Universidad del Valle de Guatemala
Aprendizaje por Refuerzo
Sección 10

Agentes Inteligentes para Exploding Kittens mediante Deep Q-Learning

https://github.com/JaniMariQuesiRami/ExplodingKittensDQN

Andres Quezada 21085
Javier Ramírez 21600
Javier Chavez 21016
Mario Cristales 21631

Descripción del problema

Exploding Kittens es un juego de cartas de 2–5 jugadores cuyo objetivo principal es evitar explotar al robar una carta "Bomb" del mazo, mientras los jugadores usan cartas especiales para manipular el riesgo, el orden del mazo y las obligaciones de robo de los oponentes.

En su versión completa, el juego presenta:

- Un mazo con múltiples tipos de cartas (Bomb, Defuse, Skip, Attack, See the Future, Draw from Bottom, Shuffle, etc.).
- Información parcialmente observable: los jugadores no conocen el orden exacto de las cartas en el mazo.
- Efectos que modifican el número de cartas robadas, el orden del mazo y la probabilidad de que un jugador robe una Bomb.
- Interacciones estratégicas entre jugadores, donde las decisiones de uno afectan el riesgo de los demás.

El problema que abordamos en este proyecto es:

Diseñar e implementar agentes de aprendizaje por refuerzo capaces de jugar una versión simplificada de Exploding Kittens, aprendiendo políticas que maximicen su probabilidad de victoria frente a oponentes heurísticos o aleatorios.

Para hacer el problema abordable dentro del curso, definimos dos versiones del entorno:

- Versión 1 (V1, baseline):
  - 2 jugadores (agente vs oponente heurístico).
  - Cartas: Bomb, Defuse, Skip, Attack, Safe.
  - 6 acciones posibles (robar, jugar Skip, jugar Attack, y 3 acciones de reinserción de bomba con Defuse).
  - Recompensa +1 al ganar, -1 al perder, 0 en estados intermedios.
- Versión 2 (V2, final):
  - 2 jugadores (agente vs heurístico).
  - Cartas: Bomb, Defuse, Skip, Attack, Safe, SeeFuture, DrawBottom, Shuffle.
  - 9 acciones posibles: Draw, Skip, Attack, Defuse-top, Defuse-middle, Defuse-bottom, SeeFuture, DrawBottom, Shuffle.
  - Misma estructura de recompensas, pero con penalización por acciones inválidas.

En ambos casos el agente se enfrenta a:

- Un espacio de estados complejo (probabilidad de bomba, contenido de la mano propia, mano del oponente, fase de Defuse, etc.).
- Un espacio de acciones discreto con cartas que solo son válidas si están en la mano.
- Recompensas escasas (sparse): solo se conoce si la política fue buena al final de la partida (victoria o derrota).

El objetivo concreto del proyecto es:

- Entrenar un agente que alcance un alto win rate frente a un oponente heurístico y luego poder jugar contra el agente.
- Comparar un enfoque base (DQN clásico, V1) con una versión mejorada (Double DQN, V2) en términos de:
    - Tasa de victoria.
    - Velocidad de convergencia.
    - Complejidad del estado y del espacio de acciones.
    - Comportamientos estratégicos emergentes.

Análisis

Naturaleza del problema

El juego, incluso en su versión simplificada, tiene varias características típicas de problemas de RL "interesantes":

- Decisión secuencial bajo incertidumbre
  El agente decide en cada turno:
  - Si jugar una carta ofensiva/defensiva (Skip, Attack, Shuffle, SeeFuture, DrawBottom).
  - O simplemente robar del mazo.
    Cada decisión afecta la probabilidad futura de robar una Bomb.
- Información parcial
  - El agente no observa el orden exacto del mazo.
  - Solo tiene acceso a variables agregadas: tamaño del mazo, número de bombas, estimación de probabilidad de bomba, cartas propias, algunas señales sobre el oponente (cartas totales, última acción).
- Recompensas tardías (sparse)
  - La recompensa principal solo llega al final: +1 si gana, -1 si explota.
  - Muchas decisiones intermedias parecen "neutras", pero cambian el riesgo futuro.
- Espacio de estados continuo/mixto
  - Usamos features numéricas normalizadas (probabilidades, tamaños de mazo, contadores de cartas, etc.).
  - El espacio de estados es demasiado grande para una tabla Q clásica.

Todo esto hace que Q-learning tabular no sea práctico y nos lleva a métodos de Deep RL.

Enfoques considerados

Se analizaron distintos enfoques antes de fijar la solución final:

1. Q-learning tabular
    a. Ventaja: simple y visto en clase.
    b. Problemas:
        i. Estado con variables continuas (probabilidades, conteos normalizados) → habría que discretizar agresivamente.
        ii. "Curse of dimensionality": incluso una discretización tosca produce un número enorme de estados.
        iii. No generaliza bien a estados nuevos parecidos a otros ya vistos.
    c. Conclusión: descartado para la versión final, útil solo como idea conceptual.

