entrenan agentes por 1000 episodios (cuando aplica), registrando por episodio: reward acumulado, ϵ, tamaño del buffer y loss promedio. La evaluación se realiza con $\epsilon = 0$ por 5 episodios, guardando un CSV con el retorno total por episodio y un gráfico de retornos. El repositorio y el cuaderno de resultados documentan el flujo de entrenamiento y evaluación (Jupyter [11]).

IV-B. Métrica

La métrica principal es el retorno total por episodio (reward acumulado), reportado como media y desviación estándar poblacional ($ddof$=0) sobre 5 episodios. Este reporte sigue el resumen implementado en el cuaderno final.

V. Resultados

V-A. Resumen cuantitativo

La Tabla I y Tabla II resumen las evaluaciones (5 episodios). Se reporta también cuántos episodios alcanzaron 1000 frames, ya que algunos episodios pueden terminar antes (por truncamiento/condición de fin del entorno) [4].

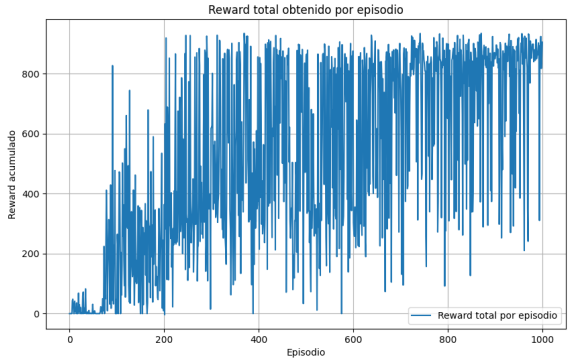

Figura 2. Entrenamiento: reward por episodio (DQN).

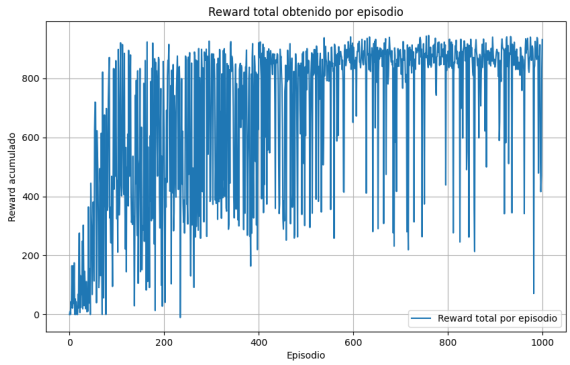

Figura 3. Entrenamiento: reward por episodio (Dueling Double DQN).

V-B. Gráficos de entrenamiento

Las Figuras 1–3 muestran la evolución del retorno por episodio durante entrenamiento. Los gráficos se generan automáticamente por el proyecto usando Matplotlib [12].

V-C. Gráficos de evaluación

Las Figuras 7–9 muestran los retornos obtenidos en evaluación (5 episodios) para distintos checkpoints. Dado

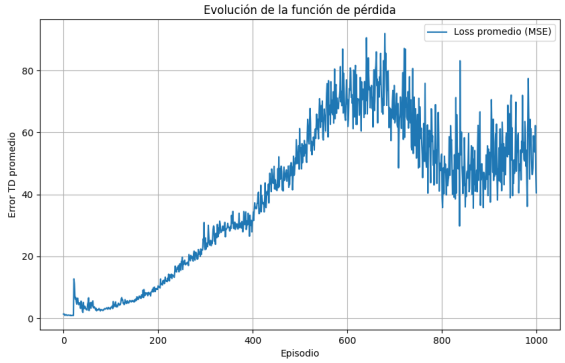

Figura 4. Entrenamiento (Double DQN): loss promedio por episodio.

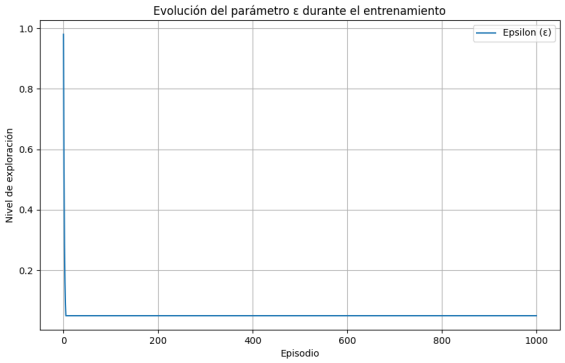

Figura 5. Entrenamiento (Double DQN): decaimiento de ϵ (exploración).

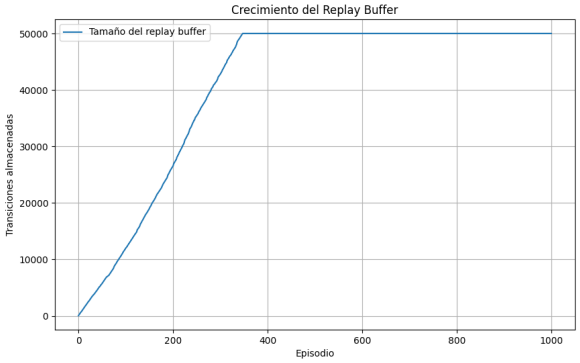

Figura 6. Entrenamiento (Double DQN): tamaño del replay buffer.

