Universidad del Desarrollo
Facultad de Ingeniería

IMPLEMENTACIÓN DE UN AGENTE DE "INVERSE REINFORCEMENT LEARNING"

Jugando el juego del Nintendo NES: "Punch-Out!"

POR: ALEXANDER RAMÓN MEDINA HERNANDEZ

Capstone Project presentado a la Facultad de Ingeniería de la Universidad del Desarrollo para optar al grado académico de Magíster en Data Science

PROFESORES GUÍA:

(Dr.) CRISTIAN CANDIA, ALONSO ASTROZA

Noviembre 2022

SANTIAGO

Para Mariale, y Ale, que les sirva de inspiración, y para mi esposa Ana, para que se sienta orgullosa de mí.

AGRADECIMIENTO

Tengo un mundo que agradecer a mis hijos, y esposa, que “sufrieron” conmigo muy de cerca el recorrido completo del magister. Les agradezco que perdonen las veces que me ausenté por atender mis deberes durante el magister.

A mi mamá y a mi papá, por tanto, que me dieron.

A todos los contribuyentes de las librerías que utilicé en el proyecto, y a los anónimos que respondieron mis preguntas en las plataformas sobre software.

TABLA DE CONTENIDO

Resumen

Se entrenará un agente de aprendizaje reforzado inverso (IRL por sus siglas en inglés) mediante la utilización de demostraciones de experto, las cuales serán generadas por un humano experto. El agente de IRL será entrenado en un ambiente de juegos de video retro clásico, más específicamente de un juego retro clásico llamado “Punch-Out!!” de la consola retro clásica Nintendo Entertainment System.

Como parte del proceso del proyecto, se realizarán cambios en las variables por defecto de los objetos de las librerías a utilizar para poder generar cambios que ayuden a aliviar las necesidades de procesamiento y tiempo de las actividades mencionadas.

Como paso final, se demostrará que los modelos de IRL a los que se les muestran ejemplos de acciones específicas bajo condiciones específicas dentro de un ambiente RL, se desempeñan mejor en tareas que requieren tener una precisión en el accionar con respecto a los algoritmos tradicionales de RL, los cuales aprenden características de sus entornos para mejorar su desempeño. Lo anterior, demostrará que la hipótesis en la que se basa el proyecto es comprobada técnicamente, y específicamente aplica en el presente caso.

A nivel técnico, se implementarán dos agentes, un agente RL, específicamente de un tipo PPO, y luego en una fase posterior, se implementará un agente GAIL (IRL), al cual se le suministraran las demostraciones de un humano experto.

1. Introducción

Posiblemente la diferencia más definitiva de los algoritmos de aprendizaje reforzado (Reinforcement Learning, por sus siglas en inglés, o RL, como se le llamará de ahora en adelante) es la capacidad que tiene uno de sus componentes (el agente, del cual se darán más detalles en la sección 2.2.1) de ejecutar acciones en un entorno, que puede ser tanto un tablero de ajedrez, como del accionador de un motor de un brazo robótico, basadas en observaciones recibidas desde el ambiente (otra parte del algoritmo, de la cual se hablará en la sección 2.2.2), y que, en promedio, busca maximizar alguna función de recompensa.

El paradigma del aprendizaje reforzado (Reinforcement Learning, por sus siglas en inglés) ha tenido un despertar reciente, hitos como los descritos en (Yazgan, 2020), y (Fawzi, 2022), donde se utilizaron algoritmos de RL para crear productos tangibles y avanzados, y que mejoran los campos donde fueron aplicados, dejaron atrás la época novel del campo, la época de DeepMind AlphaGo, que logró vencer al campeón mundial de Go. Fueron las técnicas aprendidas en esos trabajos, y similares, las que hoy en día posicionan el campo de RL como uno con alta proyección para el futuro, y del que muy seguramente hemos escuchado su última palabra.

Es por lo anterior, que el autor decidió realizar el presente proyecto, que lo ayudará a él, y también seguramente al lector, a entender más afondo los pormenores de este floreciente campo.

2. Trabajo Relacionado

2.1. Trabajos relacionados y librerías utilizadas

El trabajo realizado en (Mnih, 2013) le dio un empujón al campo del aprendizaje reforzado hacia el área del "Deep Learning" al incluir una red CNN (Convolutional Neural Network) como elemento principal en su arquitectura, que le permitió utilizar como entrada las imágenes producidas por juegos de video de la consola Atari 2600, y como salida, las acciones que deben ejecutarse para maximizar una función de recompensa previamente declarada. Este hito dio paso a un florecimiento de modelos de aprendizaje reforzado que poco a poco fueron mejorando a sus predecesores en cuanto a efectividad, eficiencia en tiempo, y metas alcanzadas.

El trabajo en (Nichol, 2018), dejó como legado la librería que será utilizada en este proyecto como elemento central, "Retro Gym" de OpenAI. Expandiendo lo que su predecesor "Gym" en (Brockman, 2016) hizo. Retro Gym permite utilizar videojuegos de consolas retro como Nintendo Entertaiment System (mayormente conocido como NES), o el Sega Genesis, entre otros, como ambiente de entrenamiento para los modelos aprendizaje reforzado. Esto último, aunque parezca trivial, permite la exploración de muchos modelos de aprendizaje reforzado, y adicionalmente presenta una plataforma para realizar "Benchmark" a cualquiera de estos modelos.

Como segundo elemento principal dentro del proyecto está el trabajo realizado en ((RM), 2020), cuyo producto es una librería de Python que permite implementar modelos de aprendizaje reforzado de manera centralizada, fácil, y rápida, "Stable Baselines 3", una

librería que toma los elementos comentados en (Mnih, 2013), (Brockman, 2016), y (Nichol, 2018), y los integra, ofreciendo una API que permite la implementación de muchos modelos de aprendizaje reforzado con una libertad de ajustes finos a los parámetros de los modelos bastante destacable.

Y como tercer y último elemento macro del proyecto, está la librería "Imitation", la cual fue creada para poder incorporar los algoritmos y modelos de aprendizaje reforzado inverso, o "Inverse Reinforcement Learning" por sus siglas en inglés, y modelos de "clonación de comportamiento" o "Behavioral Cloning" por sus siglas en inglés, a los proyectos de aprendizaje reforzado, tales como el trabajo realizado en (Abbeel, 2004), que podría tomarse como uno de los primeros intentos modernos para solucionar el problema de vehículos automotores autónomos. El enfoque tomado en (Abbeel, 2004) para enfrentar el problema de vehículos autónomos es utilizar "demostraciones de expertos", de manera de poder ser mostradas al agente RL, y que este pueda "clonar", o "imitar" dichas demostraciones, y logre completar la tarea de conducir un vehículo de manera autónoma exitosamente. El razonamiento detrás de la decisión de utilizar "Inverse Reinforcement Learning" (término acuñado por los autores), es que la elaboración de una función de recompensa robusta, y efectiva, "a mano", supone un reto difícil de completar de manera integral y general, es decir, que es difícil incluir a priori las variables que constituyan un "manejo seguro" y de "calidad" para los usuarios, en una función de recompensa de manera explícita.

2.2. Breve descripción del bucle de RL / IRL y sus elementos

Un proyecto de aprendizaje reforzado (RL) tiene varios elementos y conceptos importantes que vale la pena mencionar y comprender:

2.2.1. Agente (Agent)

Es el elemento activo dentro de un bucle de aprendizaje reforzado, y por activo nos referimos a que es el elemento que ejecuta las acciones dentro del ambiente.

Un agente de aprendizaje reforzado puede tomar muchas formas, y es evidente que la forma que adopte tendrá que ver mucho con el ambiente donde se debe desenvolver, de las acciones que tendrá que tomar, y del objetivo que tendrá el agente dentro del ambiente y del bucle de aprendizaje reforzado. Como ejemplos podemos mencionar:

- Si el objetivo en el ambiente a explorar es recoger objetos, el agente debería tener al menos un brazo con articulaciones, una cámara para poder ver sus acciones, y un mecanismo de recolección, que posiblemente tenga la forma de una mano humanoide, o de una pinza.
- Si el objetivo es conducir un automóvil de manera autónoma, el agente debe tener acceso a varios sensores dentro del automóvil, acceso a cámaras para poder ver el camino y sus alrededores, acceso a actuadores, o elementos para poder dar órdenes de aceleración o frenado del auto, y como mínimo una unidad de procesamiento central para procesar los datos recolectados por los sensores y las cámaras en

tiempo real. (Lo anterior obviamente es una simplificación del problema de manejo autónomo de automóviles).

2.2.2. Ambiente, Estado, Observación (Enviroment, State, Observation)

El ambiente dentro de un bucle de aprendizaje reforzado dicta las reglas del comportamiento del agente, sus posibles acciones, y la recompensa que puede recibir. Para poder simplificar los algoritmos, y los modelos matemáticos y estadísticos de actuación de un agente, el ambiente siempre tiene que ser reducido a un conjunto de lecturas, y métricas limitadas.

A la reducción del ambiente antes mencionada la llamamos estado, y es la representación más utilizada en la bibliografía matemática y estadística del ambiente y sus cambios durante el bucle de aprendizaje reforzado.

