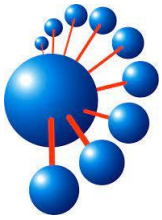
Universidad de Concepción

Facultad de Ingeniería

Departamento de Ingeniería Informática y Ciencias de la Computación



Evaluación de estrategias de recompensa en aprendizaje por refuerzo y su impacto en el desempeño y comportamiento de un agente

Francisca Hillerns Vega

Memoria presentada para la obtención del título de Ingeniero Civil Informático

Profesor Patrocinante: Julio Godoy Del Campo

Abril de 2025

Tabla de contenido

Índice de figuras

Índice de tablas

1. Introducción

Los videojuegos son una afición para millones de personas alrededor del mundo, proporcionando entretenimiento, competencia y desafíos intelectuales. Más allá de ser una fuente de diversión, los videojuegos han demostrado ser útiles en otros ámbitos como la educación, la rehabilitación y la investigación científica. Al representar una abstracción de problemas complejos del mundo real, los videojuegos permiten simular escenarios en los que se pueden desarrollar y probar soluciones a problemas de diversa índole.

En este contexto, los videojuegos han emergido como una plataforma valiosa para el desarrollo de inteligencia artificial (IA). El uso de videojuegos permite crear entornos controlados donde los agentes de IA pueden aprender a resolver desafíos simulados que luego pueden trasladarse a aplicaciones más amplias en el mundo real. Una técnica clave para este propósito es el aprendizaje por refuerzo RL *(*por sus siglas en inglés, *reinforcement learning*), que permite a los agentes aprender a optimizar su desempeño a través de interacciones con el entorno.

Esta memoria se basa en el *paper* de los autores Bakos y Davoudi llamado *"Mitigating Cowardice for Reinforcement Learning Agents in Combat Scenarios"* (2022) [9]. Al igual que en el *paper*, en esta memoria se utilizó el RL en videojuegos con escenarios de combate, donde el agente se enfrenta a un contrincante y debe pelear contra él hasta derrotarlo para ganar. Una problemática interesante que se ve específicamente en la aplicación de RL en este escenario es la llamada "cobardía", donde los agentes evitan tomar decisiones riesgosas o fallan en adoptar estrategias agresivas cuando es necesario. Las estrategias usuales funcionan para que el agente aprenda a ganar en este tipo de entornos de combate, pero la calidad de estas victorias no siempre es la mejor.

Si se recompensa al agente de la misma manera por cualquier victoria, o se le penaliza igual por cualquier derrota, sin considerar la calidad de estas, el agente no aprenderá a distinguir entre victorias o derrotas mejores. Esto significa que no necesariamente buscará alcanzar la mejor victoria posible y probablemente evitará tomar acciones riesgosas que podrían acercarlo al éxito, ya que las derrotas son penalizadas de forma uniforme, sin importar cuán cerca estuvo de lograr la victoria antes de fallar.

Este fenómeno puede limitar significativamente el desempeño del agente, llevándolo a comportamientos subóptimos, como perder combates que, de otra manera, podrían haberse ganado, o ganar combates en un periodo excesivamente largo. Este trabajo explora una propuesta de entrega de recompensas cuyo propósito es incentivar al agente a ser más valiente en su comportamiento, fomentando acciones que lo conduzcan a victorias más decisivas. Para lograr esto, se le proporciona al agente una recompensa al final del episodio que depende de su desempeño durante el mismo. En caso de victoria, se otorga una recompensa mayor cuanto mejor sea la victoria, y en caso de derrota, se aplica

una penalización menor cuanto más cerca de ganar se encontrase el agente antes de perder.

Se ha elegido aplicar esta estrategia en tres videojuegos que poseen escenarios de combate: Punch Out!!, The Legend of Zelda y Mega Man X, donde el agente se enfrenta a un contrincante y debe aprender a derrotarlo. A través del uso del algoritmo *proximal policy optimization* (PPO), se entrena al agente para tomar decisiones en tiempo real, equilibrando el dilema de exploración-explotación, donde debe decidir entre probar nuevas acciones o aferrarse a estrategias que ya han demostrado ser exitosas.

El propósito de esta memoria es estudiar cómo y en qué ámbitos varía el desempeño y comportamiento en un ambiente multiagente competitivo (de 2 agentes), en función de la estrategia de recompensa utilizada, desde recompensas estáticas hasta recompensas que dependen del estado final del juego.

2. Objetivos

2.1. Objetivo general

Implementar, evaluar y analizar el impacto de diferentes estrategias de entrega de recompensas en el desempeño y comportamiento de un agente en aprendizaje por refuerzo aplicado al ámbito de los videojuegos.