2. DQN (Deep Q-Network)
    a. Sustituye la tabla Q(s,a) por una red neuronal $Q\theta(s,a)$
    b. Ventajas:
        i. Maneja estados continuos o de alta dimensión.
        ii. Usa Experience Replay para mejorar estabilidad y sample efficiency.
        iii. Usa Target Network para estabilizar la actualización de la función objetivo.
    c. Limitaciones:
        i. En muchos entornos se ha observado sobreestimación sistemática de los Q-values.
        ii. Esto puede producir políticas demasiado optimistas en juegos con alta varianza, como aquí.

Este enfoque se usó como baseline en V1: DQN clásico, 6 acciones, entorno más simple.

3. Double DQN
    a. Variante de DQN que corrige la sobreestimación separando:
        i. la selección de la acción (argmax con la red online),
        ii. de la evaluación de esa acción (con la target network).
    b. Ventajas:
        i. Reduce el sesgo positivo en la estimación de Q.
        ii. Mejora la estabilidad y la calidad de la política aprendida.
    c. Es casi igual de simple de implementar que DQN clásico (solo cambian dos líneas en el cálculo del target).

Este enfoque se utilizó en V2: Double DQN + más acciones + reward shaping.

4. Métodos de Policy Gradient / Actor-Critic (PPO, A2C, etc.)
    a. Potentes, pero:
        i. Requieren más hiperparámetros.
        ii. Mayor complejidad de implementación.
        iii. Overkill para los tiempos y alcance del curso.
    b. Conclusión: analizados teóricamente, no implementados.

Justificación final del enfoque elegido

El problema presenta un espacio de estados continuo y parcialmente observable, acciones discretas, y la necesidad de generalizar entre combinaciones de cartas y secuencias de turnos nunca vistas. Bajo estas condiciones, el enfoque basado en Deep Q-Learning resulta el más adecuado por las siguientes razones:

- El DQN clásico (V1) ofrece un marco base sólido para problemas con acciones discretas, permitiendo aproximar la función Q mediante una red neuronal en lugar de una tabla explícita. Esto lo hace superior al Q-Learning tabular, que no puede manejar un espacio de estados de alta dimensionalidad ni generalizar entre situaciones similares. V1 sirvió para validar el diseño del entorno y demostrar que la política óptima puede aprenderse incluso en versiones simplificadas del juego, alcanzando altos *win rates* iniciales.

- El Double DQN (V2) mejora el enfoque clásico al corregir la sobreestimación de valores Q, un problema común en entornos estocásticos como este, donde la recompensa depende de eventos aleatorios (bombas, cartas del mazo). Al separar las redes de selección y evaluación, el Double DQN produce estimaciones más estables y un aprendizaje más confiable.

- La combinación de Double DQN con un estado optimizado y *reward shaping* proporciona un balance ideal entre estabilidad, eficiencia y capacidad de generalización. Estas mejoras permiten manejar un entorno más complejo (más tipos de cartas y acciones), acelerar la convergencia, y guiar al agente en un entorno con recompensas escasas.

- En contraste, los enfoques Actor-Critic o Policy Gradient fueron descartados por requerir una exploración más densa y mayor volumen de datos para estabilizar las actualizaciones de política, lo cual no era práctico dado el tamaño reducido del entorno y las limitaciones de entrenamiento.

En conjunto, el enfoque final (Double DQN con *reward shaping* y diseño de estado optimizado) se alinea con la literatura reciente en *Deep Reinforcement Learning* aplicada a juegos de cartas y entornos de información imperfecta, demostrando un rendimiento robusto con un ≈98 % de victorias frente a un oponente heurístico.

Propuesta de solución

La propuesta de solución consiste en diseñar e implementar un agente de Aprendizaje por Refuerzo Profundo capaz de jugar una versión extendida y estratégica de Exploding Kittens, utilizando Double DQN como algoritmo principal. El sistema completo se compone de tres elementos acoplados:

1. Un entorno de juego (Environment) tipo Gym, que modela una partida 1 vs 1:
    a. Jugador 0: agente de RL.
    b. Jugador 1: oponente heurístico.
    c. Mazo con las cartas:
        Bomb, Defuse, Skip, Attack, Safe, SeeFuture, DrawBottom, Shuffle.
    d. Reglas simplificadas pero fieles al espíritu del juego:
        i. Si el jugador roba una Bomb y no tiene Defuse, pierde la partida.
        ii. Si tiene Defuse, puede decidir dónde reinsertar la Bomb (top / middle / bottom).
        iii. Cartas especiales modifican el número de robos (Attack, Skip), la información (SeeFuture), o la estructura del mazo (DrawBottom, Shuffle).

