

MINESWEEPER

JUN WOO LEE, 20385
ROBERTO RIOS, 20979

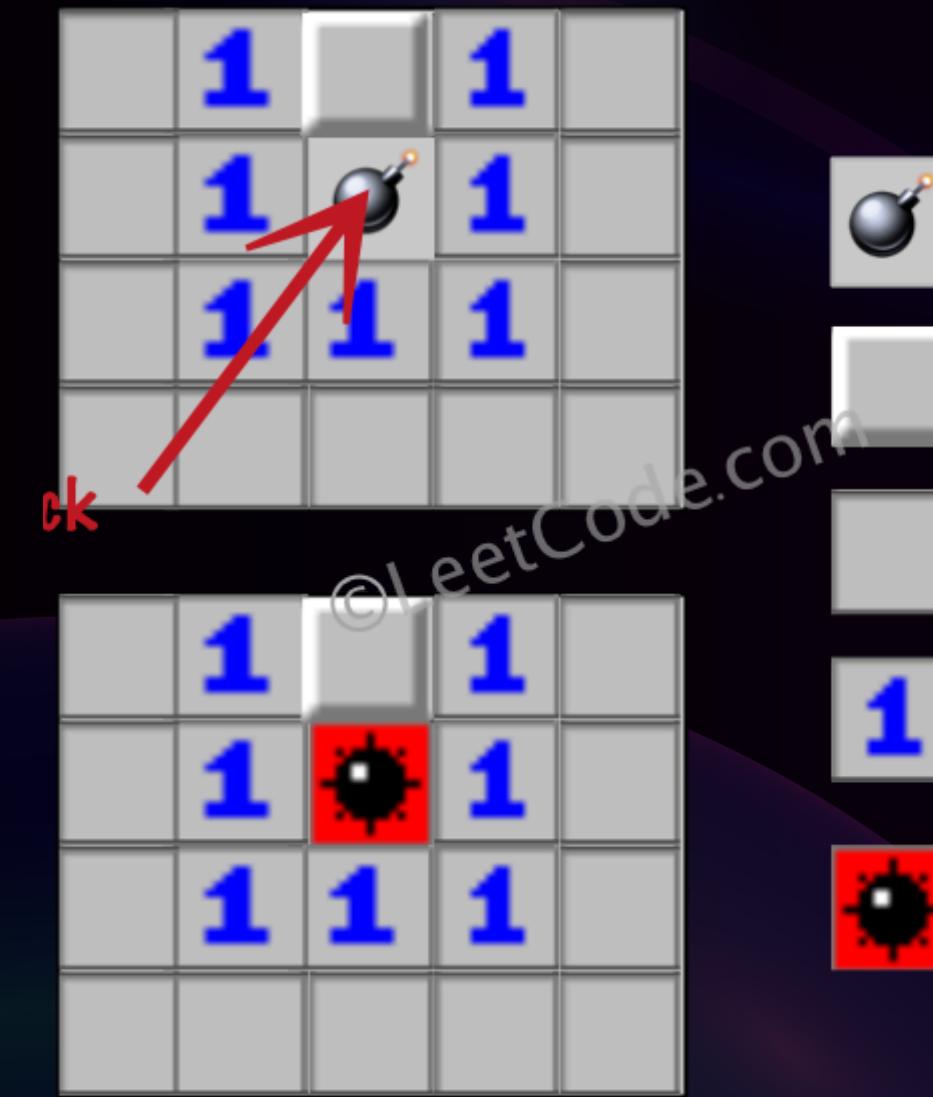


DESCRIPCION DEL PROBLEMA

El proyecto busca diseñar y entrenar un modelo de RL para jugar Minesweeper, un juego que requiere logica y deducion, de manerra autónoma y efectiva.



ANALISIS



El proyecto enfrenta la deducción imperfecta en Minesweeper, requiriendo estrategias de exploración y explotación, y un buen sistema de recompensas para que el agente tome decisiones efectivas con tres posibles acciones.

PROPUESTA DE SOLUCIÓN



 PyTorch

Solución 1: from scratch

Para desarrollar un agente que juegue Minesweeper, se diseñó un entorno en "minesweeper_env.py" con un tablero 5x5 y funciones como reset(), step(action) y render(). El agente usa un modelo de Deep-Q Network para calcular valores Q y tomar decisiones basadas en exploración y explotación, balanceadas con un enfoque epsilon-greedy.