Adicionalmente, utilizamos la palabra observación para referirnos al conjunto detallado de datos que puede tomar el estado de un ambiente, los cuales son pasados al agente durante cada transición del bucle de aprendizaje.

Las observaciones del estado del ambiente pueden ser de naturaleza continua, o de naturaleza discreta, y debido a que son una colección de datos, la convención es referirse a ellas como un espacio de observaciones ("**Observation Space**).

En resumen, el ambiente es donde se desenvuelve el agente, y se ejecuta el bucle de aprendizaje reforzado.

Como ejemplo podemos mencionar:

- En un juego de ajedrez, el ambiente es representado por el conjunto de piezas de juego, el tablero de juego, y las reglas de interacción entre los jugadores y las piezas. Dentro de las reglas es necesario mencionar que el ambiente dicta cuando un jugador es ganador, o si por el contrario (como ocurre en muchos casos de aprendizaje reforzado) el juego no puede continuar debido a que se llegó a una posición de la que ningún jugador puede continuar legalmente, por lo que el juego concluye.

2.2.3. Acciones, Política (Action, Policy)

Son el conjunto de todas las posibles decisiones que el agente puede ejecutar dentro del ambiente, y que realizan o no cambios dentro de este. Nuevamente en este caso, las acciones que pueden ser ejecutadas dependen del ambiente, de sus reglas, y del objetivo del agente dentro del ambiente.

Dependiendo del ambiente, y del agente, las acciones pueden ser discretas, o continuas, y adicionalmente pueden ser numerosas, lo que puede ser representado como un conjunto, o espacio vectorial, por lo que normalmente cuando hablamos de acciones, hablamos de espacio de acciones ("**Action Space**").

La **política**, es la distribución de las acciones más optimas, según las posibles entradas, **codificada en los pesos del modelo** de aprendizaje reforzado, es decir, la política ("policy") es el modelo de redes neuronales en sí.

2.2.4. Recompensa, Función de Recompensa (Reward, Reward Function)

La función de recompensa es la manera en la que se mapea el premio que recibe un agente según la acción que toma en cada transición de estado en el ambiente dentro del bucle de aprendizaje. La función de recompensa es elaborada por el modelador, y es en ocasiones considerada un hiper parámetro dentro del concepto global de aprendizaje reforzado. Aunque la función de recompensa es elaborada por un humano, siempre va a depender de las reglas dentro del ambiente de aprendizaje, y del objetivo que tendrá el agente dentro de este.

La recompensa es el valor numérico que toma la función de recompensa en el bucle, y puede ser instantánea, o total, es decir, la recompensa es devuelta al agente por el ambiente en cada transición del bucle de aprendizaje (instantánea), o puede ser la que se acumula durante la ejecución de un episodio de aprendizaje completo (cuando no se puede continuar ejecutando el bucle de aprendizaje, ya sea porque se cumplió la meta, o porque el agente ya no puede continuar).

La dinámica mencionada en el párrafo anterior da cabida a un fenómeno conocido dentro del aprendizaje reforzado como esparcimiento de la recompensa ("**reward sparcity**"), y es un fenómeno muy estudiado dentro del campo del aprendizaje reforzado, y por el cual se han hecho muchas modificaciones al comportamiento de los algoritmos de los modelos de aprendizaje reforzado. El fenómeno de "**reward sparcity**" ocurre debido a que, dentro del bucle de aprendizaje reforzado, el agente puede pasar muchos intentos ejecutando

acciones (o no), y no recibir recompensa alguna, incluso, el agente puede solo recibir una recompensa al concluir un episodio completo, lo que le dificulta saber cuáles acciones fueron las que lo llevaron a una conclusión exitosa dentro de un episodio, o cuales lo llevaron a un fracaso.

Es por este concepto de "**reward sparcity**" que la recompensa, y la función de recompensa toman un rol protagónico en la implementación de un modelo de aprendizaje reforzado, ya que puede significar el éxito o fracaso de un modelo/agente de aprendizaje reforzado.

2.3. Breve explicación de los algoritmos utilizados

La base de la primera etapa del proyecto será un modelo tipo PPO (Proximal Policy Optimization) como el propuesto en (John Schulman, 2017), que es un modelo de optimización de "pólizas" ("policy") que busca maximizar la función de recompensas, y cuya mejora sobre sus predecesores es la capacidad de no permitir cambios muy alejados entre la póliza actual, y la nueva, producto del ajuste de los parámetros del modelo NN durante la fase de entrenamiento. En resumen, el algoritmo PPO es ampliamente utilizado en el mundo de RL debido a su estabilidad, y su rápida convergencia hacia una póliza optima.

Para la segunda parte del proyecto se realizará la implementación de un algoritmo GAIL, propuesto en (Jonathan Ho, 2016). GAIL (Generative Adversarial Inverse Learning, por sus siglas en inglés), tiene como elemento central una red GAN, la cual a su vez está

compuesta por un elemento “Generador”, un elemento “Discriminador”, y las demostraciones de experto. La dinámica del algoritmo GAIL es la siguiente:

- El elemento “Discriminador” es alimentado de manera aleatoria con muestras tanto del elemento “Generador”, como de las demostraciones de experto.
- El elemento “Generador” (que puede ser un modelo PPO, como es el caso del presente proyecto) intenta “engañar” al elemento “Discriminador” para que “crea” que las tuplas observación-acción que le es suministrada proviene de las muestras de trayectorias de experto.
- Luego de cada paso durante el entrenamiento, la función de perdida es ajustada en todo el sistema, lo que mueve al elemento “Generador” / PPO más cerca de replicar las acciones contenidas en las trayectorias de experto, dadas las observaciones ocurridas.
- El resultado final es un modelo PPO (llamado Generador), que puede emular el comportamiento provisto en las demostraciones de experto.
- En resumen, **la función de pérdida del algoritmo GAIL busca minimizar la diferencia entre las distribuciones de tuplas observación-acción de las demostraciones de experto, y las generadas por el elemento generador**, lo que permite a la póliza converger a una distribución de acciones cercana a la del experto.

2.4. El bucle de RL

Conociendo todos los elementos básicos de un proyecto de RL, podemos entonces explicar de manera breve y concisa el bucle de un proyecto de RL en su etapa de entrenamiento, y que mejor apoyo para esto que la siguiente imagen famosa introducida por el trabajo en (Barto, 2014):

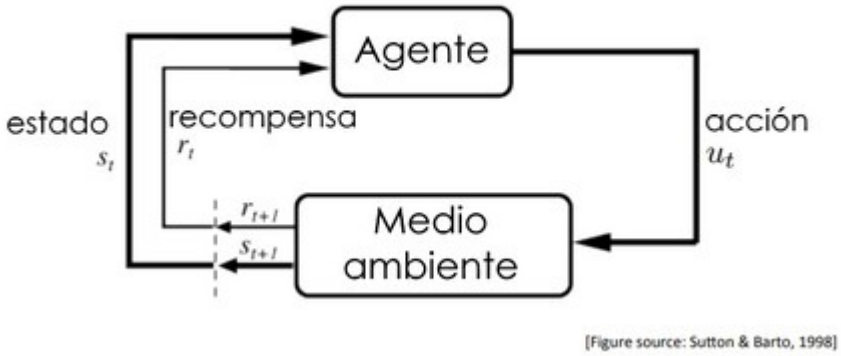


Imagen 1 Bucle aprendizaje reforzado

- El agente ejecuta una acción especifica **a**, en un tiempo **t**, en el ambiente.
- El ambiente reacciona a dicha acción, generando un estado **s** nuevo, y una recompensa **r**, en un tiempo **t+1**.
- El agente, toma el nuevo estado, y genera una nueva acción acorde para maximizar la recompensa, lo cual repite el bucle.
- El bucle es repetido hasta que se llega a un estado terminal, en el que no se puede continuar, este estado es definido por el tipo de proyecto, por el tipo de ambiente, o por el tipo de función de recompensa.

El objetivo del bucle es que el agente tenga muchas oportunidades de ver muchos estados y ejecutar muchas acciones dentro del ambiente, de manera de poder codificar dentro de

los pesos del modelo de red neuronal, una distribución de acciones a ejecutar dadas ciertas observaciones, que le permitan maximizar la recompensa obtenida al final del bucle.

3. Hipótesis y Objetivos

3.1. Hipótesis:

Un agente de aprendizaje reforzado que al ser entrenado con métodos tradicionales de RL (Proximal Policy Optimization, por ejemplo) no podrá ser capaz de ganar o anotar puntos de manera constante en el nivel King Hippo del juego retro clásico "Punch-Out!" de NES. Solo después de recibir una transferencia de aprendizaje de un experto (humano), podrá tener esa pieza de información necesaria para poder anotar puntos de manera constante, o ganarle.

En el caso del juego clásico "Punch-Out**!", la acción correcta a ejecutar debe ser una muy precisa y especifica en una ventana de tiempo reducida**, lo que dificulta su aplicación.