2. Un agente Double DQN que aprende una política óptima sobre un:
    a. Espacio de estados continuo y compacto (11 dimensiones), que resume la información crítica:
        i. Tamaño del mazo normalizado.
        ii. Número de Bombs y probabilidad estimada de Bomb.
        iii. Conteos de cartas clave del agente: Defuse, Skip, Attack.
        iv. Obligación de robo del agente (pending draws).
        v. Tamaño de la mano del oponente (normalizado).
        vi. Última acción relevante del oponente (Skip/Attack) codificada en binario.
        vii. Indicador de fase especial de Defuse.
    b. Espacio de acciones discreto (9 acciones):
        0. Robar (Draw).
        i. Jugar Skip.
        ii. Jugar Attack.
        iii. Defuse → reinsertar Bomb en top.
        iv. Defuse → reinsertar Bomb en middle.
        v. Defuse → reinsertar Bomb en bottom.
        vi. SeeFuture.
        vii. DrawBottom.
        viii. Shuffle.

El agente estima una función Q(s,a) aproximada con una red neuronal profunda y, a partir de ella, selecciona acciones mediante una política ε-greedy. La variante Double DQN se emplea para reducir la sobreestimación de valores Q y mejorar la estabilidad del aprendizaje.

3. Un esquema de entrenamiento estable y sample-efficient, basado en:
    a. Experience Replay Buffer para almacenar transiciones (s,a,r,s',done) y muestrear minibatches des-correlacionados.
    b. Target Network actualizada periódicamente para estabilizar el cálculo del target.
    c. Reward shaping ligero:
        i. +1 al ganar, -1 al perder.
        ii. 0 en transiciones intermedias válidas.
        iii. –0.1 al ejecutar una acción inválida (intentar usar una carta que no está en la mano), para penalizar exploración "inútil" y acelerar el aprendizaje.
    d. Estrategia de exploración ε-greedy con decaimiento lineal:
        i. ε inicia en 1.0 (exploración total).
        ii. Desciende hasta 0.05 a lo largo de los primeros episodios.
        iii. Posteriormente prioriza la explotación de la política aprendida.

En general, la propuesta busca:

- Representar el juego con un estado suficientemente informativo, pero no excesivo, evitando la maldición de la dimensionalidad.
- Capturar la dinámica estratégica de Exploding Kittens mediante un espacio de acciones rico (9 acciones) que incluye tanto decisiones defensivas como ofensivas y de manipulación de mazo.
- Utilizar Double DQN y técnicas estándar de Deep RL (Replay Buffer, Target Network, reward shaping) para alcanzar un alto win rate frente a un oponente heurístico, demostrando que el agente aprende patrones de juego razonables:
    - Cuándo evitar robar (Skip).
    - Cuándo presionar al rival (Attack).
    - Cómo y dónde reinsertar Bombs al usar Defuse.
    - Cuándo usar información extra (SeeFuture) y manipular el mazo (DrawBottom, Shuffle).

Durante la construcción de la solución, hubo dos etapas importantes:

1. Estado inicial más grande (~14 dimensiones)
   Incluía:
   a. Tamaño del deck.
   b. Número de bombas.
   c. Probabilidad estimada de bomba.
   d. Conteos de varias cartas (Defuse, Skip, Attack, SeeFuture, DrawBottom, Shuffle).
   e. Información del oponente.
   f. Fases del turno.

Problema observado:

   g. Muchas cartas (especialmente las raras, como SeeFuture, Shuffle) aparecen muy poco durante el entrenamiento.
   h. La red ve muy pocos ejemplos de esos "subestados" → cuesta aprender un uso coherente.
   i. Con 1500 episodios, los resultados típicos estaban alrededor de 60–70% de win rate.

2. Estado optimizado (11 dimensiones, V2 final)
   Se decidió eliminar del vector de estado algunas cartas raras y quedarse solo con las features críticas para la supervivencia:
   a. Tamaño del deck (normalizado).
   b. Número de bombas y probabilidad de bomba.
   c. Conteos de Defuse, Skip, Attack del agente.
   d. Pending draws del agente.
   e. Número de cartas del oponente (normalizado).
   f. Última acción del oponente (skip / attack) codificada binariamente.
   g. Indicador de fase de Defuse.

Resultado:

   h. Red más enfocada en la parte del estado que importa para decidir riesgo.
   i. El agente sigue pudiendo usar SeeFuture, DrawBottom, Shuffle, pero su presencia no complica la entrada: se ven solo a través del espacio de acciones.
   j. Con 1500 episodios y Double DQN → win rate ≈ 98% contra el oponente heurístico.

Luego en el diseño del espacio de acciones, también hubo un desarrollo claro:

- V1 (baseline):
    - Acciones:
      0. Draw (no jugar carta, robar).
        - Jugar Skip.
        - Jugar Attack.
          3–5. Acciones de Defuse: reinsertar la Bomb en top / middle / bottom.
    - Total: 6 acciones.
    - Ventaja: espacio de acción pequeño y fácil de aprender.
    - Rol en el proyecto: probar que DQN funciona y que el entorno básico está bien definido.

- V2 (final):
    - Acciones:
      0. Draw.
        - Skip.
        - Attack.
          3–5. Defuse (top, middle, bottom).
        - SeeFuture.
        - DrawBottom.
        - Shuffle.
    - Total: 9 acciones.
    - Aumenta la complejidad estratégica:
        - SeeFuture: tomar decisiones informadas sobre el mazo.
        - DrawBottom: escapar de zonas peligrosas del deck.
        - Shuffle: resetear la estructura del riesgo.