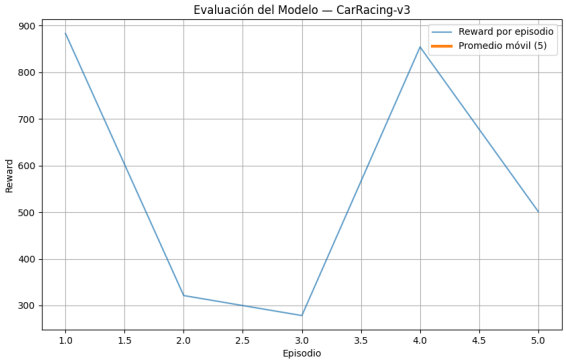

Figura 7. Evaluación (1000): Double DQN.

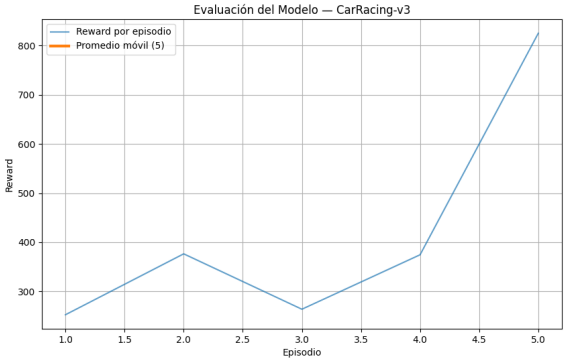

Figura 8. Evaluación (1000): DQN.

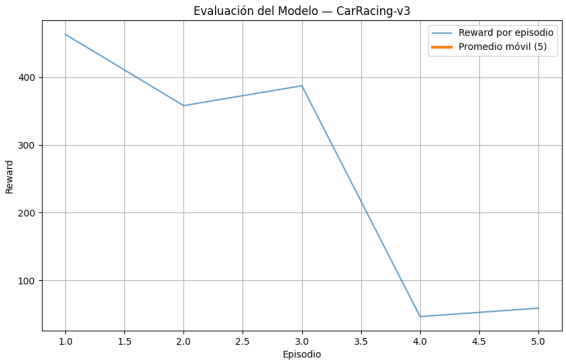

Figura 9. Evaluación (200): Dueling Double DQN.

el tamaño muestral pequeño, estas curvas deben interpretarse como evidencia indicativa y no concluyente [5].

VI. Discusión crítica

En las evaluaciones a 1000 episodios, Double DQN obtuvo el mayor retorno promedio (Tabla II), consistente con su objetivo de reducir sobreestimación en el cálculo de targets [2]. Sin embargo, a 200 episodios la variabilidad fue alta y el desempeño relativo cambió (Tabla I), lo cual puede explicarse por (i) alta varianza inherente a RL [5]; (ii) sensibilidad a hiperparámetros y etapa de entrenamiento; y (iii) tamaño de evaluación limitado (5 episodios).

VII. Especificaciones y restricciones de implementación

Restricciones: entrenamiento sin render para acelerar (modo `rgb_array`); CPU-only como configuración típica; uso de frame skipping y mini-batches para controlar costo computacional.
Lenguaje y librerías: Python [13], Gymnasium [7], PyTorch [14], NumPy [15], Matplotlib [12], OpenCV [8], pandas [16], ImageIO [17] y pygame [18].

VIII. Reproducibilidad

El repositorio del proyecto es: https://github.com/alexmancila/Proyect-RL-CarRacing-DQN/tree/main.
La configuración base (por defecto) se encuentra en `src/configuracion.py`. Para reproducir, se recomienda: (i) instalar dependencias desde `requirements.txt`; (ii) entrenar con `python main.py`; (iii) evaluar modelos con los scripts en `tests/`. La documentación de ejecución y el cuaderno final en Jupyter describen los pasos y rutas de resultados [11].

IX. Participación y contribuciones

Koc Góngora, Luis Enrique: integración y ajustes del pipeline RL, análisis de resultados, documentación técnica.
Mancilla Antaya, Alex Felipe: implementación de variantes (Double/Dueling), soporte en evaluación y automatización.
Meléndez García, Herbert Antonio: elaboración del informe y consolidación de resultados/figuras.
Paitán Cano, Dennis Jack: revisión de experimentos, organización de resultados y discusión crítica.

X. Transparencia (uso de herramientas)

Se utilizó Jupyter Notebook para análisis y reporte de resultados [11]. Para apoyo en redacción y consistencia del documento se empleó asistencia de IA generativa como herramienta editorial; las decisiones técnicas, experimentos y resultados reportados provienen del código y artefactos del repositorio.

XI. Conclusiones y trabajo futuro

Se implementó un agente basado en DQN y variantes (Double/Dueling) para CarRacing-v3. Con los resultados disponibles, Double DQN mostró el mejor retorno promedio a 1000 episodios. Como trabajo futuro: (i) aumentar episodios de evaluación y semillas para reducir varianza [5]; (ii) explorar técnicas de mejora de muestreo como *prioritized experience replay* [19] y arquitecturas más profundas; (iii) ajustar discretización de acciones o migrar a métodos continuos para control continuo nativo (p.ej., DDPG [20] o SAC [21]).

