Universidad de Margarita

Subsistema de Docencia

Decanato de Ingeniería y Afines

Coordinación de Investigación y Pasantía

Desarrollo de Agente de Inteligencia Artificial para Solución De Problemas en Entornos 2D Basado en el Q-Learning Profundo

Elaborado por: Omar Díaz

Tutora Prof. Valentina Martínez

El Valle del Espíritu Santo, Junio de 2023

INDICE

DEDICATORIA

Este trabajo de investigación está dedicado:

A mi mamá Gely, por apoyarme en todo momento y siempre estar ahí con nosotros.

A mi papá Omar, por todo su esfuerzo para que nuestra familia tenga un buen futuro

A mi hermano David, por alegrar nuestros días.

A mi hermana Eikys, por todas las cosas que me ha enseñado a través de los años.

A mi abuela Nieves (Tata), por criarme y ser una de las personas más influyentes en mi vida.

A mi tío Rommel (Titi), por todo el apoyo y trabajo duro que ha hecho.

A mis sobrinos Gerardo y Danna, por ser curiosos y siempre querer aprender más.

A toda mi familia, por siempre estar unida y ayudarnos los unos a los otros.

A todos los profesores que he tenido a lo largo de los años, por repartir su conocimiento a las siguientes generaciones y ser buenos guías durante mi vida.

A todos los amigos que he hecho durante mi vida escolar, por hacer que la misma fuera más divertida, y por todas las memorias que creamos juntos.

AGRADECIMIENTO

Agradecimientos a las siguientes personas por sus aportes al trabajo de investigación:

A mi tutora, la Profesora Valentina Martínez, por estar interesada en el tema de estudio desde el inicio y ayudarme en todo el camino hasta la conclusión del mismo.

A mi jefe Ronnie Bello, por proveer ayuda y consejos cuando más los necesitaba.

A mi compañero de trabajo Manuel Cova, por ayudar con la construcción de prototipos físicos para probar los modelos.

A Domingo Suárez, por proveer apoyo moral durante todo este tiempo en la universidad.

Al Profesor Flavio Rosales, por ser un excelente profesor y ayudar tanto con temas del trabajo de investigación como de la vida en general.

A la Profesora Yemnel Torcat, por apoyarnos a todos en nuestros trabajos de investigación y dar buenos consejos.

A la Profesora Ana Blanco, por revisar y ayudarnos en nuestros trabajos por voluntad propia.

Al señor Ricardo y la señora Antonieta, por permitirme la oportunidad de estar en la Universidad de Margarita y completar mis estudios.

Al jurado, por aprobar este trabajo de investigación.

A aquellos que trabajaron conmigo en otros proyectos durante la realización de este trabajo, pues aprendí bastantes cosas de esas experiencias, las cuales me sirvieron durante la implementación del software.

LISTA DE CUADROS

LISTA DE GRAFICOS

UNIVERSIDAD DE MARGARITA

SUBSISTEMA DE DOCENCIA

DECANATO DE INGENIERÍA Y AFINES

DESARROLLO DE AGENTE DE INTELIGENCIA ARTIFICIAL PARA SOLUCIÓN DE PROBLEMAS EN ENTORNOS 2D BASADO EN EL Q-LEARNING PROFUNDO

Autor: Omar Díaz

Tutora: Prof. Valentina Martínez

Junio de 2023

RESUMEN

Se desarrolló un framework llamado “Lagomorph” para el entrenamiento de agentes que resuelven problemas en entornos 2D, usando el Q-learning Profundo y con énfasis en la generalización de los mismos. La investigación realizada es de tipo descriptiva y documental, y culminó en un proyecto factible. Se estudiaron varios aspectos del proceso de entrenamiento, como los entornos, las estructuras de redes neuronales, las técnicas de preprocesamiento usadas y los pasos que comprenden el entrenamiento en si. Se determinó que el entorno es el factor más influyente en la generalización de los agentes, y que el Q-learning Profundo logra alcanzar un buen rendimiento en entornos 2D relativamente simples, mas requiere de esfuerzo excesivo para replicar estos resultados en entornos más complejos, como entornos 3D y de texto.

Descriptores: Inteligencia artificial, Q-learning Profundo, 2D, Reinforcement Learning, Generalización, Framework

INTRODUCCIÓN

La inteligencia artificial es un área de rápido desarrollo en la actualidad: Sistemas de procesamiento de lenguaje natural, visión computarizada, reconocimiento de audio y otras tareas más, se han beneficiado de los últimos avances, gracias a toda la atención que la comunidad les presta; sin embargo, existe un área no tan conocida, pero que provee tanta utilidad como las anteriores: El Reinforcement Learning.

Mediante el Reinforcement Learning, se pueden crear agentes de inteligencia artificial capaces de resolver problemas complejos de varios tipos: Conducción de vehículos, navegación, sistemas de recomendación, control de máquinas de ensamblaje, entre otros. Su manera única de entrenar, en comparación con otras formas de inteligencia artificial, le otorga mayor flexibilidad y capacidad de resolver tareas adicionales, a cambio de ser más complejo de utilizar.

Uno de los algoritmos más usados para entrenar agentes de Reinforcement Learning es el Q-learning Profundo, el cual ha probado ser capaz de aprender comportamientos complejos en varios entornos, y se puede implementar de una forma intuitiva y no muy compleja en relación con otras opciones disponibles.

Debido a sus características, el Q-learning Profundo se adapta fácilmente a los entornos 2D de todo tipo, lo que lo hace ideal cuando se quiere entrenar a un agente para resolver problemas que puedan representarse como ese tipo de entorno. Es por ello que este trabajo se centra en la solución de entornos 2D mediante el entrenamiento de agentes por Q-learning Profundo, tomando como base la siguiente estructura:

- Parte I: Descripción general del problema, junto con los objetivos a lograr y el valor académico que brinda esta investigación.
- Parte II: Descripción teórica, especificando las bases teóricas y legales del trabajo, antecedentes y definiciones de términos usados a través del mismo.
- Parte III: Descripción metodológica de la investigación, con su naturaleza, tipo y diseño, además de las técnicas usadas para recolectar y analizar datos.
- Parte IV: Análisis y presentación de los resultados obtenidos de la investigación.

- Parte V: Propuesta del proyecto factible, con su importancia, viabilidad y objetivos a lograr.
- Conclusiones: Resumen sintetizado de los resultados de la investigación y del proyecto factible.
- Recomendaciones: Oservaciones y consejos sobre los temas tratados, además de guías hacia futuras líneas de investigación.
- Referencias: Fuentes y documentos cuya información fue consultada durante el desarrollo de la investigación.

PARTE I

DESCRIPCIÓN GENERAL DEL PROBLEMA

En esta sección se describe la formulación del problema, las interrogantes a responder en el transcurso de la investigación, el objetivo general así como los objetivos específicos y el valor académico que provee este trabajo.

1.1 Formulación del Problema

Los sistemas automatizados fueron creados para facilitar y optimizar procesos en entornos reales y sistemas usados en producción. Durante el proceso de creación y diseño, estos se prueban en entornos simulados que se aproximan a los reales, pero esto suele resultar en sistemas que solo se ajustan a las simulaciones y no son capaces de transferir sus capacidades a los entornos reales. En palabras de Kirk, R., Zhang, A., Grefenstette, E. y Rocktaschel, T. (2022) en A Survey of Zero-shot Generalization in Deep Reinforcement Learning: “La realidad es dinámica, abierta y siempre cambiante, y los algoritmos de RL deben ser robustos frente a variaciones en sus entornos, y tener la capacidad de transferir y adaptarse a entornos no vistos (pero similares) durante su implementación.”

Se han propuesto varias técnicas y avances para resolver este dilema, lo cual resultó en la creación de la disciplina de Inteligencia Artificial. Según López, B. (s/f) en su artículo Introducción a la Inteligencia Artificial: “La IA es una rama de las ciencias computacionales encargada de estudiar modelos de cómputo capaces de realizar actividades propias de los seres humanos en base a dos de sus características primordiales: El razonamiento y la conducta.”

Así es como los agentes o modelos de inteligencia artificial son capaces de identificar patrones y reglas del entorno de forma independiente sin que estas sean declaradas explícitamente por sus creadores, de la misma manera que un ser humano lo hace usando el razonamiento y la lógica. Esas capacidades le otorgan a la inteligencia artificial una ventaja en comparación a los métodos tradicionales, ya que pueden solucionar problemas que sus propios creadores no pueden o se les dificulta resolver. Es por ello

que se suele usar para problemas que involucren patrones y tendencias ocultas, como la predicción de precios y la clasificación de imágenes.

Para ser capaz de resolver los problemas, estos se convierten en entornos matemáticos, comúnmente de una o dos dimensiones. Por ejemplo, la predicción de ventas puede representarse como un entorno 2D con las dimensiones “Tiempo” y “Ventas”, mientras que la clasificación de videos puede representarse en un entorno 3D con dimensiones “Número de fotograma”, “Altura” y “Anchura”. Debido a la complejidad de estos algoritmos, se prefiere usar la menor cantidad de dimensiones posibles en todo momento, por lo que se suelen usar entornos 2D o aplicar transformaciones para convertirlos a entornos unidimensionales.

Según Goodfellow, I., Bengio, Y. y Courville, A. (2016) en Deep Learning:

> La solución es permitir a las computadoras a aprender de la experiencia y entender el mundo en términos de una jerarquía de conceptos, con cada concepto definido mediante su relación a conceptos más simples. Al recopilar conocimiento desde la experiencia, este enfoque evita la necesidad de que operadores humanos especifiquen formalmente todo el conocimiento que la computadora necesita. (…) Si dibujamos un gráfico mostrando cómo estos conceptos son construidos uno encima de otro, el gráfico es profundo, con muchas capas. Por esta razón, llamamos a este enfoque a la IA Deep Learning.

La manera en que el agente o modelo aprende las características de estos entornos depende del tipo de algoritmo que utilice: El machine learning tradicional utiliza regresiones lineales, mas esto limita su uso a entornos lineales. Por otro lado, el Deep Learning usa redes neuronales que imitan los circuitos del cerebro humano para ser capaces de aprender y resolver problemas, incluso en entornos no lineales. Más allá de ello, existe una rama del Deep Learning, llamada Reinforcement Learning, enfocada en la resolución de problemas en entornos 2D y 3D con enfoque en la generalización.

En Playing Atari Games With Deep Reinforcement Learning (Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., Riedmiller, M., 2013), los investigadores de Google DeepMind plantearon una arquitectura de red neuronal llamada Deep Q-Learning o Q-Learning Profundo, y mostraron su efectividad al resolver problemas en entornos 2D, logrando que aprendiera a jugar varios juegos de Atari hasta

alcanzar el mismo nivel que un humano. Ellos entrenaron varios agentes basados en el mismo modelo en los distintos entornos 2D escogidos y cada uno de ellos fue efectivo en su tarea designada, mostrando mejor eficiencia que otros modelos contemporáneos.

Esto se debe a la forma en la que el Q-Learning profundo aprende conductas: Utiliza una tabla (llamada tabla Q) que forma un mapa entre el estado del entorno, las acciones que el agente puede ejecutar en ese estado y la recompensa que el agente obtendría al ejecutar esa acción en el estado indicado; relación que puede representarse como una función $Q(s, a) = r$ donde Q es la tabla Q, s es el estado, a es la acción y r es la recompensa (Mnih, V. et al. 2013:2). De esta forma puede usarse el método de descenso de gradiente para maximizar la función, resultando en un agente capaz de resolver problemas complejos.

A pesar de ello, se evidencia una clara desventaja de este modelo: Aunque todos los agentes se basan en el mismo modelo, estos tienen que ser entrenados individualmente para cada entorno. Si se tiene un agente con buen rendimiento en un entorno, este no puede usarse en otro, sino que se debe entrenar desde cero o crear otro agente para poder completar el nuevo entorno satisfactoriamente. Esto supone un uso mayor de recursos y demuestra una falta de generalización de los agentes entrenados.

Tomando en consideración lo anteriormente expuesto, en este trabajo se busca desarrollar un modelo para resolución de problemas en entornos 2D basado en el Q-Learning profundo, que sea capaz de ser usado en varios entornos con información inicial mínima sobre aquellos donde no haya sido entrenado y que requiera poco o nulo entrenamiento en los mismos para obtener buen rendimiento.

1.2 Interrogantes

1. ¿Cómo puede determinarse una estructura de red neuronal que permita la adaptación a varios entornos?
2. ¿Cuáles entornos poseen el potencial para aumentar la generalización del modelo durante y después del entrenamiento?
3. ¿De qué manera puede definirse un proceso de entrenamiento que permita la experimentación rápida y efectiva de varios modelos en los entornos elegidos?

4. ¿Cómo se puede evaluar la generalización de los modelos resultantes en comparación a los estudiados?

1.3 Objetivo General

Desarrollar un agente de inteligencia artificial para solución de problemas en entornos 2D basado en el Q-learning profundo.

1.4 Objetivos Específicos

1. Determinar una estructura de red neuronal que permita la adaptación a varios entornos.
2. Identificar entornos que tengan el potencial para aumentar la generalización del modelo durante y después del entrenamiento.
3. Definir un proceso de entrenamiento que permita la experimentación rápida y efectiva de varios modelos en los entornos elegidos.
4. Evaluar la generalización de los modelos resultantes en comparación a los estudiados.

1.5 Valor académico de Investigación

El área de la inteligencia artificial está en auge en la actualidad; proyectos como GPT-3, Stable Diffusion, DALL-E y los Transformers de Hugging Face han mostrado que la utilidad del Machine Learning y el Deep Learning no se limita a lo teorético, considerando que provee un valor práctico y tangible: Empresas como Microsoft y Amazon usan el Deep Learning para desarrollar sus agentes conversacionales (Cortana y Alexa, respectivamente); Netflix, Youtube y Spotify utilizan el aprendizaje por refuerzo para desarrollar sistemas de recomendación de videos y música a nivel masivo; y tanto Google como Tesla lo usan para sus investigaciones en torno a vehículos autónomos.

A pesar de esto, el área del aprendizaje por refuerzo parece rezagarse en comparación a las demás, probablemente debido a la popularidad de otros tipos de tareas que se adaptan más a otro tipo de redes neuronales, como la clasificación de textos, detección de objetos o generación de datos a partir de otros ya existentes.

El valor académico de esta investigación yace en su enfoque hacia la generalización, la disminución de los ciclos de entrenamiento requeridos y el reuso del mismo modelo en varios entornos; enfoque comúnmente ignorado y pasado por alto para centrarse en otros indicadores como el rendimiento o la eficiencia de muestras. Esto permitirá disminuir el tiempo utilizado en diseñar y entrenar modelos para los investigadores e ingenieros, además de fomentar el desarrollo de futuros modelos y aportes a la academia.

PARTE II

DESCRIPCIÓN TEÓRICA

En esta sección se presentan los antecedentes del trabajo, las bases teóricas y legales del mismo, y se definen varios términos relevantes para el área de investigación.

2.1 Antecedentes

Kirk, Zhang, Grefenstette y Rocktaschel (2022) realizaron un trabajo titulado: A SURVEY OF ZERO-SHOT GENERALISATION IN DEEP REINFORCEMENT LEARNING (Estudio de la Generalización Zero-shot en el Reinforcement Learning Profundo), desarrollado como una encuesta, cuyo objetivo fue presentar un panorama de los últimos avances teóricos y metodólogicos en el área del Reinforcement Learning, específicamente aquellos relacionados con la generalización, y proponer áreas de interés para estudios futuros. Durante su investigación descubrieron que el estudio de la generalización Zero-shot (definida como aquel tipo de generalización aplicada a entornos completamente nuevos) es útil para crear algoritmos base que pueden ser usados para construir soluciones a dominios específicos. También encontraron que los entornos de "caja negra" (que no pueden ser cambiados, solo reciben entrada y producen salida) no son útiles para probar formas específicas de generalización y se recomienda usar factores controlables de variación al diseñar nuevos entornos.

Esta encuesta permitió conocer el estado actual de esta disciplina y de esa manera identificar técnicas y métodos que contribuyen a la generalización de los modelos, como la modificación de datos de entrada y políticas de Zero-shot generalisation. También formuló una manera para evaluar la generalización individual sin comparar con otros modelos al usar entornos de entrenamiento y prueba de la misma forma que se hace en el aprendizaje supervisado, ahorrando recursos y tiempo para el entremiento, y la recomendación de factores controlables para los entornos formó una gran parte del diseño de los entornos personalizados usados en este trabajo de investigación.

Huang, Julien, Ye, Braga, Chakraborty, Mehta y Araújo (2022) redactaron un documento técnico titulado: CLEANRL: HIGH-QUALITY SINGLE-FILE

IMPLEMENTATIONS OF DEEP REINFORCEMENT LEARNING ALGORITHMS (CleanRL: Implementaciones de Alta Calidad en un Solo Archivo de Algoritmos de Reinforcement Learning Profundo). En este documento presentaron una librería para el lenguaje de programación Python llamada CleanRL, la cual provee implementaciones de varios algoritmos de Reinforcement Learning con capacidad de extensión, experimentación, monitoreo de experimentos, personalización y con un enfoque hacia la claridad del código. Cada implementación se encuentra contenida en su propio archivo con la menor cantidad de dependencias posible, lo que permite entender y analizar las implementaciones con más facilidad que otras librerías. Esta librería ha sido de gran ayuda en varias investigaciones y proyectos tanto académicos como de producción desde su presentación, especialmente en aquellas que necesitan modificar elementos internos como la arquitectura de la red neuronal o el proceso de entrenamiento, cambios que requieren más esfuerzo y tienen una mayor curva de dificultad para implementar en otras librerías y frameworks.

