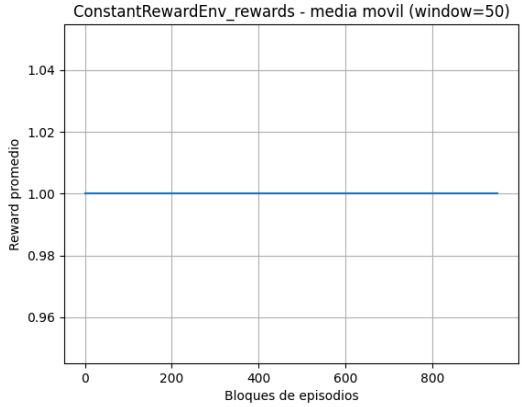
TP 2: Q-learning y DQN

Vigo Malusardi, Maria Mora

Codigo en: https://github.com/MoraVigoMalusardi/DQN.git

1. Se incluyen las curvas de recompensa media por episodio y las tablas de valores Q(s,a) aprendidas para cada uno de los tres entornos:

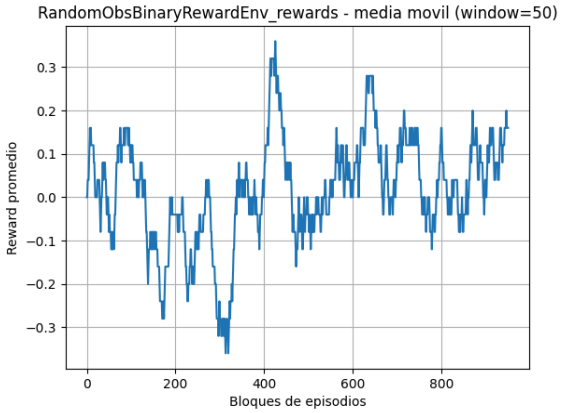
a. ConstantRewardEnv




```
Tabla de valores Q para ConstantRewardEnv:
     A0
S0  1.0
```

Figura 1: Curva de recompensa media por episodio y tabla de valores Q(s, a) aprendida para el entorno ConstantRewardEnv sobre q-learning

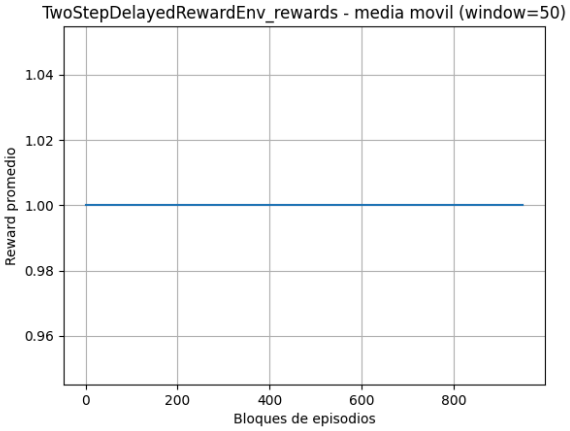
b. RandomObsBinaryRewardEnv




```
Tabla de valores Q para RandomObsBinaryRewardEnv:
     A0
S0 -1.0
S1  1.0
```

Figura 2: Curva de recompensa media por episodio y tabla de valores Q(s, a) aprendida para el entorno RandomObsBinaryRewardEnv sobre q-learning

c. TwoStepDelayedRewardEnv




```
Tabla de valores Q para TwoStepDelayedRewardEnv:
      A0
S0  0.99
S1  1.00
```

Figura 3: Curva de recompensa media por episodio y tabla de valores Q(s, a) aprendida para el entorno TwoStepDelayedRewardEnv sobre q-learning

2. ¿Qué ventajas y desventajas observó entre el método de Policy Gradient (REINFORCE) y DQN? Considerar aspectos como: estabilidad, velocidad de convergencia, sensibilidad a hiperparámetros, variabilidad de resultados, uso de memoria, etc.

3. ¿Cuál de los dos métodos logró una mejor performance en CartPole-v1 en sus experimentos? Justifique en base a las curvas de recompensa y comportamiento observado. Incluya los gráficos de la sección 4.