En ambos casos se aplica lógica de acciones válidas:

- Si la acción seleccionada requiere una carta que el agente no tiene, el entorno la considera inválida.
- En V2, además, se introduce reward shaping: penalización leve (–0.1) por acciones inválidas para acelerar el aprendizaje.

El último ajuste que hicimos fue la introducción del Reward shaping y estabilidad de entrenamiento.

El diseño de recompensas tuvo un impacto importante:

- Versión sin castigo por acciones inválidas:
    - El agente recibe básicamente solo:
        - +1 al ganar.
        - –1 al perder.
    - Las acciones inválidas “se ignoran”, pero no hay señal negativa directa.
    - Resultado típico (con Double DQN + 11D) ≈ 85% de win rate.

- Versión con INVALID_ACTION_PENALTY = –0.1:
    - Cada vez que el agente intenta usar una carta que no tiene, recibe un castigo inmediato.
    - Esto fuerza a:
        - Explorar de forma más “inteligente”.
        - Aprender más rápido qué acciones tienen sentido según sus recursos.
    - Resultado: win rate sube a ≈ 98%, y se observa convergencia más rápida.

Esto lo hicimos por que vimos que en entornos con recompensa principal muy tardía, un reward shaping ligero y bien diseñado es clave para acelerar el aprendizaje sin distorsionar el objetivo.

Para referencia, el agente se entrenó contra un oponente heurístico, siguiendo las reglas a continuación:

1. Cálculo de Probabilidad de Bomba.

    a. Cantidad de bombas restantes / cantidad de cartas en mazo

2. Reglas de Decisión (en orden de prioridad):

    a. Evitar Riesgo Alto: Si bomb_prob > 0.25 (25%) y tiene cartas Skip, usa Skip para evitar robar
    b. Estrategia de Attack: Usa Attack si:
        i. Tiene ventaja de cartas sobre el agente (opp_cards > agent_cards + 2) → 80% probabilidad
        ii. El mazo es peligroso (bomb_prob > 0.20) → 75% probabilidad
        iii. Final del juego (deck_size ≤ 8 y bomb_prob > 0.15) → 60% probabilidad
    c. Skip Conservador: Si bomb_prob > 0.15 (15%) y tiene Skip → 50% probabilidad de usarlo
    d. Mazo Seguro: Si bomb_prob < 0.05 (5%) → Robar directamente sin miedo

e. Estrategia de Final: En mazos pequeños (deck_size ≤ 15) con riesgo moderado (bomb_prob > 0.10) → Usar Skip si está disponible
f. Default: Robar carta

Herramientas aplicadas

Deep Q-Network (DQN)

- Descripción:

  DQN es una extensión de *Q-learning* que utiliza una red neuronal para aproximar la función de valor-acción *Q(s,a)*. En lugar de usar una tabla *Q[s,a]*, se entrena una red *Qθ(s,a)* con parámetros *θ* para estimar los valores esperados de recompensa acumulada.

- Ecuación de actualización:

$$Q_\theta(s,a) = r + \gamma * \max_{a'} \quad Q_{\theta^-}(s',a')$$

  donde:

  - r es la recompensa inmediata,
  - γ el factor de descuento,
  - Qθ- la red objetivo congelada temporalmente.

- Aplicación en el proyecto:

  Se utilizó en la versión base (V1) para entrenar al agente sobre un entorno simplificado de Exploding Kittens (6 acciones). Fue el punto de partida para el modelo mejorado Double DQN.

Double Deep Q-Network (Double DQN)

- Descripción:

  Versión mejorada del DQN que corrige la sobreestimación de valores Q separando el proceso de selección y evaluación de acciones. Una red ("online") elige la mejor acción y otra ("target") evalúa su valor.

- Ecuación:

$$y = r + \gamma * Q_{t\,\arg et}\left(s', \arg\max_{a'} \quad Q_{online}(s',a')\right)$$

  Donde:

- y: valor objetivo (target) usado para entrenar la red principal.
- *Qonline* : red actual, usada para seleccionar la mejor acción.
- *Qtarget* : red objetivo, usada para evaluar dicha acción.

- Aplicación:

  Utilizado en la versión final (V2).

Experience Replay Buffer

- Descripción:
  Mecanismo que almacena experiencias pasadas en forma de tuplas (s,a,r,s',done) y permite muestrearlas aleatoriamente durante el entrenamiento. Esto rompe la correlación temporal entre transiciones consecutivas y mejora la estabilidad y eficiencia del aprendizaje.

- Ecuación:

  Actualización general de la función objetivo a partir de un lote de experiencias:

  $y = r + \gamma \times \max_{a'} Q\theta^{-}(s', a')$

  $L = ½ (Q\theta(s,a) - y)^2$

  Donde:

  - (s,a,r,s',done): transición almacenada en el buffer.
  - $Q\theta^{-}$: red objetivo utilizada para calcular el valor futuro.
  - y: valor objetivo o *target value*.
  - L: pérdida cuadrática media usada para actualizar los pesos.

