Comparación de Aprendizaje entre Jugadores y un Agente Inteligente en Snake

1st Marco Ferraro Rodriguez
Escuela de Ciencias de la Computación
Universidad de Costa Rica
Alajuela, Costa Rica
marco.ferraro@ucr.ac.cr

2nd Gabriel Revillat Zeledón
Escuela de Ciencias de la Computación
Universidad de Costa Rica
San José, Costa Rica
gabriel.revillat@ucr.ac.cr

3rd Steven Nuñez Murillo
Escuela de Ciencias de la Computación
Universidad de Costa Rica
Grecia, Costa Rica
steven.nunez@ucr.ac.cr

***Resumen*—El presente trabajo emplea un ambiente de juego. Este juego fue establecido como una variación del juego Snake. Después de definir el juego se planetea evaluar el rendimiento de 5 jugadores y de un agente inteligente implementando por medio de Python. Se determina que el agente tiene mejor rendimiento que los usuarios ya que, para el agente, la toma de decisiones dentro del juego las realiza con mayor facilidad. Sin embargo, para que el agente logre tener este rendimiento, tuvo que haber un proceso de entrenamiento muy grande.**

***Palabras claves*—inteligencia artificial, agente inteligente, videojuego, aprendizaje automático, aprendizaje reforzado, Snake, Deep Q-Learning.**

Introducción

Los videojuegos en la actualidad forman parte de un gran porcentaje de personas las cuales exigen constantemente un mayor realismo e interacción en estos, estas interacciones se dan por general entre agentes, entornos de juego y personas, las cuales aportan problemas complejos para que los agentes los resuelvan. Por razones relacionadas a esas y que son entornos controlables y con cantidades infinitas de datos útiles para algoritmos de aprendizaje automático es que es los videojuegos son un área de investigación perfecta para la inteligencia artificial [1], lo que ha llevado a realizado aportes como los personajes no controlados por jugadores o NPC como son conocidos popularmente, los cuales han formado parte de una práctica esencial en esta área para aportar el extra de experiencia interactiva y realismo que tanto se solicita.

Estos NPCs han despertado gran curiosidad en los desarrolladores debido a la necesidad de crear mejores agentes en ambientes más complejos y ver hasta que punto pueden ser competentes con un ser humano, ya que como menciona [2] por lo general los NPCs definen su comportamiento a base de scripts fijos con tareas básicas que un NPC puede hacer, lo que hace que tengan comportamientos inflexibles y rígidos, por lo que continuamente se han buscado técnicas para mejorar estos agentes de tal manera que empiecen con conocimientos desde cero al igual que una jugador y a partir de ahí se vayan desarrollando.

El incorporar estos agentes inteligentes requiere de algoritmos creados especialmente para ellos lo cual conlleva un enorme trabajo y mucho tiempo sin contar el hecho de que un gran numero de estos algoritmos usan técnicas de aprendizaje automático que para lograr adaptarse a cada juego requieren de un lapso de duración bastante considerable. Lo que a lleva a buscar formas de agilizar estos procesos proponiendo por ejemplo el aprender de las acciones que realiza un jugador mientras juega utilizando así técnicas de aprendizaje reforzado en estos algoritmos [3].

Esta técnica de aprendizaje reforzado que se mencionó anteriormente es la más común en la industria de los videojuegos debido a que ofrece grandes resultados gracias a que permite aprovechar la gran cantidad de escenarios de datos utilizando algoritmos que permiten evaluar repetidamente cada uno de los escenarios de datos, obteniendo cada vez mejores resultados [4]. Un ejemplo del uso de esta técnica es la inteligencia artificial de google Deep Mind que utilizando aprendizaje por refuerzo profundo ha conseguido crear un agente competitivo capaz de derrotar a jugadores profesionales del videojuego Starcraft el cual posee una gran cantidad de variantes y complejidad.