3.2. Objetivos generales

3.2.1. Implementar un proyecto de aprendizaje reforzado completo

Como principal objetivo general se tiene poder implementar de manera exitosa un agente, un ambiente, un modelo de aprendizaje reforzado (RL), y un modelo de aprendizaje

reforzado inverso (IRL). Para esto debo dominar la funcionalidad básica de las siguientes librerías:

- Retro Gym.
- Stable Baselines 3.
- Imitation.

3.2.2. Realizar una transferencia de aprendizaje a través de Aprendizaje Reforzado Inverso (IRL) al mejor modelo básico

Una vez elegido el mejor modelo básico generado en la primera fase del proyecto, se tiene como segundo objetivo general realizar una transferencia de aprendizaje exitosa a dicho modelo, utilizando demostraciones de un experto.

3.3. Objetivos específicos

3.3.1. Integrar el juego de NES Punch-Out a la librería Retro Gym

La librería **Retro Gym** no viene con el juego clásico retro del NES **Punch-Out**, por lo que se debe realizar un proceso de integración de éste para poder ser utilizado como entorno **GYM** en el proyecto.

3.3.2. Reducir el espacio de acciones ("Action Space") y, el espacio de observaciones ("Observation Space") del ambiente gym

El ambiente de entrenamiento tiene tanto un espacio de acciones, como de observaciones muy amplios, por lo que se propone como uno de los objetivos específicos del proyecto lograr la reducción de ambos, de manera de agilizar los requisitos de tiempo, y reducir las capacidades de procesamiento necesarias para entrenar los modelos.

3.3.3. Elaborar dos funciones de recompensa, para la fase de entrenamiento

Como se mencionó anteriormente, la función de recompensa es un elemento muy importante dentro de un proyecto de aprendizaje reforzado, debido a que permite al modelo (y por ende, al agente) saber si está mejorando en su tarea de aprender a navegar a través del entorno, y lograr su meta, por lo que se propone como un objetivo específico la elaboración de dos funciones de recompensa, de manera de poder realizar comparaciones entre ambas, y así tener más datos que avalen los resultados obtenidos durante el proyecto.

3.3.4. Entrenar un modelo para cada función de recompensa definida, y elegir el que tenga mejor comportamiento

Una vez encuentre los mejores hiper parámetros al modelo elegido, proceder a entrenar dicho modelo RL para cada función de recompensa definida por una cantidad elevada de pasos (de un millón de pasos en adelante) se vuelve un objetivo específico.

3.3.5. Elaborar el dataset con trayectorias de experto para realizar la transferencia de aprendizaje

Para poder realizar la transferencia de aprendizaje desde las trayectorias de un experto al modelo elegido, primero debemos crear un dataset con tuplas observación-acción, que deben ser creadas por un humano experto en el nivel King Hippo. Una vez creado el dataset de demostraciones de experto utilizaremos la librería “Imitation” para realizar la transferencia de aprendizaje.

3.3.6. Implementación del algoritmo de IRL GAIL

Como última fase del proyecto, se tomarán las demostraciones de experto generadas por un humano (mi hijo), y se utilizarán en conjunto con el algoritmo GAIL, para “enseñar” a dicho modelo como anotar puntuaciones elevadas, o incluso vencer al oponente “King

Hippo” de manera constante. Si se logra cualquiera de las condiciones podríamos decir que el proyecto es exitoso.

4. Datos y Metodología

Los datos en un proyecto de RL, a diferencia de un proyecto de aprendizaje supervisado, o no supervisado, no están formados por un conjunto de tuplas “imágenes-etiquetas”, ni por un conjunto de documentos, los cuales deben ser clasificados según su contenido, sino que deben ser generados durante la fase de entrenamiento, lo que comúnmente se llama la fase “Online Learning”, y están compuestos principalmente por:

- El **espacio de observaciones**, en el caso del proyecto en curso, el espacio de observaciones está compuesto de imágenes.
- El **espacio de acciones**, que está compuesto de acciones discretas, es decir, se codifican los combos de botones a un número entero.
- El **estado** (state), que está formado por algunas variables propias del entorno de entrenamiento, y por ambos espacios, el de observaciones, y el de acciones.

La metodología que se utilizará en el desarrollo del proyecto es una combinación de varias:

- Para la utilización de la libraría Retro-Gym, se utilizarán las recomendaciones de trabajo contenidas en la documentación de esta.
- Para la utilización de la librería “Stable Baselines 3”, se seguirán las recomendaciones del trabajo en ((RM), 2020).

- Para la utilización de la librería “Imitation”, de nuevo, se seguirán las recomendaciones propuestas en su propia documentación (AI, 2019).
- Como influencia general, se utilizará el método científico, y algunos videos tutoriales de YouTube como (Renotte, 2022) que implementan proyectos similares.

En el apartado de metodología se ahondará en los detalles a seguir.

4.1. Datos

4.1.1. Breve descripción del juego clásico de NES Punch-Out

La consola clásica llamada Nintendo Entertainment System, (**NES** como es mayormente conocida) es considerada la consola que hizo posible el resurgimiento de la industria de los video juegos luego de la debacle de 1983 causada principalmente por la compañía Atari, y sus adversarios, al sobrepoblar el mercado con copias baratas de los mismos juegos.

Consta de una consola con procesador de 8 bits (largo máximo de cada palabra dentro del sistema computacional), un procesador NMOS de 1,78 MHz de velocidad de procesamiento, una capacidad de memoria RAM de 2KB, una bahía para introducir los cartuchos intercambiables de los juegos, dos entradas para dos controles, una salida de audio/video RF, y una entrada para energía DC (proveniente de un adaptador de corriente).

Una breve descripción del juego de Punch-Out se da a continuación:

- Salió al mercado americano en 1987.
- Es un juego de boxeo, en el cual el jugador controla a **"Little Mac"**, para poder golpear a, y esquivar los golpes del oponente.
- Tuvo dos iteraciones dentro de su ciclo de vida: una conocida como "Mike Tyson's Punch-Out!!" (la utilizada en este proyecto, y la más querida), y la otra, llamada "Punch-Out!!" En la que se prescinde del nombre del atleta Mike Tyson.
- Fue creado por el mítico equipo de Investigación y Desarrollo 3 dentro de la compañía Nintendo.
- Es un juego de un solo jugador.
- Consta de 13 niveles, teniendo 10 distintos peleadores oponentes, de los cuales tres se repiten.
- El nivel se gana al tumbar tres veces dentro de un mismo round al enemigo, o bajo ciertas condiciones específicas a cada enemigo.
- El nivel 5: **King Hippo**, es el nivel que será utilizado durante el proyecto, y en el cual se basa la hipótesis de este. El nivel tiene una particularidad, y es que el oponente **solo puede ser golpeado cuando su short se cae**, y esto solo sucede cuando el jugador golpea a King Hippo en una posición especifica, **que solo dura fracciones de segundo**, y que adicionalmente, **solo sucede al menos cinco veces** dentro d un asalto de tres minutos, lo que lo hace un reto para los jugadores menos experimentados.

4.1.2. Elaboración de las funciones de recompensa de los modelos RL

El juego clásico de NES Punch-Out, no viene integrado en la librería Gym-Retro por defecto, por lo cual hay que ejecutar un proceso de integración de este, en el cual se incorporan los elementos del juego, y adicionalmente se crean las funciones de recompensa a utilizar.

En (Brockman, 2016) se menciona la existencia de dicha herramienta de integración, y en (Videogames.ai, 2021) se tiene tutorial comprensivo, en video, de cómo utilizar dicha herramienta.

Este proceso, se realiza mediante un mapeo de las variables dentro de la memoria RAM del juego, permitiendo extraer cualquier variable dentro del entorno del juego, como:

- Puntaje del personaje, salud de nuestro personaje
- Cuantas veces ha recibido golpes, etc.

las cuales son específicas para cada juego. Para comenzar el proceso, se debe activar la herramienta “Integration Tool”, desde el entorno Python, que se encuentra en el directorio de la librería Gym-Retro. A continuación, una imagen de la interfaz GUI de la herramienta:

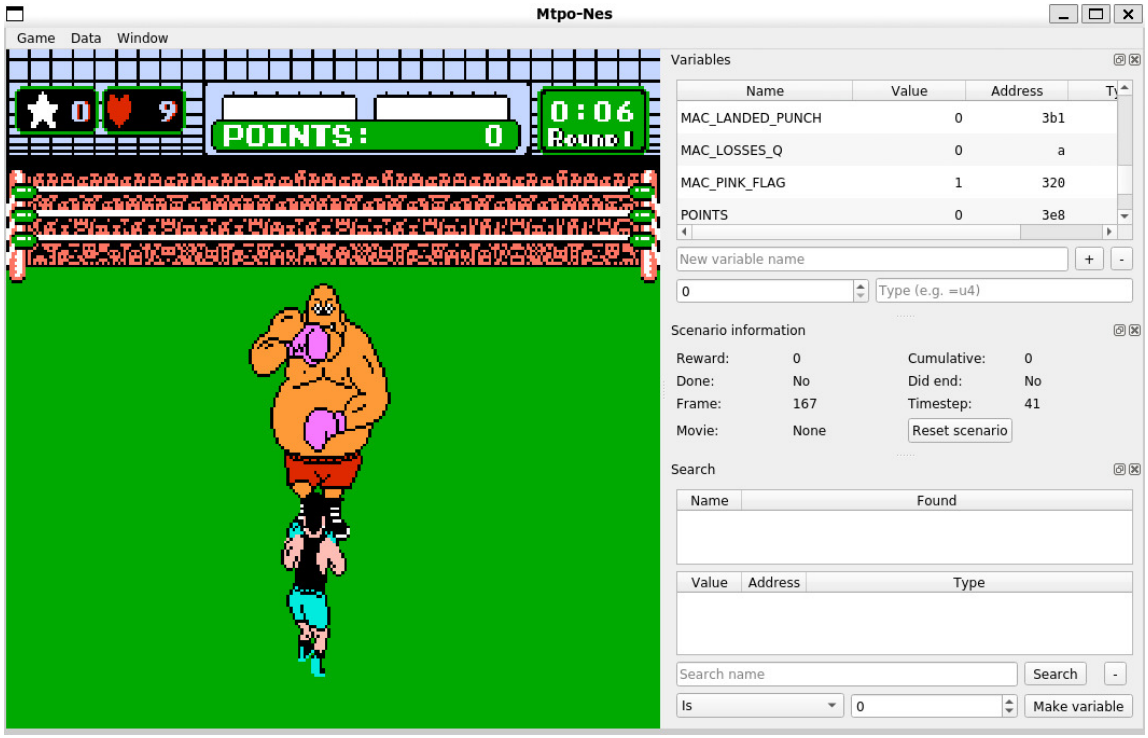


Imagen 2 GUI de la herramienta de integración

El resultado del proceso de integración es la posibilidad de utilizar el juego retro clásico Punch-Out! como un entorno de aprendizaje RL, con todas sus características como:

- La capacidad de crear un entorno Gym-Retro utilizando el nombre del juego Punch-Out.
- La capacidad de comenzar el entrenamiento en el nivel deseado, a través del concepto de "**game states**", que en resumen son una "foto" que puede ser cargada al comienzo del entrenamiento, para dejar al agente en el comienzo de la acción, y así evitar que deba aprender cómo utilizar la pantalla de inicio, y los menús del comienzo del juego. Como es de esperarse, todos los entrenamientos realizados como parte del proyecto se hicieron en el nivel "King Hippo", sin embargo, como

adicional, también se exploró el primer nivel, para poder hacer comparaciones de los resultados.

- La creación de un archivo tipo JSON llamado “data.json” en formato de tabla, el cual contiene la información de las variables extraídas del Gym, y que será pasado al agente durante cada transición del bucle de RL con el nombre de variable “info”. A continuación, se muestra el contenido del archivo “data.json” en forma de tabla que será utilizado en el proyecto, el cual sirve como base para la creación de las funciones de recompensa:

Tabla 1 Variables del entorno Gym-Retro del proyecto

Nombre	Dirección en RAM	Tipo de dato	Breve descripción
ENEMY_DOWNED_Q	977	<d1	Cantidad de caídas del enemigo
POINTS	1000	>n6	Puntaje dentro del juego
FIGHT_INIT	13	<d1	"Flag" de inicio de pelea
HEART_LOSS	805	<d1	"Flag" de pérdida de corazón (stamina), o de recepción de un golpe
MAC_KO_Q	976	<d1	Cantidad de caídas de Little Mac
MAC_LANDED_PUNCH	945	<d2	"Flag" de golpe exitoso de Little Mac
UPPER_FLAG	613	|u1	"Flag" de upper-cut exitoso de Little Mac

Es entonces a partir de estas variables del entorno Gym-Retro que se crean las funciones de recompensa:

4.1.2.1. Función de recompensa “POINTS”

Esta función de recompensa realiza un mapeo 1:1 de la cantidad de puntos dentro del juego, a la recompensa del agente, asignando el valor de la puntuación actual dentro del juego en cada transición al valor de la recompensa del agente, la cual a su vez será utilizada por el modelo de RL en todos sus cálculos. Podemos decir que es una recompensa semi densa, debido a que el agente está en constante retroalimentación de su desempeño en el juego, aunque sea por lo menos con una sola variable.

4.1.2.2. Función de recompensa “PUNCHES”

Se denominó a esta función de recompensa “PUNCHES” debido a que utiliza los puñetazos que Little Mac conecta, y también los que recibe, como variables para tomar en cuenta para su recompensa. Esta función de recompensa realiza una retroalimentación más activa del desempeño del agente (y por ende el modelo) dentro del entorno, tal y como lo vemos resumido en la siguiente tabla:

Tabla 2 Variables de la función de recompensa "PUNCHES"

Acción	Recompensa	Breve descripción
ENEMY_DOWNED_Q	200	Otorga 200 puntos por cada derribo al enemigo
HEART_LOSS	-5	Quita 5 puntos por cada golpe recibido, o por cada golpe que el enemigo logra bloquear
MAC_KO_Q	-200	Quita 200 puntos por cada caída que recibe Little Mac
MAC_LANDED_PUNCH	5	Otorga 5 puntos por cada golpe acertado correctamente
UPPER_FLAG	20	Otorga 20 puntos por cada upper-cut acertado correctamente

4.1.3. Espacio de Acciones

El espacio de acciones del proyecto está formado por combos de botones que deben ser presionados para ejecutar las acciones dentro del juego. Una consola NES tiene un control con dos botones de acción, un pad de direcciones, y dos botones de selección.

En total, el control de NES, tiene un pad direccional (cuatro direcciones individuales), dos botones de acción (A, y B), y dos botones de selección (START, SELECT), si codificamos dichos botones como un vector multibinario, tenemos 2^8 combinaciones, es decir, 256 posibles combos de botones. Como veremos más adelante, no todas las combinaciones son necesarias para poder jugar de manera correcta el juego.

Los objetos Retro Gym de la consola NES tienen varios tipos de espacios de acción:

1. Multibinarios.
2. Multidiscretos.
3. **Discretos.**

Con cada uno de ellos se puede representar las acciones dentro del juego.

4.1.4. Espacio de observaciones

El espacio de observaciones es como se le conoce al contenido visual de un algoritmo/ambiente de un proyecto de RL. En el caso de este proyecto, el espacio de observación es la pantalla del juego, que consiste en una imagen de 224 pixeles de ancho, por 240 pixeles de alto, y tres canales de color, como se muestra a continuación:

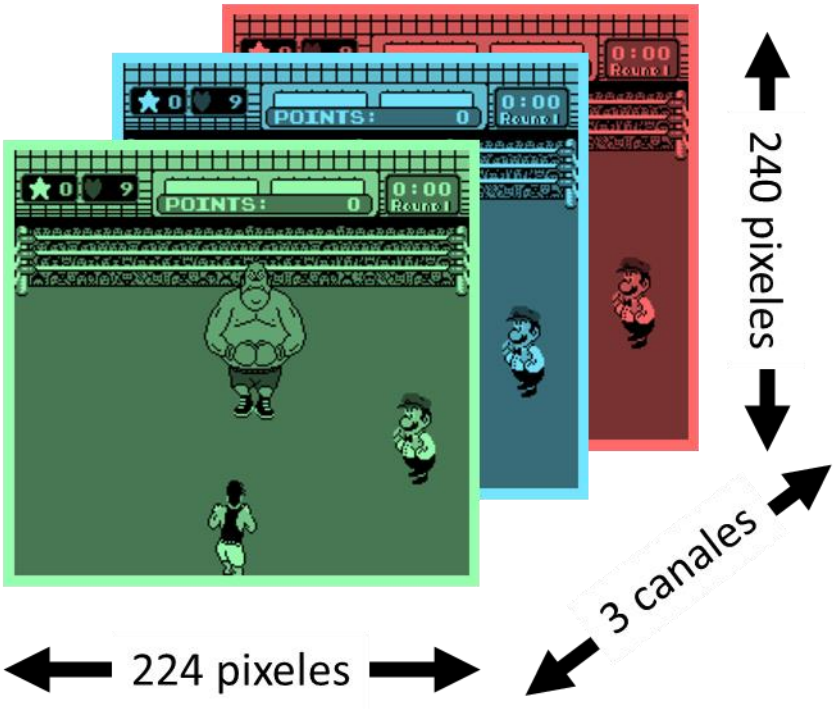


Imagen 3 Espacio de observaciones completo

4.1.5. Reducción de los espacios de acción, y observación

Una práctica recomendada en los proyectos de RL es reducir los espacios de acción y de observación debido a que esto permite disminuir las exigencias de capacidades de procesamiento, y esto a su vez, se traduce en reducciones considerables del tiempo de ejecución de los proyectos.

