Diseño de un Agente Inteligente en un Entorno Simulado

El aprendizaje por refuerzo es una técnica de machine learning que busca que un agente se desenvuelva y genere una serie de acciones en un entorno para lograr los mejores resultados bajo un sistema de recompensas (AWS, s.f.), esto le permite mediante la exploración y explotación descubrir las mejores rutas de procesamiento para maximizar la ganancia (AWS, s.f.); es decir, el aprendizaje resulta de una interacción continua entre la inteligencia y los entes externos a ella. Una de sus aplicaciones fundamentales es el desarrollo de agentes que resuelvan juegos, tal es el ejemplo de 'LunarLander', un entorno cuyo objetivo es aterrizar una nave en una zona delimitada bajo 4 posibles acciones: no hacer nada, impulsar el motor a la izquierda, a la derecha o a arriba. Esto es particularmente útil cuando se identifica el alto impacto del aprendizaje por refuerzo en la aeronáutica, abordando problemas como la prevención de colisiones, gestión del tráfico o control de altitudes (Razzaghi et al., 2024), en esta medida, simular estos entornos permite desarrollar e implementar soluciones automatizadas que asistan acciones complejas como el alunizaje, donde se requiere un aterrizaje muy suave en un superficie altamente accidentada (Desmarais, 2024).
El objetivo de 'LunarLander' es generar un agente capaz de aterrizar suavemente en un entorno donde se puede configurar tanto la gravedad (-10 por defecto) como la velocidad del viento (15 por defecto), y mediante las acciones que se decidan sobre la nave, se le asignan diferentes valores para 8 estados: coordenadas (x,y), velocidades lineales (x,y), ángulo, velocidad angular y dos booleanos que señalan si la nave está en contacto con la superficie. Con base en estos estados el sistema asigna recompensas según las acciones que tome durante el proceso y terminación (éxito [cuando la nave aterriza sin golpes y mantiene su posición durante un periodo prolongado], colisión o inactivo).
Ahora bien, entrenar a un agente para la resolución de 'LunarLander' requiere la integración de diversas técnicas según las características del entorno cuyas acciones son discretas (es decir, no se puede implementar Q - Learning o similares que suponen continuidad en estados y acciones); por tanto, el objetivo de este proyecto es desarrollar un agente inteligente capaz de aprender este entorno y generar soluciones deseables mediante técnicas de aprendizaje automático. Además, se quieren comparar diversos enfoques y evaluar su rendimiento para identificar la solución más adecuada; para ello, se implementaron modelos enfocados en dos ejes principales del entorno.

Metodología

Aproximación a la Función de Valor

Las funciones de valor son fundamentales para el aprendizaje por refuerzo, permitiendo identificar zonas que maximizan la recompensa a partir de condiciones del entorno; sin embargo, su cálculo es altamente costoso, por tal motivo, el uso de modelos supervisados es muy útil. Se ejecutó LunarLander 500 veces con políticas aleatorias, permitiendo la recolección de valores para entrenar 3 modelos de aprendizaje supervisado.

Árbol de Decisión

Este método implementa una estrategia en donde se realiza una búsqueda para identificar los puntos que pueden ser óptimos dentro del árbol, repitiendo el proceso de forma descendente hasta que con todos los registros se terminara la regresión (IBM, 2025). En este caso, es útil porque es un modelo sencillo, en donde su ventaja radica en la facilidad de interpretabilidad, permitiendo comprender relaciones entre los datos de entrada y el resultado; sin embargo, su complicaciones están en que se puede sobreajustar, perdiendo la capacidad de generalización; o en caso contrario, presentar underfitting según la configuración de los hiperparametros, los cuales, en este caso, se seleccionaron

tras múltiples pruebas, encontrando el balance para un mejor rendimiento y satisfacer el problema sin sobreajustar el modelo tal como se presenta en la Tabla 1.

XG Boost

Se seleccionó XG Boost para aproximar la función de valor debido a su alto rendimiento y capacidad de generalización. La elección se basó en:

- Alto poder predictivo: Es eficaz en problemas de regresión, lo cual es esencial para estimar con precisión los valores esperados de recompensa.
- Manejo de sesgo – varianza: Maneja boosting, lo cual permite reducir el sesgo sin incrementar significativamente la varianza, lo cual es ideal en modelos RL.
- Regularización incorporada: XG Boost implementa regularización L1 y L2, lo cual permite controlar el sobreajuste y mejora la generalización.