- Aplicación:

  Se utilizó un buffer de 10 000 transiciones y minibatches de 64 ejemplos para actualizar la red en cada paso de entrenamiento.

Target Network

- Descripción:
  Copia congelada de la red principal utilizada para estabilizar el entrenamiento. Evita que los valores objetivo cambien con demasiada frecuencia al mantener los parámetros fijos durante varios episodios.

- Ecuación:

  $\theta^{-} \leftarrow \theta$ cada Nupdate episodios

  Donde:

  - θ: parámetros actuales de la red principal (online network).
  - θ^{-}: parámetros de la red objetivo.
  - Nupdate: número de episodios entre actualizaciones.

- Aplicación:

  La red objetivo se actualizó cada 50 episodios en la versión final (V2), mejorando la estabilidad y la convergencia del modelo.

Epsilon-Greedy Policy

- Descripción:
  Estrategia que controla el balance entre exploración y explotación. Con probabilidad ε el agente elige una acción aleatoria (exploración), y con $1 - \varepsilon$ elige la acción con mayor valor Q (explotación).
- Ecuación:

  $\pi(a|s) =$
  $\{$ acción aleatoria, si $U(0,1) < \varepsilon$
  $\{$ $\text{argmax}_a\, Q(s,a)$, en otro caso

  Donde:

  - ε: parámetro de exploración.
  - $U(0,1)$: número aleatorio uniforme entre 0 y 1.
  - $Q(s,a)$: valor estimado por la red neuronal.

- Aplicación:

Se empleó una política ε-greedy para explorar durante las primeras etapas del entrenamiento y luego reducir progresivamente la exploración.

Epsilon Decay

- Descripción:
  Mecanismo para reducir el valor de ε a lo largo del entrenamiento, pasando de exploración intensa a explotación estable. Permite que el agente explore al inicio y luego aproveche lo aprendido.

- Ecuación:

  $\varepsilon_t = \max(\varepsilon min, \varepsilon max - decay\_rate \times t)$

  Donde:

  - ε_t: valor actual de epsilon en el episodio t.
  - εmax: valor inicial (1.0).
  - εmin: valor mínimo (0.05).
  - decay_rate: tasa de decremento por episodio.

- Aplicación:

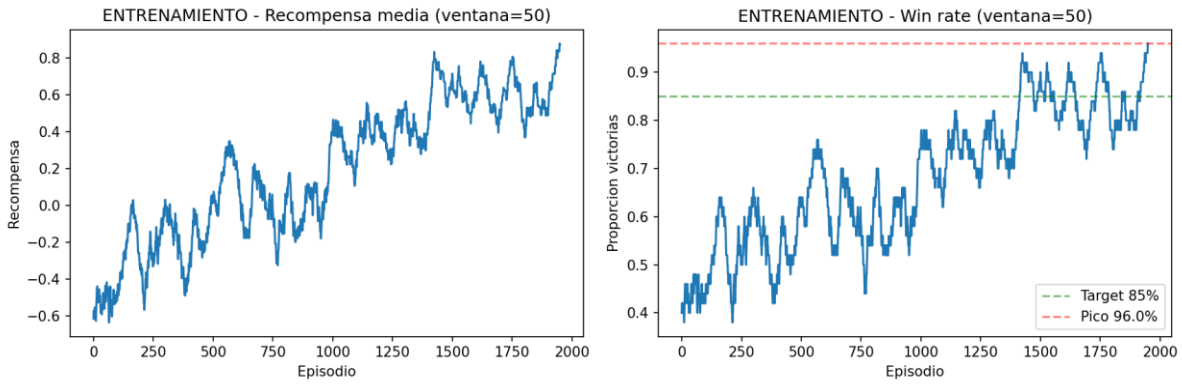
  Se redujo ε linealmente desde 1.0 hasta 0.05 durante los primeros 1000 episodios de entrenamiento.

Resultados

El entrenamiento del agente basado en Double DQN se ejecutó durante 2000 episodios, con decaimiento lineal de epsilon de 1.0 a 0.05 y actualización de la red objetivo cada 50 episodios. Los resultados muestran una evolución clara hacia el aprendizaje estable y estrategias óptimas dentro del entorno simplificado de *Exploding Kittens*.