Para el espacio de acciones, se pasará de un espacio multi binario (que viene por defecto), a un espacio discreto de un combo de botones a la vez. A continuación, se muestra el espacio de acciones final:

```
DODGE = [
[], # Sin movimiento
['RIGHT'], # Esquiva a la derecha
['LEFT'], # Esquiva a la izquierda
['DOWN'], # Se cubre
['UP', 'A'], # Golpea a la cara con un derechazo
['UP', 'B'], # Golpea a la cara con un izquierdazo
['A'], # Golpea al cuerpo con un derechazo
['B'], # Golpea al cuerpo con un izquierdazo
['START'], # Utiliza súper poder
]
```

Imagen 4 Espacio de acciones reducido

Para el espacio de observaciones, se realizará una reducción de las imágenes, utilizando la librería **OpenCV** en su versión de Python, que permite la manipulación de imágenes. Esta manipulación de imágenes se realizará en cada pasada del buble de RL, ejecutando los siguientes pasos:

- Generamos una zona de enfoque, que corta dos tercios de la pantalla en los extremos laterales, dejando la imagen con 196 pixeles de alto, 80 de ancho, y tres canales de color:

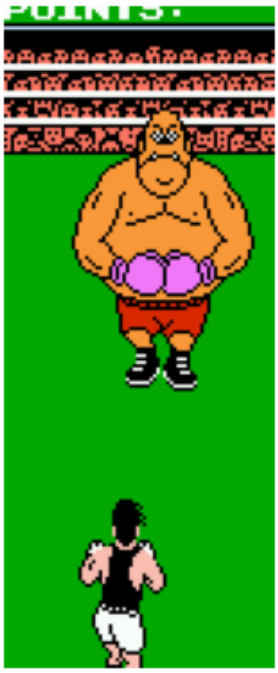


Imagen 5 "Área de enfoque" con tres canales de color

- Reducimos a uno la cantidad de canales de colores.

Imagen 6 "Área de enfoque" con un canal de color

- Como último paso, reducimos la imagen a 84 pixeles de alto, 84 pixeles de ancho, y un solo canal de color:

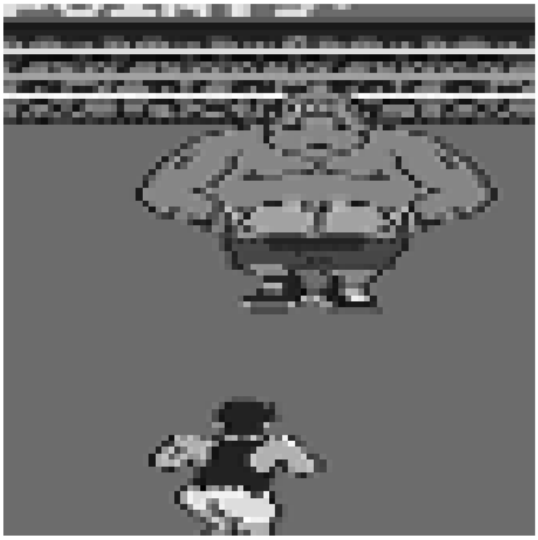
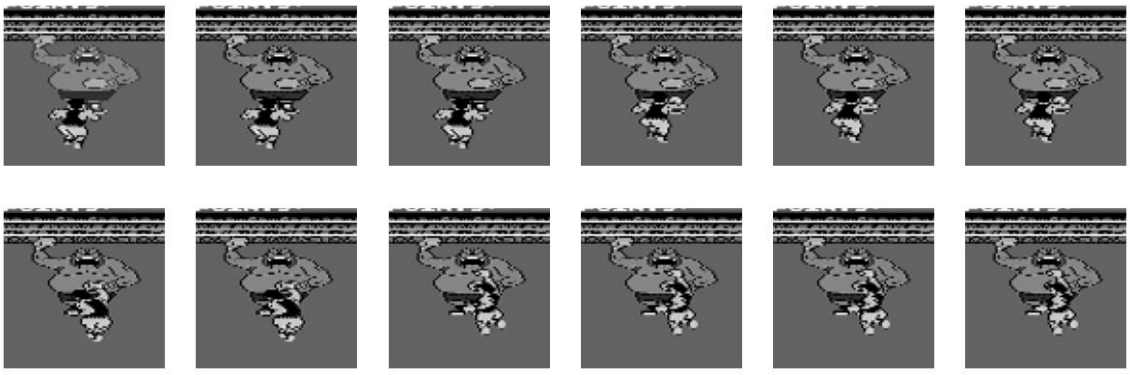


Imagen 7 "Área de enfoque" reducida

Las operaciones anteriores se realizan modificando el funcionamiento por defecto la clase que crea el entorno GYM RETRO, e introduciendo métodos y variables adicionales a la misma para lograr el comportamiento deseado de los elementos de la librería.

4.1.6. Trayectorias/demostraciones de experto

Las demostraciones de experto, o trayectorias de experto como también son conocidas, consisten en la recolección de episodios completos jugados por expertos humanos con sus acciones, observaciones, y datos adicionales, y que son guardados en un archivo formato **NPZ**. A las observaciones se les debe hacer un procesamiento extra, que consiste en apilar 12 cuadros seguidos, para simular la impresión de movimiento de los elementos dentro del juego, tal y como se ve a continuación:



1 Imágenes de una trayectoria experta apiladas

En las que se puede apreciar ya al final, que Little Mac está comenzando a lanzar un "uppercut".

Lo ideal es contar con la mayor cantidad de muestras posibles, pero debido a restricciones de memoria RAM de las maquinas utilizadas para la implementación del presente proyecto, se utilizaron 20 trayectorias de experto, aunque se recolectaron 100 en total.

La recolección de las trayectorias de experto se realizó modificando una clase de la librería Gym Retro, que permite "wrappear" un ambiente GYM en una función que habilita el control de un humano del agente, y permite guardar en un objeto "**Pickle**" las tuplas observación-acción-información recolectadas, transformadas a un objeto "**Trajectory**", utilizado en la librería "**Imitation**".

4.2. Metodología

4.2.1. Modificación de las clases base de las librerías utilizadas

Para poder llevar a cabo las reducciones de los espacios de acción, y observaciones anteriormente mencionadas, se deben realizar modificaciones a las clases base de las librerías utilizadas.

Adicionalmente, se crearon herramientas para poder recolectar las demostraciones de experto, y poder generar todos los cambios requeridos en la mismas.

4.2.2. Ejecución de experimentos de los distintos algoritmos RL

Llamaremos a cada ejecución de un entrenamiento de durante al menos un millón de pasos, un experimento. Realizaremos experimentos con un agente PPO en un ambiente RL tanto contra el enemigo mencionado en la hipótesis, como contra un enemigo del primer nivel del juego, para tener muestras variadas del agente, y tener contra qué compararlo. La cantidad de experimentos por nivel se limitará a dos debido a las necesidades de procesamiento exigidas por los algoritmos (cada experimento puede tomar hasta 72 horas en CPU, que es hardware utilizado en el proyecto). Estos experimentos iniciales permitirán comprobar la primera parte de la hipótesis propuesta, que es:

> *"Un agente de aprendizaje reforzado que al ser entrenado con métodos tradicionales de RL (Proximal Policy Optimization, por ejemplo) no podrá ser capaz de ganar o anotar puntos de manera constante en el nivel King Hippo del juego retro clásico "Punch-Out!" de NES"*

Esto lo podremos constatar debido a que el mismo tipo de agente, uno PPO, entrenado para vencer al enemigo y anotar puntos de manera constante en el primer nivel, lo logrará sin problemas.

4.2.3. Ejecución de los experimentos IRL

Una vez culminada la primera parte, se procederá a la recolección de las demostraciones de experto humano, las cuales serán ingresadas al algoritmo GAIL. Se ejecutará un experimento GAIL que cumplirá con la segunda parte de la hipótesis propuesta:

> *"Solo después de recibir una transferencia de aprendizaje de un experto (humano), podrá tener esa pieza de información necesaria para poder anotar puntos de manera constante, o ganarle.".*

Pudiendo ganarle, o anotar una cantidad elevada de puntos, de manera constante al enemigo mencionado en la hipótesis.

4.2.4. Evaluación de los agentes RL e IRL en los experimentos durante la fase de aprendizaje

Para evaluar a los agentes en la etapa de aprendizaje se tomará la salida que muestran los algoritmos durante dicha fase. Se utilizará una combinación de gráficas de Tensorboard (donde están presentes), y de los KPI de aprendizaje de cada uno de los algoritmos.

4.2.5. Evaluación de los agentes RL e IRL en los experimentos después de la fase de aprendizaje

Para evaluar la efectividad de los agentes y las pólizas entrenadas durante los experimentos RL, e IRL, se utilizará una función creada para dicho propósito, la cual

tomará como entrada el agente, el ambiente, y realizará 1000 episodios seguidos, de los cuales se extraerá la siguiente variable:

- La cantidad de puntos obtenidos por el agente.

Estos valores, nos permitirán realizar una comparación entre los distintos agentes, y los distintos paradigmas, con lo cual estaremos en condiciones para confirmar o refutar la hipótesis inicial de manera objetiva, es decir, con datos.

Existe otra manera de evaluar el desempeño de un agente, una manera subjetiva, de mera apreciación humana de cómo se desenvuelve un agente en un combate, y que tiene que ver con lo siguiente:

- Si el agente es capaz de encontrar un "exploit" en alguno de los combates, que le permita ganar fácilmente.
- Si el agente se comporta de manera parecida a la "humana", es decir, si evita recibir golpes, y da varios golpes seguidos cuando tiene la oportunidad.