El modelo fue configurado con los hiperparámetros de la Tabla 2, seleccionados empíricamente para alcanzar un equilibrio entre precisión y generalización.

Red Neuronal

El uso de redes neuronales en el aprendizaje por refuerzo en una práctica muy extendida; por ejemplo, (Arranz et al., 2019) estudiaron el aprendizaje profundo aplicado a entornos simples de OpenIA Gym obteniendo excelentes resultados, por otro lado, CyberLympha (2022) señala la gran eficacia de las RNN para reemplazar las Tablas - Q y generar salidas y comportamientos deseables. Lo anterior sumado a su alta flexibilidad, siendo capaces de aprender patrones no lineales muy complejos (InteractiveChaos, s.f.) hizo considerar este algoritmo para la aproximación de la función de valor.

La red neuronal se configuró con la librería Tensorflow Keras, la cual proporciona herramientas para el preprocesamiento y entrenamiento. Inicialmente se realizó la partición train - test en una proporción 80% - 20% respectivamente, posteriormente, se ingresaron los datos a una red con la arquitectura de la Figura 1. La selección de los hiperparámetros se realizó de acuerdo con el conocimiento del problema y un ajuste a prueba y error, igualmente, la activación relu fue elegida dado que es la que suele reportar mejores resultados en la literatura.

Aprendizaje por Refuerzo

Hay múltiples formas de aproximarse al aprendizaje por refuerzo, algunos algoritmos integran modelos de redes neuronales o ensambles, mientras que otros hacen uso de tablas o fórmulas. En este caso se programaron 2 modelos de Reinforcement Learning con el fin de comparar y analizar su rendimiento. Los hiperparámetros de RL utilizados fueron los presentados en la Tabla 3. Los modelos de aprendizaje por refuerzo implementados fueron:

Deep Q-Network (DQN)

Ante gran variedad de estados en el entorno, la actualización de tablas - Q se vuelve altamente ineficiente y muy costoso computacionalmente, por tal motivo, es necesario implementar estrategias alternativas que faciliten este procedimiento; por ejemplo, las redes neuronales para la aproximación de la función de valor (Sanz, 2020). En este caso se presenta otra red neuronal profunda integrada a su funcionamiento en el 'LunarLander' cuya arquitectura se relaciona en la Figura 2.

La complejización de la arquitectura se debió por un lado a la dificultad para lograr éxitos en el entorno y por otro al requerimiento de implementar aprendizaje profundo (tres capas ocultas o más), nuevamente se utilizó únicamente ReLu debido a su buen rendimiento en la literatura y, que en pruebas donde se reemplaza la activación por otras funciones como Sigmoide o Tanh el resultado no

nota mejoría. Adicionalmente se utilizó Adam, permitiendo adaptar la tasa de aprendizaje según cada parámetro.

Fitted Q-Iteration

Es un algoritmo de aprendizaje por refuerzo que se usa con un valor de Q el cual ayuda a la estimación de la recompensa futura en problemas de decisión secuencial, en donde, esto hace encontrar una política óptima. Por parte del objetivo para la predicción del valor esperado es útil gracias a la función Q ya que como su definición lo indica es el "valor esperado de tomar una acción en un estado y actuar óptimamente después", esto hace que con ayuda del árbol no solo prediga si no que codifique el conocimiento para llevar a cabo una estrategia óptima (Gaeta, 2016).

En este caso, se decidió probar con dos configuraciones distintas que permitieran comparar el rendimiento, una primer aproximación integrada con árboles de decisión y una segunda con XGBoost. Las diferencias de hiperparámetros se detallan en la Tabla 3.

Análisis de Resultados

Aproximación a la Función de Valor

Árbol de Decisión

El modelo de árbol de decisión aplicado mostró un rendimiento moderado, puesto que, el MSE fue de (54.53) y el MAE de (1.82), lo que indica que las predicciones del valor Q tienen una predicción razonable respecto a los valores reales, el R2 fue de (0.53), reflejando que el modelo logra explicar aproximadamente el 54% de la variabilidad en los valores Q, lo cual es bastante bueno. En cuanto a las decisiones aprendidas, se observa que el modelo tiende a asignar acciones óptimas consistentes a estados similares, lo que sugiere cierta estabilidad en la política aprendida, sin embargo, algunos estados presentan valores Q negativos, lo que puede indicar predicciones poco favorables en ciertas situaciones. En general, aunque el modelo muestra una política razonable y un nivel aceptable de precisión, aún hay espacio para mejorar, sería recomendable experimentar con modelos más complejos y robustos, o aumentar la cantidad y diversidad de los episodios utilizados para entrenar el modelo.