2.2. Objetivos específicos

1. Evaluar el desempeño del agente en términos de eficacia y eficiencia al utilizar diferentes estrategias de entrega de recompensas en los videojuegos seleccionados.

2. Analizar cómo la utilización de distintos tipos de recompensas influye en la velocidad de aprendizaje y la calidad de las decisiones tomadas por el agente en entornos de juegos con contrincantes.

3. Investigar la estabilidad y consistencia del desempeño del agente a lo largo del tiempo al aplicar distintas estrategias de recompensa en entornos de juego dinámicos.

4. Comprender a profundidad el método de PPO, su teoría y la implementación en PyTorch de Stable Baselines3.

5. Determinar qué tipos de recompensas son más efectivos en diferentes ambientes de juego, en base a los experimentos realizados.

3. Marco Teórico

3.1. Aprendizaje por refuerzo

El aprendizaje por refuerzo (RL, por sus siglas en inglés) es un enfoque dentro del **machine learning** que permite a un agente aprender un comportamiento óptimo en un entorno con el objetivo de maximizar una recompensa. Este comportamiento se desarrolla a través de interacciones continuas con el entorno y la observación de sus respuestas, de manera similar a cómo los seres vivos exploran y se adaptan al mundo que los rodea.

En este contexto, el aprendizaje por refuerzo se modela mediante un **Proceso de Decisión de Markov** (MDP, por sus siglas en inglés), el cual se define por los siguientes elementos clave [15]:

- **Espacio de estados (S)**: representa todas las posibles situaciones en las que puede encontrarse el agente.
- **Espacio de acciones (A)**: conjunto de decisiones que el agente puede tomar en cada estado.
- **Función de transición (P(s′ | s,a))**: describe la probabilidad de moverse de un estado s a un nuevo estado s′ tras ejecutar una acción a.
- **Función de recompensa (R(s,a))**: asigna un valor a cada acción tomada en un estado determinado, proporcionando retroalimentación sobre su utilidad.
- γ: factor de descuento (un valor entre 0 y 1) que refleja la importancia de las recompensas futuras.

Bajo este marco, el **entrenamiento** se refiere al proceso mediante el cual el agente interactúa con el entorno para aprender a tomar decisiones óptimas. Durante este proceso, el agente observa el estado actual, selecciona una acción de acuerdo con una política de decisión, recibe una recompensa o penalización en función de la acción tomada y transita a un nuevo estado. A través de estas interacciones y utilizando mecanismos de prueba y error, el agente ajusta su comportamiento con el objetivo de maximizar la recompensa acumulada.

Este proceso de aprendizaje está caracterizado por los siguientes elementos clave [1]:

- **Política (π)**: estrategia que utiliza el agente para determinar la siguiente acción en función del estado actual. En este trabajo, se emplea una **política estocástica** $\pi_\theta(a|s)$, que representa la probabilidad de seleccionar la acción *a* dado el estado *s*, donde Θ son los parámetros del modelo que se ajustan durante el entrenamiento.
- **Función de recompensa (R(s,a))**: Proporciona una señal de retroalimentación escalar basada en el estado y la acción, indicando el valor de una decisión tomada.
- **Función de valor (V(s)):** Estima la recompensa acumulada esperada desde un estado dado, ayudando al agente a evaluar la calidad de diferentes estados.

- **Modelo de transición (P(s′ | s,a))**: Una representación del entorno que permite al agente planificar prediciendo estados y recompensas futuros.

Dado un estado del entorno (una representación de su condición en un momento específico), la política guía al agente para decidir qué acción realizar (es decir, un paso que el agente lleva a cabo para navegar por el entorno) con el objetivo de maximizar la recompensa acumulada final, que corresponde a la suma de todas las recompensas obtenidas (Figura 3.1) [2].

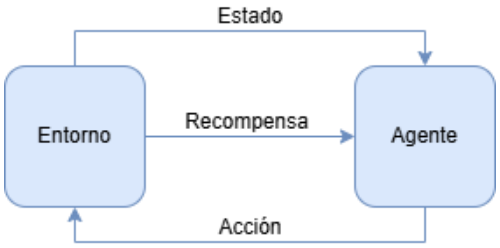


Figura 3.1. Diagrama de aprendizaje por refuerzo

Para mejorar esta política, el agente no solo debe seleccionar acciones de alto rendimiento ya conocidas, sino que también debe explorar más el entorno para descubrir nuevas estrategias y oportunidades de recompensa. Esto se conoce como el **dilema entre exploración y explotación**.