Referencias

[1] V. Mnih, K. Kavukcuoglu, D. Silver, A. A. Rusu, J. Veness, M. G. Bellemare, A. Graves, M. Riedmiller, A. K. Fidjeland, G. Ostrovski, S. Petersen, C. Beattie, A. Sadik, I. Antonoglou, H. King, D. Kumaran, D. Wierstra, S. Legg, and D. Hassabis, "Human-level control through deep reinforcement learning," *Nature*, vol. 518, no. 7540, pp. 529–533, 2015.

[2] H. van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Proceedings of the AAAI Conference on Artificial Intelligence*, 2016. [Online]. Available: https://ojs.aaai.org/index.php/AAAI/article/view/10295

[3] Z. Wang, T. Schaul, M. Hessel, H. van Hasselt, M. Lanctot, and N. de Freitas, "Dueling network architectures for deep reinforcement learning," in *Proceedings of the 33rd International Conference on Machine Learning (ICML)*, 2016. [Online]. Available: http://proceedings.mlr.press/v48/wangf16.html

[4] Farama Foundation, "Gymnasium carracing-v3 environment documentation," Online documentation, accessed: 2026-01-14. [Online]. Available: https://gymnasium.farama.org/environments/box2d/car_racing/

[5] R. S. Sutton and A. G. Barto, *Reinforcement Learning: An Introduction*, 2nd ed. MIT Press, 2018. [Online]. Available: http://incompleteideas.net/book/the-book-2nd.html

[6] E. Catto, "Box2d: A 2d physics engine for games," Software, accessed: 2026-01-14. [Online]. Available: https://box2d.org/

[7] Farama Foundation, "Gymnasium: A standard api for reinforcement learning environments," Software, accessed: 2026-01-14. [Online]. Available: https://gymnasium.farama.org/

[8] G. Bradski, "The opencv library," in *Dr. Dobb's Journal of Software Tools*, 2000. [Online]. Available: https://opencv.org/

[9] L.-J. Lin, "Self-improving reactive agents based on reinforcement learning, planning and teaching," in *Machine Learning: Proceedings of the Eighth International Workshop*, 1992.

[10] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *International Conference on Learning Representations (ICLR)*, 2015. [Online]. Available: https://arxiv.org/abs/1412.6980

[11] T. Kluyver, B. Ragan-Kelley, F. Pérez, B. Granger, M. Bussonnier, J. Frederic, K. Kelley, J. Hamrick, J. Grout, S. Corlay, P. Ivanov, D. Avila, S. Abdalla, and C. Willing, "Jupyter notebooks—a publishing format for reproducible computational workflows," in *Positioning and Power in Academic Publishing: Players, Agents and Agendas*. IOS Press, 2016, pp. 87–90.

[12] J. D. Hunter, "Matplotlib: A 2d graphics environment," *Computing in Science & Engineering*, vol. 9, no. 3, pp. 90–95, 2007.

[13] Python Software Foundation, "Python language reference, version 3.x," Software, accessed: 2026-01-14. [Online]. Available: https://www.python.org/

[14] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala, "Pytorch: An imperative style, high-performance deep learning library," in *Advances in Neural Information Processing Systems (NeurIPS)*, 2019. [Online]. Available: https://pytorch.org/

[15] C. R. Harris, K. J. Millman, S. J. van der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, N. J. Smith *et al.*, "Array programming with NumPy," *Nature*, vol. 585, no. 7825, pp. 357–362, 2020.

[16] W. McKinney, "Data structures for statistical computing in python," in *Proceedings of the 9th Python in Science Conference (SciPy)*, 2010, pp. 56–61.

[17] A. Silvester and contributors, "Imageio: Python library for reading and writing image data," Software, accessed: 2026-01-14. [Online]. Available: https://imageio.readthedocs.io/

[18] pygame contributors, "pygame," Software, accessed: 2026-01-14. [Online]. Available: https://www.pygame.org/

[19] T. Schaul, J. Quan, I. Antonoglou, and D. Silver, "Prioritized experience replay," in *International Conference on Learning Representations (ICLR)*, 2016. [Online]. Available: https://arxiv.org/abs/1511.05952

[20] T. P. Lillicrap, J. J. Hunt, A. Pritzel, N. Heess, T. Erez, Y. Tassa, D. Silver, and D. Wierstra, "Continuous control with deep reinforcement learning," in *International Conference on Learning Representations (ICLR)*, 2016. [Online]. Available: https://arxiv.org/abs/1509.02971

[21] T. Haarnoja, A. Zhou, P. Abbeel, and S. Levine, "Soft actor-critic: Off-policy maximum entropy deep reinforcement learning with a stochastic actor," in *International Conference on Machine Learning (ICML)*, 2018. [Online]. Available: http://proceedings.mlr.press/v80/haarnoja18b.html