Aplicación de Algoritmos de Reforzamiento para desempeñarse en un juego simple

Escuela de Ingeniería
Ingeniería de Ciencias Computacionales
Metodología de la Investigación
Avance Proyecto Final
Professor: Moises Sánchez Adame
Tijuana, B.C, 30 de Noviembre del 2022

Gerardo Meneses
29902
CETYS Universidad
Escuela de Ingeniería
Ciencias Computacionales
gerardo.meneses@cetys.edu.mx

Daniel Agraz
33293
CETYS Universidad
Escuela de Ingeniería
Ciencias Computacionales
daniel.agraz@cetys.edu.mx

David Roldán
27292
CETYS Universidad
Escuela de Ingeniería
Ciencias Computacionales
davide.roldan@cetys.edu.mx

1. Introducción:

El presente documento tiene como intención demostrar al lector la propuesta de investigación que tiene como objetivo la implementación de algoritmos de reforzamiento. Demostrando primeramente la parte teórica y como dichos fundamentos se aplican en el mundo real. Donde parte del enfoque de la investigación es analizar su rendimiento en aplicaciones sencillas como son los videojuegos 2D, integrando algoritmos de aprendizaje por refuerzo clásicos demostrando resultados.

El resto del documento se encuentra estructurado de la siguiente manera. Apartir del punto numero 2. Se expone el Plantamiento del problema de la realización del proyecto, donde se introducen el tema de la investigacion, el juego que se utilizara como prueba sera el ”breakout” y los algoritmo que se empleara para obtener los resultados. También se presentan el Marco Referencial en el capitulo 4 y la estrategia metodologica donde ya se explica mas a detalle la emplimentación y hace donde queremos llegar con esta investigación y por ultimo se explicara los resultados que se tuvieron con esta investigación experimental.

2. Planteamiento del Problema

2.1. Delimitación del Tema

En esta investigación se busca abarcar el desempeño del algoritmo de reforzamiento Deep-Q Network (DQN) como una combinación de Q-Learning con Redes Neuronales que da apertura a que un agente evalúe el estado actual de un ambiente controlado para producir una decisión autónoma. Este estudio promueve obtener un contraste con respecto a la clásica implementación que actualmente existe con aprendizaje por refuerzo sobre varios juegos virtuales.

El ambiente que se va a utilizar para evaluar dicho algoritmo va a ser en el juego de Atari Breakout. Con el propósito de dirigir el enfoque no tanto en el desarrollo gráfico de los juegos sino en la lógica que ejecuta del programa.

Además se concentra en presentar los conceptos que engloban al aprendizaje por reforzamiento, para tener un mejor entendimiento sobre el proceso de enseñanza con este tipo de aprendizaje que conforma parte de la rama de “Machine Learning”. Como es de esperar, se debe referir a la Inteligencia artificial con el motivo de explicar la estrecha relación que tiene con el proyecto.

2.2. Preguntas de investigación

- ¿Cómo se implementan los algoritmos por reforzamiento?
- ¿Que aplicaciones reales tiene aprendizaje por refuerzo en el mundo real
- ¿Qué tan eficientes son los algoritmos de aprendizaje por refuerzo para ganar juegos sencillos en comparación a algoritmos clásicos?
- ¿Cuáles son los algoritmos por refuerzo más comunes?
- ¿Cuáles son los algoritmos clásicos que se han usado para ganar juegos sencillos?

2.3. Objetivos de investigación

Objetivo Principal:

- Determinar en qué métricas son más eficientes los algoritmos de aprendizaje con deep learning por refuerzo en comparación con los clásicos

Objetivo Específicos:

- Identificar el ambiente y los agentes que se involucran en cada uno de los juegos a evaluar.
- Implementar una de las librerías disponibles que existen para el diseño y desarrollo de Algoritmos de Reforzamiento con Redes Neuronales.
- Definir las métricas en las que el aprendizaje por refuerzo con Redes Neuronales puede ser superior a algoritmos clásicos aplicados.
- Conocer el impacto que tiene el aprendizaje por refuerzo con Redes Neuronales en esta era digital.
- Hacer que un algoritmo Deep Q Learning tenga más aprendizaje que un algoritmo de aprendizaje clásico (learning) en el juego del breakout.