Debido a que esta librería contiene varias implementaciones de Deep Q Learning, fue escogida como la librería de Reinforcement Learning a usar en este trabajo. La facilidad que posee para ser extendida resultó ser un factor determinante para escogerla en comparación a Stable Baselines o RL-Zoo, librerías modulares que ofrecen varias facilidades para el uso de modelos de Reinforcement Learning en producción, mas no poseen la flexibilidad necesaria para la experimentación extensiva.

Juliani, Berges, Teng, Cohen, Harper, Elion, Goy, Gao, Henry, Mattar, y Lange (2020) realizaron un documento de referencia titulado: UNITY: A GENERAL PLATFORM FOR INTELLIGENT AGENTS (Unity: Una Plataforma General para Agentes Inteligentes). Este documento estudia la posibilidad de usar Unity como motor de simulaciones para el entrenamiento de agentes de inteligencia artificial, examina el Unity ML-Agents Toolkit, una librería que permite esto al proveer una api compatible con OpenAI Gym fácilmente accesible desde Python; y discute toda clase de investigaciones que pueden realizarse con estas tecnologías. La popularidad de Unity como motor de simulación ha resultado en varios usos de esta librería por compañías para implementar modelos de

Reinforcement Learning en producción, como lo hizo Jam City con su juego Snoopy Pop. (Teng, E. 2019)

Este trabajo permitió conocer cómo usar estas herramientas para la creación de entornos relativamente complejos que tomarían demasiado tiempo para implementarse en pygame, cómo comunicarse e interactuar con estos entornos usando Python y varios casos en los que se ha usado esta librería y cómo fue implementada. Debido a estas y otras consideraciones se escogió a Unity como el motor de simulación para crear entornos personalizados a estudiar en este trabajo de investigación, usando el Unity ML-Agents Toolkit como una capa de abstracción para conectarse con la implementación en CleanRL mediante la API de OpenAI Gym y el Gymnasium de Farama Foundation.

2.2 Bases Teóricas

2.2.1 Inteligencia Artificial

López, B. (s/f) define a la inteligencia artificial como "(…) una rama de las ciencias computacionales encargada de estudiar modelos de cómputo capaces de realizar actividades propias de los seres humanos en base a dos de sus características primordiales: El razonamiento y la conducta."

Esta definición muestra la diferencia entre los algoritmos de inteligencia artificial con los tradicionales: La capacidad de los primeros de resolver actividades que requieren razonamiento humano, lo que les provee de una gran importancia, ya que abre las puertas para la solución y automatización de una gran cantidad de problemas que resultaban ser imposibles o sumamente costosos.

Por ejemplo, Pino, R., Gómez, A. y Martínez, N. (2001) proveen una lista de varias tareas en las que la inteligencia artificial posee ventajas y beneficios en comparación a los algortimos tradicionales, la cual incluye:

- Procesamiento de Lenguaje Natural. (NLP)
- Visión Artificial.
- Resolución de Problemas.
- Representación del Conocimiento y Razonamiento.

- Aprendizaje Automático.

La implementación de la inteligencia artificial define cómo ésta se enfrenta a los problemas, y existen varias orientaciones que guían este proceso. Álvarez, L. (1994) explica en su libro FUNDAMENTOS DE INTELIGENCIA ARTIFICIAL varias orientaciones de la inteligencia artificial, algunas de las cuales son:

- Orientación Simbólica: Planteada por Turing. Una máquina pensante que simule un cerebro infantil y pueda aprender para convertirse en un cerebro adulto.
- Orientación Neoconexionista: Simula la actividad del cerebro humano mediante redes neuronales. No busca reemplazar la orientación simbólica, mas bien perfeccionarla y fundamentar su diseño.
- Orientación Biocomputacional: Plantea la construcción de biocircuitos con moléculas biológicas que no requieran computadoras para operar, sino que actúen por cuenta propia como verdaderas computadoras biológicas.

Actualmente se estudia la orientación neoconexionista debido a que aún se requieren más avances en la ingeniería molecular para implementar la biocomputacional, y la simbólica no se adapta a entornos que requieran razonamiento humano de alta complejidad.

Los entornos tratados en este trabajo de investigación entran en esta categoría, y son demasiado complejos para ser resueltos con algoritmos tradicionales. Solo es posible completarlos mediante el uso de inteligencia artificial, más específicamente con el uso de modelos que pertenecen al área del Reinforcement Learning, la cual usa redes neuronales de la forma descrita en la orientación neoconexionista.

2.2.1.1 Reinforcement Learning

Sutton, R. y Barto, A. (2020) sostienen que:

> El Reinforcement Learning es un acercamiento computacional a entender y automatizar el aprendizaje dirigido por objetivos y la toma de decisiones. Se distingue de otros acercamientos computacionales por su énfasis en el aprendizaje del agente

mediante interacción directa con su entorno, sin requerir supervision de ejemplos o modelos completos del entorno.

Lo que diferencia al Reinforcement Learning de los demás tipos de inteligencia artificial es su enfoque a trabajar en entornos donde no se conoce toda la información de antemano, basando su aprendizaje en el ensayo y error y la interacción directa con el entorno.

Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., Lanctot, M., Sifre, L., Kumaran, D., Graepel, T., Lillicrap, T., Simonyan, K., y Hassabis, D. (2017) mostraron su eficacia en la demostración de Google DeepMind titulada MASTERING CHESS AND SHOGI BY SELF-PLAY WITH A GENERAL REINFORCEMENT LEARNING ALGORITHM (Dominando ajedrez y shogi mediante auto-juego con un algoritmo general de Reinforcement Learning), donde explican al algoritmo AlphaZero, el cual puede jugar tanto ajedrez, shogi y go a un mayor nivel que los mejores algoritmos contemporáneos en solo 24 horas usando técnicas de Reinforcement Learning, sin necesidad de estudiar jugadas humanas o conocer cualquier otra cosa sobre los juegos más allá de sus reglas y el estado actual del tablero.

2.2.1.2 Q-Learning profundo

Mnih, V. et al.(2013) plantearon el algoritmo base del Q-Learning profundo con el siguiente pseudocódigo:

Algorithm 1 Deep Q-learning with Experience Replay

Initialize replay memory $\mathcal{D}$ to capacity N
Initialize action-value function Q with random weights
for episode $= 1, M$ **do**
  Initialise sequence $s_1 = \{x_1\}$ and preprocessed sequenced $\phi_1 = \phi(s_1)$
  for $t = 1, T$ **do**
    With probability ϵ select a random action a_t
    otherwise select $a_t = \max_a Q^*(\phi(s_t), a; \theta)$
    Execute action a_t in emulator and observe reward r_t and image x_{t+1}
    Set $s_{t+1} = s_t, a_t, x_{t+1}$ and preprocess $\phi_{t+1} = \phi(s_{t+1})$
    Store transition $(\phi_t, a_t, r_t, \phi_{t+1})$ in $\mathcal{D}$
    Sample random minibatch of transitions $(\phi_j, a_j, r_j, \phi_{j+1})$ from $\mathcal{D}$
    Set $y_j = \begin{cases} r_j & \text{for terminal } \phi_{j+1} \\ r_j + \gamma \max_{a'} Q(\phi_{j+1}, a'; \theta) & \text{for non-terminal } \phi_{j+1} \end{cases}$
    Perform a gradient descent step on $(y_j - Q(\phi_j, a_j; \theta))^2$ according to equation 3
  end for
end for

Lo que hace este algoritmo es usar el algoritmo tradicional de Q-Learning y reemplazar la tabla Q con una red neuronal, convirtiéndolo en Q-Learning profundo. El algoritmo explora el entorno, guarda los estados que observa junto con las recompensas que obtuvo y procede a identificar las características del entorno que llevan a mayores recompensas dados el estado y un set de acciones. De esta manera, si el entorno puede generar una cantidad excesiva de estados, no hace falta popular toda una tabla para entrenar el modelo, ya que al usar una red neuronal se pueden extrapolar estados desconocidos mediante la experimentación en estados ya conocidos.

Comúnmente se suele usar el Q-learning junto con el Prioritized Experience Replay, técnica formulada por Schaul, T., Quan, J., Antonoglou, I., y Silver, D. (2016) en el trabajo del mismo nombre que acelera el entrenamiento al darle más importancia a las experiencias más recientes; y junto con la técnica propuesta por Hasselt, H., Guez, A., y Silver, D. (2015) en DEEP REINFORCEMENT LEARNING WITH DOUBLE Q-LEARNING, que permite mejorar la estabilidad del entrenamiento al crear una copia intermedia de la red neuronal que es modificada durante cada paso en vez de la principal, y las dos son sincronizadas cada cierta cantidad de pasos.

2.2.2 Red Neuronal

Ruiz, C. y Basualdo, M. (2001) definen a la red neuronal como: “Un nuevo sistema para el tratamiento de la información, cuya unidad básica de procesamiento está inspirada en la célula fundamental del sistema nervioso humano: la neurona”.

Las redes neuronales, como su nombre indica, se basan en el cerebro humano y sus neuronas para aprender un comportamiento complejo. Las neuronas guardan información, y las interacciones entre ellas resultan en comportamiento inteligente que posee memoria y puede corregir sus propios errores sin intervención externa.

Montesinos, O., Montesinos, A., y Crossa, J. (2022) dicen en su libro FUNDAMENTALS OF ARTIFICIAL NEURAL NETWORKS AND DEEP LEARNING (Fundamentos de las redes neuronales artificiales y aprendizaje profundo):

> La implementación práctica de las redes neuronales es posible debido al hecho de que son sistemas de cómputo

> masivamente paralelos hechos de un gran número de unidades básicas de procesamiento (neuronas) que están interconectadas y aprenden de su entorno (…) las redes neuronales no son más que simplemente modelos estadísticos generalizados no lineales.

La base teórica detrás de las redes neuronales se compone tanto del comportamiento del cerebro como de la estadística y el cálculo diferencial. La combinación interdisciplinaria de técnicas como el descenso de gradiente (cálculo de derivadas), ciclos de entrenamiento inspirados en los modelos heurísticos de la estadística y unidades básicas simples similares a las neuronas permite replicar el proceso de aprendizaje presente en la naturaleza, de forma artificial.

2.2.2.1 Elementos de una Red Neuronal

Ruiz, C. y Basualdo, M. (2001) explican que la red neuronal se compone de tres niveles o capas:

- Capa de Entrada: “Es la capa que recibe directamente la información de las fuentes externas de la red.”
- Capas Ocultas: “Son internas a la red y no tiene contacto directo con el entorno exterior. (…) Las neuronas de las capas ocultas pueden estar interconectadas de distintas maneras, lo que determina, junto con su número, las distintas topologías de redes neuronales.”
- Capas de Salida: “Transfieren información de la red hacia el exterior.”

Todas estas capas son necesarias con excepción de las ocultas, pues su número puede variar de cero hasta miles o millones de capas. De esta manera, las observaciones entran por la capa de entrada, son procesadas por las capas ocultas y finalmente se convierten en una acción, predicción o valor práctico por las capas de salida.

Cada capa está compuesta por varias neuronas o nodos. Montesinos, O. et al (2022) explican los elementos internos de la neurona:

- Valores de entrada: Los datos a ser procesados por la neurona. Pueden venir del exterior o de otras neuronas en una capa anterior.

- Pesos: Las variables que modifican a los valores de entrada. El valor de los pesos define la clase de transformación que se aplicará a los datos, y es la parte más importante, a tal punto que puede reconstruirse toda la red neuronal y su comportamiento aprendido con solo poseer los pesos de cada neurona. El proceso de “entrenar una red neuronal” se refiere a encontrar los pesos correctos para cada neurona.
- Bias o sesgo: La variable que indica el valor que debe tener el resultado de la transformación para que logre activar a la función de activación. Suele juntarse con los pesos debido a que cumplen un rol similar y se entrenan al mismo tiempo.
- Función de activación: Decide si el resultado de la transformación se transmitirá a la siguiente capa o se descartará, dependiendo de un límite establecido por el bias o sesgo. Hay varios tipos de funciones de activación, y es el elemento que permite que la red pueda resolver problemas no lineales.
- Valor de salida: El valor que se transmite a la siguiente capa, o al exterior si la neurona se encuentra en la capa de salida. Puede ser el resultado de la transformación de los valores de entrada por los pesos, o un valor nulo (cero) en caso de que este no supere el límite de la función de activación.

2.2.3 Entorno

El entorno y el agente son dos conceptos altamente relacionados. Según Sutton, R. y Barto, A. (2020):

> El que aprende y toma decisiones es llamado el agente. Las cosas con las que interactúa, comprendiendo todo fuera del agente, es llamado el entorno. Estos interactúan continuamente, el agente seleccionando acciones y el entorno respondiendo a estas acciones y presentando nuevas situaciones al agente. El entorno también da lugar a recompensas, valores numéricos especiales que el agente busca maximizar sobre el tiempo mediante sus acciones.

El entorno comprende todo con lo que el agente interactúa, todo lo que puede afectarlo o que este puede afectar. El entorno es tan importante como el propio agente, pues define el problema que se quiere resolver y un entorno mal implementado no puede ser resuelto por ningún agente, no importa que tan efectivo sea.

Cabe destacar que para convertir un problema a un entorno solo se requiere formular una función de recompensa y formato de observaciones apropiados. Debido a esto, también se pueden resolver problemas analíticos en áreas como la matemática económica y el procesamiento de imágenes al interactuar con ellos como entornos.

Por ejemplo, Cote, M., Kadar, A., Yuan, X., Kybartas, B., Barnes, T., Fine, E., Moore, J., Hausknecht, M., Tao, R., Asri, L., Adada, M., Tay, W., y Trischler, A. (2019) de Microsoft Research desarrollaron un entorno llamado TextWorld en TEXTWORLD: A LEARNING ENVIRONMENT FOR TEXT-BASED GAMES (TextWorld: Un entorno de aprendizaje para juegos basados en texto), el cual está inspirado por juegos de texto como Zork y provee una manera de entrenar a agentes en tareas basadas principalmente en texto usando los algoritmos de Reinforcement Learning existentes, los cuales no fueron planteados para tales tareas, pero aun así logran resolver los problemas planteados en el entorno de TextWorld.

2.2.3.1 Tipos de Entornos

Brockman, G., Cheung, V., Pettersson, L., Schneider, J., Schulman, J., Tang, J. y Zaremba, W. (2016) implementaron el OpenAI Gym, considerado una de las librerías más importantes para el Reinforcement Learning. Este define una manera estándar para interactuar con varios tipos de entornos. Existe una gran cantidad y continuamente se siguen creando más (pues existen tantos entornos como problemas que resolver), como ejemplo de algunos, estos son los tipos de entornos implementados en la versión inicial del OpenAI Gym:

- Control Clásico y Toy Text: “Tareas pequeñas de la literatura de Reinforcement Learning.”
- Algorítmicos: “Realiza cómputos tales como sumar números de varios dígitos y revertir secuencias.”
- Atari: “Juegos clásicos de Atari, con imágenes de pantalla o RAM como entrada, usando el Arcade Learning Environment.”
- Juegos de Mesa: La versión inicial del OpenAI Gym incluye el juego de Go en tableros de 9x9 y 19x19.

- Robots 2D y 3D: Simulaciones de control para robots implementadas con el motor físico MuJoCo.

Como dicho anteriormente, estos son solo una parte de los tipos de entornos existentes. Farama Foundation, la mayor organización de investigación dedicada al Reinforcement Learning en la actualidad, desarrolla y mantiene en la actualidad más de 20 proyectos, todos ellos enfocados exclusivamente en los entornos y sus tipos. Entre ellos se encuentran MiniWorld (Entornos 3D con observaciones visuales), MiniGrid (Entornos 2D basados en cuadrículas), ViZDoom (Entornos basados en el videojuego de 1993, Doom), Minari (Librería que permite recolectar observaciones de entornos y usarlas para entrenar agentes de forma offline) y PettingZoo, el cual fue el primer proyecto de la Farama Foundation; hecho por Terry, J., Black, B., Grammel, N., Jayakumar, M., Hari, A., Sullivan, R., Santos, L., Perez, R., Horsch, C., Dieffendahl, C., Williams, N., Lokesh, Y., y Ravi, P. (2021) e introducido en PETTINGZOO: GYM FOR MULTI-AGENT REINFORCEMENT LEARNING como el equivalente a Gym (en ese tiempo mantenido por OpenAI) para entornos con más de un agente.

2.2.4 Entrenamiento

Sutton, R. y Barto, A. (2020) hacen referencia a las cualidades que diferencian al entrenamiento de modelos de Reinforcement Learning: “La característica más importante que distingue al Reinforcement Learning de otros tipos de aprendizaje es que usa información de entrenamiento que evalúa las acciones tomadas en vez de instruir al dar las acciones correctas.”

En el Reinforcement Learning no es posible conocer la acción correcta a tomar la mayor parte del tiempo (Y algunas veces no hay una acción correcta), por lo que su entrenamiento toma un enfoque distinto. En vez de dejar que el agente realice una acción y luego decirle la correcta, el agente la realiza y esta es evaluada según la recompensa que obtiene del entorno; luego se le dice al agente si la decisión que tomó fue correcta o no basado en esa evaluación.