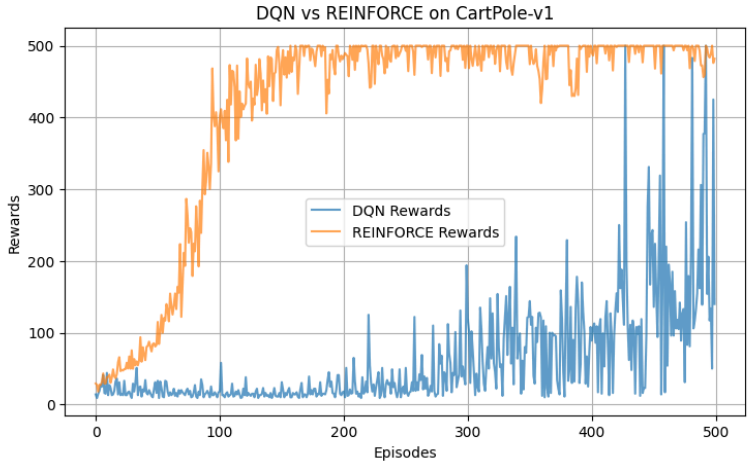


Figura 4: Curvas de recompensa por episodio para un agente entrenado con el algoritmo DQN y para un agente entrenado con el algoritmo REINFORCE, ambos sobre el entorno CartPole-v1

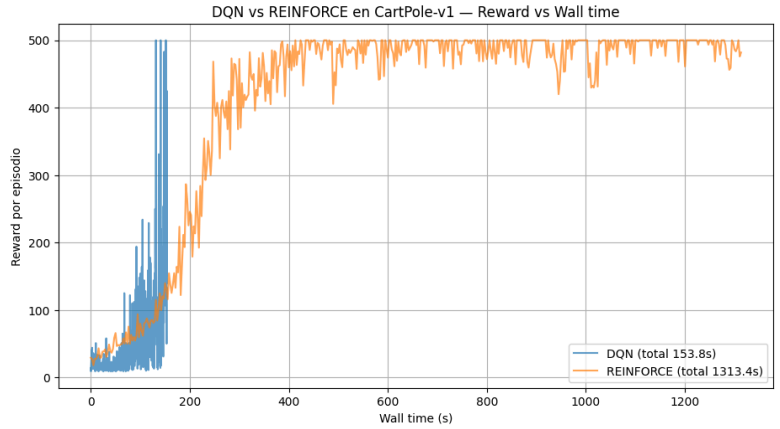


Figura 5: Curvas de recompensa por episodio vs. wall time para un agente entrenado con el algoritmo DQN y para un agente entrenado con el algoritmo REINFORCE, ambos sobre el entorno CartPole-v1

El máximo de recompensa de CartPole es 500, que es cuando se logra +1 punto por cada instante en el que el palo está parado, donde 500 además es el largo máximo del episodio (por ende, el agente logró mantener el palo parado durante todo el episodio).

En este caso, se puede observar de la figura 4 que el agente entrenado con el algoritmo REINFORCE supera significativamente en performance al agente entrenado con DQN. No solo logra efectivamente alcanzar el reward máximo posible, sino que además comienza a mejorar las recompensas obtenidas mucho más rápido que el otro agente. No obstante, el tiempo de entrenamiento del agente de DQN fue significativamente menor al del agente de reinforce.

Sin embargo, la performance del agente de DQN me resulta extraña y entiendo que puede haber un error de implementación. Intenté debuggear mi código en busca de un error, pero no tuve éxito y este fue el resultado final sobre el entorno de CartPole.

4. Para el caso de ALE/Breakout-v5: ¿Qué impacto tuvo el uso del replay buffer y la target network en DQN? ¿Qué ocurre si se omite alguno de estos mecanismos? Experimente y justifique.