2.4. Justificación

El propósito del proyecto es demostrar nuestra hipótesis primeramente que él los algoritmos de refuerzo tendrán un mejor desempeño que los algoritmos convencionales, adicción de esto el interés en particular de los integrantes del equipo por esta área de conocimiento derivada de Inteligencia Artificial, complementando el gran crecimiento de aplicaciones que se están creando y cambiando nuestra forma de vida, nos hizo profundizar en un área que conocemos, la cual es los videojuegos.

3. Marco Rerencial

3.1. Estado del arte

En la actualidad la inteligencia artificial juega un papel importante en las áreas científicas, industria aeroespacial, en áreas de seguridad cibernética y demás. Donde el estado del arte de aprendizaje por refuerzo implica generalmente el uso de librerías como Tensorflow o Pytorch que facilitan en gran medida la experimentación, investigación, desarrollo y mejora de algoritmos de aprendizaje profundo de la mano con aprendizaje por refuerzo. Actualmente empresas reconocidas en esta área de investigación son OpenAI y DeepMind.

3.2. Marco Teórico

Los antecedentes del aprendizaje por refuerzo tiene como fundamento teórico el aprendizaje estadístico. Siendo aprendizaje estadístico una rama del área de las matemáticas, la cual se enfoca en encontrar relaciones entre variables ya sean cualitativas o cuantitativas, para poder generar un resultado correlacionado. Posterior a ello, se combina aprendizaje estadístico con aprendizaje máquina, está siendo una área de las ciencias computacionales que tiene como objetivo implementar la teoría matemática de aprendizaje estadístico a instrucciones que una computadora pueda realizar de una forma eficiente. Con el pasar de los años fue evolucionando a ser conocida como inteligencia artificial. En la actualidad, existe una nueva subrama de la inteligencia artificial conocida como aprendizaje profundo, esta subrama rompe con lo convencional de aprendizaje estadístico/máquina e integra nuevas ideas y conceptos acerca de redes neuronales artificiales, neurona artificial, transformaciones no lineales aplicadas por neurona, transformación de datos topológicos por medio de operación de convolución, y más. Por otro lado, aprendizaje por refuerzo es uno de los 4 tipos de aprendizaje máquina, los cuales son, aprendizaje supervisado, aprendizaje no supervisado, aprendizaje semi-supervisado y aprendizaje por refuerzo.

Algorithm 1: Q-Learning

Result: Off-policy control for estimating $\pi \simeq \pi_*$
Parameters: step size $\alpha \in (0, 1]$, small $\epsilon > 0$
Initialize $Q(s, a)$ $\forall s \in \mathcal{S}^+, a \in \mathcal{A}(s)$ arbitrarily.
Set $Q(terminal, \cdot) = 0$
for *each episode* **do**
  Initialize S;
  for $step = 0$ **to** T **do**
    Choose A from S using policy derived from Q (e.g., ϵ-greedy);
    Take action A, observe R, S';
    $Q(S, A) \leftarrow Q(S, A) + \alpha[R + \gamma \max_a Q(S', a) - Q(S, A)]$;
    $S \leftarrow S'$;
  end
end

Esta imagen buscar explicar los pasos y formulas que se utilizan para poder cumplir con el algoritmo de Q learning.

Algorithm 1 Deep Q-learning from Demonstrations.