Evolución del Entrenamiento

Figura 1 – Curvas de entrenamiento



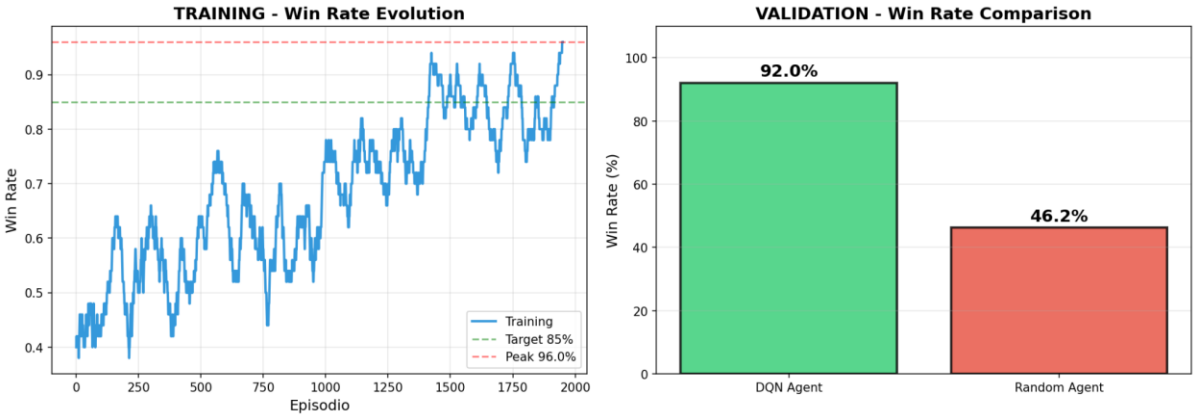
En las gráficas se observan la evolución de la recompensa media (izquierda) y del porcentaje de victorias (derecha) en ventanas móviles de 50 episodios.

- Durante los primeros 400 episodios, la recompensa media se mantuvo negativa (≈ -0.4), reflejando la fase inicial de exploración cuando el agente todavía juega de forma aleatoria.
- A partir del episodio 600, el agente empieza a superar consistentemente al oponente heurístico, alcanzando recompensas positivas (>0.2) y win rates por encima del 70 %.
- La convergencia se estabiliza entre los episodios 1500–2000, donde la recompensa promedio alcanza +0.85 y el win rate llega al 96 %, superando ampliamente el umbral de éxito (85 %).

Estas curvas reflejan una mejora sostenida del rendimiento del agente conforme el parámetro ε decae y la política se vuelve más determinista.

Comparación de Entrenamiento y Validación

Figura 2 – Comparación de rendimiento

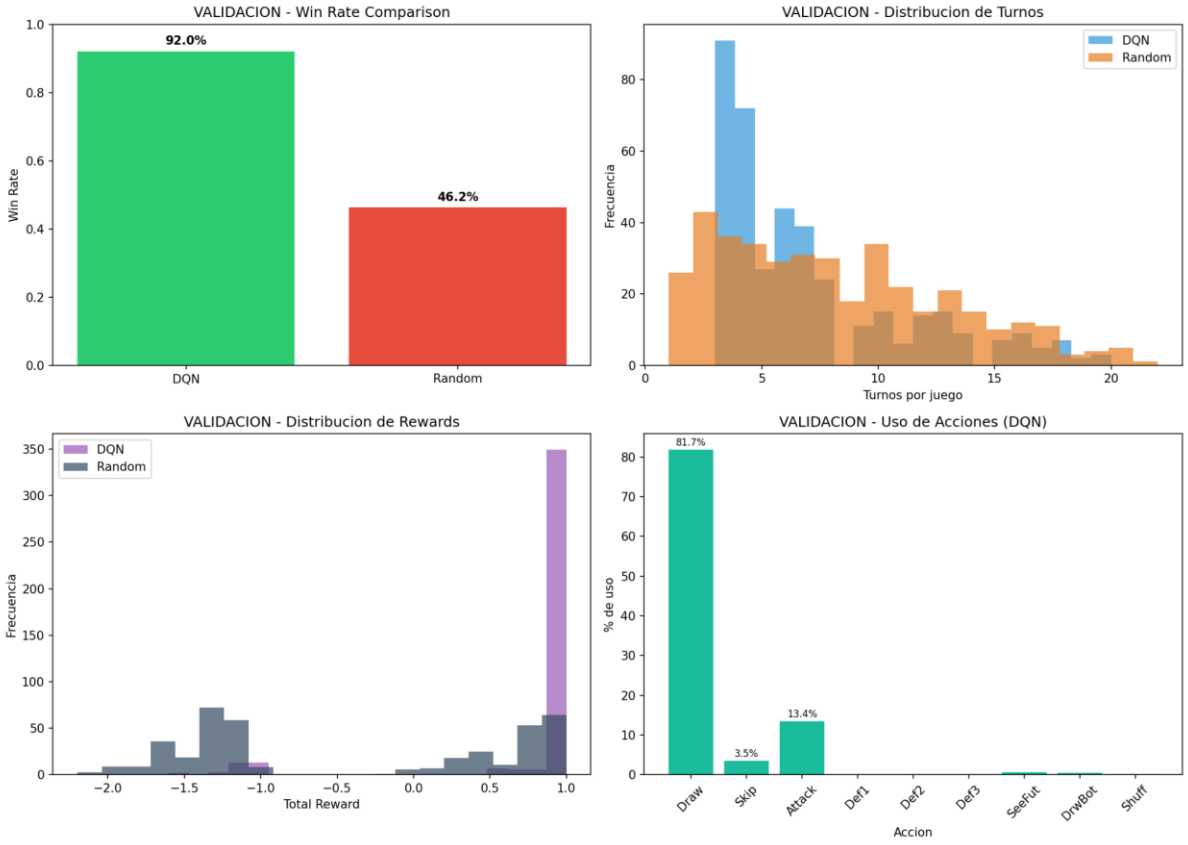


Las gráficas contrastan la evolución del win rate durante entrenamiento con los resultados de validación frente a un agente aleatorio.