Sin embargo, es importante dejar claro que el aprendizaje automático ofrece otra gran variedad de algoritmos diferentes aparte del ya mencionado, como lo son el aprendizaje supervisado, no supervisado, multi-tarea entre otros, los cuales también poseen capacidad de entrenar a estos agentes inteligentes, por lo que la alternativa que desee utilizar el desarrollador va a depender del contexto para el cual esté siendo desarrollado este agente, los requisitos y las capacidades con las que deba contar dicho agente.

Debido a que el objetivo general de este articulo es el comparar el rendimiento que tiene un agente inteligente al competir contra jugadores reales en el videojuego Snake se decidió utilizar un algoritmo de aprendizaje de refuerzo profundo para desarrollar el agente inteligente esto con el fin de que el agente aprendiera desde cero realizando movimientos aleatorios y fuera aprendiendo de estos conociendo únicamente que su objetivo es maximizar las recompensas según los estados posibles.Entonces para lograr este objetivo se llevó un proceso de construcción del agente inteligente, posterior a esto un entrenamiento del agente del cual se iban recolectando información de las partidas del agente, luego hubo un proceso de recolección de la información de las partidas de los jugadores para por último definir que aspectos evaluar y obtener los resultados para su futuro análisis.

METODOLOGÍA

Para la realización de los objetivos de este proyecto se va a llevar a cabo una metodología que consiste en 6 etapas de trabajo.

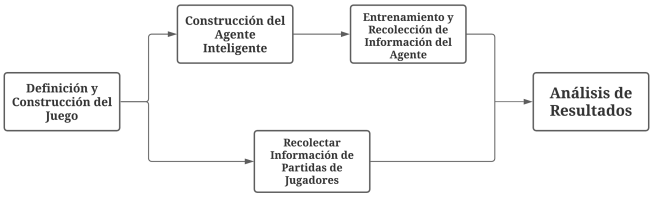

Figura 1. Etapas de la metodología de trabajo

- **Definición y Construcción del Juego:** En la primera sección de la metodología se definirá e implementará un juego para el desarrollo del proyecto.
- **Construcción del Agente Inteligente:** Una vez definido el ambiente de juego se implementará un agente inteligente para que funcione y aprenda sobre el juego previamente definido.
- **Entrenamiento y Recolección de Información del Agente:** Para el siguiente paso, teniendo el agente inteligente implementado, se plantea realizar el entrenamiento del agente inteligente. Esto se hace con la intención de que el agente logre desarrollarse sobre el juego. Después de entrenar al agente inteligente, se definirá los pasos para recolectar información que se obtendrán a partir de partidas evaluadas.
- **Recolectar Información de Partidas de Jugadores:** En esta sección, apartado del desarrollo de las 2 etapas anteriores, se recolectará información de partidas del juego previamente definido. Se definirá la forma en que se recolecta los resultados, obtenidos por medio de jugadores que interactúen con el juego.
- **Análisis de Resultados:** Finalmente, en esta etapa, se definirá como interpretar los datos que se obtienen, tanto del agente como de los jugadores.

A. *Definición y Construcción del Juego*

El juego que se implementó es una variación del juego clásico de *Snake*. El objetivo del juego es que un jugador controle una serpiente y logre comer la mayor cantidad de alimentos posible sin perder el juego. Para implementar el juego se tomó de base el proyecto de [5]. Una de la variaciones que se le hizo al juego es que, al momento de comer u obtener 1 punto, la serpiente va a incrementar su velocidad.

En esta implementación del juego *Snake*, la serpiente tendrá la posibilidad de realizar cualquiera de las siguientes cuatro acciones:

- Cambiar de dirección hacia arriba.
- Cambiar de dirección hacia abajo.
- Cambiar de dirección hacia la izquierda.
- Cambiar de dirección hacia la derecha.

Cabe mencionar, que en esta variación de *Snake*, si la serpiente hace contacto la pared el juego se terminará.