Inputs: $\mathcal{D}^{replay}$: initialized with demonstration data set, θ: weights for initial behavior network (random), θ': weights for target network (random), τ: frequency at which to update target net, k: number of pre-training gradient updates
for steps $t \in \{1, 2, \dots k\}$ **do**
  Sample a mini-batch of n transitions from $\mathcal{D}^{replay}$ with prioritization
  Calculate loss $J(Q)$ using target network
  Perform a gradient descent step to update θ
  if $t \bmod \tau = 0$ **then** $\theta' \leftarrow \theta$ **end if**
end for
for steps $t \in \{1, 2, \dots\}$ **do**
  Sample action from behavior policy $a \sim \pi^{eQ_\theta}$
  Play action a and observe (s', r).
  Store (s, a, r, s') into $\mathcal{D}^{replay}$, overwriting oldest self-generated transition if over capacity
  Sample a mini-batch of n transitions from $\mathcal{D}^{replay}$ with prioritization
  Calculate loss $J(Q)$ using target network
  Perform a gradient descent step to update θ
  if $t \bmod \tau = 0$ **then** $\theta' \leftarrow \theta$ **end if**
  $s \leftarrow s'$
end for

La imagen que se muestra muestra los pasos que contiene uno de nuestros algoritmos expuestos el cual es Deep Q learning dicho este cuenta con redes neuronales, en imagen pasada explica las formulas y pasos que con llevan para cumplir con este aprendizaje con éxito.

3.3. Diseño metologico

El motivo de nuestra elección de este proyecto, es la demostración de nuestro tema, así como también cumplir con los objetivos que nos establecimos comprobar nuestra hipótesis, con la investigación profunda que se tuvo entre Deep Q learning y Q learning ,donde con ello tendremos que entender sobre los agentes que utilizaremos donde será nuestro el algoritmo de aprendizaje el cual será el que tendrá que aprender y implementar, así mismo como el ambiente nuestro juego simple conocido como "breakout", este sumerge, este principio de acercamiento con las redes neuronales, nos enseñara los impactos que está teniendo y tendrá sobre nuestra vida a futuro, entre los beneficios y prejuicios. Todo esto suma al hecho que se tiene que realizar un algoritmo de ambas implementaciones el cual tenga como entorno del entorno simple, para medir la eficiencia de nuestros algoritmos tendremos que declarar unas métricas para esta eficiencia, las cuales se tendrán que aplicar.

4. Estrategia Metodológica

4.1. Depedencias

En cuestión de la implementación realizada fue mediante un código en python como se mencionó anteriormente, el cual como se tiene en Colab, para un mejor manejo de recursos.
Empezando con las dependencias del proyecto, las librerías que se utilizaron fueron las siguientes:

- Gym
- typing
- Autorom
- Mushroom
- piglet
- Numpy
- Tensorflow
- Matplotlib
- IPython
- Math
- Random

4.2. Deep Q learning Approach

Para la implementación moderna, se empleó "BreakoutNoFrameskip-v4" un modelo entrenado de un agente DQN que utiliza la librería de stable-baselines3, lo que posibilita que la máquina se desenvuelva en el ambiente del videojuego Breakout. Después se buscó configurar dicho ambiente para utilizarlo en un modelo determinado para el propósito del proyecto. Con esto se estableció la política de 'LnCnnPolicy', que permite la implementación de un actor crítico que hace uso de redes neuronales convolucionales con una capa de normalización, dentro de esta política del DQN, se cargan como parámetros la variable env, que incluye el ambiente en el que interactuara el modelo; $prioritized_replay =$ True, esta característica garantiza que la experiencia de juego sea acumulativa, lo que muestra el concepto de "experience replay" que fomenta la práctica de la computadora en el juego. Habiendo establecido el modelo y la política del mismo, se procedió a poner a aprender al modelo con la función de model.learn, en donde se definió 15000 $total_timestamps$, que viene siendo el número de pasos en total que el agente hará sobre el entorno, por lo que, por cada partida la máquina interactuara por 15000 frames, siendo ese el tiempo que tiene para lograr una puntuación máxima. Para finalizar se guarda el modelo con model.save, asegurando que la configuración del modelo se haya guardado.

4.3. Temporal Difference Learning

Empezando hacemos una demostración de cómo quedaría nuestro modelo impreso en el programa, en pocas palabras el tablero donde se podrá apreciar los ejes así como el puntaje en la parte superior, posterior a ello, declaramos nuestras variables de las posiciones.

Para esta parte de la implementación clásica, se lograron representar cuatro espacios: Los bloques de colores, que si los destruyes te otorgan un punto; el espacio en el que el jugador visualiza la pelota moverse; los límites que permiten al jugador mover la barra de izquierda a derecha para hacer que rebote con la pelota y por último, el espacio que detecta si el jugador perdió.