Por ejemplo, en un videojuego, un agente que prioriza la **explotación** podría repetir una estrategia conocida que le garantiza una victoria, aunque no sea óptima. En contraste, la **exploración** podría llevarlo a probar movimientos arriesgados que, si resultan efectivos, le otorgarían mayores recompensas a largo plazo. Este aprendizaje autónomo es crucial en ausencia de supervisión directa, requiriendo que el agente descubra por sí mismo las acciones que maximizan la recompensa a través de un proceso de **ensayo y error**.

Las **recompensas** son un componente esencial del aprendizaje por refuerzo, ya que guían al agente indicándole qué acciones son valiosas y motivándolo a aprender de ellas. En este paradigma, el agente es recompensado por tomar decisiones que conducen a estados exitosos.

Las recompensas pueden ser:
- **Inmediatas**, como recibir un punto por cada paso dado en la dirección correcta.
- **Diferidas**, como recibir una recompensa solo al final del episodio si se alcanza un objetivo.

Existen múltiples formas de definir las recompensas, y la elección de la función de recompensa puede tener un impacto significativo en el proceso de aprendizaje. Encontrar el equilibrio adecuado entre diferentes tipos de recompensas es una parte fundamental en el diseño de un algoritmo de aprendizaje por refuerzo exitoso.

3.2. Proximal Policy Optimization

Cuando los problemas de aprendizaje por refuerzo son más complejos, como es el caso de los videojuegos elegidos, se recurre a redes neuronales profundas, dando lugar al campo conocido como **aprendizaje por refuerzo profundo** (*Deep Reinforcement Learning*). Estas redes son capaces de manejar grandes espacios de estados y acciones, lo que permite a los agentes aprender patrones más sofisticados.

Uno de los algoritmos más utilizados en este contexto es **Proximal Policy Optimization** (PPO), debido a su simplicidad y efectividad en entornos complejos. PPO se clasifica como un **método de gradiente de política**, optimizando directamente la política del agente. En este enfoque, una red neuronal recibe el estado actual del entorno y genera una distribución de probabilidad sobre las posibles acciones. Su objetivo es **maximizar la recompensa acumulada** ajustando la política de forma iterativa, manteniéndola cerca de la política anterior para evitar cambios abruptos y mejorar la estabilidad del entrenamiento. Este control sobre las actualizaciones de la política es lo que distingue a PPO de otros métodos [3].

Función Objetivo en PPO

En **aprendizaje por refuerzo profundo**, la **función objetivo** es un elemento clave que guía el ajuste de los parámetros de la política y de la función de valor. Su propósito es optimizar la toma de decisiones del agente de manera que maximice la recompensa esperada a largo plazo. En el caso de PPO, esta función se construye a partir de varios términos que permiten estabilizar el aprendizaje y mejorar la eficiencia del entrenamiento.

En particular, PPO no optimiza directamente la recompensa acumulada, sino que emplea una **función objetivo sustituta** (*surrogate objective function*), diseñada para mejorar la estabilidad del entrenamiento. A diferencia de otros métodos de optimización de políticas, PPO impone restricciones sobre cuánto puede cambiar la política en cada iteración, lo que previene actualizaciones drásticas que podrían degradar el desempeño del agente.

La función objetivo de PPO se compone de los siguientes elementos fundamentales:

1. **Función objetivo sustituta recortada (L^{CLIP})**: limita los cambios en la política para garantizar estabilidad.
2. **Estimación de ventaja ($\hat{A}_t$)**: evalúa qué tan buena fue una acción en comparación con la expectativa general en ese estado.
3. **Función de valor ($V(s)$)**: ayuda al agente a predecir la recompensa futura esperada.
4. **Función de entropía ($S(\pi_\theta)$)**: fomenta la exploración evitando que la política se vuelva demasiado determinista.

A continuación, se describen cada uno de estos términos en detalle.

Función Objetivo Sustituta Recortada (L^{CLIP})

La función objetivo de PPO se basa en la razón de probabilidad entre la política nueva y la política anterior. Se define como:

$$L^{CLIP}(\theta) = \hat{\mathrm{E}}_t[\min(r_t(\theta)\hat{\mathrm{A}}_t, clip(r_t(\theta), 1-\varepsilon, 1+\varepsilon)\hat{\mathrm{A}}_t)] \quad (1)$$

Donde:

- $r_t(\theta)$ es la **razón de probabilidad** entre la política nueva y la política anterior:

$$r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{old}}(a_t|s_t)} \quad (2)$$

  Donde:
  - $\pi_\theta(a_t|s_t)$ es la probabilidad de tomar la acción a_t dado el estado s_t bajo la política nueva.
  - $\pi_{\theta_{old}}(a_t|s_t)$ es la probabilidad de tomar la acción a_t dado el estado s_t bajo la política anterior.