XG Boost

Con el fin de evaluar la calidad de la aproximación de la función de valor Q(s,a) mediante el modelo XG Boost, se utilizaron tres métricas principales: MAE, MSE y R2 dando así un resultado de 1.23, 36.68 y 0.69 respectivamente. Estos valores reflejan un buen desempeño del modelo. En particular, el bajo valor del MAE indica que, en promedio, el error entre la recompensa real y la predicción del modelo es pequeño. Además, en la Figura 4 se observa una gráfica de dispersión entre las predicciones y la recompensa real, en ella se observa que el modelo es capaz de capturar correctamente la relación general entre los estados, acciones y recompensas. Aunque se identifica una concentración de puntos especialmente alrededor de -100, lo cual corresponde posiblemente a estados penalizados. A pesar de esto, el modelo mantiene una buena capacidad de generalización en la mayoría de los casos, ofreciendo un balance entre precisión, robustez y capacidad para capturar tendencias en los datos. En general, el ensamble XGBoost logra captar los patrones y relaciones entre los datos.

Red Neuronal

Evaluando la red neuronal se identificó un buen ajuste y modelamiento de los datos, con un MSE de 47.3, un MAE de 2.19 y un R2 de 0.61 refleja no solo una baja tasa de errores en la predicción sino una buena capacidad para replicar los patrones de predicción y la variabilidad presente en los datos.

Además, la Figura 5 muestra que no existe sobreajuste, dado que con el paso de las épocas el error es relativamente consistente tanto para entrenamiento como para validación, mostrando en ambos casos un descenso natural y deseable. En general, la red neuronal con una arquitectura relativamente simple es muy buena aproximando la función de valor, lo que la hace deseable para su implementación en el juego. Ahora bien, sería de utilidad identificar que tanto estos resultados se hacen visibles en el 'LunarLander', enfoque que se detalla más en el apartado 'Aprendizaje por Refuerzo' en 'DQN'.

Comparativamente, todos los modelos logran aproximaciones aceptables a los valores Q, generando resultados deseables; sin embargo, hay diferencias importantes que se deben tener en cuenta; por ejemplo, el árbol de decisión, aunque ajusta bastante bien gran parte de la variabilidad de los datos ($R2 = 0.53$) se ve superado con creces tanto por el XGBoost como por la Red Neuronal, este resultado es esperable teniendo en cuenta los niveles de complejidad de cada uno de estos modelos. No obstante, el que más destaca es el XGBoost, probablemente en su arquitectura para reducir el sesgo progresivamente e implementación de la regularización muestra un desempeño muy bien ajustado a las complejidades de la relación presente. En general, estos resultados muestran que efectivamente si es buena idea la implementación de modelos de aprendizaje supervisado para aproximar la función de los valores Q; sin embargo, ¿cómo se ve esto de manera aplicada? ¿en la aplicación continuará siendo el XGBoost el mejor modelo en términos de optimización de políticas?

Aprendizaje por Refuerzo

Deep Q-Network (DQN)

En general la DQN tuvo un rendimiento aceptable, llegando a cumplir con el objetivo del juego en 7 ocasiones y convergiendo hacia las acciones deseadas, además, mostró una mejoría continua a medida que se privilegia la explotación sobre la exploración. El monitoreo del entrenamiento mostró 3 aspectos fundamentalmente:

- A medida que pasan los episodios, se requieren más pasos, lo que hace que el algoritmo necesite bastante tiempo, recursos computacionales y gran cantidad de iteraciones hasta generar el comportamiento deseado.
- Es altamente sensible a la inicialización, dado que en varias pruebas que se realizaron no se logró ningún éxito y las acciones que tomaba no parecían apuntar hacia la solución del problema; mientras que en otras la convergencia ocurría rápidamente y era más una cuestión de continuar iterando
- Es altamente sensible a las condiciones del entorno, dado que, si se aumenta la gravedad y el viento, el algoritmo se vuelve más propenso a quedarse estancado (la nave se queda flotando y nunca aterriza) o si se disminuye la gravedad y el viento no se logra ningún éxito.