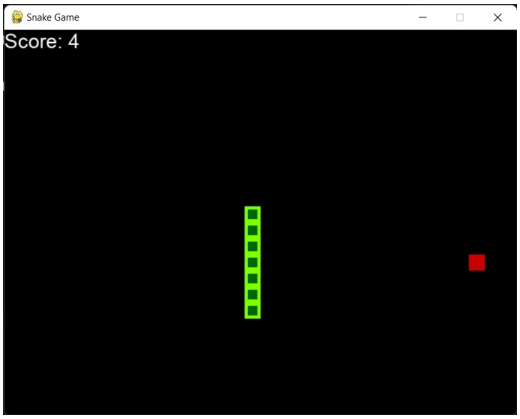

Figura 2. Implementación del juego *Snake* llevada a cabo en Python

Teniendo en claro los conocimientos adquiridos durante el curso de Inteligencia Artificial, un aspecto importante importante del juego es tener una representación del estado actual. Esto es con la intención de que un eventual **Agente Inteligente** logre tomar acciones con base en este *input* [6]. Asimismo, para las siguientes etapas, tener la representación del estado nos ayudará para recolectar datos para la toma de decisiones de los jugadores.

TABLA I
REPRESENTACIÓN DEL ESTADO DEL JUEGO DE FIGURA 2

Peligro Adelante	Si
Peligro Derecha	No
Peligro Izquierda	No
Dirección Actual	Arriba
Comida Izquierda	No
Comida Derecha	Si
Comida Arriba	No
Comida Abajo	Si
Velocidad Actual	15
Puntaje Actual	4

Ya una vez definido como se quiere representar como un estado del juego, hay que fijar como almacenarlo para que sea legible por los programas Python. Para hacer esto, cada estado que se genere será guardado en una lista con variable numéricas que nos permitan representar variables cualitativas como **Derecha**, **Izquierda**, etc..

B. *Construcción del Agente Inteligente*

En el siguiente paso de la metodología se implementará un agente inteligente usando la librería *PyTorch*. Esta librería nos ofrece muchas herramientas para desarrollar proyectos en el área de inteligencia artificial. Con base en los antecedes, y como menciona [4] el tipo de aprendizaje más común en la industria de los video juegos es el aprendizaje reforzado, ya que usa los algoritmos para evaluar repetidamente un escenario de datos. Asimismo, el aprendizaje reforzado nos

ayuda a que un agente logre realizar tareas complejas, fuera de problemas de clasificación [6]. Con base en este conocimiento, y tomando de base la implementación de [5] se definió un agente inteligente que logre analizar todos los estados que genera y utilice la **Ecuación de Bellman** para asignarle valor a los posibles estados que genera de una manera óptima.

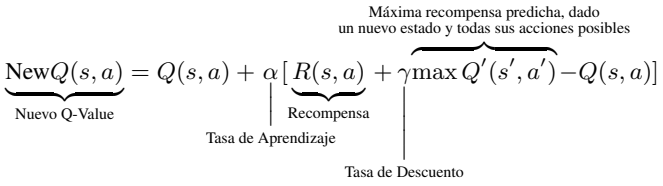

Figura 3. Ecuación de Bellman

Mediante la ecuación de Bellman, se evaluarán las acciones que realice el agente, todo en busca de estados óptimos que maximicen la recompensa. Cabe mencionar, que la una partida con mayor puntaje, brindará más recompensa.

C. Entrenamiento del Agente

El siguiente paso, ya definido el agente, es el entrenamiento del mismo. Al lidiar con un agente inteligente que aprende mediante aprendizaje reforzado no determinar con claridad la cantidad de partidas que se necesite de entrenamiento para que el agente realice las partidas óptimas, siempre. Dicho esto, y teniendo en claro que vamos a usar la ecuación de Bellman como política de decisión, el proceso de aprendizaje se llevará a cabo mediante un modelo de **Red Neuronal**, proporcionada por PyTorch. Al enlazar la ecuación de Bellman con la red neuronal hacemos una práctica de aprendizaje reforzado llamado *Deep Q-Learning* o aprendizaje por refuerzo profundo.