La forma en que el agente logra aprender luego del proceso anterior es debido a una técnica llamada propagación hacia atrás o retropropagación, formulada por Rumelhart,

D., Hinton, G., y Williams, R. (1986) en LEARNING REPRESENTATIONS BY BACK-PROPAGATING ERRORS (Aprendiendo representaciones al retropropagar errores). Esta técnica forma la base del entrenamiento de redes neuronales y permite aproximar el impacto de cada neurona del agente en la decisión final mediante el cálculo de derivadas parciales, y ajustar los pesos de cada una apropiadamente.

2.2.5 Generalización

Cobbe, K., Klimov, O., Hesse, C., Kim, T. y Schulman, J. (2019) hablan sobre los problemas de generalización que se presentan en el Reinforcement Learning:

> Generalizar entre tareas sigue siendo difícil para algoritmos de vanguardia de Deep Reinforcement Learning. Aunque agentes entrenados pueden resolver tareas complejas, les cuesta transferir su experiencia a nuevos entornos. Agentes que han amaestrado diez niveles en un videojuego suelen fallar catastróficamente al encontrar el onceavo por primera vez. (…) En resumen, los agentes se especializan en sobremanera a los entornos encontrados durante el entrenamiento.

El Reinforcement Learning se basa en resolver problemas en entornos complejos difíciles de analizar y definir, y la generalización juega un factor importante en el proceso pues, sin ella, los agentes “fallan catastróficamente”; en otras palabras, son incapaces de actuar frente a escenarios nuevos, lo cual rompe con el propósito del Reinforcement Learning.

Dicho esto, existen varios tipos de generalización y formas de aumentarla. Un ejemplo de esto son la generalización Zero-shot y el Transfer Learning, este último usado con frecuencia en varias ramas de la inteligencia artificial. Según Zhuang, F., Qi, Z., Duan, K., Xi, D., Zhu, Y., Zhu, H., Xiong, H., y He, Q. (2019):

> El Transfer Learning busca mejorar el rendimiento de los alumnos objetivo en los dominios de destino al transferir los conocimientos obtenidos en dominios de origen diferente pero relacionados. De esta manera, la dependencia en una gran cantidad de datos del dominio de destino puede reducirse al construir alumnos.

Se puede entrenar un agente en un entorno, y luego entrenarlo de nuevo en otro entorno similar. Esto se conoce como Transfer Learning y se usa para reducir el tiempo y recursos requeridos en el entrenamiento, ya que permite usar modelos entrenados previamente como base para tareas más específicas, con la condición de que los dos entornos deben ser lo suficientemente similares como para que pueda haber transferencia de conocimientos entre ellos.

2.3 Bases Legales

2.3.1 Ley Orgánica de Ciencia, Tecnología e Innovación

Art. 27.- A los fines de la presente Ley, las siguientes actividades serán consideradas como factibles de ser llevadas a cabo con los aportes a la ciencia, la tecnología, la innovación y sus aplicaciones:

1. Proyectos de innovación relacionados con actividades que involucren la obtención de nuevos conocimientos o tecnologías en el país, con participación nacional en los derechos de propiedad intelectual, en las áreas prioritarias establecidas por la autoridad nacional, con competencia en materia de ciencia, tecnología, innovación y sus aplicaciones:
    a. Sustitución de materias primas o componentes para disminuir importaciones o dependencia tecnológica.
    b. Creación de redes productivas nacionales.
    c. Utilización de nuevas tecnologías pura incrementar la calidad de las unidades de producción.
    d. Participación, investigación e innovación de las universidades y centros de investigación e innovación del país, en la introducción de nuevos procesos tecnológicos, esquemas organizativos, obtención de nuevos productos o de procedimientos, exploración de necesidades y, en general, procesos de innovación con miras a resolver problemas concretos de la población venezolana.
    e. Formación de cultores o cuadros científicos y tecnológicos en normativa, técnicas, procesos y procedimientos de calidad.

f. Procesos de transferencia de tecnología dirigidos a la producción de bienes y servicios en el país, que prevean la formación de cultores o cuadros científicos y tecnológicos en lo técnico, operativo, profesional y científico.

De acuerdo a la norma precitada, la Ley Orgánica de Ciencia, Tecnología e Innovación fomenta y regula las actividades de investigación tecnológica del país. Varias de estas actividades se refieren a sistemas de producción y procesos de calidad, dos áreas donde la inteligencia artificial provee una gran cantidad de ventajas. Los sistemas de producción involucran procesos repetitivos y controlados con muchas oportunidades para recolectar información que puede usarse para entrenar agentes de inteligencia artificial, lo que los vuelve un objetivo ideal para la implementación de los mismos. No solo eso, uno de los principales usos que se le da al Reinforcement Learning es el control de robots como los usados en las plantas de ensamblaje y fábricas, y aumentar la generalización de estos sistemas permite disminuir la probabilidad de que ocurra un error durante su funcionamiento debido a circunstancias inusuales.

2.4 Definición de Términos

Deep Learning

"El deep learning es un tipo de machine learning que entrena a una computadora para que realice tareas como las hacemos los seres humanos, como el reconocimiento del habla, la identificación de imágenes o hacer predicciones. En lugar de organizar datos para que se ejecuten a través de ecuaciones predefinidas, el deep learning configura parámetros básicos acerca de los datos y entrena a la computadora para que aprenda por cuenta propia reconociendo patrones mediante el uso de muchas capas de procesamiento." (SAS)

Generalización

"Acción y efecto de generalizar." (RAE)

Generalizar

“Abstraer lo que es común y esencial a muchas cosas, para formar un concepto general que las comprenda todas.” (RAE)

Inteligencia Artificial

“Disciplina científica que se ocupa de crear programas informáticos que ejecutan operaciones comparables a las que realiza la mente humana, como el aprendizaje o el razonamiento lógico.” (RAE)

Machine Learning

“El ‘machine learning’ –aprendizaje automático– es una rama de la inteligencia artificial que permite que las máquinas aprendan sin ser expresamente programadas para ello.” (BBVA)

Red Neuronal

“Una red neuronal es un procesador distribuido en paralelo de forma masiva con una propensión natural a almacenar conocimiento experimental y convertirlo en disponible para su uso.” (IBM)

Reinforcement Learning

“Reinforcement learning es una técnica de machine learning en la que un agente informático aprende a realizar una tarea a través de repetidas interacciones de prueba y error con un entorno dinámico.” (Mathworks)

PARTE III

DESCRIPCIÓN METODOLÓGICA

En esta sección se describe la naturaleza, tipo y diseño de la investigación realizada, la población y muestra de estudio, y las técnicas usadas durante la recolección y análisis de datos.

3.1 Naturaleza de la investigación

Se escoge un modelo cuantitativo para la investigación, pues su naturaleza permite que la recolección de información sea un proceso simple. Según Ortega, C. (2022), “La investigación cuantitativa consiste en recolectar y analizar datos numéricos. Este método es ideal para identificar tendencias y promedios, realizar predicciones, comprobar relaciones y obtener resultados generales de poblaciones grandes.”

La mayor parte de la información tratada en este trabajo es cuantitativa, debido a que las variables de estudio se comprenden de datos numéricos, lo que hace que trabajar con un modelo cuantitativo sea lo ideal en este trabajo.

3.1.1 Tipo de investigación

Se trata de una investigación descriptiva, ya que busca obtener nuevos datos al experimentar, crear y evaluar modelos, para luego analizarlos. Según Muguira, A. (2022):

> La investigación descriptiva se encarga de puntualizar las características de la población que está estudiando. (…) su objetivo es describir la naturaleza de un segmento demográfico, sin centrarse en las razones por las que se produce un determinado fenómeno. Es decir, “describe” el tema de investigación, sin cubrir “por qué” ocurre.

En este trabajo se estudian modificaciones que pueden mejorar la generalización de los modelos, mas no estudia el por qué o cómo logran mejorarla; se limita a describir sus efectos y sacar conclusiones a partir de ello.

El trabajo también corresponde a un proyecto factible debido a que el modelo se va a implementar en código. Según Pérez, J. y Merino, M. (2013): “La noción de proyecto

factible refiere a aquellas propuestas que, por sus características, pueden materializarse para brindar solución a determinados problemas.”

Como el modelo tiene su implementación, este puede ser usado en la práctica como un programa más; solo se debe cargar en un framework de inferencia de inteligencia artificial (como Tensorflow o Pytorch) y de esa forma está listo para ser integrado en cualquier sistema.

3.1.2 Diseño de la investigación

La investigación tiene el diseño de una investigación documental, debido a que se basa en el conocimiento y técnicas propuestas en trabajos previos del área. Según Ortega, C. (2022): “La investigación documental es una técnica de investigación cualitativa que se encarga de recopilar y seleccionar información a través de la lectura de documentos, libros, revistas, grabaciones, filmaciones, periódicos, bibliografías, etc.”

Durante el trabajo se revisa la documentación de las librerias y frameworks a utilizar, así como otros trabajos recientes que introduzcan nuevos enfoques al Reinforcement Learning; estos se toman como referencia para la implementación, y todo este proceso constituye una investigación documental.

3.1.3 Objeto de estudio

Según Pérez, J. y Merino, M. (2019): “La noción de objeto de estudio se emplea en el ámbito de la ciencia para referirse a un tema de investigación. El objeto de estudio puntualiza qué se analizará y cómo se llevará a cabo la tarea.”. Es decir, es el tema que se está investigando y el alcance del mismo.

El objeto de estudio son los modelos de Reinforcement Learning que usan Q-learning profundo y los entornos en los que estos pueden actuar. Más específicamente, se investigan las implementaciones de Q-learning profundo hechas por CleanRL y Stable Baselines, y aquellos entornos 2D que permitan su uso mediante la API de OpenAI Gym o Gymnasium. Estos son elegidos mediante la revisión de documentación, basándose en el rendimiento de los modelos y los objetivos de cada entorno.

3.1.4 Técnicas de recolección de datos

Los datos se obtienen mediante la revisión documental, pues se modifican activamente los modelos y entornos durante su investigación. Según Mendez, D. (2010):

> Es una técnica de observación complementaria, en el caso de un registro de acciones y programas. La revisión documental permite hacer una idea del desarrollo y las características de los procesos y también la información que se confirma o se pone en duda.

Los datos resultantes se almacenan en las plataformas de Weights & Biases y Hugging Face, las cuales proveen herramientas de registro y monitoreo de experimentos especializadas para la inteligencia artificial.

Cuando empieza el proceso de entrenamiento de un modelo, el programa guarda los parámetros necesarios para recrear el experimento (tasa de aprendizaje, tasa de exploración, código fuente, etc.). Mientras se está entrenando, se guardan registros locales de varias estadísticas e indicadores que permiten monitorear el progreso y ofrecen la información necesaria para el análisis del efecto de la estructura de red neuronal, el entorno y el entrenamiento en sí en la generalización (Algunos de estos indicadores son la recompensa obtenida, la pérdida o “loss”, la longitud de los episodios, utilización de recursos del sistema, videos de demostración, etc.). Una vez terminado este proceso, el programa se comunica con el servidor de Weights and Biases y le envía todos estos datos junto con el modelo ya entrenado para su almacenamiento y análisis.

Para evaluar la generalización de los modelos y recolectar datos para compararlos, éstos se pueden descargar del servidor directamente y realizar un proceso de evaluación similar al de entrenamiento sin modificar los pesos o la estructura interna de la red; solo analizando su rendimiento en la tarea asignada y guardando los resultados obtenidos dentro del mismo servidor de Weights and Biases.

Si las características del modelo muestran una mejora significativa de la generalización al igual que alta estabilidad en las pruebas, este se puede subir a un repositorio en Hugging Face para que pueda usarse con mayor facilidad y esté disponible al público mediante demostraciones interactivas y descargas directas.

3.1.5 Técnicas de análisis de datos

Los datos se analizan mediante pruebas estadísticas, gráficos y tablas comparativas. El objetivo es identificar las técnicas y modificaciones aplicadas que produzcan una mejor generalización y la magnitud de tales mejoras. Se utilizan las librerías de Python numpy, pandas y scipy para los cálculos y análisis; mientras que se usan las librerías seaborn y plotly para los gráficos, tablas y visualizaciones gráficas.

Weights and Biases provee visualizaciones gráficas de datos almacenados por defecto, por lo que estas son usadas para los parámetros de entrenamiento y evaluación que permiten analizar los efectos de la red neuronal, los entornos y el proceso de entrenamiento.

Para las comparaciones más complejas, tablas y pruebas estadíticas (como las necesarias para evaluar la generalización de los modelos) se utilizan Jupyter Notebooks alojados en Google Colab y Deepnote. Estos servicios proveen entornos de python enfocados a la ciencia de datos y a la demostración y explicación de información cuantitativa. El notebook se conecta con el servidor de Weights and Biases para obtener los datos de los entrenamientos y las evaluaciones, analizarlos con scipy y luego visualizarlos con plotly y seaborn para su análisis cuantitativo.

Para el análisis cualitativo, se descarga el modelo directamente desde el servidor de Weights and Biases o el repositorio de Hugging Face, se carga dentro de Pytorch y luego se le da acceso al modelo para que interactúe con un entorno de prueba y se pueda verificar su rendimiento en situaciones prácticas.

PARTE IV

ANÁLISIS Y PRESENTACIÓN DE RESULTADOS

En esta sección se presentan los resultados del análisis de los datos obtenidos durante el proyecto de investigación.

4.1 Determinación de una estructura de red neuronal que permita la adaptación a varios entornos

Este objetivo consiste en identificar una de las formas que puede tener una red neuronal, con el fin de que pueda ser empleada en varios entornos sin que tenga que ser modificada.

Para llevar a cabo dicho procedimiento, se recolectaron varias arquitecturas y estructuras de redes neuronales usando las plataformas de GitHub y Hugging Face, seleccionando aquellas que cumplieran las siguientes propiedades específicas:

- Permite el entrenamiento mediante el Q-learning profundo.
- Permite la modificación de su código interno.
- Permite ser integrado con herramientas de registro de experimentos como Weights and Biases, de manera que se puede registrar información personalizada.

Para su análisis, estas estructuras fueron implementadas en Google Colab y Azure Machine Learning, tomando en cuenta sus requisitos, su facilidad de integración y su eficacia al resolver los problemas planteados. Estas plataformas representan las dos maneras principales en que se interactúan con modelos de inteligencia artificial (Jupyter Notebooks y máquinas virtuales), por lo que el proceso de implementación en las mismas puede generalizarse a otras plataformas como servidores físicos.

En efecto, se evaluaron las estructuras presentadas por 3 frameworks distintos:

Tabla 1: Comparación de frameworks de Reinforcement Learning

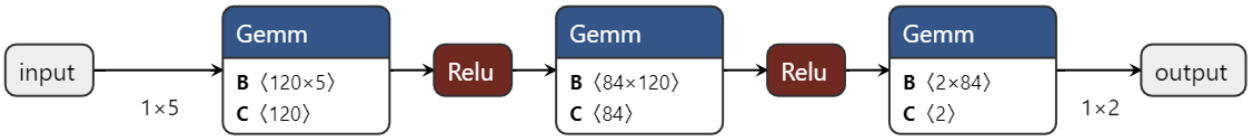
Framework	Facilidad de uso	Capacidad de modificación

RL Petting Zoo	Fácil de usar e implementar. (solo requiere 2 comandos)	Limitada, debido a que es un framework grande con varios módulos.
Stable Baselines 3	Más difícil de usar que Petting Zoo.	Mayor grado de libertad, pero aún así sigue siendo grande con varios módulos.
CleanRL	Relativamente fácil de usar.	Provee el mayor grado de libertad ya que encapsula sus implementaciones en archivos independientes

Fuente: Elaboración propia

Como resultado, se escogieron las estructuras presentadas por CleanRL como base para el framework personalizado y especializado en el Q-learning profundo que se creó y usó en este proyecto de investigación, con el nombre de "Lagomorph" (https://github.com/odiaz1066/lagomorph), el cual usa dos estructuras de red neuronal, cada una para tipos de entornos distintos:

Figura 1: Estructura de red neuronal para observaciones vectoriales.

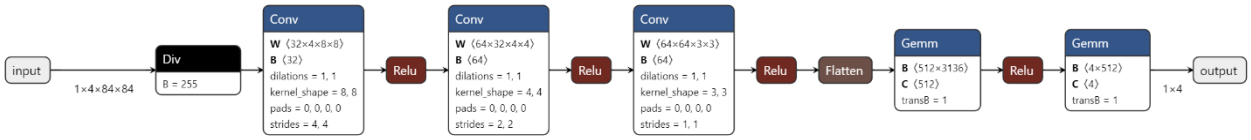


Fuente: Elaboración propia. (2023)

Esta primera estructura se usa en entornos con observaciones vectoriales, es decir, las observaciones son puramente numéricas y directamente representan el estado del entorno. Por ejemplo, este gráfico permite visualizar la estructura del modelo final entrenado para el entorno "CartPoleGoal-v1". Como puede apreciarse, el modelo recibe un vector de 5 números como entrada (representando la posición y velocidad del carro, ángulo y velocidad angular del poste y la posición de la meta), procesa esta entrada a través de varias capas que realizan operaciones de multiplicación de matrices y filtran sus resultados mediante la función Relu; y por último se presenta la salida del sistema

como un vector de 2 números (las dos acciones disponibles en el entorno) que representan las probabilidades de que su respectiva acción lleve a una recompensa mayor.

Figura 2: Estructura de red neuronal para observaciones visuales.