En promedio se requirieron 227 pasos por episodio y cada uno tuvo una recompensa acumulada promedio de -181.32. Esto parece indicar que el algoritmo requiere más tiempo de explotación; sin embargo, debido al alto costo computacional no fue posible continuar el procedimiento; sin embargo, esta aproximación al problema parece ser bastante buena y lograr el comportamiento deseado a pesar de su gran limitación en la baja velocidad de convergencia y la baja estabilidad en el aprendizaje.

Fitted Q-Iteration

Por parte del primer modelo de este tipo, se puede observar que durante las iteraciones se analizaron 10 estados únicos para cada uno de ellos se identificó la acción con el mayor valor Q, el cual está en

un rango entre 8,1 y 9,7, esto refleja que el modelo ha aprendido a tomar decisiones que conducen a una mayor recompensa acumulada. En cuanto a la recompensa promedio, se obtuvo un valor de 2, que es positivo, respecto a los aterrizajes exitosos, se logró 1 lo cual refleja una convergencia hacia el aprendizaje correcto por parte del agente, pero siendo altamente costosa computacionalmente; es decir, de 300 episodios apenas 1 logró el éxito. En general, parece ser una buena aproximación, pero con un alto coste de recursos y con convergencia muy lenta.

Por parte del segundo modelo (FQI con un modelo XG Boost), se obtuvieron los siguientes resultados sobre los episodios de prueba:

- Aterrizajes exitosos: 2 de 20 (10%)
- Recompensa promedio: -324

Estos resultados sugieren que la política actual logra un desempeño satisfactorio en el entorno Lunar Lander. La baja tasa de aterrizajes exitosos sugiere que el agente aún no ha aprendido una política efectiva; sin embargo, comparando con el otro modelo FQI, la diferencia es abismal, en apenas 20 intentos logró 2 éxitos, siendo algo que no logró el otro en 300 iteraciones. El valor negativo de la recompensa promedio indica que la mayoría de los episodios el agente realiza acciones que son penalizadas, como choques, uso excesivo del motor o pérdida de control.

Para mejorar la política aprendida, se recomienda:

1. Aumentar el número de episodios de entrenamiento lo cual mejora la generalización del modelo, lo cual implica adicionalmente, mejorar bastante los recursos computacionales.
2. Aumentar las iteraciones de FQI para permitir una mejor convergencia.
3. Aplicar una política de exploración más estructurada para evitar exploración completamente aleatoria

Comparativamente, se detecta que en todos los casos existe una muy importante limitante: los recursos computacionales, esto era un factor esperable, teniendo en cuenta que el aprendizaje por refuerzo se caracteriza por ser costoso y lento; sin embargo, en unos modelos este factor fue más marcado que en otros. Por el lado de la DQN, aunque logró 7 aterrizajes exitosos de 300 y un notorio avance hacia la convergencia, también mostró un alta sensibilidad tanto a la inicialización como a los cambios en el entorno, lo que lo hace inestable, lento en convergencia y muy costoso; mientras tanto, el primer FQI tuvo tasas de aprendizaje muy bajas en el mismo número de iteraciones que la red. El mayor avance lo muestra la FQI con XGBoost, que logro una tasa de éxito del 10% (muy superior a los dos anteriores) a costa de un costo computacional demasiado alto (generando hasta bloqueos en el dispositivo), esto parece mostrar que efectivamente, el modelo con mejor predicción en la política también produce mejores resultados en el aprendizaje por refuerzo; sin embargo, sería ideal buscar alternativas más eficientes computacionalmente.

Conclusiones

Este proyecto permitió conocer y explorar las integraciones del aprendizaje por refuerzo con el supervisado, comparando el rendimiento de múltiples modelos y visualizando sus efectos en las acciones tomadas por agente. En general, hubo 3 aspectos generales sobre los que es relevante comentar:

- Los modelos supervisados son una base sólida para aproximar la función de valor; sin embargo, hay algunos que logran mejores resultados según la complejidad de la relación entre los estados y la recompensa; por ejemplo, en este caso se intentó con un árbol de

decisión, y aunque logra resultados aceptables, su diseño simple no es suficiente para captar los patrones intrínsecos, en comparación al XGBoost, cuyos resultados son bastante buenos.

- A la hora de integrar los modelos al aprendizaje por refuerzo, debe considerarse el trade – off entre una convergencia lenta o un alto coste computacional; por ejemplo, FQI con XGBoost logró excelentes resultados, pero un consumo de recursos insostenible por más de 20 episodios, mientras que FQI con Árbol de Decisión logró apenas un éxito en 300 iteraciones.
- Otro factor muy importante es la estabilidad del entrenamiento y consistencia en los resultados, dado que, aunque DQN podría parecer una muy buena aproximación al lograr 7 éxitos, al cambiar las condiciones del entorno o inicialización los resultados eran muy cambiantes y deficientes.