Se llevarán a cabo 135 partidas de entrenamiento. Adicionalmente, para poder desarrollar las etapas posteriores, se almacenará en un archivo csv los resultados de cada partida. Vamos a almacenar 2 variables, el puntaje y los *frames* como unidad de tiempo.

TABLA II
EJEMPLO DE RESULTADOS DE UNA PARTIDA

score	frame_count
0	23
0	135
2	45
0	44

Para evitar que en la fase de exploración el agente decida acciones que resulte en que la serpiente se encicle, se decidió tener un contador de *frames* que se inicializa en 0 cada vez que la serpiente obtiene un alimento. Si este contador alcanza el largo de la serpiente, multiplicado por 100, se terminará la partida.

Una vez entrenado el agente, se plantea tener 15 partidas de prueba. Esto con el propósito de recolectar los datos para la evaluación del rendimiento del agente en las siguientes etapas.

D. Recolectar Información de Partidas de Jugadores

Separado de las 2 etapas anteriores, para cumplir los objetivos establecidos se necesita recolectar información de partidas. Para este experimento, se definió la siguiente metodología; vamos a recolectar la información de las partidas de **5 jugadores**. Cada jugador va a realizar 3 sesiones, en cada sesión se realizarán 5 partidas. Por lo tanto, cada participe del experimento realizara 15 partidas. Para cada partida, se guardará en un archivo csv todos los estados que se generen a lo largo de la partida. Estos estados fueron definidos en la Tabla I. Asimismo, después de finalizar cada sesión, se generará otro archivo csv conteniendo los datos definido en la Tabla II, de esta manera, al tener tuplas con los mismos *labels*, vamos a poder comparar con más facilidad el rendimiento de los jugadores en contra del agente inteligente.

E. Análisis de Resultados

Una vez obtenido los datos de las partidas de los jugadores y los del agente inteligente, la intención para analizar los resultados será, identificar la **media aritmética** de los puntajes y de los *frames* a lo largo de todas las partidas para cada uno de los jugadores, y del agente inteligente. Recordemos que los *frames* de una partida es nuestro indicio de tiempo (consideramos una partida ideal como obtener el mayor puntaje posible en la menor cantidad de *frames* posibles). El siguiente paso que vamos a realizar es, por medio de una herramienta de generación de gráficos, realizar gráficos lineales con una serie de puntajes y media de puntajes para cada jugador y el agente. Después de realizar este gráfico lineal, se seguirá el mismo proceso pero con la cantidad de *frames* y media de *frames*. Los gráficos lineales con estos ejes nos ayudarán a descubrir tendencias y la **tasa de rendimiento** dentro de las partidas. Estos gráficos serán realizados con Excel, por la simplicidad que ofrece este programa para realizar gráficos con datos de archivos tipo csv.

Finalmente, se analizará la cantidad de *frames* y el puntaje de la mejor y peor partida de cada jugador, asimismo como la del agente inteligente. En cada uno de estos pasos, la intención es estar comparando el rendimiento de los jugadores, como una población aparte, contra la del agente inteligente.

RESULTADOS

TABLA III
MEDIA DE PUNTAJES Y FRAMES DE LOS JUGADORES Y DEL AGENTE INTELIGENTE

Jugador	Media de puntajes	Media de *frames*
Usuario 1	4,53	154,2
Usuario 2	12,87	523,8
Usuario 3	10,8	433,6
Usuario 4	7,4	268,4
Usuario 5	7,27	270,8
Agente	22,67	476

En la Tabla III se puede observar la media de puntajes y la media de *frames* de cada jugador, así como las del agente inteligente. Podemos notar en estos datos que la media de

puntajes del agente inteligente supera con gran diferencia a las medias de puntajes de todos los jugadores, y vemos cómo la media de *frames* recorridos por el agente inteligente no llegó a ser muy alta para poder alcanzar los puntajes obtenidos, en comparación a los jugadores donde la media de *frames* es muy alta respecto a sus puntos, lo que nos indica que el agente inteligente logra un mejor rendimiento en las partidas que una persona.