Solución 2: gymnasium

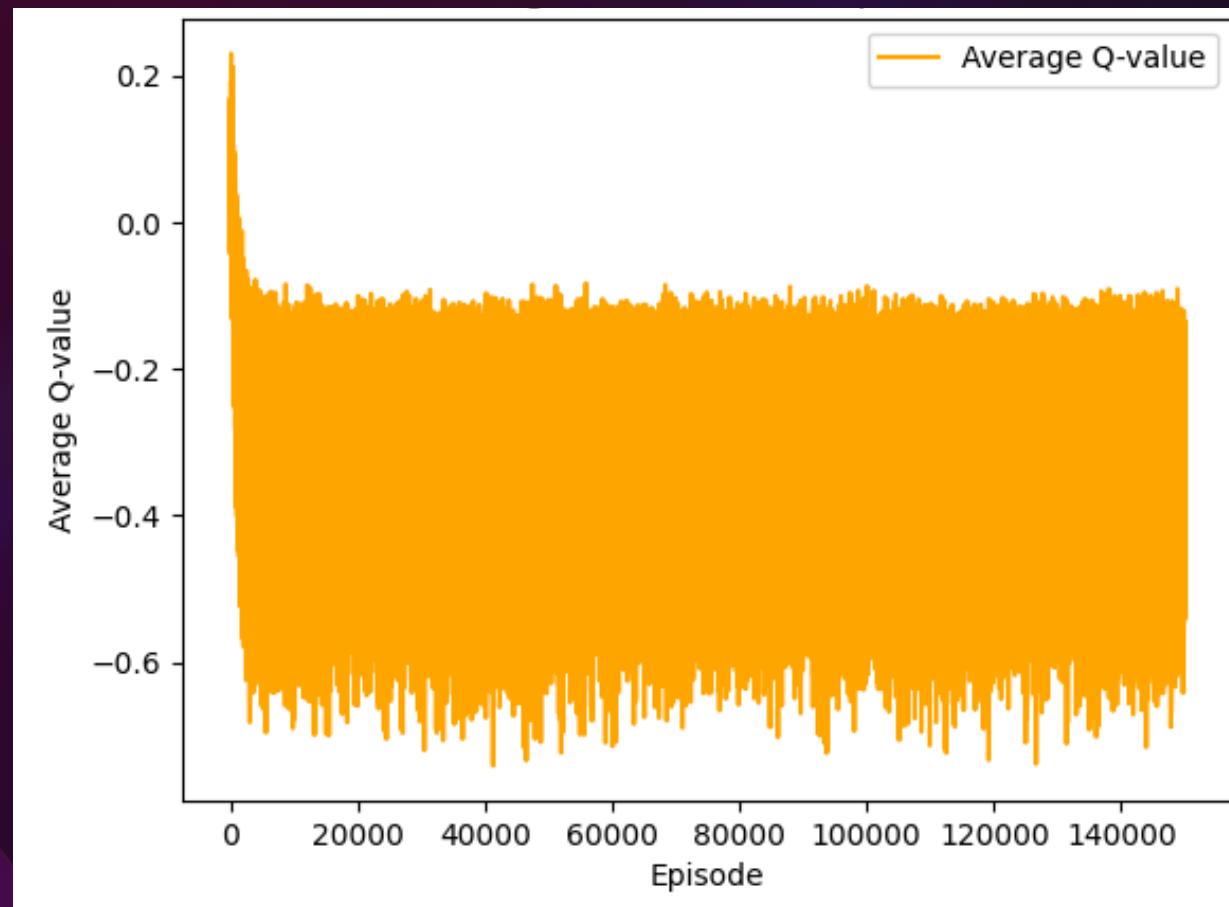
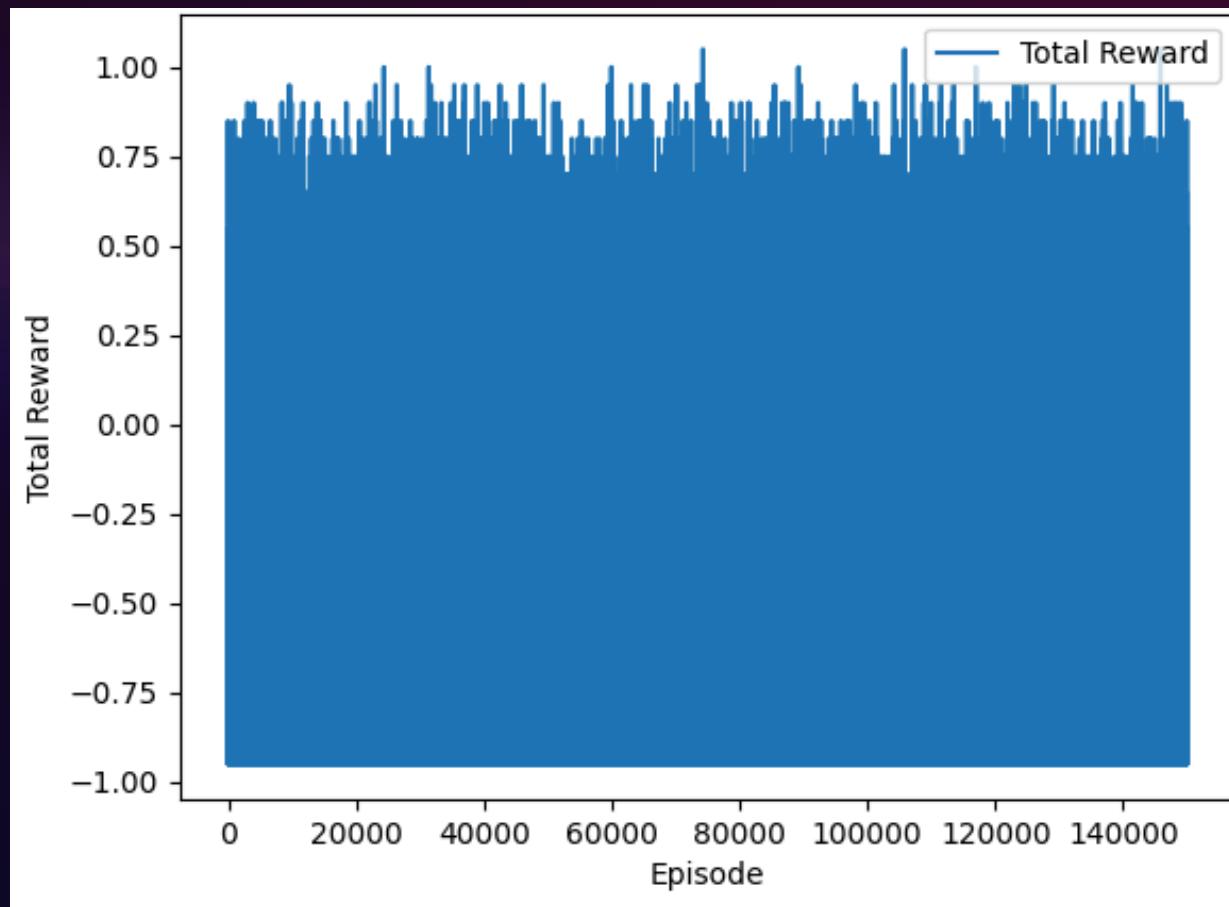
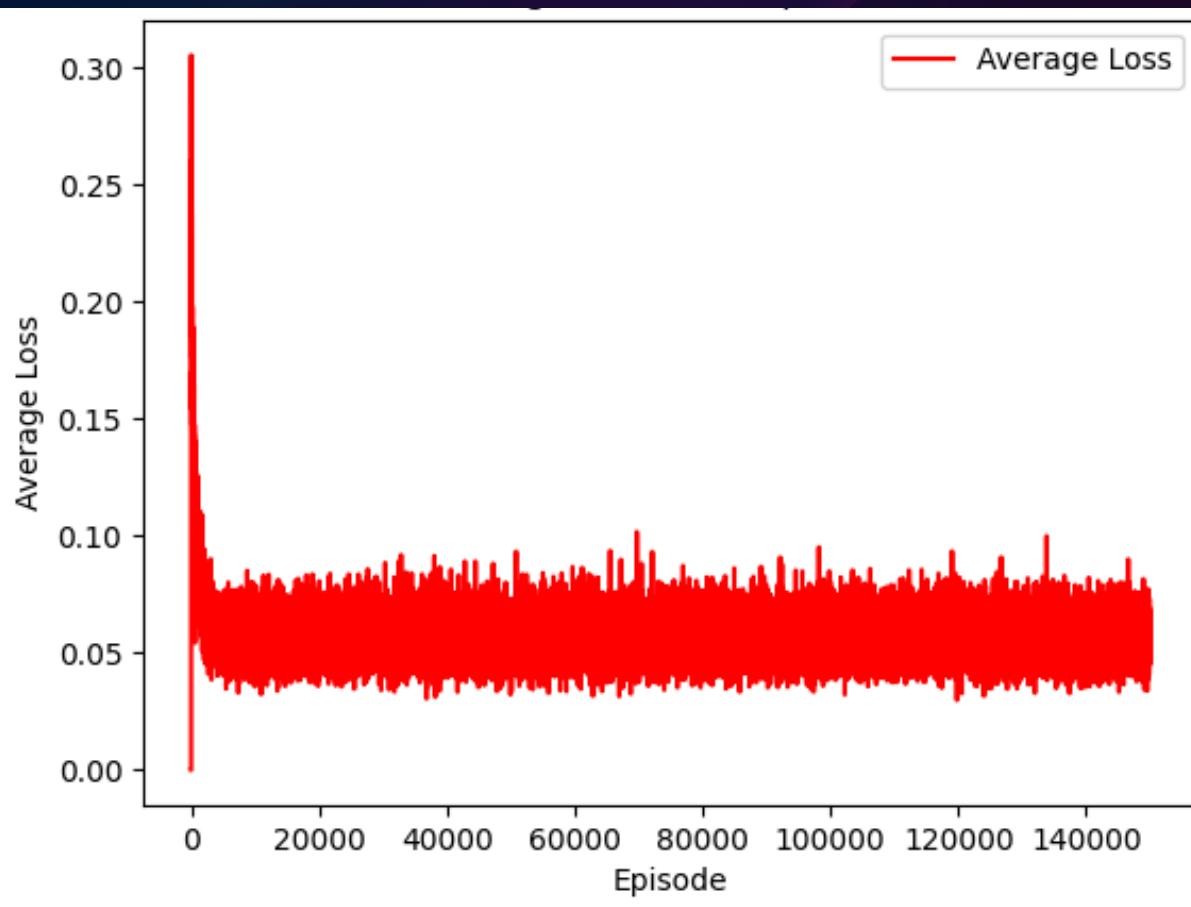
La segunda implementación usó Gymnasium de OpenAI para el entrenamiento, con un entorno en "minesweeper_env.py" y recompensas específicas. El agente utiliza el modelo PPO de "stable_baselines3" con la política "MlpPolicy" para entrenarse como una red Actor-Crílico.