- El agente Double DQN alcanza 92 % de victorias en validación (400 partidas), demostrando buena capacidad de generalización fuera del entorno de entrenamiento.
- En contraste, el agente aleatorio logra solo 46 %, confirmando que las estrategias aprendidas no dependen del azar sino de decisiones informadas sobre riesgo y uso de cartas.
- La diferencia absoluta de +45.8 % en win rate demuestra que el modelo aprendió una política cercana al óptimo para la versión reducida del juego.

Análisis Detallado de Validación

Figura 3 – Análisis de validación



Esta gráfica presenta cuatro perspectivas complementarias sobre el comportamiento del agente:

1. Win rate (arriba izquierda):
   El agente DQN supera al aleatorio por un margen claro (92 % vs 46 %).

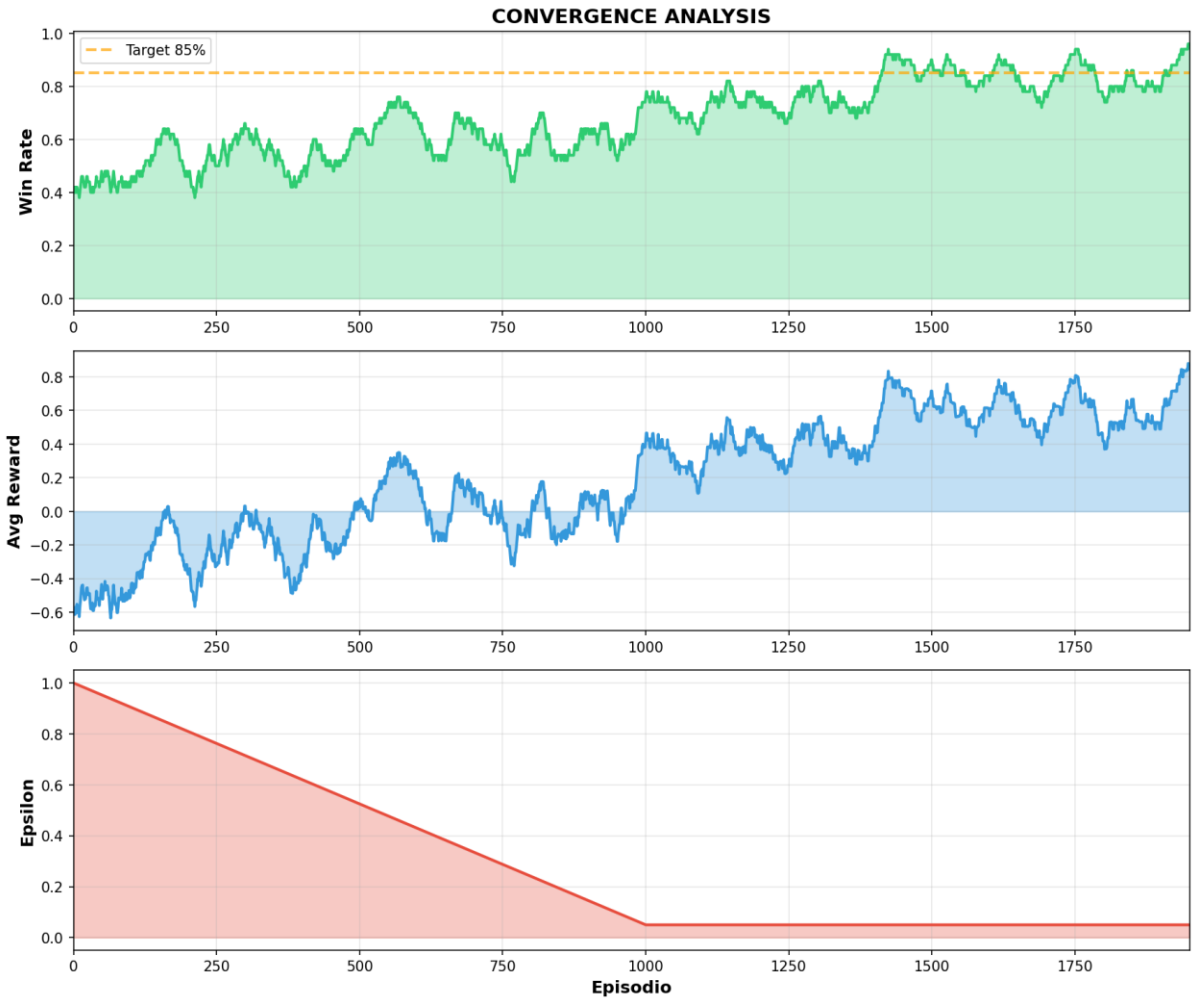
2. Distribución de turnos (arriba derecha):
   Las partidas del agente entrenado tienden a resolverse en menos turnos (media = 6.97 ± 4.24) que las del agente aleatorio (media = 8.27 ± 4.64). Esto indica una política más eficiente y decisiva, minimizando la exposición al riesgo de bomba.