Según la Tabla III, podemos determinar que, entre todos los usuarios, el usuario 1 fue quien registro media de puntajes más baja, mientras que el usuario 2 fue quien registro la media más alta. A continuación se muestran los gráficos resultantes de los usuario 1 y 2, respectivamente.

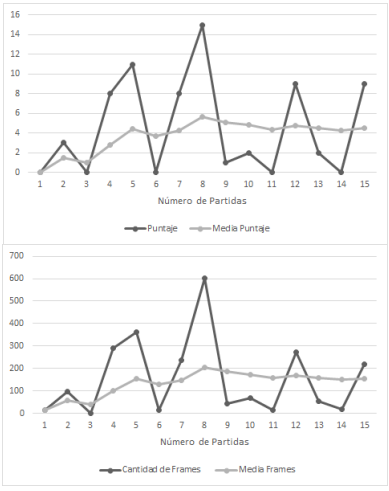

Figura 4. Gráficos del usuario 1.

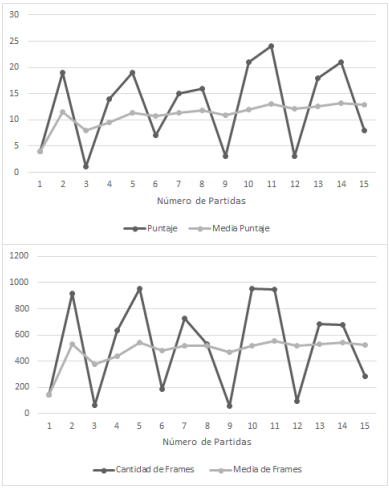

Figura 5. Gráficos del usuario 2.

Al tener estás gráficas, que muestran la linea de medias de puntaje más baja y más alta respectivamente, podemos compararlas con el gráfico lineal generado con los datos del agente inteligente.

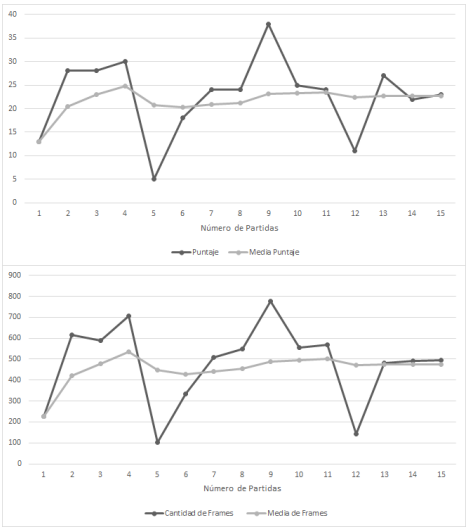

Figura 6. Gráficos del agente.

Notese que, al comparar los gráficos, los gráficos de la Figura 6 muestran un mejor rendimiento dentro del juego que los de la Figura 5. Podemos ver como la linea de media de puntajes es, efectivamente superior que al de la Figura 5. Adicionalmente, podemos ver que en ninguno de los gráficos presentados, existe una tendencia explicita. Podemos ver que tanto los jugadores, como el agente, obtienen valores de puntajes y *frames* de forma inconsistente entre partidas.