- $\hat{\mathrm{A}}_t$ es la **estimación de la ventaja** en el tiempo t, que mide qué tan buena fue la acción a_t en comparación con otras acciones disponibles en el mismo estado.

- ε es un hiperparámetro que controla cuánto puede cambiar la política en una iteración, limitando el cambio en la razón de probabilidad.

- La función **clip** asegura que $r_t(\theta)$ no se desvíe demasiado de 1. Si la razón cae fuera del rango $[1-\varepsilon, 1+\varepsilon]$, se penaliza el cambio, evitando actualizaciones inestables en la política.

Estimación de Ventaja ($\hat{\mathrm{A}}_t$)

La ventaja se utiliza para estimar qué tan buena fue una acción a_t en comparación con la expectativa general en ese estado. Se calcula como:

$$\hat{\mathrm{A}}_t = \sum_{i=0}^{T-t} (\gamma\lambda)^i \, \delta_{t+1} \quad (3)$$

Donde:

- δ_t es la **diferencia temporal** (*TD error*) en el tiempo t, definida como:

$$\delta_t = R_t + \gamma V(s_{t+1}) - V(s_t) \quad (4)$$

Donde:

- R_t representa la recompensa obtenida en el tiempo *t* tras ejecutar la acción a_t en el estado s_t.
- γ es el **factor de descuento**.
- $V(s_t)$ es la **función de valor**, que estima la recompensa futura esperada desde el estado s_t.

- λ es un hiperparámetro que controla la suavidad de la estimación de ventaja. Si $\lambda = 1$, se obtienen estimaciones más exactas (pero con mayor varianza), mientras que valores más bajos de λ hacen que la estimación de la ventaja sea más estable, pero con mayor sesgo.

Función de Valor ($V(s_t)$)

En PPO, la función de valor se ajusta durante el entrenamiento para hacer más precisas las predicciones de la recompensa futura. Se optimiza mediante la siguiente ecuación de regresión:

$$L^{VF}(\theta) = \hat{\mathrm{E}}_t[(V_\theta(s_t) - \hat{R}_t)^2] \quad (5)$$

Donde:

- $V_\theta(s_t)$ es la salida de la red neuronal, que representa la estimación de la función de valor.

- $\hat{R}_t$ es la recompensa acumulada descontada, utilizada como objetivo para mejorar la estimación de $V_\theta(s_t)$.

Función de Entropía ($S(\pi_\theta)$)

La entropía incentiva la exploración, evitando que la política se vuelva demasiado determinista. Se define como:

$$S(\pi_\theta) = -\hat{\mathrm{E}}_t[\pi_\theta(a_t|s_t)\log\pi_\theta(a_t|s_t)] \quad (6)$$

Agregar un término de entropía en la función objetivo fomenta la exploración y reduce el riesgo de que el agente se estanque en un comportamiento subóptimo.

Función de Pérdida Final

Finalmente, la función de pérdida de PPO combina los tres términos anteriores:

$$L^{PPO}(\theta) = L^{CLIP}(\theta) - c_1 L^{VF}(\theta) + c_2 S(\pi_\theta) \quad (7)$$

Donde:

- c_1 es el coeficiente que controla la importancia de la función de valor.
- c_2 es el coeficiente que controla la importancia de la entropía.

Debido a su estabilidad, eficiencia y capacidad para resolver problemas complejos, así como su comprobada eficacia en el contexto de esta investigación, PPO fue seleccionado como el algoritmo para entrenar a los agentes en los entornos de batalla de los tres videojuegos escogidos [5].

3.3. Mitigating Cowardice for Reinforcement Learning Agents in Combat Scenarios

Esta memoria está basada en la propuesta hecha por Bakos y Davoudi en su paper *"Mitigating Cowardice for Reinforcement Learning Agents in Combat Scenarios"* (2022) [9]. Cuando se entrena un agente en escenarios de combate, el propósito del agente es derrotar a un enemigo disminuyendo su vida a cero, lo común es otorgar una recompensa o castigo al final del episodio, dependiendo del resultado del combate: ganar o perder.