Todas las dimensiones de estos espacios de referencia

se realizó, especificando sus coordenadas en tuplas de enteros en un plano 2D, para luego representarlo con plt.figure() de la librería matplotlib.pyplot.

También se logró tener una representación matricial de números binarios que señalen este tipo de espacios anteriormente mencionados, así como, indicadores de si la pelota está fuera de los límites del juego, o si tuvo impacto con la barra deslizante.

4.4. Diseño Metodológico

Cuantitativo y Experimental El motivo de esta opción fue debido a que se cuentan con resultados cuantitativos, ya que tenemos la oportunidad de analizar y más que todo contar, por otra parte experimental por la manera de comprobar nuestra hipótesis es mediante un experimento, el cual se explicará más adelante.

4.5. Hipótesis

El algoritmo de aprendizaje por refuerzo combinado con redes neuronales tiene un mejor desempeño en términos de alta precisión y baja tasa de error, a la hora de terminar un juego, en comparación al desempeño de un algoritmo de refuerzo clásico.

4.6. Variables e Indicadores

Variable independiente: Tiempo (Iteraciones)
Variable dependiente: Precisión

Indicadores

Se ejecutará el algoritmo para elaborar 5 entrenamientos, donde en cada entrenamiento haremos uso de las siguientes métricas para determinar el desempeño de cada algoritmo y comprobar la hipótesis son las siguientes:

- El tiempo en microsegundos o segundos que se toma para resolver cada juego.
- La puntuación obtenida en cada iteración.
- La tasa de error.
- La complejidad temporal del código del algoritmo.

4.7. Técnicas e instrumentos de recolección

Las técnicas que se utilizaran en el proyecto, será el análisis y observación de resultado obtenidos siguiendo nuestros indicadores que nos servirán a detectar el algoritmo con mejor desempeño. Los algoritmos serán implementados mediante el lenguaje Python, y se realizará una implementación por cada juego propuesto: Breakout, Minesweeper y Pong,

5. Análisis de Resultados

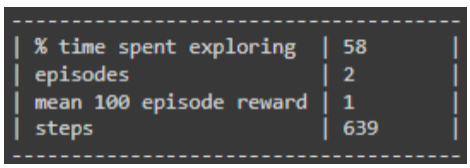
Al tratarse de un proyecto de investigación cuantitativo y experimental, el enfoque del mismo no requiere transitar por una etapa de recolección de datos. Se considera que no es viable realizar este proceso debido a que el criterio de comparación de desempeño entre la implementación moderna y clásica del algoritmos, en este caso es el puntaje máximo que la máquina consiga en un determinado lapso de tiempo, así como los movimientos o iteraciones que requiere el algoritmo para conseguir dicho resultado.

Así pues para obtener estas métricas, se tendrá que generar varias pruebas de entrenamiento que evalúen el desempeño del algoritmo por cada iteración, tal que, se tenga información sobre los resultados de cada partida que juegue la computadora, para así rastrear a detalle el desarrollo de la inteligencia artificial e identificar en que punto se empieza a notar una mejoría en la interacción de la máquina.

5.1. Técnicas de Análisis de Datos

Hasta ahora para este avance la técnica que se aplicó es evaluar el modelo por un número determinado de episodios, en el que tendrá la posibilidad de irse acostumbrando al modo de juego de breakout, para que comience a aprender, y tenga la capacidad de que en futuros episodios mejore su partida.

Algunas de las métricas que se tienen son el % de tiempo de exploración, el número de episodio, el promedio de recompensa por cada 100 episodios y el número de pasos que se ejecutarón en el mismo.



Para este avance se logró realizar la evaluación de 14 episodios, de los cuales se puede visualizar en cada uno las métricas explicadas previamente.