A continuación, se muestra una tabla resumen de los experimentos a realizar:

Tabla 3 Experimentos a realizar para la evaluación de los agentes

Nivel	**Agente**	**Evaluación**
Glass Joe (1er nivel)	• Random Agent • PPO	Objetiva y subjetiva
King Hippo (5to nivel)	• Random Agent • PPO • GAIL	Objetiva y subjetiva

Se ha incluido un "Random Agent", que ejecuta acciones al azar, como muestra de control.

5. Resultados

Los resultados se dividen en dos partes, primero, los resultados durante la etapa de entrenamiento de RL, que muestran principalmente la evolución de los mejores modelos entrenados durante dicha fase, y segundo, los resultados de la fase de evaluación de los mejores modelos de la fase IRL.

5.1. Resultados durante la fase de entrenamiento

5.1.1. Resultados del entrenamiento del modelo PPO (RL) para el nivel 1: Glass Joe

La implementación de la librería SB3, permite la utilización de la herramienta Tensorboard, la cual muestra mediante un gráfico, la evolución de la etapa de aprendizaje. Para el primer nivel (Glass Joe), con un modelo PPO entrenado durante 2 millones de pasos para la función de recompensa POINTS, y sobre 800 mil pasos para la función de recompensa “PUNCHES”, se muestra a continuación los gráficos de aprendizaje de las funciones de recompensa “POINTS” y “PUNCHES”:

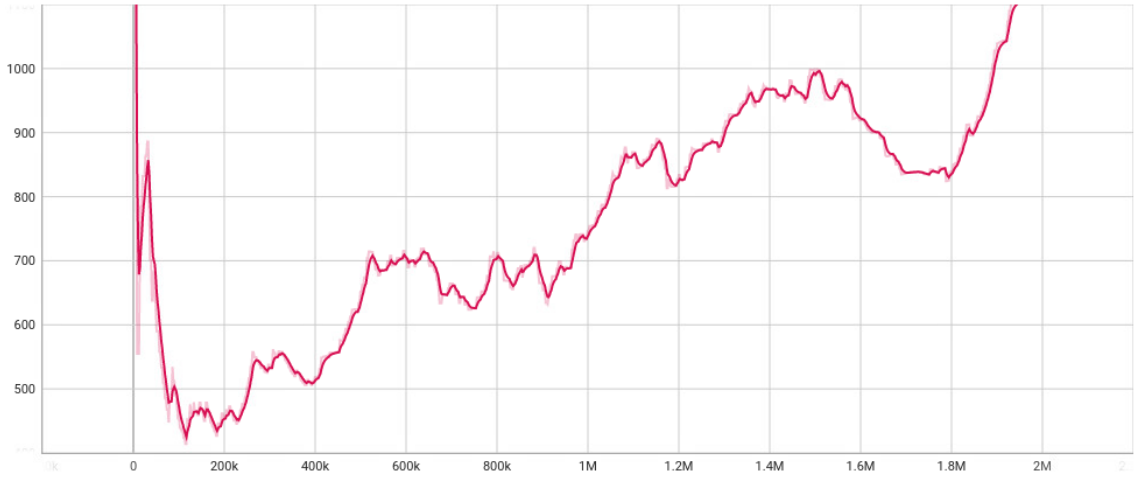


Imagen 8 Evolución de la recompensa "POINTS" en el nivel Glass Joe

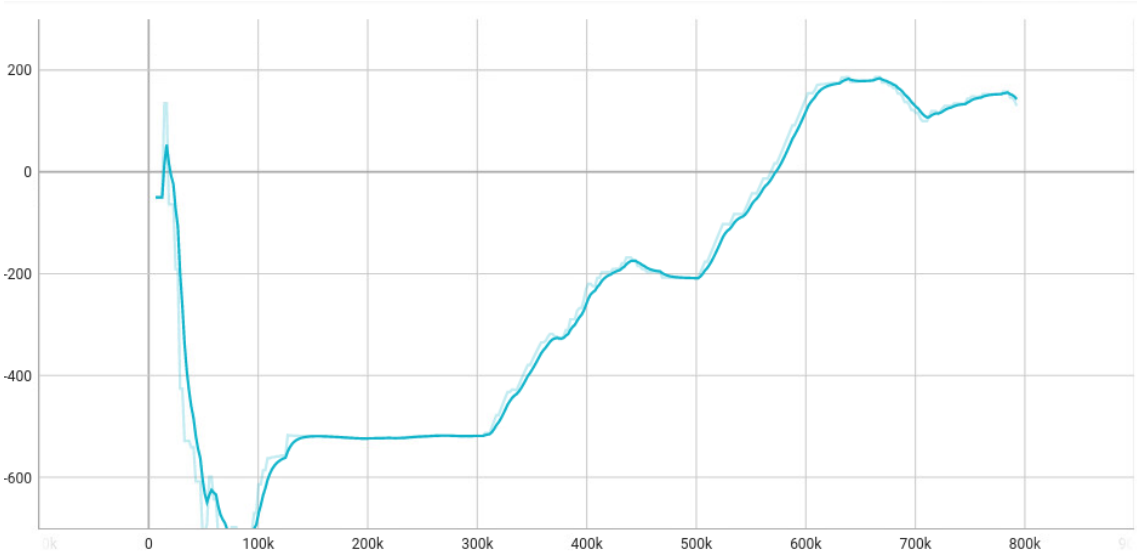


Imagen 9 Evolución de la recompensa "PUNCHES" en el nivel Glass Joe

Es evidente que para ambas funciones de recompensas hay una clara tendencia de aumento de la recompensa promedio a medida que avanza el tiempo de entrenamiento. Sin duda se puede asegurar que ambos modelos son exitosos y logran su cometido, sin embargo, el modelo con la función de recompensa “POINTS”, logra una puntuación más alta.

Para el nivel King Hippo, utilizando un modelo PPO, se muestra a continuación se muestran los gráficos evolución de recompensa media:

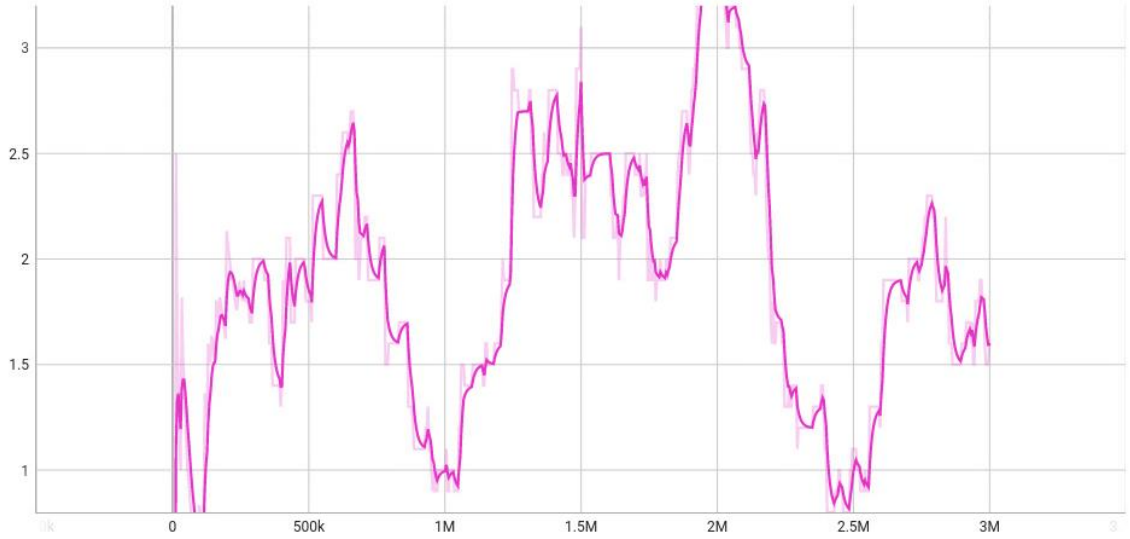


imagen 10 King Hippo evolución recompensa "POINTS"