Fuente: Elaboración propia. (2023)

La figura 2 muestra otra estructura, usada en entornos con observaciones visuales. Por ejemplo, este gráfico representa la estructura que usa el modelo entrenado para el entorno “BreakoutNoFrameskip-v4”. Como puede verse, esta estructura es más larga y posee más capas debido a que los entornos visuales son más complejos que los vectoriales. Su entrada es un vector de 4 observaciones visuales, y cada una corresponde a una matriz de 84x84, haciendo referencia al valor de los píxeles visualizados en escala de grises (Se usa dicha cantidad a la vez para que el modelo pueda inferir el estado actual del entorno en el tiempo); estas visualizaciones son procesadas por 3 capas convolucionales que se encargan de transformarlas y reducirlas a las propiedades relevantes para resolver el problema, el resultado es procesado por otras capas de multiplicación por matrices similares a la primera estructura, y al final la salida es un vector de 4 números representando las 4 acciones posibles en el entorno.

Las dos estructuras presentadas anteriormente han probado ser útiles y efectivas en un alto rango de entornos, tanto vectoriales como visuales, pues están basadas en los experimentos de Mnih, V. et al.(2013). La estructura encargada de las observaciones visuales está basada en la presentada en el documento original, modificada por el equipo de CleanRL para optimizar el entrenamiento y obtener mejores resultados con menos pasos; mientras que la encargada de las observaciones vectoriales es una versión más simple de la visual que comparte los detalles de su implementación a excepción de los hiperparámetros predeterminados. Debido a esto, se determinó que estas son idóneas para la adaptación a varios entornos.

4.2 Identificación de entornos que tengan el potencial para aumentar la generalización del modelo durante y después del entrenamiento

Los entornos son tan importantes como la estructura del modelo, por lo que este objetivo consiste en identificar entornos que permitan que el modelo pueda generalizarse y ser usado en varias tareas.

Para lograr esto, se recolectaron varios entornos encontrados en Gymnasium, una librería basada en el OpenAI Gym y mantenida por Farama Foundation para proveer una API estándar con la cual entrenar modelos de Reinforcement Learning. También se estudiaron otros que proveen Unity ML Agents y MiniWorld, este último también mantenido por Farama Foundation.

Estas librerías fueron escogidas debido a que son usadas y referenciadas frecuentemente: Una gran parte de las librerías de Reinforcement Learning siguen la API de Gymnasium para interactuar con ellas, Unity ML Agents es la solución más popular para para crear agentes dentro del motor de Unity, y MiniWorld posee un motor relativamente simple, además de que puede integrarse a Gymnasium fácilmente.

Se analizó la representación, tanto vectorial como visual, de varios entornos en busca de características idóneas para la generalización. Algunas de las propiedades deseadas son:

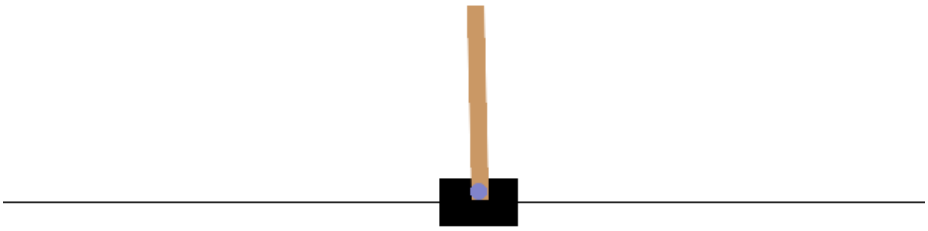
- Baja complejidad, para que se pueda experimentar rápidamente con los mismos.
- Alta similaridad en la representación matemática de los entornos seleccionados.

Estas propiedades resultan en un set de entornos relativamente simples con mecánicas suficientemente similares como para que lo aprendido en uno pueda ser usado en otro, pero también con los aspectos necesarios diferentes, que permitan que la tarea a resolver no sea exactamente la misma.

Dado el criterio mencionado anteriormente, se decidieron usar los siguientes entornos para el entrenamiento:

- **CartPole-v1**

Figura 3: Imagen del entorno CartPole-v1



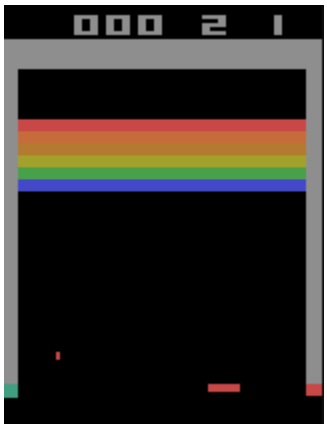
Fuente: Farama Foundation. (2022)

En este entorno, el agente controla un carro con un poste inestable en su centro. El objetivo del agente es equilibrar el poste al mover el carro a la izquierda o a la derecha, evitando que se caiga. Este es un entorno vectorial.

  - Observación: Vector de 4 valores (Posición del carro, velocidad del carro, ángulo del poste, velocidad angular del poste).
  - Acciones: 2 (Mover carro a la izquierda o a la derecha)

- **BreakoutNoFrameskip-v4**

Figura 4: Imagen del entorno BreakoutNoFrameskip-v4



Fuente: Farama Foundation. (2022)

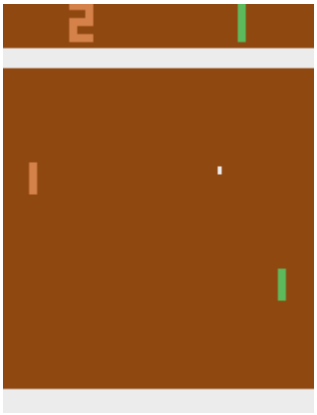
Este entorno está basado en el juego de Atari “Breakout”. El agente controla una paleta y su objetivo es hacer rebotar una pelota para que esta choque y destruya todos

los bloques en la pantalla, evitando que la pelota caiga fuera de la misma. Este es un entorno visual.

- Observación: Vector de 4x84x84 valores(4 imágenes, 84 píxeles de ancho y 84 píxeles de alto, escala de grises).
- Acciones: 4 (Hacer nada, Mover izquierda, Mover derecha, Acción primaria).

- **PongNoFrameskip-v4**

Figura 5: Imagen del entorno PongNoFrameskip-v4



Fuente: Farama Foundation. (2022)

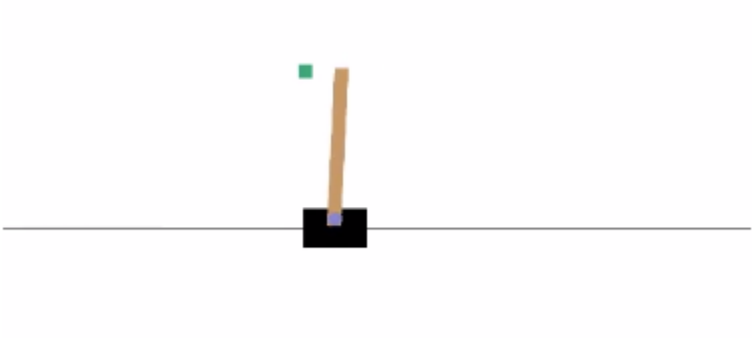
Este entorno está basado en el juego de Atari “Pong”. El agente controla una paleta y su objetivo es devolver la pelota hacia el lado de su oponente, buscando que este falle en devolvérsela y conseguir un punto, al mismo tiempo que protege su lado de la pantalla para que su oponente no consiga puntos.

- Observación: Vector de 4x84x84 valores(4 imágenes, 84 píxeles de ancho y 84 píxeles de alto, escala de grises).
- Acciones: 6 (Hacer nada, Mover izquierda, Mover derecha, Acción primaria, Gatillo izquierdo, Gatillo derecho).

Además de estos, durante el desarrollo del proyecto de investigación se decidió crear un entorno personalizado basado en CartPole-v1 y que ha sido agregado a Lagomorph:

- **CartPoleGoal-v1**

Figura 6: Imagen del entorno CartPoleGoal-v1



Fuente: Elaboración propia. (2023)

Este entorno es una modificación del entorno “CartPole-v1”. El agente controla un carro conectado a un poste inestable, y su objetivo es mover el carro a la meta (cuya posición es indicada por un punto verde) y balancear el poste allí, sin dejar que el poste se caiga en ningún momento. Este es un entorno vectorial.

- Observación: Vector de 5 valores (Posición del carro, velocidad del carro, ángulo del poste, velocidad angular del poste, posición de la meta).
- Acciones: 2 (Mover carro a la izquierda o a la derecha)

Estos entornos fueron elegidos debido a su similaridad en representación matemática (el rango de movimiento del agente puede representarse en un vector unidimensional incluso en los entornos visuales) y su baja complejidad, lo cual ayuda a su capacidad de generalización. Esto es importante ya que el mundo real no se adhiere a las situaciones matemáticamente perfectas encontradas en los entornos simulados, por lo que se requiere que el agente pueda generalizar los conocimientos obtenidos en los entornos que observa, para así aplicarlos a otros que no haya visto.

4.3 Definición de un proceso de entrenamiento que permita la experimentación rápida y efectiva de varios modelos en los entornos elegidos

El proceso de entrenamiento también influye en el rendimiento final de los modelos y la capacidad de estudiarlos. Es por ello que definir un proceso que permita experimentar rápidamente con varios modelos y recopilar información sobre los mismos es fundamental.

Por tal motivo, se implementaron flujos de trabajo en varias plataformas como Google Colab, Azure Machine Learning, Kaggle y equipos locales, y se registraron observaciones en cuanto a las características de cada flujo de trabajo, sus ventajas y desventajas.

Una vez probadas varias maneras de realizar el entrenamiento y recolectadas las observaciones, estas se analizaron y compararon en relación a las siguientes características: Facilidad de instalación, facilidad de uso, estabilidad, límites y recursos necesarios:

Tabla 2: Comparación de plataformas para entrenar inteligencia artificial.

Plataforma	Facilidad		Estabilidad	Límites	Recursos necesarios
	Instalación	Uso			
Equipo Local	Difícil de instalar, especialmente si se tienen otros proyectos en el equipo que usen python. Además, el soporte para GPUs es limitado en la actualidad y requiere instalación y mantenimiento delicado de drivers.	Relativamente fácil debido a que se tiene control completo de la máquina; una vez instalado, puede ejecutarse fácilmente y permite crear prototipos rápidamente.	Depende de la carga de trabajo del equipo. Si solo se usa para entrenamiento, es estable. Si se usa para otras cosas mientras se entrena el modelo, entonces el rendimiento puede ser inestable si no se tiene suficiente	Se limita al hardware instalado en el equipo, lo que significa que para mejorar el rendimiento se debe comprar e instalar partes al equipo con el riesgo de romperlo si se hace incorrectamente, o de que estos se averíen en	Los modelos de reinforcement learning requieren una cantidad relativamente baja de memoria en comparación con otros tipos de inteligencia artificial, pero aún así requiere una cantidad considerabl

			memoria o capacidad de procesamiento.	algún momento.	e. 8 GB de RAM como mínimo. También es altamente recomendable instalar un GPU con soporte para CUDA ya que la mayor parte de las librerías y frameworks de inteligencia artificial están optimizadas para trabajar con ellos y agilizar enormemente el entrenamiento; de lo contrario, este puede ser

					excesivamente lento.
Google Colab	Extremadamente fácil de instalar, solo requiere 2 comandos para instalar Lagomorph (3 o 4 si se requieren observaciones visuales).	Fácil de usar para entrenamiento; si se necesitan editar archivos (al crear prototipos, por ejemplo) se vuelve más complejo e incómodo. No tiene persistencia, por lo que deben reinstalarse todos los archivos y librerías cada vez que se reinicia.	Desconexiones y paradas repentinas son comunes en la plataforma, por lo que se requieren hacer respaldos externos de vez en cuando para entrenamientos más largos.	Como dicho anteriormente, el sistema de archivos no es persistente, por lo que se requieren tomar medidas preventivas para entrenamientos largos. También posee límites de cómputo a menos que se use algunos de los planes pagos, pero estos son suficientes para modelos de Reinforcement Learning.	Conexión a internet y una cuenta de Google.

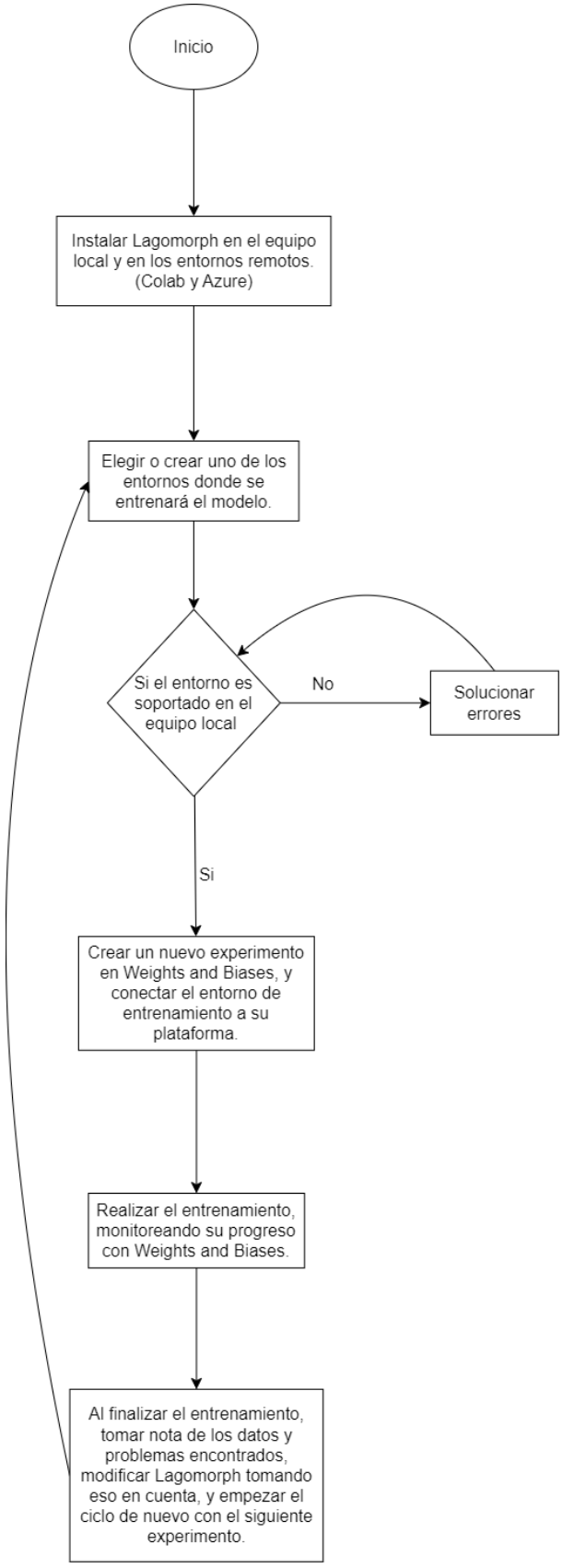
Azure Machine Learning	Complicado de configurar e instalar, pues requiere navegar la plataforma general de Azure encima de la instalación de las librerías necesarias.	Relativamente fácil de usar una vez instalado, mas aún así tiene otros inconvenientes, como la dificultad para administrar el almacenamiento usado.	Moderadamente estable. La plataforma en sí posee buena infraestructura y no tiende a detenerse repentinamente, pero en el área de usuario hay que prestar atención pues es fácil que aparezcan problemas allí.	Azure limita bastante las configuraciones de máquinas disponibles para estudiantes, pero las disponibles son suficientes para realizar Reinforcement Learning.	Conexión a internet, cuenta de Microsoft. Créditos limitados para estudiantes.
Kaggle	Fácil de instalar, 3 o 4 comandos y listo.	Es un poco más complicado de usar que Colab, pero su uso es similar.	Inestable. Puede detener ejecución en cualquier momento y su capacidad de respaldo externo no es tan buena como la de	Poco tiempo de procesamiento disponible, almacenamiento persitente pero difícil de acceder.	Conexión a internet, cuenta de Kaggle, número telefónico.

			Colab, por lo que es más difícil entrenar un modelo complejo.		

Fuente: Elaboración propia.

Basándose en estas observaciones, se escogieron las plataformas de Colab y Azure Machine Learning para entrenamiento y equipos locales para el desarrollo y programación de prototipos. Para ello se creó un framework basado en CleanRL llamado "Lagomorph" (https://github.com/odiaz1066/lagomorph) y se realizaron los experimentos de la siguiente manera:

Figura 7: Diagrama de flujo de los experimentos realizados con Lagomorph.


Inicio
Instalar Lagomorph en el equipo local y en los entornos remotos. (Colab y Azure)
Elegir o crear uno de los entornos donde se entrenará el modelo.
Si el entorno es soportado en el equipo local
No
Solucionar errores
Si
Crear un nuevo experimento en Weights and Biases, y conectar el entorno de entrenamiento a su plataforma.
Realizar el entrenamiento, monitoreando su progreso con Weights and Biases.
Al finalizar el entrenamiento, tomar nota de los datos y problemas encontrados, modificar Lagomorph tomando eso en cuenta, y empezar el ciclo de nuevo con el siguiente experimento.

Fuente: Elaboración propia.

4.4 Evaluación de la generalización de los modelos resultantes en comparación a los estudiados.

Para que los modelos puedan usarse en situaciones prácticas, estos deben ser capaces de actuar de forma correcta en una gran variedad de situaciones. Para ello, se debe evaluar su capacidad de generalización en varios entornos.