% time spent exploring	2
episodes	14
mean 100 episode reward	1.5
steps	11235

5.2. Resultados

Algoritmo Q learning

Primero se mostrarán los resultados obtenidos con la implementación clásica de ”Q-Learning” que se ejecuto una cantidad de nueve veces siendo que por cada iteración se llegaron a jugar entre 250 y 500 juegos para entrenar a la máquina, juntando en conjunto un total de alrededor de 3500 iteraciones.

Por cada una de las nueve ejecuciones que se realizaron se empleó un riguroso proceso de entrenamiento en el que se entrenaba el modelo hasta que terminara 500 juegos y por cada ejecución se guardaron todas las partidas del agente a cuadro por cuadro y la póliza en una archivo de formato .JSON en donde se fueron acumulando los pesos de los estados y la actualización de estados nuevos.

Un factor a destacar es que el puntuaje que se presentará en las siguientes gráficas no concuerda en su totalidad a la puntuación que obtuvo el agente en la partida real, esto ocurre debido a que en el juego de Breakout los bloques que se encuentran hasta el fondo del contenedor otorgan más puntos, sin embargo con la finalidad de visualizar estos resultados solo se contó cuando la pelota hace contacto con los bloques.

La primer métrica que se tomó en cuenta fue la puntuación obtenida por juego. Con la implementación clásica en un lapso de 3500 iteraciones, la máquina logró alcanzar la puntuación máxima de 23 puntos en la iteración 3433 que perteneció a la novena ejecución de entrenamiento.

Dentro de cada juego el agente se le dió el objetivo de anotar la mayor cantidad de puntos posibles a lo largo de 5 oportunidades por juego. El juego finaliza cuando el agente haya utilizado sus cinco vidas y se hace recuento de la puntuación.

Esta gráfica comprende la puntuación de cada juego en el que participó el agente. Como se puede notar está gráfica marca un incremento gradual en la primera mitad de las iteraciones y luego ya se mantiene estable con ciertos picos de resultados.

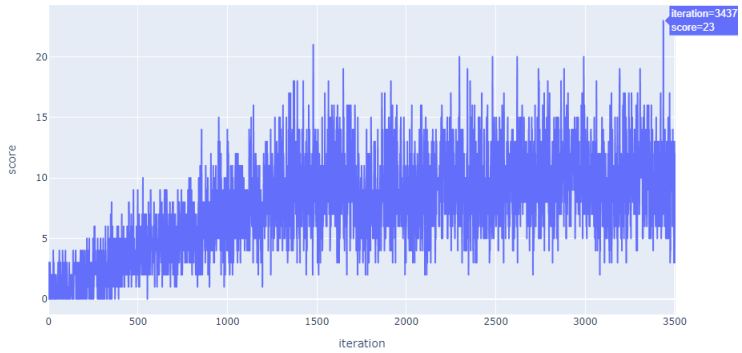


Figure 1. Gráfica de la Puntuación por iteración(juego)

En la siguiente gráfica se presentan la frecuencia de puntajes únicos en un rango de iteraciones, en este visualizador se priorizaron solo las primeras 1000 iteraciones de las 3500 iteraciones. Se puede notar como en la última sección de iteraciones el color de la barra es más variada, esto indica que los resultados cada vez más tienden a ser de puntuaciones cada vez más altas, siendo que ya no solo predominand 4 o 7 colores, sino más de 10.

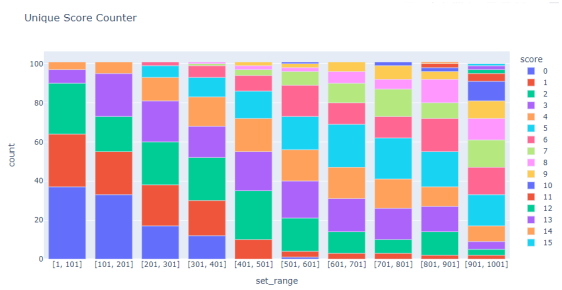


Figure 2. Gráfica de Puntuación Única en función de un rango de iteraciones

La figura 3. muestra el registro de puntuaciones máximas, esto es uno de los principales indicadores para determinar si el modelo en verdad esta aprendiendo. En este caso con la implementación clásica ocurrió que el algoritmo aprendió de manera escalonada teniendo un rápido progreso conforme cada ejecución.