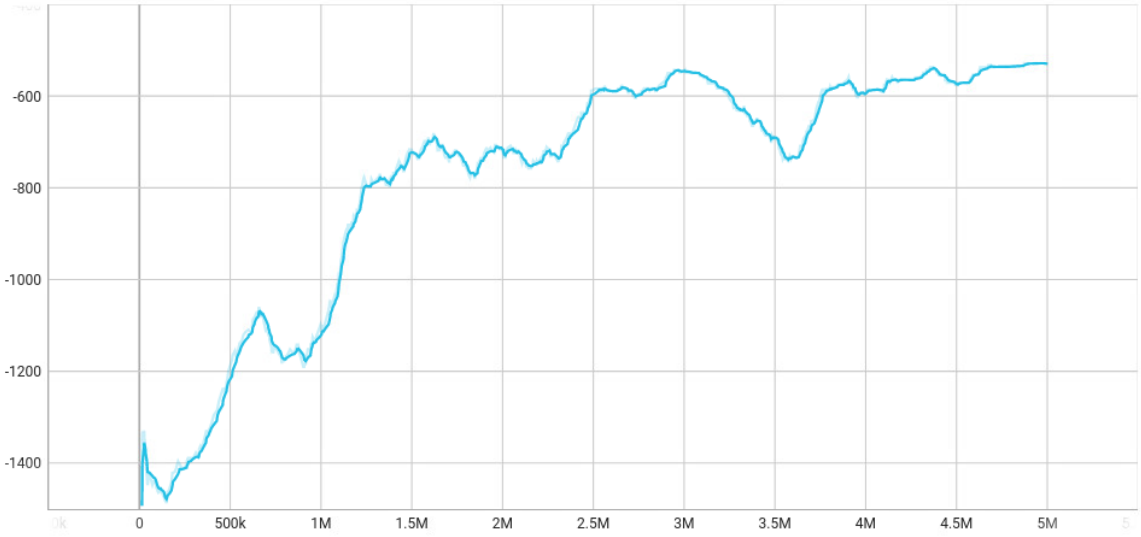


imagen 11 King Hippo evolución recompensa "PUNCHES"

En este caso, para el nivel de King Hippo, la función de recompensa PUNCHES es la que tiene una clara tendencia al alza, partiendo desde un valor cercano a -1400, y llegando hasta un valor de -600, donde hay una meseta sin aumento considerable. Sin embargo, en el caso de la función de recompensa POINTS, la evolución es irregular, apenas pasando sobre 3 puntos. Ninguno de las funciones de recompensa logra una puntuación alta de manera regular.

5.1. Resultados durante la fase de evaluación

5.1.1. Resultados para el nivel 1: Glass Joe

A continuación, se muestra la distribución de las puntuaciones obtenidas por los agentes RANDOM, y PPO en el primer nivel del juego (nivel de control) a lo largo de **1000** episodios completos jugados por ambos agentes:

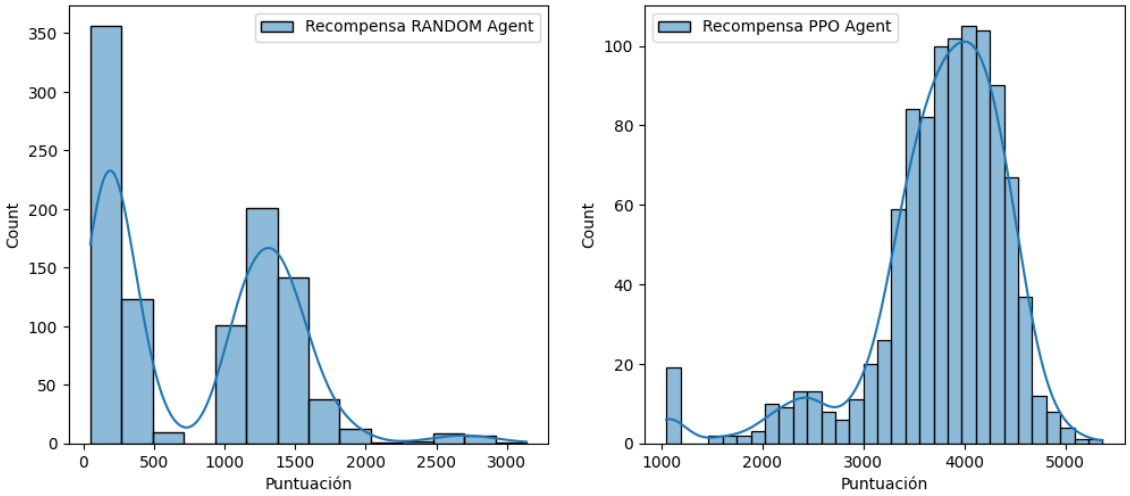


Imagen 12 Histogramas de puntuaciones (Glass Joe)

Visualmente se podría decir que las distribuciones provienen de distintos procesos. El resumen de cada distribución se muestra a continuación:

	Recompensa RANDOM Agent
count	1000.000000
mean	813.060000
std	636.942802
min	50.000000
25%	180.000000
50%	1050.000000
75%	1342.500000
max	3140.000000

	Recompensa PPO Agent
count	1000.000000
mean	3764.650000
std	689.298782
min	1050.000000
25%	3497.500000
50%	3880.000000
75%	4210.000000
max	5370.000000

Imagen 13 Resumen de las variables RANDOM y PPO Agents

La media de puntuación del agente RANDOM es de 813,06, y la del agente PPO es de 3764,65.

Para contar con un dato estadísticamente significativo que permita asegurar que existe una diferencia entre ambas distribuciones se ejecutará una prueba de hipótesis, una prueba T-STUDENT para dos distribuciones, utilizando la librería **SCIPY** de Python.

El resultado es el siguiente:

```
t_result_glassjoe = stats.ttest_ind(random_glassjoe_rew_df, ppo_glassjoe_rew_df)
alpha = 0.05

if (t_result_glassjoe[1] < alpha):
    print('Los agentes RANDOM y PPO son generados por distintos procesos.')
else:
    print('Los agentes RANDOM y PPO son generados por el mismo proceso.')
print(t_result_glassjoe)
```

```
Los agentes RANDOM y PPO son generados por distintos procesos.
Ttest_indResult(statistic=array([-99.45124906]), pvalue=array([0.]))
```

Imagen 14 Resultado prueba de hipótesis Glass Joe

Es entonces seguro decir, con una significancia del 95 %, que las puntuaciones de ambos agentes **provienen de procesos distintos**.

5.1.2. Resultados nivel 5: King Hippo (Hipótesis)

Luego de completar 1000 episodios, los resultados de cada agente (RANDOM, PPO, GAIL) para el nivel King Hippo son:

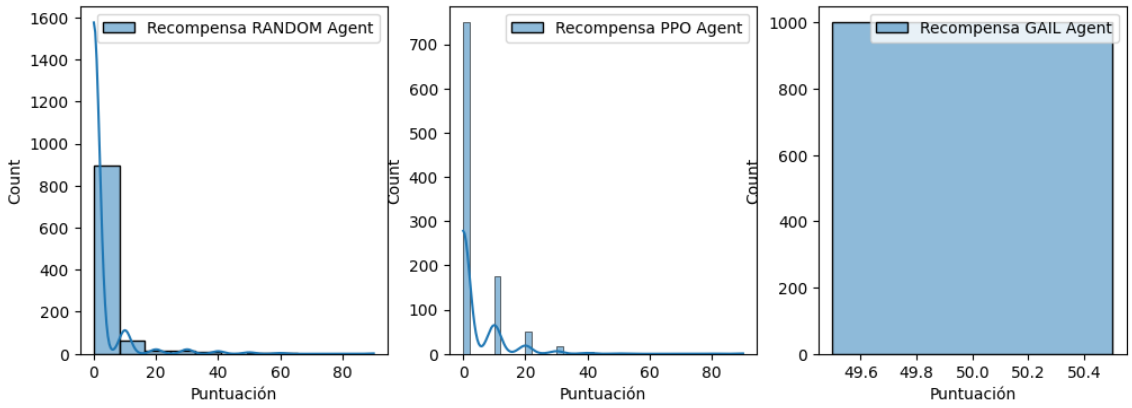


Imagen 15 Histogramas de puntuaciones (King Hippo)

El mejor modelo GAIL que fue entrenado durante 500.000 pasos, es el modelo que logra una puntuación más elevada y constante con una media de 50 puntos, en TODAS las ocasiones que juega, es decir es completamente constante durante su ejecución. Como resumen de cada distribución tenemos:

	Recompensa RANDOM Agent
count	1000.000000
mean	1.930000
std	7.404723
min	0.000000
25%	0.000000
50%	0.000000
75%	0.000000
max	90.000000

	Recompensa PPO Agent
count	1000.000000
mean	3.790000
std	8.556232
min	0.000000
25%	0.000000
50%	0.000000
75%	10.000000
max	90.000000

	Recompensa GAIL Agent
count	1000.0
mean	50.0
std	0.0
min	50.0
25%	50.0
50%	50.0
75%	50.0
max	50.0

Imagen 16 Resumen de las variables RANDOM, PPO, y GAIL Agents

De nuevo, utilizaremos una prueba estadística para corroborar que las distribuciones provienen de distintos procesos, es decir, son significativamente diferentes, vamos a utilizar una prueba ANOVA (específicamente Oneway ANOVA, cuya hipótesis nula busca medir si dos o más variables tienen la misma media de población). Lo que da como resultado:

```
t_result_kinghippo = stats.f_oneway(random_kinghippo_rew_df, ppo_kinghippo_rew_df, gail_kinghippo_rew_df)
alpha = 0.05

if (t_result_kinghippo[1] < alpha):
    print('El agente RANDOM, el PPO, y el GAIL son generados por distintos procesos.')
else:
    print('El agente RANDOM, el PPO, y el GAIL son generados por el mismo proceso.')
print(t_result_kinghippo)
```

```
El agente RANDOM, el PPO, y el GAIL son generados por distintos procesos.
F_onewayResult(statistic=array([17375.74974553]), pvalue=array([0.]))
```