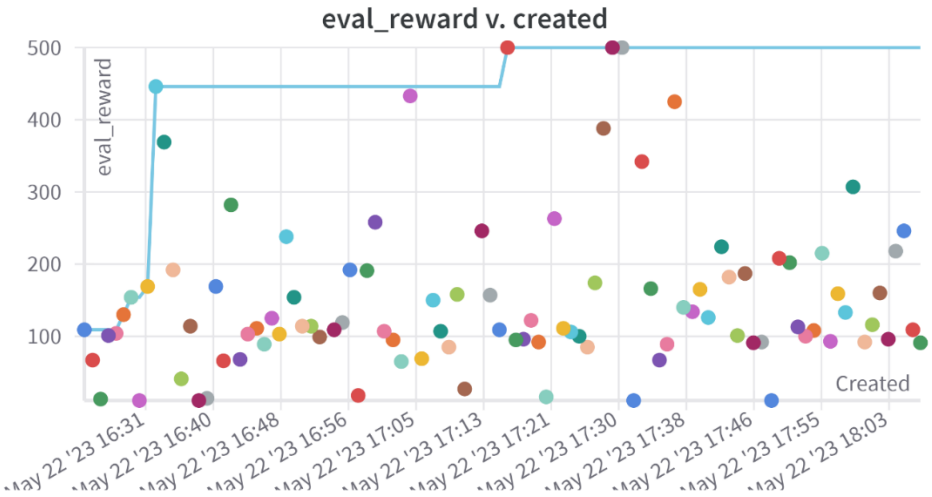
Se recolectaron los datos de cada experimento usando la plataforma de Weights and Biases. Se registraron las recompensas en cada episodio, los valores de pérdida y valores Q, recompensas de evaluación y videos de cada modelo resolviendo un entorno.

Una vez estos datos fueron almacenados en Weights and Biases, se graficaron usando las funciones nativas de la plataforma para ser analizadas. Además, se extrajo esta información a Deepnote para procesarla con Scipy y Seaborn y obtener más conclusiones y conocimiento sobre los modelos.

A continuación se presentan los resultados de los entrenamientos realizados y los modelos producidos:

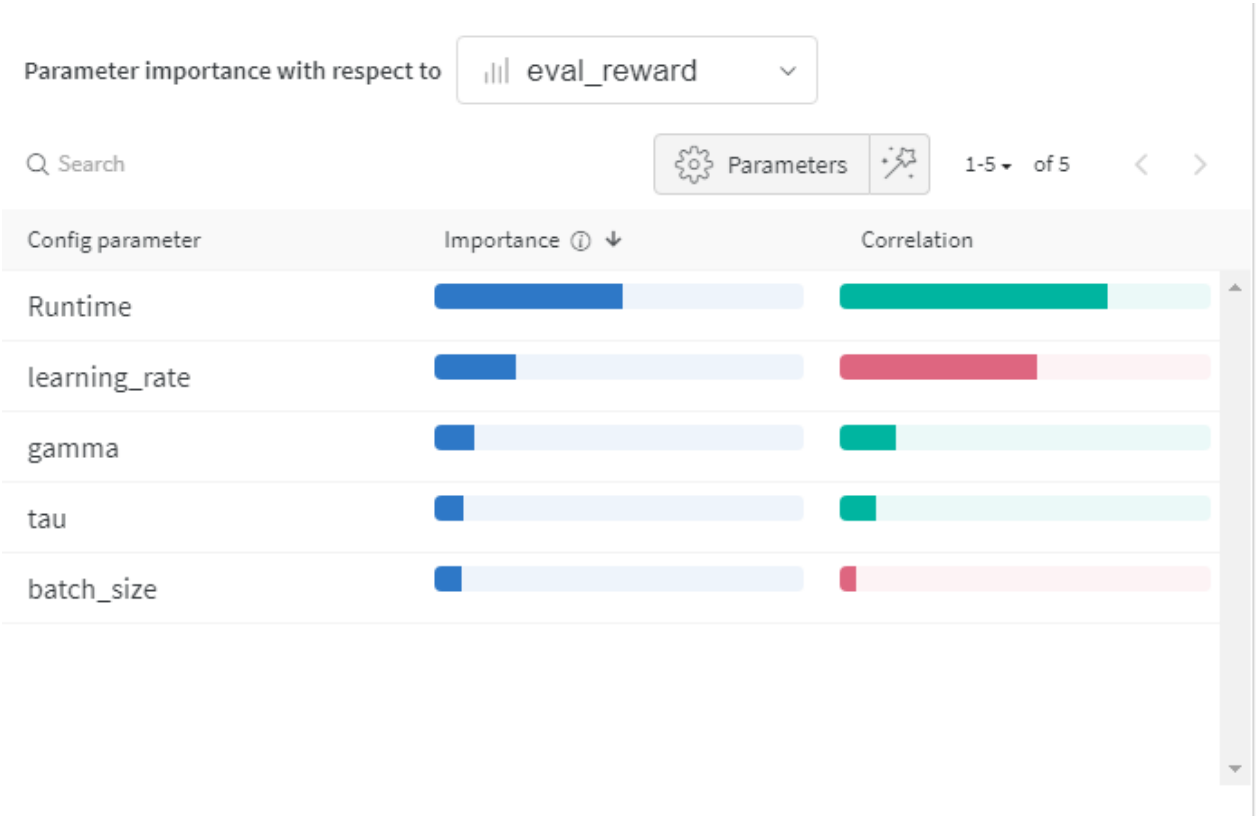
- CartPole-v1: Al inicio del entrenamiento se realizó una sesión de optimización de hiperparámetros con los siguientes resultados:

Figura 8: Experimentos de optimización de hiperparámetros, graficados por fecha de inicio y recompensa obtenida.



Fuente: Elaboración propia. (2023)

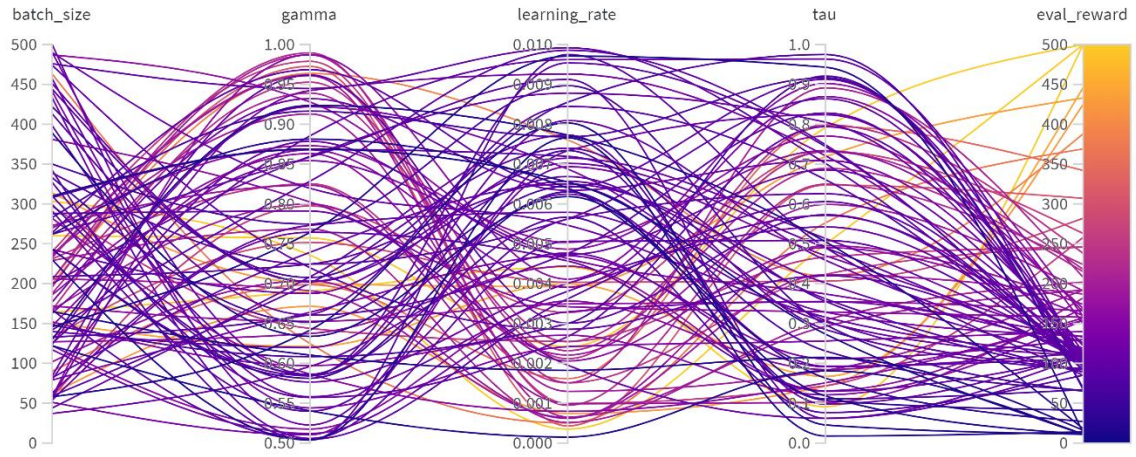
Figura 9: Correlación entre hiperparámetros y recompensa obtenida.



Fuente: Elaboración propia. (2023)

En la primera figura se muestran las diferentes pruebas hechas al igual que la recompensa máxima que alcanzaron. A partir de los datos de estas pruebas se pudo establecer la correlación entre la recompensa y los hiperparámetros, la cual se puede ver en la segunda figura. Según los datos obtenidos, el parámetro con más influencia es la tasa de aprendizaje (learning rate), que tiene una relación inversa con la recompensa (mayor tasa de aprendizaje resulta en recompensas menores), lo cual tiene sentido debido a que una tasa de aprendizaje alta hace que el entrenamiento se vuelva inestable y no converja en una solución. A continuación se presenta un gráfico con todas las pruebas hechas junto con sus parámetros y recompensas finales:

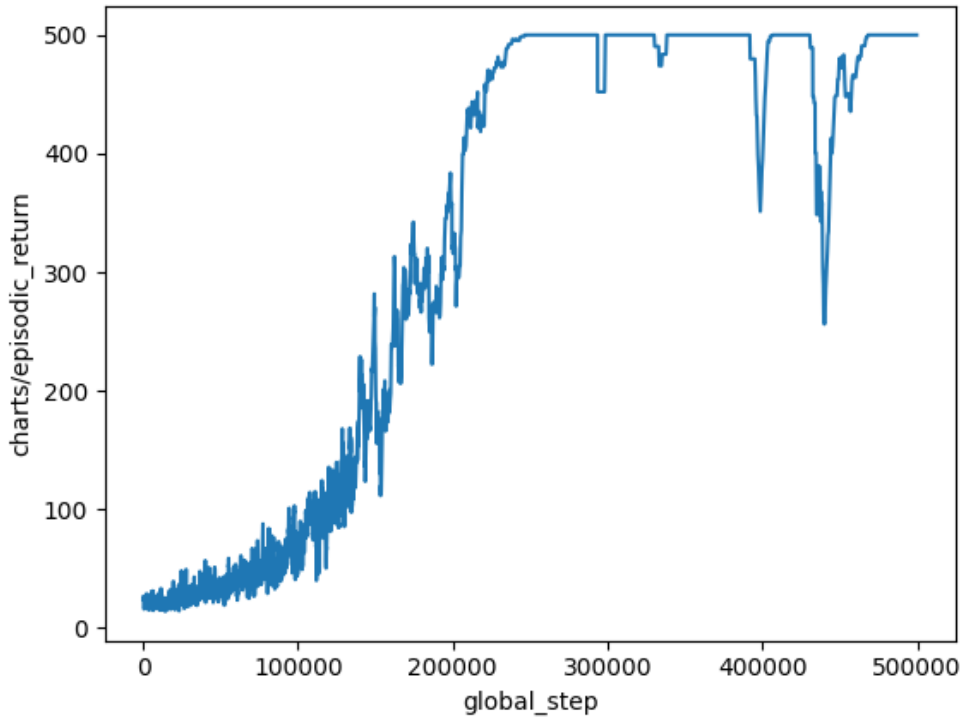
Figura 10: Representación gráfica de todos los experimentos de optimización realizados.



Fuente: Elaboración propia. (2023)

Al realizar el entrenamiento con estos parámetros óptimos, el modelo final llega a la recompensa máxima posible en el entorno de 500.00 ± 0. El gráfico de recompensas a lo largo del entrenamiento se evidencia a continuación:

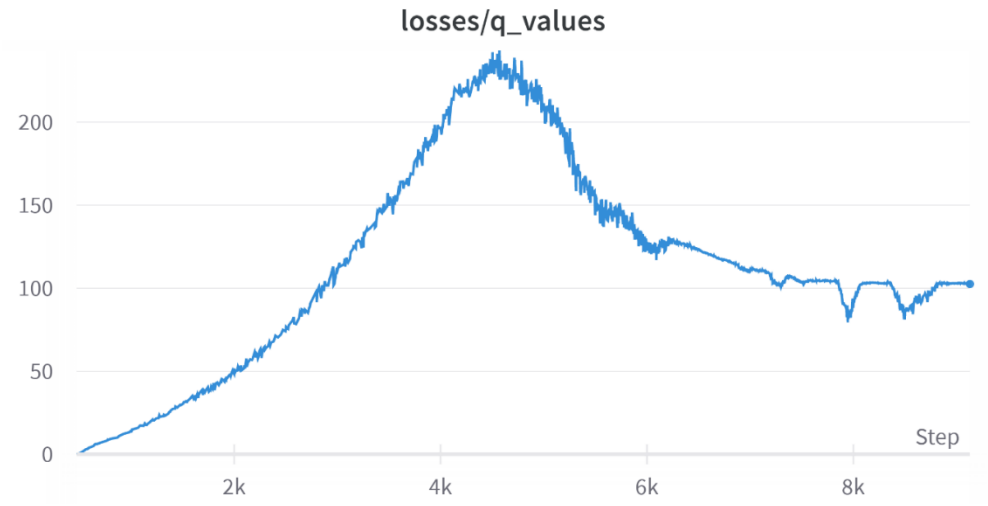
Figura 11: Recompensas a lo largo del entrenamiento en el entorno CartPole-v1.



Fuente: Elaboración propia. (2023)

Se puede apreciar que el modelo continuamente mejora durante el entrenamiento hasta que llega al rendimiento máximo del entorno entre los 200 mil y 300 mil pasos. Para entender más a fondo lo que sucede es necesario observar el gráfico de los valores Q:

Figura 12: Valores Q a lo largo del entrenamiento en el entorno CartPole-v1.

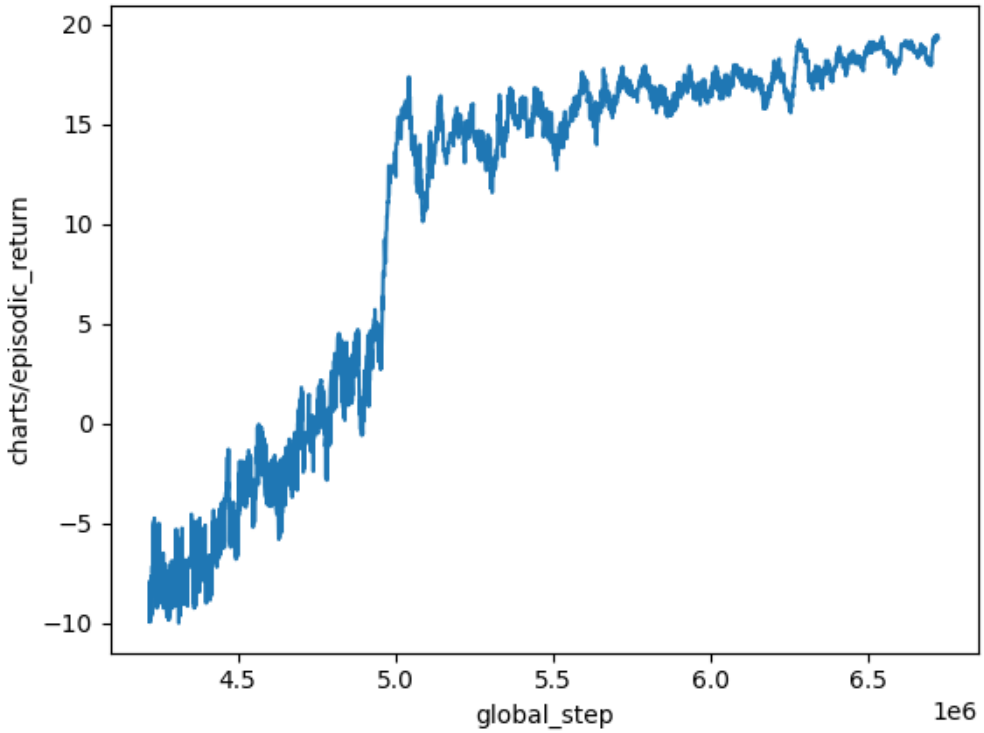


Fuente: Elaboración propia. (2023)

Como se puede ver, el modelo aprende del entorno exitosamente hasta que llega al límite máximo posible, momento en el que ya no tiene nada mas que aprender y sus valores Q empiezan a degradarse poco a poco, lo que indica que no hay retorno en entrenarlo por más de 300.000-400.000 pasos. Esta clase de sobreentrenamiento no suele causar problemas, pero en algunos casos puede empeorar el rendimiento o hasta deshacer lo aprendido por el modelo final, dependiendo de los parámetros usados. En este caso, el modelo fue entrenado por 500.000 pasos y aún logra alcanzar la recompensa máxima de 500 consistentemente.

- PongNoFrameskip-v4: Este entorno es más complejo que CartPole-v1, ya que usa observaciones visuales, por lo que requiere más pasos para aprender. El modelo final se entrenó por 6.5 millones de pasos y alcanza una recompensa de 18.80 ± 1.25, lo cual se acerca bastante a la recompensa máxima del entorno de 21.

Figura 13: Recompensas a lo largo del entrenamiento en el entorno PongNoFrameskip-v4.