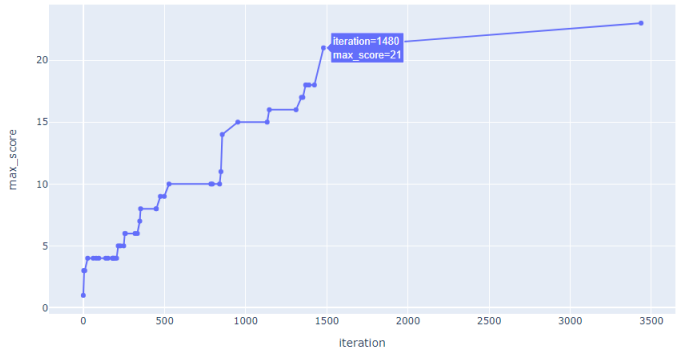


Figure 3. Gráfica de Puntuación Máxima en base a una determinada iteración

La figura 4. indica el crecimiento de las puntuaciones promedio obtenidas por conjuntos iguales de 100 iteraciones, y busca reafirmar que en efecto el algoritmo esta aprendiendo.

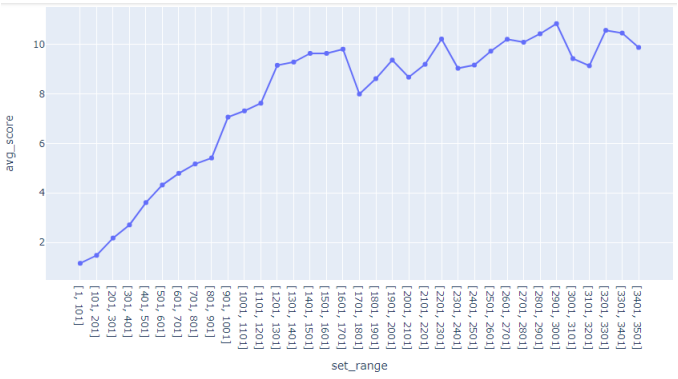


Figure 4. Gráfica de la Puntuación Promedio en un rango de iteraciones

La figura 5. representa el número de frames jugados por juego, esta métrica resulta útil, ya que facilita saber que agente fue el que más tiempo estuvo interactuando con el ambiente. Siendo que de la misma manera que la puntuación se obtuvo una gráfica con un crecimiento gradual. Tal que en una ejecución se puede percibir un buen desempeño por parte de determinada iteración.

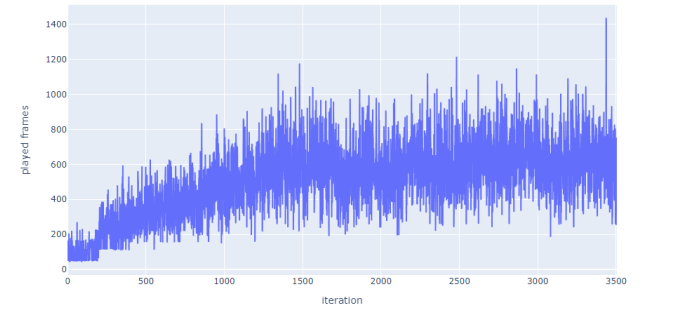


Figure 5. Gráfica de Frames Jugados en función de una iteración dada

Otra métrica interesante que expone la curva de aprendizaje que tiene el agente con el ambiente es el descubrimiento de nuevos estados y la longitud total de estados que la póliza actualiza conforme cada ejecución. Tanto en la figura 6 y 7, se observa como en las primeras etapas de entrenamiento el agente recibió un incremento en la cantidad de estados nuevos aprendidos por iteración, mientras que, en etapa finales el algoritmo redujo la cantidad de estados nuevos.