Imagen 17 Resultado prueba de hipótesis King Hippo

5.1.3. Resumen de los resultados

A continuación, veremos una tabla con el resumen de los resultados:

Tabla 4 Resultados globales de la etapa de evaluación

Agente	Cantidad de episodios	Recompensa promedio Glass Joe	Recompensa promedio King Hippo
Random	1000	813,06 (639 std)	1,93 (7,4 std)
PPO	1000	3764,65 (689 std)	3,79 (8,5 std)
GAIL	1000	X[1]	50 (0 std)

- Para tener una base con qué comparar los resultados de cada algoritmo, se realizaron 1000 experimentos en cada nivel (Glass Joe, y King Hippo), con un agente al azar, el cual elige sus acciones estocásticamente. Su desempeño es, entonces, la base para medir a los demás algoritmos. Sus resultados fueron:
  - En el nivel 1 el agente Random tuvo:
    - 813 puntos de promedio, con una desviación estándar de 639.
  - En el nivel 5, tuvo:
    - 1,93 puntos de promedio, con una desviación estándar de 7,4.
- Los resultados del algoritmo PPO, que permitirán tener un contraste con el RL tradicional fueron:
  - En el nivel 1 el agente PPO, luego de 1000 episodios tuvo:
    - 3764,65 puntos de promedio, con una desviación estándar de 689.
  - En el nivel 5, tuvo:
    - 3,79 puntos de promedio, con una desviación estándar de 8,55.
- Finalmente, para el algoritmo GAIL, que es el foco de la hipótesis del proyecto, el cuál basa su accionar en las demostraciones de expertos humanos tuvo el siguiente resultado:
  - En el nivel 5:
    - Por las razones explicadas anteriormente, no se realizaron experimentos en el nivel 1 con el algoritmo GAIL.
  - En el nivel 5, tuvo:

[1] Para el nivel 1 no se realizaron experimentos para GAIL debido a que el algoritmo PPO por sí solo tiene extraordinarios resultados. El uso de experimentos en el nivel 1 es principalmente para tener un grupo de muestras de control que faciliten contrastar los resultados entre el algoritmo RL tradicional PPO, y el algoritmo IRL GAIL.

- 50 puntos de promedio, con una desviación estándar de 0.

Este último resultado es la culminación del proyecto como tal, para poder obtenerlo, se debió entrenar el modelo GAIL con demostraciones de experto, lo que en sí mismo, no es trivial.

Es entonces evidente que los algoritmos de RL, e IRL son mejores que la base, que un agente Random, sin embargo, en la sección 6 se ahondará en dichos resultados, y se planteará que estos avalan la hipótesis presentada en la sección 3.

6. Conclusiones

Como se mencionó anteriormente, se evalúan los algoritmos RL, y IRL de manera objetiva, con los datos obtenidos y pruebas estadísticas sobre los mismos, y, de manera subjetiva, es decir, con la apreciación de un humano experto.

6.1. Conclusiones objetivas

Según lo observado en la sección 5.1.1, se puede hacer la siguiente declaración:

- Un modelo PPO, entrenado durante suficiente tiempo en un proyecto RL, es capaz de desarrollar una póliza que puede lograr puntuaciones altas en el primer nivel: Glass Joe, de manera constante, lo que se traduce en que **el agente PPO es mucho mejor que el azar**. Lo anterior sin la necesidad de recibir una transferencia de aprendizaje de algún experto.

Esta conclusión es respaldada por los datos obtenidos en dicha sección, que son resultado de la fase de entrenamiento.

Basados en los resultados mostrados en 5.1.2, se puede indicar que:

- El resultado de la prueba estadística ANOVA, que asegura que las muestras de los 1000 episodios jugados por cada agente son diferentes, y, por ende, el agente GAIL con un promedio de 50 puntos por episodio es el que mejor desempeño tiene, Lo anterior **permite corroborar satisfactoriamente la hipótesis principal del proyecto**, lo que significan buenas noticias, debido a que el modelo GAIL al final genera una póliza muy cercana a las trayectorias de experto mostradas.

6.2. Conclusiones subjetivas

Cuando un humano ve el desempeño de los agentes en ambos niveles, se le hace evidente que en el caso de los algoritmos PPO, el agente busca acertar la mayor cantidad de golpes que le sea posible, sin embargo, este comportamiento no asegura la victoria, y la mayoría de las veces los agentes solo anotan altas puntuaciones sin ganar los encuentros.

En el caso del modelo GAIL, para el nivel King Hippo, es indiscutible que este es una mejora sobre el modelo PPO, y sobre el agente RANDOM, lo que permitió decir en la sección anterior que la hipótesis propuesta se cumple, sin embargo, el agente solo logra anotar una cantidad específica de puntos de manera constante, **sin obtener la victoria nunca**.

En general, se puede decir entonces, que, aunque los modelos tienen comportamientos constantes y buenos, no logran igualar o sobre pasar a un jugador humano experimentado, en cuanto a cantidad y efectividad de victorias en el nivel 5: **King Hippo**.

6.3. El problema con los horizontes variables

Lo mencionado en (AI, 2019), donde se recomienda ejecutar los algoritmos de IRL en experimentos con episodios de horizonte fijo, es la razón por la que el agente GAIL del proyecto tuvo una convergencia a alargar su permanencia en el ambiente, en vez de golpear con más frecuencia durante la apertura de la ventana de acción mencionada en la sección 3.1, esto se conoce como fuga de información. La recomendación del trabajo mencionado no podía ser satisfecha en el presente proyecto, debido a la naturaleza impredecible de un combate dentro del juego, el cual puede terminar mucho antes de cualquier horizonte fijo que se haya podido establecer.

6.4. Trabajo futuro y limitaciones

Los modelos entrenados en el proyecto estaban siendo ejecutados solo en un nivel cada uno, por lo que una acción a futuro que podría aportar mucho valor es intentar entrenar un modelo que pueda competir de manera efectiva en varios niveles del juego, incluso uno que pueda jugar y ganar el juego entero.

En una futura iteración con acceso a aceleradores gráficos de alto rendimiento se podría recolectar muchas más demostraciones de experto, que contengan muchas instancias del "truco" necesario para pasar el nivel King Hippo, de manera que dicha acción pueda ser codificada en los pesos del modelo.

A futuro se podría realizar la incorporación de señales auditivas al modelo, debido a que el enemigo King Hippo hace un sonido muy particular, indicando al humano que es el momento para golpear, y ejecutar el “truco”.

La principal limitación a la hora de realizar el proyecto fue técnica, y tiene que ver con la falta de capacidad de cómputo, y de memoria RAM. Cada experimento de entrenamiento tomaba más de 72 horas utilizando dispositivos CPU, lo que dificultaba saber en un tiempo prudente, si los hiper parámetros seleccionados daban los mejores resultados. Este tema quedó más en evidencia en el apartado de las trayectorias de experto, de las cuales inicialmente se generaron 100, y que solo se pudieron utilizar 20, debido a que si se intentaba ejecutar los entrenamientos del algoritmo GAIL con más de 20 trayectorias la maquina colapsaba, y daba error de memoria.

Otra limitación bastante inesperada fue la falta de apoyo bibliográfico y explicativo sobre como recolectar las trayectorias de humano. Todas las herramientas de las librerías para recolectar las trayectorias eran sin interacción humana. Para sortear esta limitación se debió modificar algunas funciones de las librerías utilizadas.

7. Bibliografía

(RM), G. A.-I. (2020). *Stable-Baselines3 Docs - Reliable Reinforcement Learning Implementations*. Obtenido de Stable Baselines 3 Documentation: https://stable-baselines3.readthedocs.io/en/master/

Abbeel, N. (2004). Apprenticeship Learning via Inverse Reinforcement Learning.

AI, H.-C. (2019). *Imitation Documentation*. Obtenido de Limitations on Horizon Length: https://imitation.readthedocs.io/en/latest/getting-started/variable-horizon.html

Barto, R. S. (2014). *Reinforcement Learning.* MIT Press.

Brockman, C. P. (2016). OpenAI Gym. *Arxiv*.

Fawzi, B. H. (2022). Discovering faster matrix multiplication algorithms with reinforcement learning. *Nature*, 47-53.

John Schulman, F. W. (2017). Proximal Policy Optimization Algorithms. *Arxiv*.

Jonathan Ho, S. E. (2016). Generative Adversarial Imitation Learning. *Arxiv*.

Mnih, K. S. (2013). Playing Atari with Deep Reinforcement Learning. *Arxiv*.

Nichol, P. H. (2018). Gotta Learn Fast: A New Benchmark for Generalization in RL. *Arxiv*.

Renotte, N. (9 de Febrero de 2022). *Youtube*. Obtenido de https://www.youtube.com/watch?v=rzbFhu6So5U&t=6231s

Videogames.ai. (Diciembre de 2021). *Youtube*. Obtenido de OpenAI game integration tool - part 1: https://www.youtube.com/watch?v=lPYWaUAq_dY&t=3079s

Yazgan, J. S. (2020). Chip Design with Deep Reinforcement Learning. *Arxiv*.

FIN DEL DOCUMENTO