OpenAI Gym



RESULTADOS FROM SCRATCH



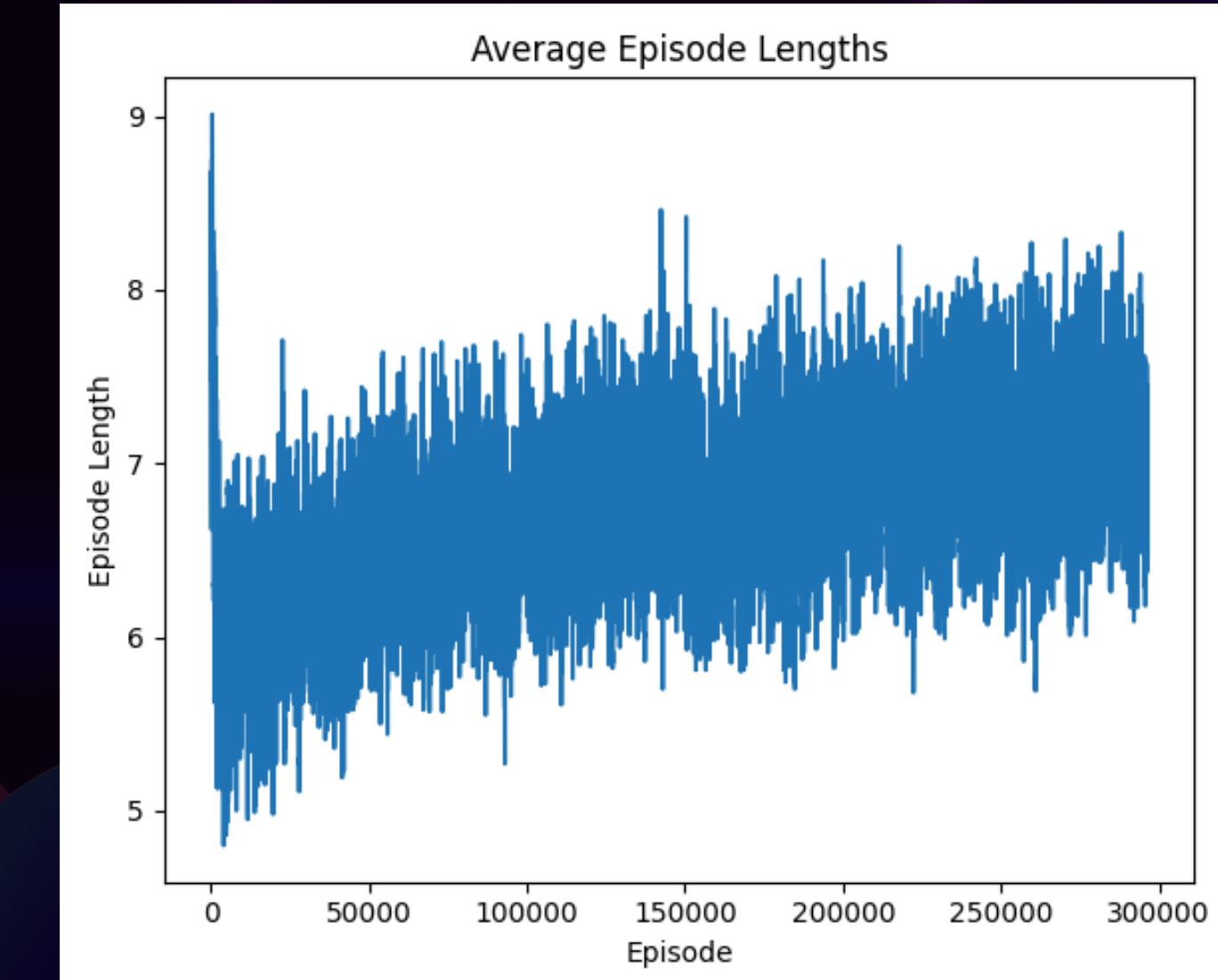