En general, el aprendizaje por refuerzo requiere un trade – off entre precisión, robustez, complejidad y eficiencia computacional, por tanto, la selección del modelo supervisado debe obedecer a pruebas previas, como fue en este caso, el XGBoost reportó excelentes resultados aproximando la función de valor, lo cual se evidenció a al hora de entrenar el agente.

Referencias

Arranz, R., Echeverría, L., Del Caño, J., Ponce, F., & Romero, J. (2019). *Aprendizaje por refuerzo profundo aplicado a juegos sencillos.* Madrid: Universidad Complutense de Madrid.

AWS. (s.f.). *Amazon Web Services*. Obtenido de ¿Qué es el aprendizaje mediante refuerzo?: https://aws.amazon.com/es/what-is/reinforcement-learning/#:~:text=El%20aprendizaje%20por%20refuerzo%20(RL,utilizan%20para%20lograr%20sus%20objetivos

CyberLimpha. (21 de Febrero de 2022). *Medium*. Obtenido de Recurrent Neural Networks in Reinforcement Learning: https://medium.com/@cyberlympha/recurrent-neural-networks-in-reinforcement-learning-11600819ede4

Desmarais, A. (2024). Un gran paso adelante: ¿Por qué es tan difícil aterrizar con éxito en la Luna? *EuroNews*.

Gaeta, M. (2016). *Fitted Q-iteration by Functional Networks for control problems.* Applied Mathematical Modelling.

IBM. (2025). *IBM*. Obtenido de ¿Qué es un árbol de decisión?: https://www.ibm.com/es-es/think/topics/decision-trees

IBM. (s.f.). *IBM*. Obtenido de ¿Qué es XGBoost?: https://www.ibm.com/mx-es/think/topics/xgboost

InteractiveChaos. (s.f.). *InteractiveChaos*. Obtenido de Ventajas y desventajas: https://interactivechaos.com/es/manual/tutorial-de-machine-learning/ventajas-y-desventajas-0

Klimov, O. (s.f.). *Gymnasium Documentation*. Obtenido de Lunar Lander: https://gymnasium.farama.org/environments/box2d/lunar_lander/

Razzaghi, P., Chen, S., Bregeon, A., Tabrizian, A., & Taye, A. (2024). *A Survey on Reinforcement Learning in Aviation Applications.* Estados Unidos: Arxiv.

Sanz, M. (3 de Abril de 2020). *Medium*. Obtenido de Introducción al aprendizaje por refuerzo. Parte 3: Q-Learning con redes neuronales, algoritmo DQN.: https://markelsanz14.medium.com/introducci%C3%B3n-al-aprendizaje-por-refuerzo-parte-3-q-learning-con-redes-neuronales-algoritmo-dqn-bfe02b37017f

XGBoost Developers. (2022). *DMLC XGBoost*. Obtenido de Python API Reference: https://xgboost.readthedocs.io/en/stable/python/python_api.html#xgboost.XGBRegressor

Diseño de un Agente Inteligente en un Entorno Simulado

Anexos

Tablas.

Tabla 1. *Hiperparámetros del Árbol de Decisión*

Hiperparámetro	Valor	Descripción
max_depth	14	Permite capturar relaciones complejas y evitar sobreajuste
min_sample_split	27	Número muestras mínimo para split, mejora la generalización
min_samples_leaf	15	Permite que se capten patrones específicos sin ajustar el ruido
max_feature	log2	Características para dividir un nodo, permite reducir la varianza

Tabla 2. *Hiperparámetros de XGBoost*

Hiper Parámetro	Valor	Justificación
n_estimators	300	Suficientes árboles para aprendizaje estable
max_depth	6	Profundidad a la que va a llegar el árbol
learning_rate	0.1	Estándar para un aprendizaje gradual
subsamble	0.8	Reduce el sobreajuste a remuestrear los datos
reg_alpha	0.1	Regularización L1 para ayudar con ruido
reg_lambda	0.1	Regularización L2 para estabilizar el entrenamiento

Tabla 3. *Hiperparámetros de RL*

Hiperparámetro	DQN	FQI 1	FQI 2
Épsilon	Épsilon - Greedy (Decay = 0.995)	0.995	0.995
Gamma	0.99	0.6	0.9
Iteracion	300	300	20

Figuras.