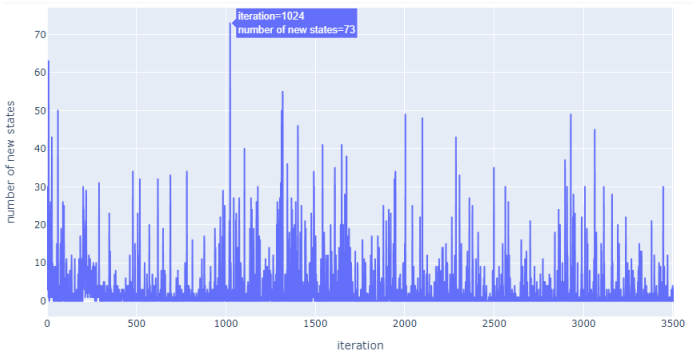


Figure 6. La siguiente gráfica demuestra el número de nuevos estados aprendidos por iteración

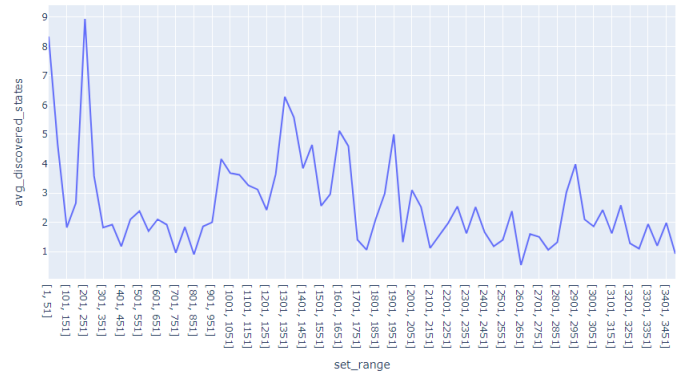


Figure 7. Esta gráfica indica el Promedio de Estados Descubiertos en base a un rango de iteraciones

Y por último, el número de estados totales termina siendo una de las últimas métricas que resaltan el progreso de aprendizaje del modelo, en la figura 8. se presenta como de forma casi lineal, la gráfica crece con un comportamiento estable siendo que cada ejecución se va incrementa el número de estados totales en la póliza más reciente.

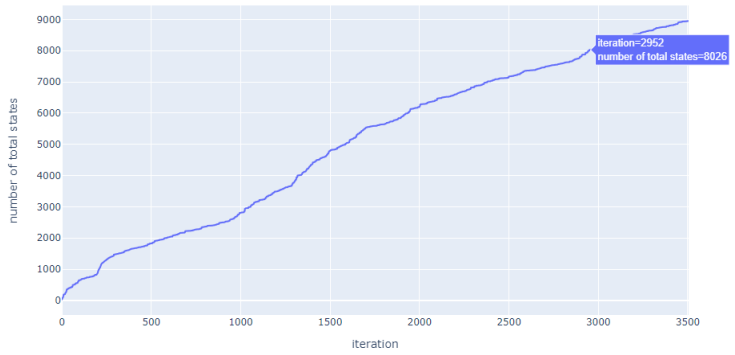


Figure 8. Gráfica del número de Estatos Totales por iteración

Para cerrar con los resultados de la implementación Q-Learning se logró capturar el juego con puntuación máxima de la iteración 3433. En esta iteración el agente logró obtener 24 puntos, tomando en cuenta solo el contacto por bloque.

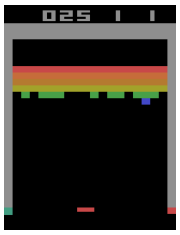


Figure 9. Gráfica del número de Estatos Totales por iteración

Algoritmo Deep Q learning

En cuanto a los resultados de la implementación moderna bajo el algoritmo de Deep Q-Learning, dicho algoritmo maneja redes neurales convolucionales para abstraer las características de la imágen como bordes, colores entre otros aspectos de la imágen y lo que busca este acercamiento es traducirle al agente cada cambio en el ambiente por medio de la modificación de estados y pesos.

rollout/	
ep_len_mean	865
ep_rew_mean	2.57
time/	
fps	27
iterations	30000
time_elapsed	5523
total_timesteps	150000
train/	
entropy_loss	-0.0982
explained_variance	0.98
learning_rate	0.0007
n_updates	29999
policy_loss	-4.45e-05
value_loss	8.99e-06