TABLA IV
DATOS DE LAS PEORES PARTIDAS DE LOS JUGADORES Y DEL AGENTE INTELIGENTE

Jugador	**Puntaje**	*frames*
Usuario 1	0	16
Usuario 2	1	61
Usuario 3	3	104
Usuario 4	0	2
Usuario 5	0	33
Agente	5	104

TABLA V
DATOS DE LAS MEJORES PARTIDAS DE LOS JUGADORES Y DEL AGENTE INTELIGENTE

Jugador	**Puntaje**	*frames*
Usuario 1	15	601
Usuario 2	24	949
Usuario 3	24	1094
Usuario 4	15	474
Usuario 5	18	682
Agente	38	776

Finalmente, al analizar la Tabla IV y Tabla V podemos ver que el agente inteligente tiene una mayor rendimiento entre las peores partida. Análogamente, al analizar las mejores partidas el agente también logró realizar el mayor puntaje sin hacer la mayor cantidad de *frames*.

DISCUSIÓN

Teniendo los resultados presente pensar que el rendimiento del agente, después de su debido entrenamiento, se debe a varias razones. Tengamos en cuenta que a una persona que nunca ha jugado le irá mejor que al agente si este no está entrenado ya que a la persona se le pueden explicar las reglas y puede comprender con rapidez el entorno del juego; por el contrario, el agente necesita ser entrenado para saber si sus acciones son buenas o no, asimismo como conocer las recompensas y castigos. En la práctica del aprendizaje reforzado no es recomendado darle todas las condiciones del ambiente al agente, ya que el agente debería de entrar en una fase de **exploración** para reconocer el valor de sus acciones, siempre y cuando ya hayan sido realizadas. Una vez que el agente está entrenado, este sabe donde está la comida y va directo hacia ella; en cambio una persona aunque esté enfocada en dirigirse hacia la comida se podría confundir y desviarse antes de alcanzarla, lo que causa que tenga que recorrer más *frames* innecesarios, caso que no ocurre con el agente porque sabe cómo moverse por el espacio de juego una vez entrenado. También podemos mencionar que la decisión de incrementar la velocidad al ingerir un alimento dentro del juego representa un obstáculo más grande para los jugadores que para el agente. Pareciera que este incremento de velocidad es indiferente para el agente, gracias a que basa sus decisiones en la representación del estado. En contra parte, para los jugadores tener una velocidad más alta requiere más concentración y mejores destrezas de coordinación.

CONCLUSIÓN

Como conclusión de este proyecto, y como áreas de mejora, no se pudo garantizar que el modelo que utiliza el agente no este sobre-ajustado con la gran cantidad de partidas de entrenamiento que se realizaron y eso genere un rendimiento tan óptimo en el juego. Sin embargo, al tener el agente entrenado con todas estas partidas, consideramos provechoso ver las diferencias y el proceso de decisión que el agente ha realizado. Adicionalmente, no podemos garantizar que los hiperametros que se establecieron para definir la función de *Deep Q Learning* sean los mejores ya que no se hizo un proceso de *hypertuning* [6].

Adicionalmente, para futuros proyectos, se plantea la posibilidad de utilizar más juegos para analizar el rendimiento del agente en otros juegos, comparándolo con otros jugadores. Asimismo, con los *sets* de datos recolectados, se podría hacer una comparación de rendimiento entre diferentes modelos de aprendizaje, aparte de aprendizaje reforzado.

REFERENCIAS

[1] K. Shao, Z. Tang, Y. Zhu, N. Li, and D. Zhao, "A survey of deep reinforcement learning in video games," *arXiv preprint arXiv:1912.10944*, 2019.

[2] E. B. Montes Chaparro, "Inteligencia artificial adaptativa en videojuegos con procesos estocásticos," 2014.

[3] D. Alcocer Soto, "Aprendizaje por refuerzo aplicado a los videojuegos cooperativos," B.S. thesis, Universitat Politècnica de Catalunya, 2018.

[4] J. D. Duarte Antolinez, "Estudio e implementación de machine learning en el desarrollo de videojuegos," 2019.

[5] P. Loeber, "Snake-ai pytorch," https://github.com/python-engineer/snake-ai-pytorch, 2021.

[6] S. Russell and P. Norvig, "Artificial intelligence: a modern approach," 2002.