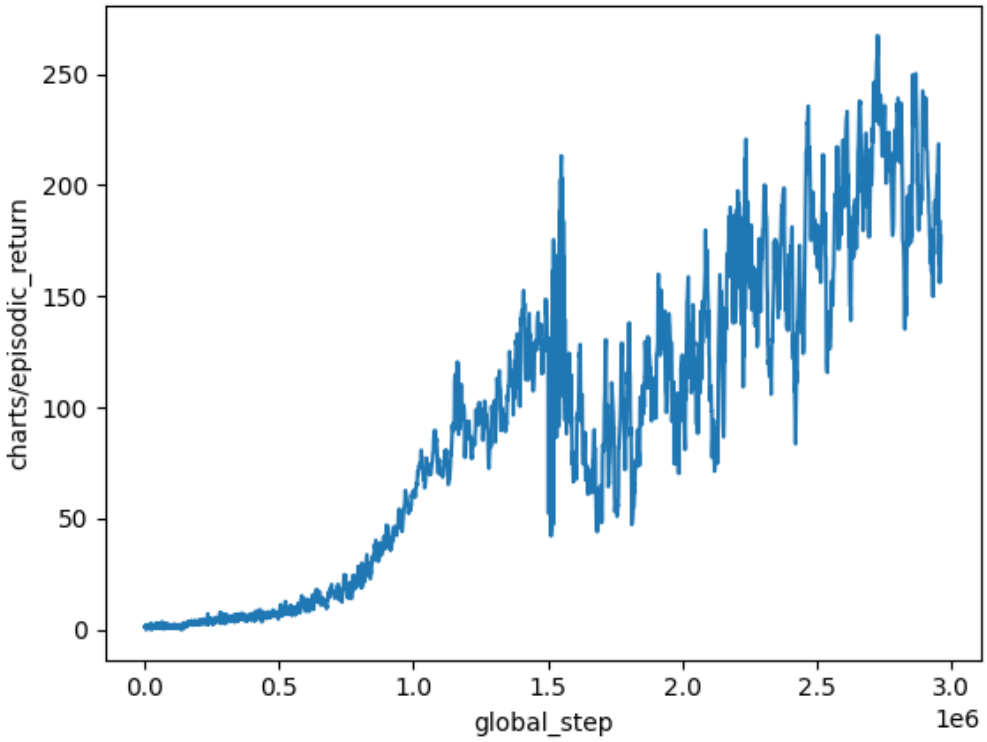


Fuente: Elaboración propia. (2023)

El entrenamiento avanzó lentamente hasta alrededor de los 4.5 millones de pasos, donde empezó a conseguir recompensas positivas y logró un aumento repentino en rendimiento, considerando que pasó de obtener un promedio de recompensas entre 0 y 5, a uno entre 10 y 15 en menos de 100 mil pasos. Este tipo de "explosión" de recompensa o pérdida es común en Reinforcement Learning e indica el momento en que el agente ha logrado aprender cómo resolver el problema presentado; en los siguientes pasos del entrenamiento el agente buscó maximizar y optimizar esta solución hasta llegar al resultado final de 18.80.

- BreakoutNoFrameskip-v4: La complejidad de este entorno es similar a Pong, pero la estrategia óptima es distinta; aún así, el agente logró aprender cómo jugar en menos de 3 millones de pasos, obteniendo una recompensa final de 221.40 ± 32.78.

Figura 14: Recompensas a lo largo del entrenamiento en el entorno BreakoutNoFrameskip-v4.



Fuente: Elaboración propia. (2023)

La gráfica demuestra que el entrenamiento de este agente fue más estable que el de Pong, lentamente incrementando la recompensa obtenida a lo largo del proceso. A pesar de que el modelo final alcance solo la mitad de la recompensa máxima del entorno (432), es interesante descubrir que aún así logra replicar la misma estrategia óptima presentada en Playing Atari Games With Deep Reinforcement Learning (Mnih, V. et al. 2013), donde el agente se concentra en crear un túnel a un lado y logra pasar la pelota al otro lado de los bloques, de manera que esta queda rebotando por un largo tiempo sin necesidad de intervención por parte del agente:

Figura 15: Agente usando la estrategia óptima en el entorno BreakoutNoFrameskip-v4.

Fuente: Elaboración propia. (2023)

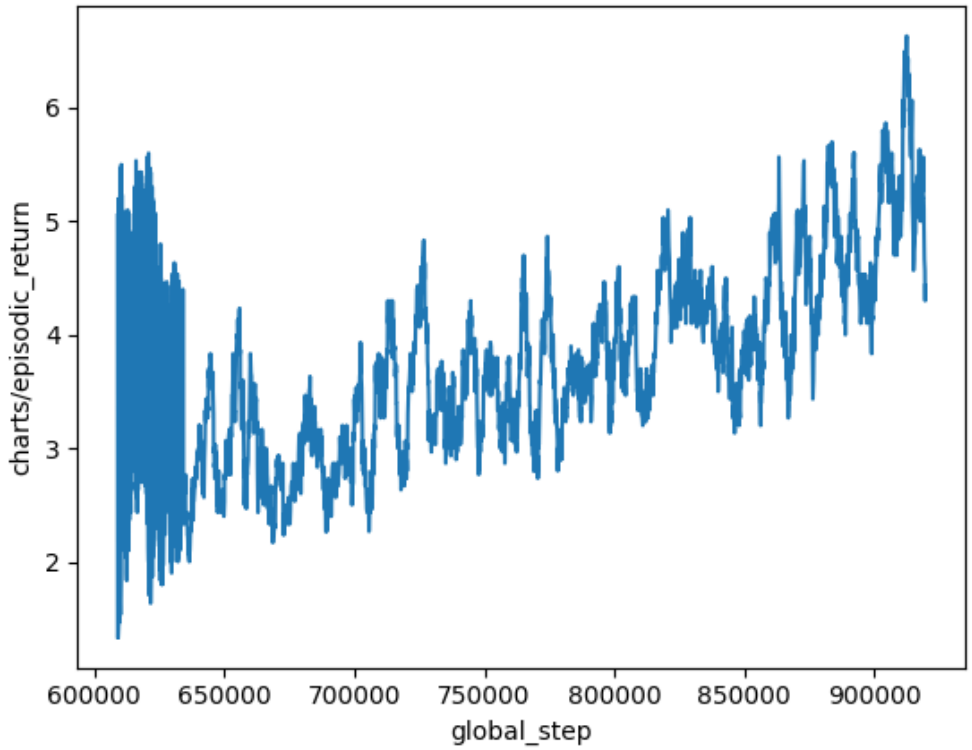
Nótese cómo el agente priorizó hacer camino a los lados y pasó la pelota al otro lado para que esta pudiera destruir varios bloques por si sola sin temor a que se caiga. Este

es comportamiento aprendido de forma completamente independiente por el agente, usando solamente observaciones visuales sin intervención humana.

Aunado a estos entornos base, se entrenaron modelos en los siguientes entornos nuevos o modificados:

- BreakoutNoFrameskip-v4 (Rotado): Este entorno es el mismo que el anterior con la diferencia de que cada observación tiene 50% de probabilidad de ser rotada por 90 grados en sentido contrarreloj, con el objetivo de aprender a rebotar la pelota con una paleta horizontal y una vertical. De esta manera, el modelo sería capaz de aprender a jugar tanto Breakout como Pong habiendo sido entrenado únicamente en Breakout. Debido a que esto aumenta la complejidad considerablemente, el modelo se entrenó por poco más de 900 mil pasos, para confirmar que el agente logre cumplir la tarea. Como es de esperar, la complejidad añadida hizo que el agente no hiciera progreso considerable hasta el paso 600mil:

Figura 16: Recompensas a lo largo del entrenamiento en el entorno BreakoutNoFrameskip-v4.



Fuente: Elaboración propia. (2023)

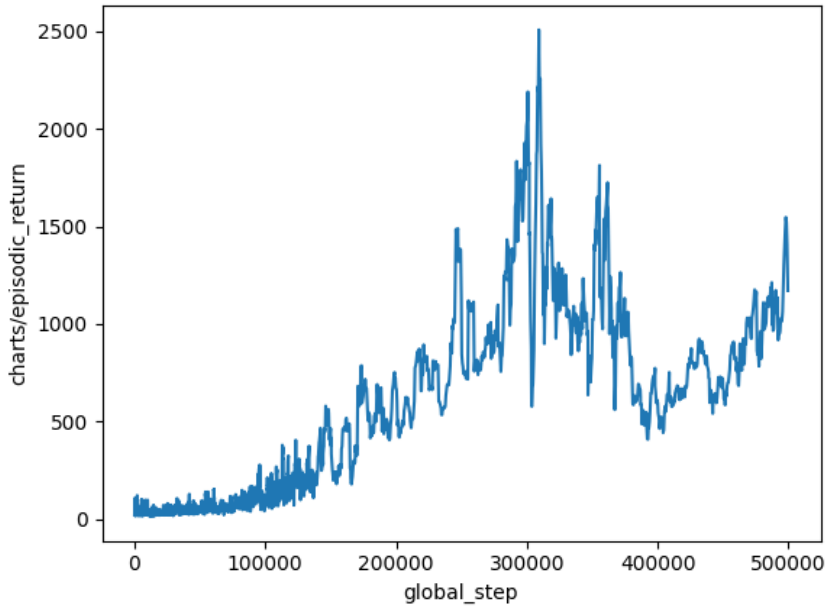
La rotación continua del entorno hace que el entrenamiento sea altamente inestable, como puede apreciarse en la parte izquierda del cuadro, no obstante, el agente logra adaptarse y aprender poco a poco. El modelo final obtuvo las siguientes recompensas en promedio:

- Breakout base: 18
- Breakout rotado en 90 grados: 21
- Breakout rotado en 180 grados: 2
- Breakout rotado en 270 grados: 1
- Pong: -21
- Breakout alternando rotación entre 0 y 90 grados: 23

Sorprendentemente, el modelo final no sabe cómo actuar en Pong a pesar de su similitud, tanto visual como práctica, con Breakout rotado por 90 grados. Aún así, se puede observar que el modelo logra jugar Breakout correctamente en las dos rotaciones distintas que ha visto, tanto por separado como juntas; de hecho, su rendimiento aumenta cuando se alterna entre las dos rotaciones, probablemente debido a que se entrenó en este tipo de observaciones y adaptó su comportamiento de acuerdo a ello.

- CartPoleGoal-v1: Este entorno se creó para este proyecto de investigación. Al ser una extensión de CartPole-v1, tiene una complejidad ligeramente mayor, pero aún así puede resolverse con el mismo modelo usado para CartPole-v1, obteniendo una recompensa de 3420.00 ± 1874.80 al ser entrenado por 500 mil pasos. Estas fueron sus recompensas durante el entrenamiento:

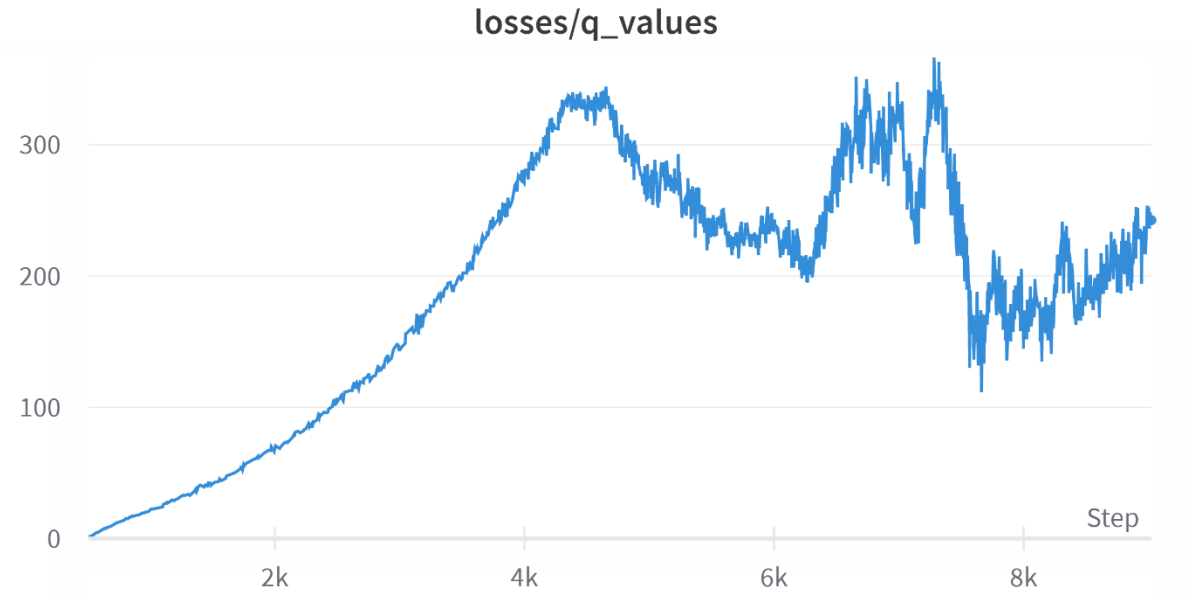
Figura 17: Recompensas a lo largo del entrenamiento en el entorno CartPoleGoal-v1.



Fuente: Elaboración propia. (2023)

El proceso puede parecer inestable, pero de hecho es bastante estable, como puede apreciarse en la gráfica de sus valores Q:

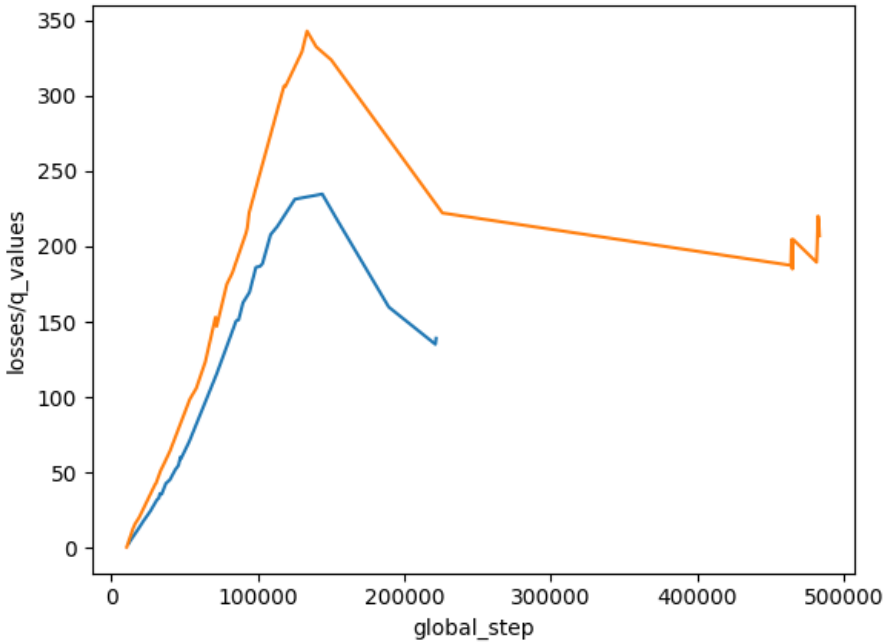
Figura 18: Valores Q a lo largo del entrenamiento en el entorno CartPoleGoal-v1.



Fuente: Elaboración propia. (2023)

Su entrenamiento comprende una curva natural muy similar a la vista en el entorno base (CartPole-v1 en azul, CartPoleGoal-v1 en naranja):

Figura 19: Valores Q del entrenamiento en los entornos CartPole-v1 y CartPoleGoal-v1.



Fuente: Elaboración propia. (2023)

La curva de CartPoleGoal es mayor debido a que el entorno es más complejo y por lo tanto el límite de comportamientos que puede aprender el agente sobre el mismo es mayor. (De hecho, desde un punto de vista matemático se puede decir que CartPoleGoal es un superset de CartPole, o que CartPole es un subset de CartPoleGoal con una dimensión menos)

Hay dos cosas que vale la pena destacar de estos resultados:

- La complejidad añadida vale la pena, pues un agente entrenado en CartPoleGoal puede resolver cualquier problema de CartPole sin problemas, mas lo inverso no es cierto.
- A pesar de esta diferencia en complejidad, los dos modelos entrenados parecen llegar al límite de su entrenamiento en el mismo número aproximado de pasos, lo cual prueba que es posible crear un modelo más

generalizable sin necesidad de alterar su estructura o entrenarlo por más tiempo, solo con incrementar ligeramente la complejidad del entorno y/o agregar otra dimensión al problema, al menos en el entorno de CartPole-v1.

Según los resultados, una misma estructura de red neuronal puede usarse para resolver una gran cantidad de entornos, y las dos definidas anteriormente fueron suficientes para resolver todos los entornos estudiados sin problemas.

Los entornos existentes son muy variados y hay tantos como problemas que resolver, por lo que, para garantizar la generalización, estos deben tener características en común que el agente pueda usar para decidir que acción usar.

Por otro lado, el proceso de entrenamiento puede afectar la generalización mediante el preprocesamiento. Como se evidenció en el entorno de Breakout rotado, el simple hecho de rotar las observaciones logró que el modelo se adaptara a más situaciones, con menor rendimiento en el entorno base. Más pasos de entrenamiento y experimentación podrían mejorar este proceso y anular esta disminución en rendimiento y eficacia.

Cabe destacar que se logró aumentar la generalización de forma efectiva al incluir una dimensión más al problema en el entorno de CartPoleGoal, utilizando la misma cantidad de pasos para resolverlo. Este es un fenómeno interesante que vale la pena estudiar más a fondo, y posiblemente se deba a que la estructura escogida tiene mucha más habilidad que la requerida por los dos entornos, y es por ello que los puede aprender con la misma dificultad.

PARTE V

PROPUESTA

5.1 Importancia de la aplicación de la propuesta

El beneficio principal radica en la automatización de procesos: El Reinforcement Learning permite realizar de forma automática tareas y procedimientos que requieren razonamiento, lógica y reconocimiento de patrones, los cuales son aspectos problemáticos de resolver mediante la heurística o la programación tradicional.

No solo ello, la implementación de estos sistemas también ayuda a tener un mayor entendimiento de los problemas que tratan: La inteligencia artificial entra dentro de la disciplina de la Ciencia de Datos (Data Science) y por lo tanto requiere manipular grandes cantidades de datos para identificar su naturaleza y patrones dentro de los mismos. Los desarrolladores pueden descubrir propiedades nuevas dentro de los datos incluso antes de empezar la implementación de la inteligencia artifical en sí; y han ocurrido casos en que se han descubierto patrones y características específicas sobre ciertos problemas debido a comportamientos aparentemente extraños en agentes de Reinforcement Learning.