Figure 10. Métricas obtenidas en la iteración número 150,000

rollout/	
ep_len_mean	834
ep_rew_mean	2.26
time/	
fps	25
iterations	55000
time_elapsed	10885
total_timesteps	275000
train/	
entropy_loss	-0.55
explained_variance	0.818
learning_rate	0.0007
n_updates	54999
policy_loss	-0.0115
value_loss	7.66e-05

Figure 11. Métricas obtenidas en la iteración número 275,000

rollout/	
ep_len_mean	801
ep_rew_mean	2.22
time/	
fps	24
iterations	72000
time_elapsed	14814
total_timesteps	360000
train/	
entropy_loss	-0.223
explained_variance	1.19e-07
learning_rate	0.0007
n_updates	71999
policy_loss	-5.2e-05
value_loss	1.43e-06

Figure 12. Métricas obtenidas en la iteración número 360,000

A manera de ejemplificación se muestra en la figura 9. la partida que produció la mayor puntuación al entrenar el modelo con DQN, este juego se dió en la iteración 350,000 con una puntuación de 30 puntos, si solo se toma en cuenta los contactos por bloque.

Figure 13. Mejor iteración aplicando algoritmo de DQN

5.3. Discusión

Ahora en este apartado se busca generar un contraste entre los resultados obtenidos por parte de ambos algoritmos. A priori los resultados pueden parecer un tanto por la suposión que poner en la cual aprende de una manera muy parecida a la del sur humano mientras que por otra parte el Deep Q learning este aprende mediante ya la utilización de redes neuronales, con ello se puede apreciar una notable diferencia de puntos.
Entrando ya a la comparación de los resultados entre los dos tipos, para mostrar los resultados precios, reiterando nos hemos basado en su puntuación(score) máxima de cada partida. El algoritmo de Q learning alcanzo una puntuación máxima de 24 con un total de 3500 iteraciones, su contraparte el algoritmo de Deep learning alcanzo una puntuación máxima de 35 con un total de 35 con un total de interacciones de $350,000$ iteraciones en total

Al final tambien podemos concluir en base a los puntajes maximos obtenidos por ambas implementaciones, que el

algoritmo de aprendizaje deep learning al llegar a las 350,000 iteraciones de entrenamiento es alrededor de un 20% mejor. Dado que:

%mejora = $(1 - QLearning/DeepQLearning) * 100$
%mejora = $(1 - 24/30) * 100$
%mejora = 20%
Pero a costo que el algoritmo de Deep Q Learning tuvo que entrenar 10,000% mas iteraciones que el clásico.

% diferencia de iteraciones = DeepQLearning / QLearning * 100
% diferencia de iteraciones = 350,000 / 3500*100
% diferenica de iteraciones = 10,000%

6. Conclusión

No cabe que la inteligencia artificial esta siendo un campo de desarrollo emergente e innovador tecnologicamente hablando, donde solo unos pocos están familiarizados con ello, es ahí donde la elaboración de este escrito ayudo a comprobar nuestra hipótesis de una manera muy exhaustiva, esto nos hizo llegar a la concluisión que nuetra hipotesis fue comprobada correctamente. Los objetivos investigacion experimental por la forma que se plantea, la demostracion de nuestra de los resultados, el desarrollo de las pruebas hizo alcanzar el logro de los objetivos mencionados anteriormente.
Esto nos hace llegar a que las redes neuronales necesitan un gran cantidad de recurso computacional, esto genera que se requiere de muchas iteraciones para entrenar un modelo efectivo, que fuera contundente y representativo con sus resultados.

7. Bibliografía

1) Russell, S. J., Norvig, P. (2016). Artificial intelligence: a modern approach. Malaysia; Pearson Education Limited.

2) Dayan, P., Niv, Y. (2008). Reinforcement learning: the good, the bad and the ugly. Current opinion in neurobiology, 18(2), 185-196.

3) Li, Y. (2017). Deep reinforcement learning: An overview. arXiv preprint arXiv:1701.07274.

4) Sutton, R. S., Barto, A. G. (2018). (2nd ed.). The MIT Press.