Figura 1. *Arquitectura de la Red Neuronal*

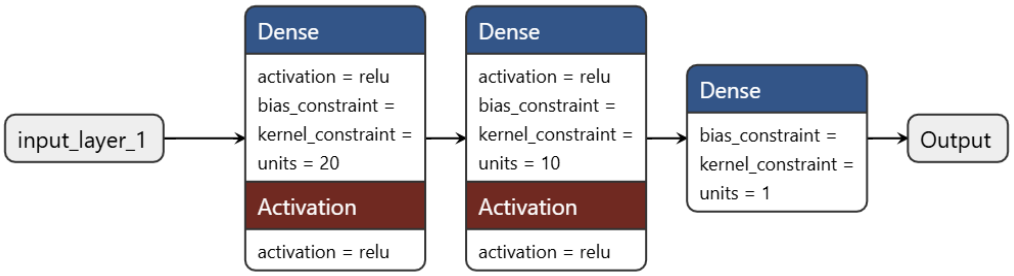

Figura 2. *Arquitectura de la Red Neuronal para DQN*

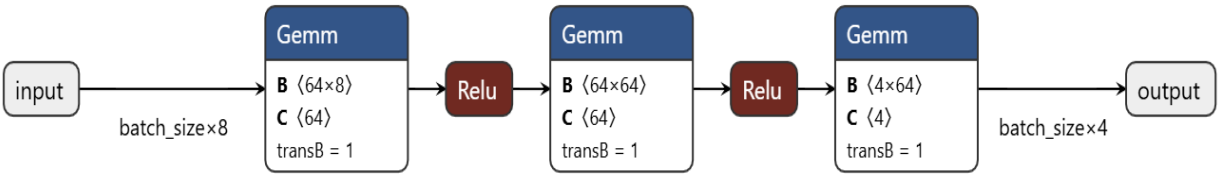

Figura 3. *Predicción vs. Recompensa Real del Árbol de Decisión*

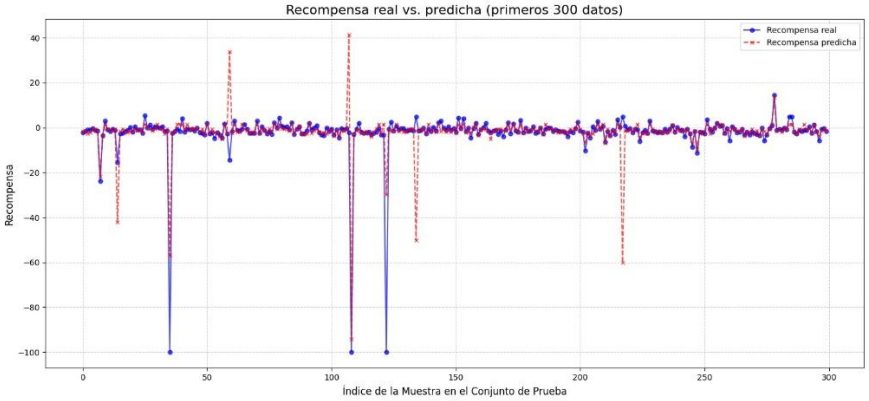

Figura 4. *Predicción vs. Recompensa Real del XGBoost*

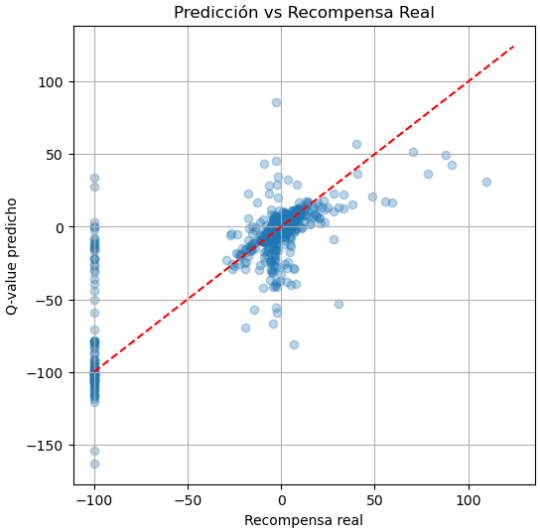

Figura 5. *Pérdida en Entrenamiento y Prueba de la Red Neuronal*