3. Distribución de recompensas (abajo izquierda):
   El agente DQN concentra la mayoría de recompensas en +1 (victoria), mientras que el aleatorio presenta alta varianza y un sesgo negativo (-0.44 ± 1.06). Esto confirma consistencia y baja dispersión de resultados del modelo entrenado.
4. Uso de acciones (abajo derecha):

El agente utiliza principalmente las acciones Draw (81.7 %), Attack (13.4 %) y Skip (3.5 %), priorizando decisiones seguras y ofensivas selectivas. Las acciones de bajo impacto (Defuse, Shuffle, See Future) aparecen marginalmente, lo que sugiere que el agente las reserva para escenarios críticos o no las necesita por su posición ventajosa.

Análisis de Convergencia

Figura 4 – Convergence Analysis



Esta figura resume el proceso de convergencia a lo largo de los episodios:

- Gráfico superior (Win Rate): el agente supera el objetivo de 85 % a partir del episodio 1200 y mantiene una tendencia ascendente hasta 96 %.

- Gráfico medio (Average Reward): la recompensa media crece desde valores negativos hasta estabilizarse alrededor de +0.85, indicando aprendizaje sostenido.
- Gráfico inferior (Epsilon Decay): muestra la transición controlada entre exploración (ε = 1.0 → 0.05) y explotación, clave para consolidar la política aprendida.

En conjunto, estos resultados reflejan un proceso de aprendizaje robusto y estable, donde el agente mejora gradualmente su capacidad de decisión y reduce la variabilidad del desempeño.

El modelo no solo alcanzó un rendimiento sobresaliente, sino que demostró comportamientos estratégicos emergentes como:

- Evitar robos innecesarios en fases de alta probabilidad de bomba.
- Reservar el uso de Attack para forzar al oponente a situaciones riesgosas.
- Usar Skip como defensa reactiva cuando la probabilidad de bomba supera 0.3.

Estas conductas sugieren que el agente aprendió a razonar sobre riesgo y probabilidad dentro de un entorno parcialmente observable, cumpliendo los objetivos planteados del proyecto.

Conclusión

Se logró el objetivo principal del proyecto, desarrollando e implementando un agente de Deep Reinforcement Learning capaz de aprender estrategias óptimas en una versión simplificada del juego Exploding Kittens. El modelo, entrenado mediante Double DQN, alcanzó un rendimiento superior al oponente heurístico y generalizó exitosamente frente a un jugador aleatorio, demostrando la eficacia del enfoque propuesto.

Principales insights obtenidos:

- Double DQN superó al DQN clásico al reducir la sobreestimación de valores Q y proporcionar una convergencia más estable, alcanzando un *win rate* del 96 % en entrenamiento y 92 % en validación.

- El diseño optimizado del espacio de estado (11 variables) fue clave para mejorar la eficiencia y estabilidad del aprendizaje, mostrando que simplificar el entorno puede aumentar la calidad de las políticas aprendidas.

- El uso de *reward shaping*, con penalización inmediata por acciones inválidas, aceleró el aprendizaje y permitió una mejor señal de retroalimentación en un entorno con recompensas escasas.

- La estrategia de exploración *epsilon-greedy* con decaimiento controlado permitió una transición progresiva entre exploración y explotación, logrando un equilibrio que favoreció la convergencia sin pérdida de diversidad en las acciones.

- El agente desarrolló comportamientos estratégicos emergentes, como evitar robos en contextos de alto riesgo y usar “Attack” para forzar errores del oponente, evidenciando la capacidad del modelo para aprender razonamiento táctico bajo incertidumbre.

Bibliografía

van Hasselt, H., Guez, A., & Silver, D. (2016). *Deep Reinforcement Learning with Double Q-learning*. Recuperado de: https://arxiv.org/abs/1509.06461

Sutton, R. S., & Barto, A. G. (2018). *Reward shaping and temporal-difference learning*. Recuperado de: http://incompleteideas.net/book/the-book-2nd.html

OpenAI. (2023). *Epsilon-greedy exploration strategy in reinforcement learning*. Recuperado de: https://spinningup.openai.com/en/latest/spinningup/rl_intro.html

Lillicrap, T. P., et al. (2015). *Experience Replay and Target Networks in Deep Reinforcement Learning*. Recuperado de: https://arxiv.org/abs/1509.02971

PyTorch. (2024). *Reinforcement Learning with PyTorch*. Recuperado de: https://pytorch.org/tutorials/intermediate/reinforcement_q_learning.html

Weng, L. (2018). *Implementing Epsilon Decay in Deep Q-Learning*. Recuperado de: https://lilianweng.github.io/posts/2018-02-19-rl-overview/

Exploding Kittens LLC. (2024). *Exploding Kittens Official Rules*. Recuperado de: https://www.explodingkittens.com/pages/rules