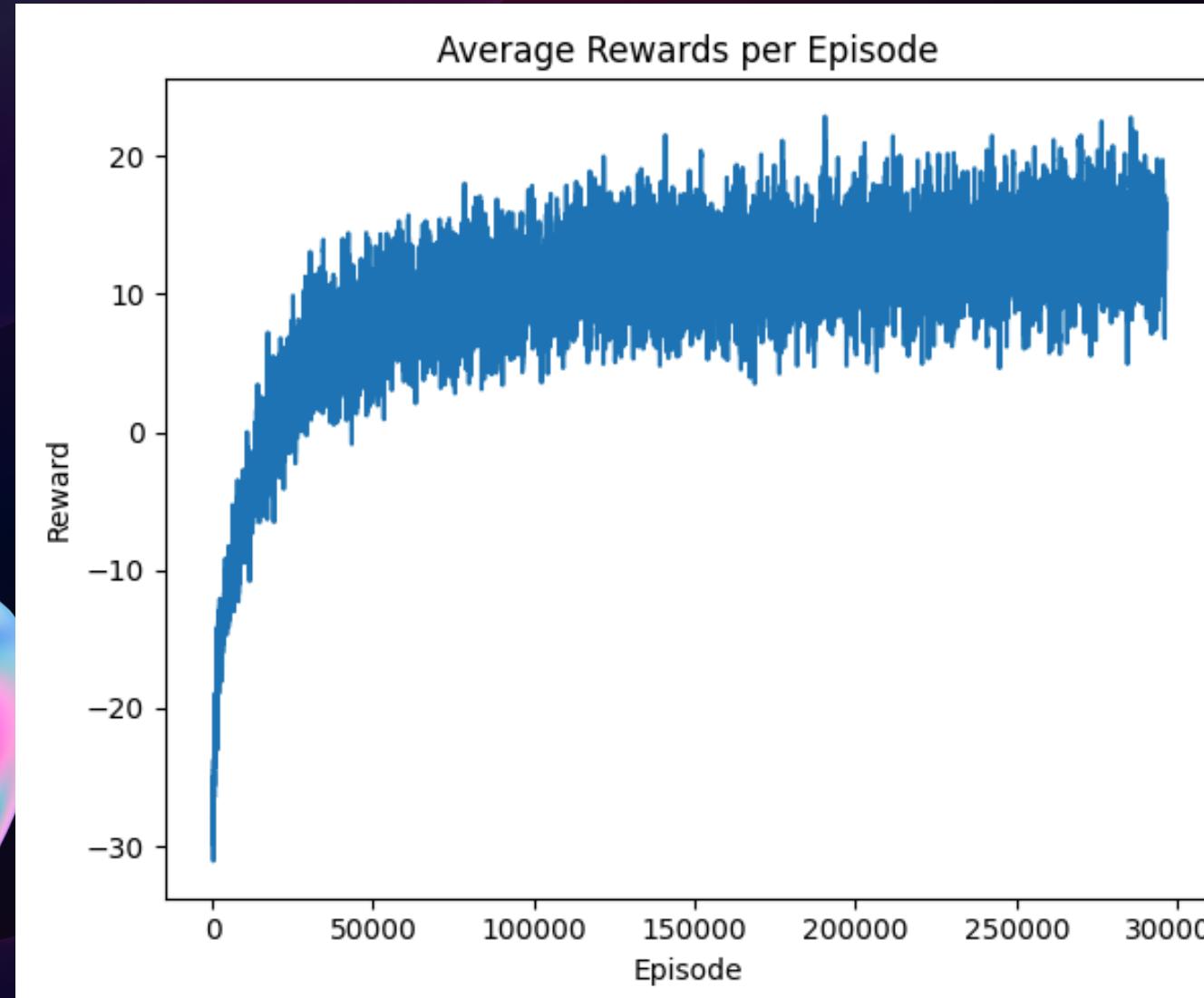
RESULTADOS FROM SCRATCH