Por último, también hay que destacar que estos agentes nos permiten descubrir soluciones que no se habrían tomado en cuenta de otra manera; un ejemplo de esto puede ser el comportamiento que mostró el agente de Breakout durante el entrenamiento, pues las personas comúnmente se centran en rebotar la pelota y romper los bloques más cercanos, mientras que el agente se centró en pasar la pelota al otro lado de los bloques para alcanzar una solución óptima.

5.2 Viabilidad de la aplicación de la propuesta

5.2.1 Técnica

Los requisitos técnicos dependen principalmente del tamaño de los modelos a entrenar, como se presenta en la siguiente tabla:

Tabla 3: Requisitos técnicos para el entrenamiento de los modelos.

Factor requerido	Modelos pequeños	Modelos grandes
Poder de cómputo	No requiere de mucho, hasta pueden entrenarse en una laptop.	Dependiendo del tamaño puede necesitar desde una PC de última generación hasta un clúster de computadoras.
Memoria	Depende del entorno; puede exigir una cantidad moderada para entornos altamente visuales. (4 GB o menos)	Suficiente memoria para soportar el modelo, el entorno, el framework de entrenamiento y cualquier otro proceso en la máquina. (De 8 GB en adelante)
Conocimientos	Conocimientos intermedios de programación para estructuras ya establecidas.	Matemática avanzada y estadística para modelos y/o entornos nuevos o modificados.

Fuente: Elaboración propia.

5.2.2 Operativa

Para llevar un modelo a capacidad operativa se requiere realizar una serie de etapas, descritas a continuación:

Tabla 4: Etapas del entrenamiento de modelos de inteligencia artificial mediante Reinforcement Learning.

Etapa	Descripción	Notas
Preparación	Se estudia a fondo el problema a resolver, analizan los datos y se crea el entorno o simulación. Es	Esta etapa define la eficacia de todo el proceso, pues la información obtenida

	la etapa que requiere más atención.	guiará el desarrollo del modelo final. Es por ello que debe asegurarse que todos los datos sean de buena calidad. Se recomienda consultar a expertos en matemática, estadística y ciencia de datos.
Entrenamiento	Se entrena el agente en el entorno designado, usando el framework escogido.	Tener en cuenta el uso final que se le dará al modelo a la hora de entrenar, e implementar su entrenamiento adecuadamente.
Operación	Se integra el modelo entrenado en un sistema nuevo o uno ya establecido.	Probar y evaluar el modelo extensivamente, e implementar medidas para solventar accidentes o errores que puedan ocurrir durante su operación.

Fuente: Elaboración propia.

5.2.3 Económica

Los recursos económicos requeridos dependen del problema a solventar o tarea a realizar. Estas tareas pueden clasificarse en 3 áreas:

- Ligeras: Análisis numérico. (Predicción de ventas, análisis exploratorio, inteligencia empresarial)
- Medianas: Simulaciones y tareas basadas en acciones. (Simulaciones de vehículos, manipulación de brazos robóticos, entornos de Atari)
- Pesadas: Procesamiento de lenguaje natural o generación de imágenes. (Fine-tuning de modelos GPT o LlaMA mediante Reinforcement Learning with Human Feedback, modelos de difusión como Dall-E o Stable Diffusion)

Tabla 5: Recursos necesarios para el entrenamiento de inteligencia artificial, por tarea.

Tarea	RAM	GPU (VRAM)	Entornos en la nube	Costo Inicial	Gasto Mensual
Análisis numérico	2 GB o menos. Dependiendo de la complejidad puede realizarse incluso con menos de 100 MB.	Opcional.	No son necesarios.	200$ o menos (Hardware de consumidor)	0-10$ /mes
Simulaciones y tareas basadas en acciones	8 GB o más.	Altamente recomendada, preferiblemente con 4 GB o más de VRAM y de la marca NVIDIA*.	Alternativa razonable a comprar una GPU si no se usará con frecuencia.	300$ GPU 800-900$ Equipo completo	10-50$ /mes (Entornos en la nube)
Procesamiento de lenguaje natural o	14-60 GB o hasta más, dependiend	Necesaria, 14-60 GB de VRAM o hasta	Altamente recomendados a menos	500-1000$ GPU	200-500$ /mes (Entornos

generación de imágenes	o del modelo a usar.	más, dependiendo del modelo a usar.	que se requiera usar la GPU frecuentemente.	1500-3500$ Equipo completo	en la nube)

Fuente: Elaboración propia.

5.3 Objetivos de la Propuesta

5.3.1 Objetivo General

Desarrollar un agente de inteligencia artificial para solución de problemas en entornos 2D basado en el Q-learning profundo, creando un framework llamado “Lagomorph” para tal fin.

5.3.2 Objetivos Específicos

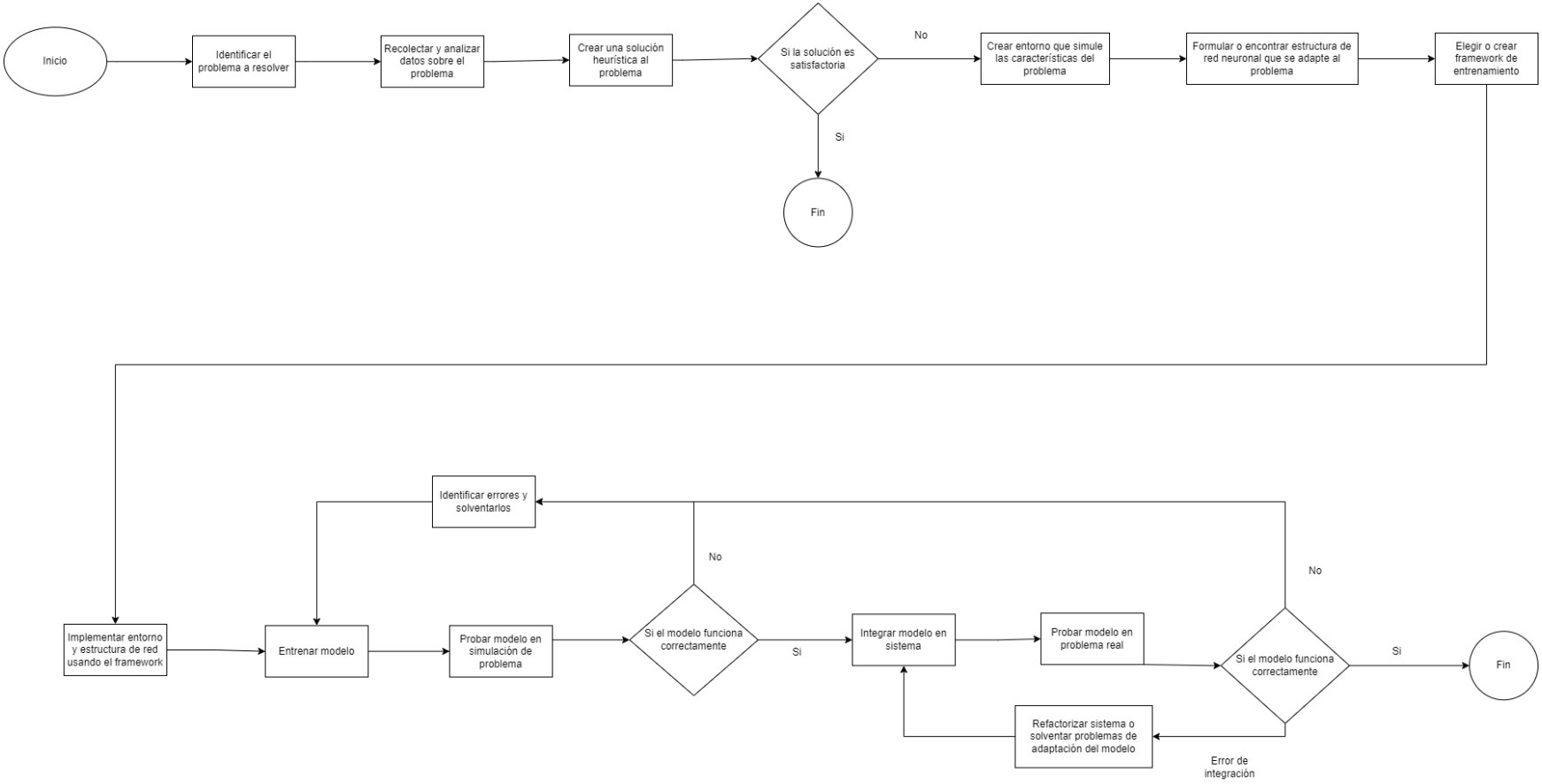
1. Implementar un proceso de entrenamiento de agentes que sea reproducible y fácil de usar.
2. Permitir la extensión del framework para incluir entornos y pasos de preprocesamiento personalizados según el usuario lo requiera.
3. Proveer una manera de instalar el framework de forma fácil y sencilla.
4. Integrar el framework con repositorios de registro de experimentos para monitorear los entrenamientos y guardar sus resultados.
5. Implementar una manera de evaluar modelos propios o de terceros fácilmente dentro del propio framework.

5.4 Representación Gráfica y Estructura de la Propuesta

El framework puede encontrarse en el siguiente repositorio en GitHub: https://github.com/odiaz1066/lagomorph.

Además, a continuación se presenta el diagrama de flujo para el desarrollo de un agente de inteligencia artificial mediante Reinforcement Learning:

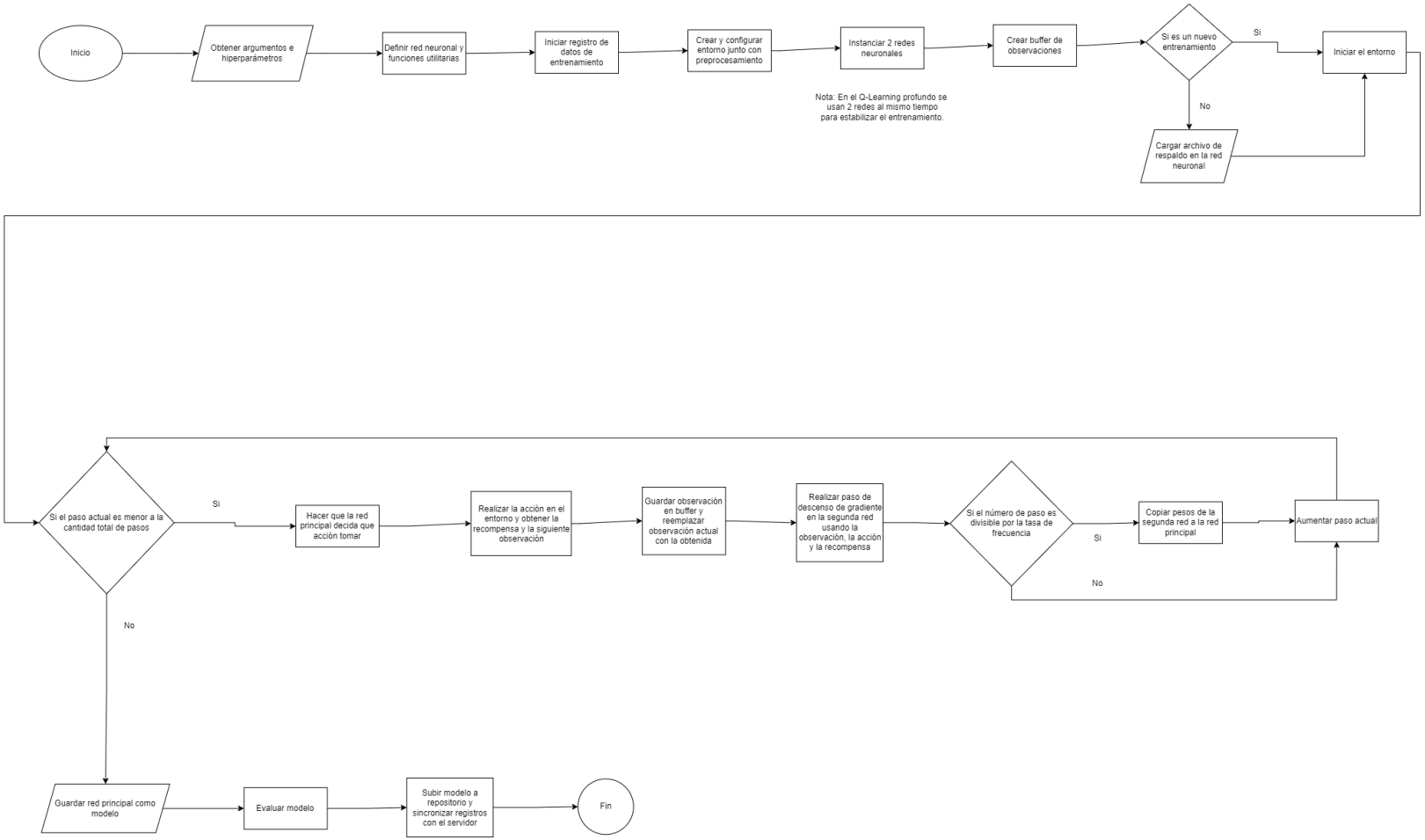
Figura 20: Diagrama de flujo sobre el proceso de desarrollo de un agente de inteligencia artificial mediante Reinforcement Learning



Fuente: Elaboración propia.

El siguiente diagrama explica a fondo cómo se realiza el paso de “Entrenar modelo”, de la manera en que se realiza dentro de CleanRL y Lagomorph:

Figura 21: Diagrama de flujo sobre el proceso de entrenamiento de un modelo de Reinforcement Learning, mediante Lagomorph



Fuente: Elaboración propia.

CONCLUSIONES

Durante el transcurso del presente trabajo de investigación, se logró desarrollar el agente de inteligencia artificial para solución de problemas en entornos 2D, y se creó un framework diseñado para tal fin, denominado Lagomorph, que está disponible al público en el siguiente repositorio de GitHub: https://github.com/odiaz1066/lagomorph.

Como se ha demostrado, este framework es capaz de entrenar agentes para resolver varios entornos, incluyendo juegos de Atari, solo con adaptarlos a la API de Gymnasium. En tal sentido, las estructuras de red neuronal presentadas no son las únicas capaces de producir agentes útiles y pueden existir otras que den mejores resultados en entornos específicos, las escogidas están optimizadas y han sido extensivamente probadas para asegurar un rendimiento óptimo en una gran variedad de entornos de forma general.

Asimismo, se puede usar Lagomorph para entrenar agentes en otros entornos no presentados en este proyecto; ya que no se limita a un set específico, y puede integrarse con otros frameworks y librerías para extender sus capacidades. Adicionalmente, dicha herramienta posee integraciones para Unity ML-Agents y Farama Miniworld en su código de forma predeterminada, y se pueden agregar más de forma simple implementando una interfaz que cumpla con la API de Gymnasium. (Debe soportar los métodos “__init__”, “step” y “reset” como mínimo)

Por otro lado, el framework fue creado para ser sencillo de usar en varias plataformas, considerando que soporta tanto Windows, Linux y entornos de Jupyter; y ha sido probado en Windows 10, Google Colab y Azure Machine Learning. Además, la organización del código hace que sea fácil de modificar para el uso que se requiera. Por ejemplo, para ser usado como una librería e integrarlo en un sistema más grande solo hace falta copiar un único archivo dependiendo de los entornos a usar (vectoriales o visuales), mover las variables y argumentos globales dentro de una clase y convertir el bucle de ejecución principal a un método dentro de la misma clase.

Con respecto a los modelos resultantes, estos son capaces de generalizar su conocimiento de manera efectiva, y los resultados demuestran que el entorno donde ha sido entrenado juega el mayor rol a la hora de desarrollar capacidades de generalización.

En efecto, preprocesar las observaciones del entorno de Breakout no contribuyó mucho a su rendimiento en el entorno de Pong, pero la adición directa de un nuevo objetivo al entorno de CartPole logró que el agente fuera capaz de resolver una variedad más grande de problemas que los presentes en el entorno base. Aun así, todos los modelos fueron capaces de resolver los entornos estudiados usando las estructuras de red y procesos de entrenamiento descritos, por lo que no se requieren modificaciones extensas para entrenar nuevos modelos; de hecho, el framework permite entrenar nuevos modelos en una gran variedad de entornos con un solo comando.

Dicho todo esto, se puede afirmar que se pudo desarrollar el agente de solución de problemas en entornos 2D de forma exitosa, usando el Q-Learning Profundo; y gracias a ello ahora se tiene un framework que la comunidad puede usar para desarrollar agentes por su propia cuenta, sin necesidad de poseer de muchos recursos o conocimientos en el tema.

RECOMENDACIONES

A continuación, se presentan algunas recomendaciones prácticas basadas en las observaciones hechas durante el proceso de investigación:

- Identificar si los beneficios de implementar un agente mediante Deep Learning contrarrestan los costos adicionales. El Deep Learning (y el Reinforcement Learning, por extensión) requiere de una gran capacidad de cómputo en comparación a técnicas heurísticas como los algoritmos evolutivos, por lo que es recomendable primero implementar una alternativa tradicional que resuelva el problema a un grado satisfactorio, entrenar un agente usando Deep Learning y, por último, comparar el rendimiento de los dos para concluir si es conveniente o no.
- Usar otros frameworks para tareas más especializadas, como entornos 3D, procesamiento de lenguaje natural, múltiples agentes o entornos con una cantidad de estados excesiva o cuya función de recompensa esté vagamente definida. A pesar de que Lagomorph ha probado ser efectivo, estos entornos son muy complejos o poseen características que hacen que su entrenamiento sea

impráctico o imposible sin modificaciones previas. Algunos frameworks que pueden usarse para estas tareas son Hugging Face Transformers para el procesamiento de texto, o Petting Zoo para entornos multiagentes.