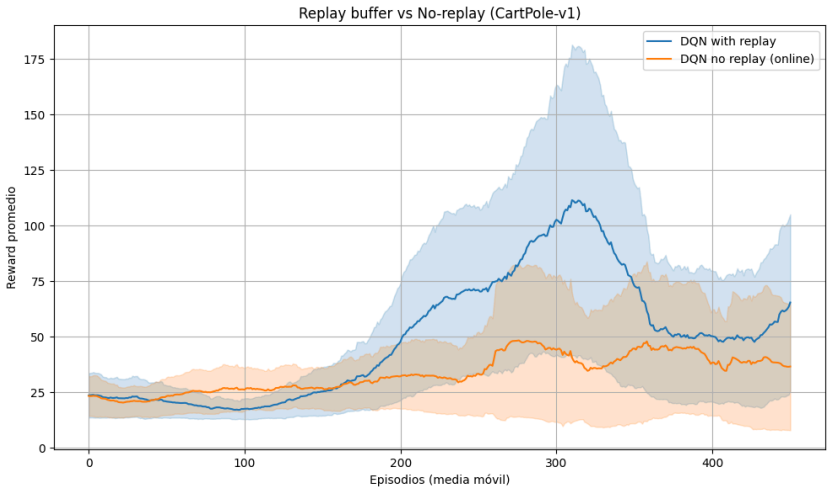


Figura 5: Curvas de recompensa promedio por episodio, sobre 3 ejecuciones con distintas seeds, para un agente entrenado con el algoritmo DQN con replay buffer y un agente sin replay buffer.

Se puede ver de la figura 5 que al utilizar replay buffer la recompensa sube más rápido y se alcanzan rewards promedio mayores. Esto es porque el replay buffer ayuda a mitigar el problema de correlación de las muestras.

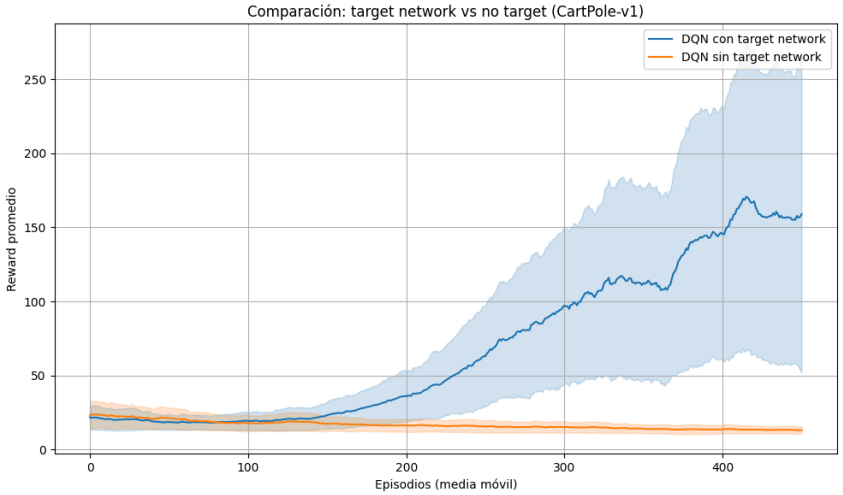


Figura 6: Curvas de recompensa promedio por episodio, sobre 3 ejecuciones con distintas seeds, para un agente entrenado con el algoritmo DQN con target network y un agente sin target network.

Se puede ver de la figura 6 que en el caso del agente sin target network, el aprendizaje está estancado y no logra mejorar el retorno promedio por episodio a lo largo de las ejecuciones. Por el contrario, en

el caso del agente que si utiliza una target network, se puede ver como la curva de reward promedio va en aumento con tendencia creciente a lo largo de las ejecuciones. Esto evidencia los beneficios que tiene el utilizar una red objetivo. Usar una target network permite mitigar el problema de 'moving target' durante el entrenamiento.

5. ¿Qué dificultades encontró al implementar y entrenar cada uno de los algoritmos? Pueden referirse a temas como: debugging, elección de hiperparámetros, uso de TensorBoard, etc.

Al comienzo tuve errores por la implementación de los entornos de prueba, para los cuales use como base los entornos hechos para el trabajo 1, en donde las observaciones eran del tipo 'Box' (continuas), y eran incompatibles con mis implementaciones de este trabajo.

Me costó conceptualmente entender cómo adaptar los datos para utilizar MinAtar. Para esto tuve que implementar un wrapper (SimpleFrameStack), que además de apilar los últimos 4 frames, transpone las dimensiones de cada observación y las normaliza. Además, agregue una verificación en la implementación de DQN para distinguir entre los datos de entrada y usar como arquitectura un MLP o una CNN según corresponda. Los resultados obtenidos sobre este entorno fueron los siguientes:

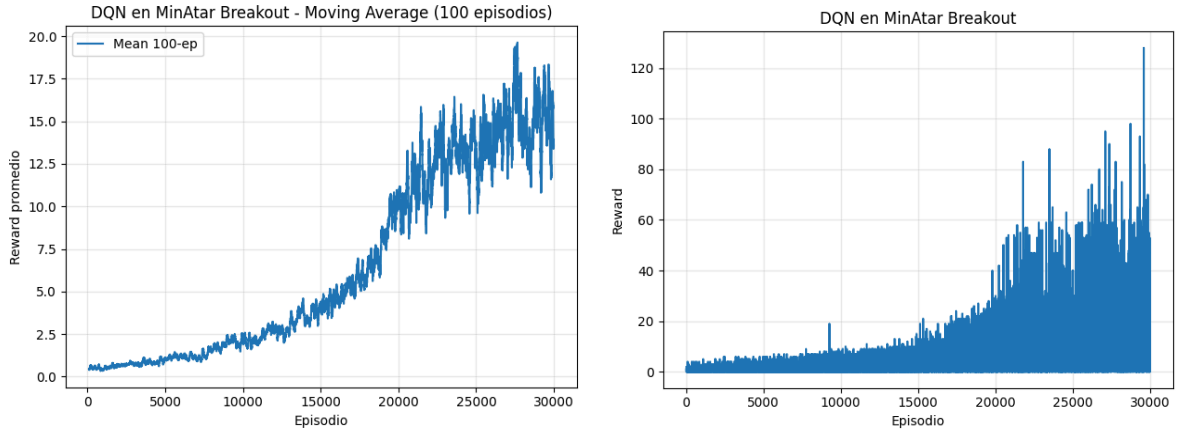
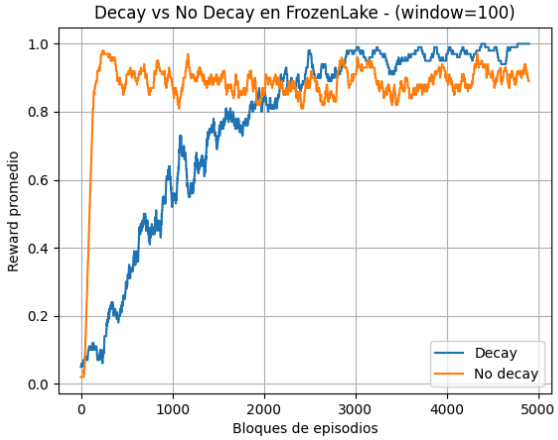


Figura 7: Curvas de recompensa promedio por episodio con moving average y sin moving average para un agente entrenado con DQN sobre MinAtar Breakout, durante treinta mil episodios.

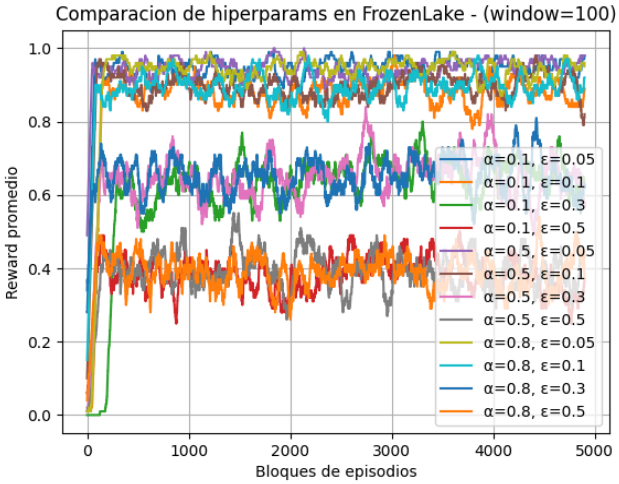
Otra dificultad que tuve fue el debugging de DQN para el caso de CartPole-v1, en el cual no logre obtener buenos resultados y no logré descubrir por qué.

<u>Otros gráficos pedidos</u>

- **Experimento: comparación de utilización de decay vs. no decay en epsilon**



- **Experimento: tuneo de hiperparametros para el agente de Q-learning sobre FrozenLake**



De este análisis pude concluir que el algoritmo, al menos para el caso del entorno estudiado, no es tan sensible a cambios en α (learning rate), pero si es muy sensible a cambios en los valores de ε (probabilidad de tomar una acción aleatoria en lugar de la óptima). En este caso, los valores de epsilon mas chicos (por lo tanto, menor exploración) lograron mejores resultados.