Tamaño del Tablero	Número de Minas	Densidad de Minas	Juegos Simulados	Tasa de Victoria	Observaciones
5x5	3	12%	100	0%	El modelo no logró aprender una estrategia efectiva.
5x5	2	8%	100	20%	La baja densidad de minas facilitó el juego, pero el rendimiento sigue siendo bajo.



RESULTADOS GYMNASYUM



RESULTADOS GYMNASYUM

Tamaño del Tablero	Número de Minas	Densidad de Minas	Juegos Simulados	Tasa de Victoria
5x5	3	12%	100	50% - 60%
5x5	4	16%	100	20% - 25%



CONCLUSIONES

Limitaciones

La implementación de DQN mostró limitaciones: el agente no logró aprender una estrategia efectiva, con recompensas constantes y baja tasa de éxito en pruebas, posiblemente debido a restricciones en el tamaño del tablero.

Gym vs PPO

La implementación con Gymnasium y PPO permitió al agente aprender y adaptarse eficazmente, alcanzando tasas de victoria del 50-60% en un tablero 5x5 con 3 minas y manteniendo un 20-25% con 4 minas.

Mejoras

El uso de Gymnasium y PPO mejoró el desempeño del agente, permitiéndole aprender y enfrentar mayores densidades de minas, mientras que el DQN desde cero mostró limitaciones en adaptación y exploración.



BIBLIOGRAFIA



Gym Documentation. (2022). Spaces. Gym Library.
<https://www.gymlibrary.dev/api/spaces/#discrete>

Paszke, A., & Towers, M. (2023). Reinforcement Learning (DQN) Tutorial.
PyTorch.
https://pytorch.org/tutorials/intermediate/reinforcement_q_learning.html

PyTorch. (n.d.). Reinforcement Learning (PPO) with TorchRL Tutorial.
https://pytorch.org/tutorials/intermediate/reinforcement_ppo.html

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J., & Zaremba, W. (2016). OpenAI Gym. arXiv preprint arXiv:1606.01540.
<https://arxiv.org/abs/1606.01540>

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., & Klimov, O. (2017). Proximal Policy Optimization Algorithms. arXiv preprint arXiv:1707.06347.
<https://arxiv.org/abs/1707.06347>