- Probar exhaustivamente el agente si se planea usarlo en un entorno real no simulado, y procurar que las simulaciones sean representativas del entorno real. En caso contrario, puede que el agente no realice el trabajo deseado o, en sistemas importantes, puede presentar un riesgo en la operación diaria del mismo. Implementar medidas preventivas y correctivas es un paso esencial de todo sistema, y su importancia es mayor cuando se involucra a la inteligencia artificial.
- Estudiar a fondo el efecto del entorno en la generalización de los entornos finales como línea de investigación a futuro, pues se evidencia que este presenta un efecto mayor del esperado en la misma. El entorno comprende todo un sistema por sí mismo, debido a que, tanto su creación como operación, involucran a varias variables (Las acciones que pueden realizarse, el tipo y cantidad de posibles observaciones, las técnicas de procesamiento involucradas, interfaces públicas si el entorno se conecta a un sistema externo, etc.), por lo que, el estudio de su influencia en la generalización puede ayudar a la creación de modelos más robustos y eficientes.
- Tener una visión a futuro a la hora de implementar estos sistemas: La industria de la inteligencia artificial progresa rápidamente y sistemas de última tecnología pueden quedar rezagados en cuestión de meses o hasta semanas. Es por ello que los sistemas que la integren deben ser flexibles y adaptables para que puedan ser actualizados y extendidos con facilidad.
- Prestar atención, tanto a los modelos como a los datos necesarios para entrenarlos, ya que estos son factores determinantes para manipular de forma apropiada a los agentes inteligentes. Si un entorno no es representativo de la realidad, no va a poder realizar su trabajo adecuadamente; si se entrena un modelo predictivo con datos erróneos o de mala calidad, este no va a poder identificar los patrones inherentes en los mismos.
- Probar una selección de varios algoritmos de Reinforcement Learning antes de enfocarse en uno solo. Existen otros además del Q-Learning Profundo, como PPO,

Dueling DQN, REINFORCE, entre otros. Cada uno es mejor en distintas clases de entornos, algunos requieren más pasos de entrenamiento, parte de ellos solo funcionan en entornos con acciones discretas y otros con acciones continuas, etc. La elección de cual usar depende del problema a resolver.

Referencias

Álvarez, L. (1994). *Fundamentos de Inteligencia Artificial.* Recuperado el 14 de Marzo de 2023, de https://books.google.co.ve/books?id=UfccXvwzIOUC

Brockman, G. C. (05 de Junio de 2016). *OpenAI Gym.* Recuperado el 03 de Marzo de 2023, de https://arxiv.org/pdf/1606.01540.pdf

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., . . . Sigler, E. (2020). *Language Models are Few-Shot Learners.* Recuperado el 12 de Febrero de 2023, de https://arxiv.org/pdf/2005.14165.pdf

Cobbe, K. K. (26 de Julio de 2020). *Leveraging Procedural Generation to Benchmark Reinforcement Learning.* Recuperado el 02 de Marzo de 2023, de https://arxiv.org/pdf/1912.01588.pdf

Cote, M., Kadar, A., Yuan, X., Kybartas, B., Barnes, T., Fine, E., . . . Trischler, A. (08 de Noviembre de 2019). *TextWorld: A Learning Environment for Text-based Games.* Recuperado el 11 de Junio de 2023, de https://arxiv.org/pdf/1806.11532.pdf

Goodfellow, I., Yoshua, B., y Courville, A. (2016). *Deep Learning.* Recuperado el 27 de Febrero de 2023, de https://www.deeplearningbook.org/

Hasselt, H., Guez, A., y Silver, D. (2015 de Diciembre de 08). *Deep Reinforcement Learning with Double Q-learning.* Recuperado el 11 de Junio de 2023, de https://arxiv.org/pdf/1509.06461.pdf

Huang, S. J. (Julio de 2022). *CleanRL: High-quality Single-file Implementations of Deep Reinforcement Learning Algorithms.* Recuperado el 04 de Marzo de 2023, de https://www.jmlr.org/papers/volume23/21-1342/21-1342.pdf

Juliani, A. B. (06 de Mayo de 2020). *Unity: A General Platform for Intelligent Agents.* Recuperado el 02 de Marzo de 2023, de https://arxiv.org/pdf/1809.02627.pdf

Kirk, R., Zhang, A., Grefenstette, E., y Rocktäschel, T. (30 de Enero de 2022). *A Survey of Zero-shot Generalisation in Deep Reinforcement Learning.* Recuperado el 09 de Noviembre de 2022, de https://arxiv.org/pdf/2111.09794.pdf

López, B. (s.f.). *Introducción a la Inteligencia Artificial.* Recuperado el 20 de Febrero de 2023, de http://www.itnuevolaredo.edu.mx/takeyas/Articulos/Inteligencia%20Artificial/ARTICULO%20Introduccion%20a%20la%20Inteligencia%20Artificial.pdf

Mendez, D. (2010). *Revisión Documental.* Recuperado el 27 de Marzo de 2023, de https://adsucc.fandom.com/es/wiki/Revision_Documental

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., y Riedmiller, M. (01 de Enero de 2013). *Playing Atari with Deep Reinforcement Learning.* Recuperado el 09 de Noviembre de 2022, de Google DeepMind: https://arxiv.org/pdf/1312.5602v1.pdf

Montesinos, O., Montesinos, A., y Crossa, J. (05 de Enero de 2022). *Fundamentals of Artificial Neural Networks and Deep Learning.* Recuperado el 11 de Junio de 2023, de https://www.researchgate.net/profile/Osval-Montesinos-Lopez/publication/357820885_Fundamentals_of_Artificial_Neural_Networks_and_Deep_Learning/links/623345ffb4db545f4731e274/Fundamentals-of-Artificial-Neural-Networks-and-Deep-Learning.pdf

Muguira, A. (2022). *¿Qué es la investigación descriptiva?* Recuperado el 24 de Marzo de 2023, de https://www.questionpro.com/blog/es/investigacion-descriptiva/

Murphy, K. (2012). *Machine Learning: A Probabilistic Perspective.* Recuperado el 12 de Febrero de 2023, de http://noiselab.ucsd.edu/ECE228/Murphy_Machine_Learning.pdf

Ortega, C. (2022). *¿Qué es la investigación documental?* Recuperado el 24 de Marzo de 2023, de https://www.questionpro.com/blog/es/investigacion-documental/

Ortega, C. (2022). *Investigación cuantitativa. Qué es y cómo realizarla.* Recuperado el 24 de Marzo de 2023, de https://www.questionpro.com/blog/es/que-es-la-investigacion-cuantitativa/

Pérez, J. M. (03 de Septiembre de 2013). *Proyecto factible - Qué es, definición, características y surgimiento.* Recuperado el 24 de Marzo de 2023, de https://definicion.de/proyecto-factible/

Pérez, J. M. (12 de Abril de 2019). *Objeto de estudio - Qué es, definición y concepto.* Recuperado el 24 de Marzo de 2023, de https://definicion.de/objeto-de-estudio/

Pino, R., Gómez, A., y Martínez, N. (2001). *Introducción a la Inteligencia Artificial: Sistemas Expertos, Redes Neuronales Artificiales y Computación Evolutiva.* Recuperado el 14 de Marzo de 2023, de https://books.google.co.ve/books?id=RKqLMCw3IUkC

Ramesh, A., Pavlov, M., Goh, G., Voss, C., Radford, A., Chen, M., y Sutskever, I. (2021). *Zero-Shot Text-to-Image Generation.* Recuperado el 12 de Febrero de 2023, de https://arxiv.org/pdf/2102.12092.pdf

REAL ACADEMIA ESPAÑOLA. (s.f.). *Diccionario de la lengua española, 23.ª ed., [versión 23.6 en línea].* Recuperado el 03 de Marzo de 2023, de https://dle.rae.es

Rombach, R., Blattmann, A., Lorenz, D., Esser, P., y Ommer, B. (2021). *High-Resolution Image Synthesis with Latent Diffusion Models.* Recuperado el 12 de Febrero de 2023, de https://arxiv.org/pdf/2112.10752.pdf

Ruiz, C. B. (Marzo de 2001). *Redes Neuronales: Conceptos Básicos y Aplicaciones.* Recuperado el 03 de Marzo de 2023, de https://www.frro.utn.edu.ar/repositorio/catedras/quimica/5_anio/orientadora1/monograias/matich-redesneuronales.pdf

Rumelhart, D., Hinton, G., y Williams, R. (09 de Octubre de 1986). *Learning representations by back-propagating errors.* Recuperado el 11 de Junio de 2023, de https://www.iro.umontreal.ca/~vincentp/ift3395/lectures/backprop_old.pdf

Schaul, T., Quan, J., Antonoglou, I., y Silver, D. (25 de Febrero de 2016). *Prioritized Experience Replay.* Recuperado el 11 de Junio de 2023, de https://arxiv.org/pdf/1511.05952.pdf

Silver, D., Hubert, T., Schrittwieser, J., Antonoglou, I., Lai, M., Guez, A., . . . Hassabis, D. (05 de Diciembre de 2017). *Mastering Chess and Shogi by Self-Play with a General Reinforcement Learning Algorithm.* Recuperado el 11 de Junio de 2023, de https://arxiv.org/pdf/1712.01815.pdf

Sutton, R. y. (2020). *Reinforcement Learning: An Introduction.* Recuperado el 03 de Marzo de 2023, de http://incompleteideas.net/book/RLbook2020.pdf

Teng, E. (11 de Noviembre de 2019). *Training your agents 7 times faster with ML-Agents.* Recuperado el 13 de Marzo de 2023, de https://blog.unity.com/technology/training-your-agents-7-times-faster-with-ml-agents

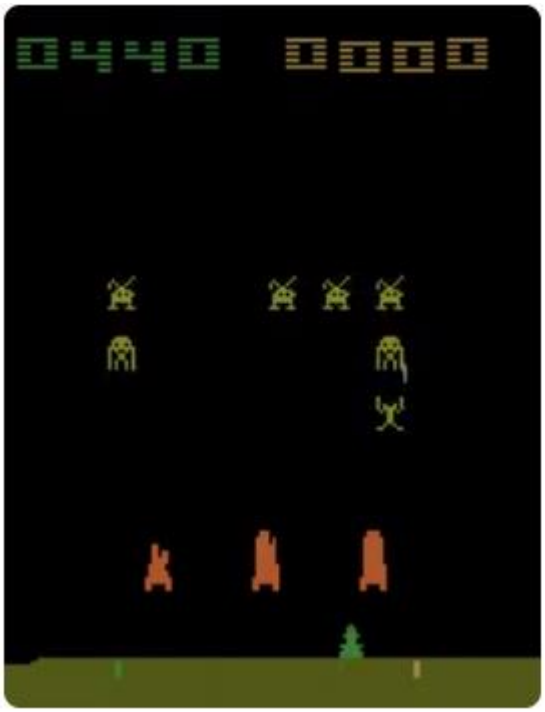
Terry, J., Black, B., Grammel, N., Jayakumar, M., Hari, A., Sullivan, R., . . . Ravi, P. (26 de Octubre de 2021). *PettingZoo: Gym for Multi-Agent Reinforcement Learning.* Recuperado el 11 de Junio de 2023, de https://arxiv.org/pdf/2009.14471.pdf

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., . . . Rush, A. (2019). *HuggingFace's Transformers: State-of-the-art Natural Language Processing.* Recuperado el 12 de Febrero de 2023, de https://arxiv.org/pdf/1910.03771.pdf

Zhuang, F., Qi, Z., Duan, K., Xi, D., Zhu, Y., Zhu, H., . . . He, Q. (17 de Diciembre de 2019). *A Comprehensive Survey on Transfer Learning.* Recuperado el 11 de Junio de 2023, de https://arxiv.org/pdf/1911.02685.pdf

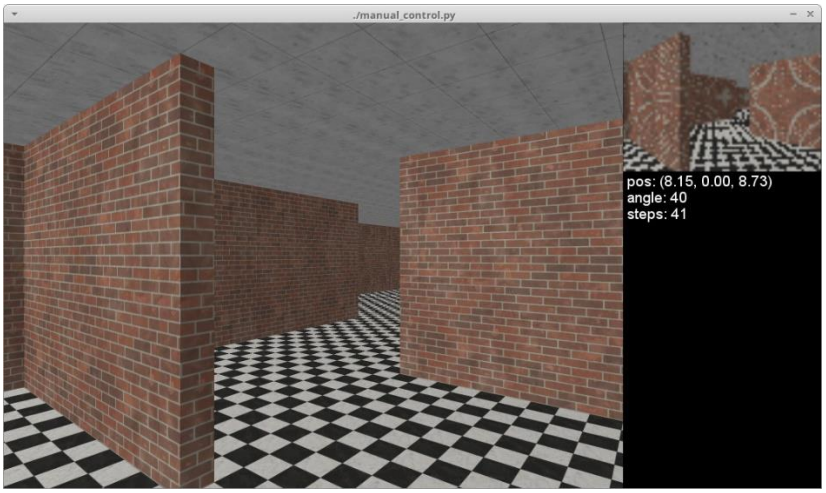
Anexos

Anexo 1: Entrenamiento de agente en Space Invaders usando Stable Baselines 3, durante la evaluación preliminar de entornos y frameworks. Recompensa final de 606.5.



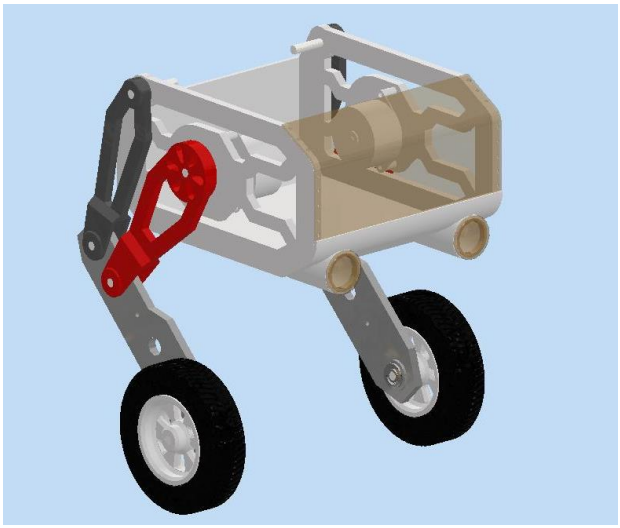
Fuente: Elaboración propia.

Anexo 2: Entorno MazeS3Fast-v0 de Miniworld. Ejemplo de entorno 3D no soportado por Lagomorph.



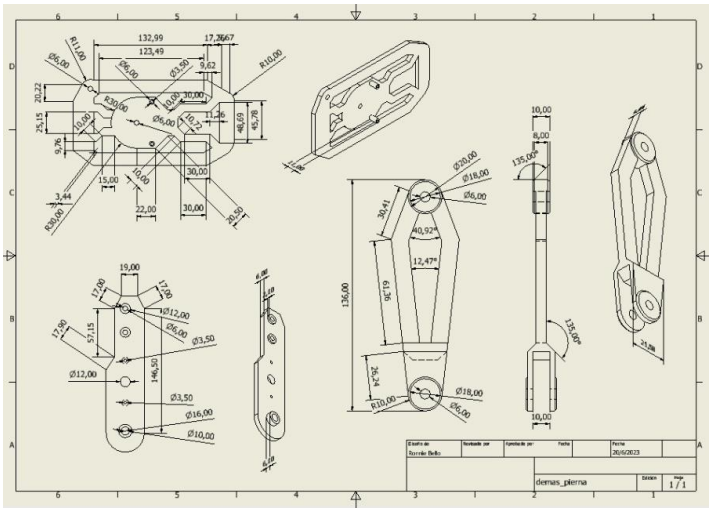
Fuente: Farama Foundation. (2022)

Anexo 3: Modelo 3D de prototipo de robot autobalanceador, para probar el agente entrenado en CartPoleGoal-v1 en un entorno real.



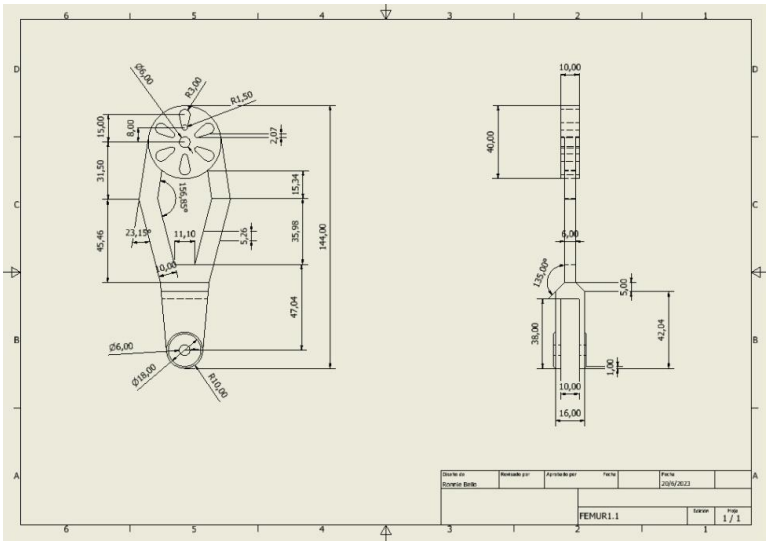
Fuente: Manuel Cova (2023)

Anexo 4: Plano de prototipo de robot autobalanceador.



Fuente: Ronnie Bello (2023)

Anexo 5: Plano de pierna de prototipo de robot autobalanceador.



Fuente: Ronnie Bello (2023)