Según Bakos y Davoudi [9], este tipo de recompensa estática puede fomentar un comportamiento de "cobardía" en los agentes, que en este caso se refiere al miedo a ser castigados por fallar, lo que genera estrategias subóptimas. Por ejemplo, el agente puede adoptar un comportamiento evasivo, evitando interactuar con el enemigo y tomando decisiones defensivas excesivas, lo que resulta en un juego lento y poco eficiente.

Este comportamiento es especialmente problemático en entornos de combate donde el objetivo es derrotar al oponente minimizando el número de acciones necesarias para lograrlo. En lugar de involucrarse de manera efectiva en la batalla, el agente tiende a huir o evitar al oponente, lo que no solo reduce su rendimiento, sino que también limita su capacidad para aprender estrategias más complejas.

Para abordar este problema, Bakos y Davoudi proponen una **técnica de recompensa decaída**, que ajusta la recompensa o penalización al final del episodio en función del desempeño del agente. Este enfoque permite al agente diferenciar entre una buena y una mala victoria, o entre una derrota buena y una mala. No es lo mismo que el agente gane con toda su vida intacta a que lo haga al borde de la derrota. Del mismo modo, hay derrotas que son mejores que otras: no es igual perder cuando se estuvo cerca de vencer al enemigo, que hacerlo sin haberle causado ningún daño.

Los autores proponen que, en lugar de recibir una recompensa fija por ganar o perder, el valor de la recompensa o penalización se ajuste en función de parámetros específicos, como la cantidad de vida restante del agente o del oponente al final del combate.

Por ejemplo, si el agente gana un combate con toda su vida intacta, recibe una recompensa mayor que si lo hace con poca vida. De igual forma, si pierde, pero estuvo cerca de derrotar a su oponente, la penalización será menor que si hubiera perdido sin causar daño. Esta técnica no solo mejora el rendimiento del agente, sino que también mitiga la cobardía, alentando al agente a participar activamente en la batalla en lugar de adoptar estrategias evasivas.

En caso de derrota, la técnica de recompensa decaída busca disminuir la cobardía. Esta se produce por el miedo a la penalización que el agente recibe al final del episodio. Dado que el objetivo en los combates es disminuir la vida del oponente a cero, la vida restante del enemigo se utiliza como parámetro para reducir el castigo, permitiendo al agente distinguir entre derrotas buenas y malas, alentando al agente a luchar a pesar de sus derrotas.

En caso de victoria, se busca la "masterización" (mejor resultado posible) del parámetro utilizado para la disminución de la recompensa. Por ejemplo, si el parámetro es el tiempo restante, el agente recibirá una mayor recompensa por ganar más rápidamente, lo que lo inducirá a comportarse de manera más agresiva. Si el parámetro es la vida restante del agente, terminar el combate con más vida implicará una mayor recompensa, fomentando un estilo más defensivo.

Este *paper* busca comparar los resultados de dos formas de otorgar recompensas al finalizar un episodio: una recompensa estática y una recompensa decaída.

- **Recompensa Estática:** La recompensa estática es un tipo de recompensa que se otorga al final de un episodio, dependiendo simplemente del resultado del combate: ganar o perder. Este tipo de recompensa no tiene en cuenta el desempeño específico del agente durante el combate, como la cantidad de daño infligido al oponente, el tiempo que tardó en derrotarlo o cuánta vida le quedó al final del episodio.

- **Recompensa Decaída:** La recompensa decaída ajusta la recompensa o penalización al final del episodio según el desempeño específico del agente, permitiéndole diferenciar entre una victoria buena y una mala, o una derrota buena y una mala. Este enfoque considera parámetros específicos, como la cantidad de vida restante del agente o del oponente, o el tiempo transcurrido durante el combate. Por ejemplo, si el agente derrota a su oponente sin sufrir daño, la recompensa será mayor que si lo hace con poca vida restante. De manera similar, si el agente pierde el combate, pero estuvo cerca de derrotar al oponente, la penalización será menor que si no causó daño alguno.

La fórmula para calcular el valor de la recompensa decaída utiliza como base a la recompensa estática y es la siguiente:

$$V_{net} = \pm V_{ter}^{\frac{P_{obs}}{P_{max}}} \tag{8}$$

Donde:

- P_{obs} es el valor del parámetro al finalizar el episodio
- P_{max} es el máximo valor que el parámetro puede tomar
- V_{ter} es la recompensa estática que se le daría al agente al finalizar el episodio
- V_{net} es la recompensa decaída que se le da al agente al finalizar el episodio

Bakos y Davoudi probaron el enfoque de la recompensa decaída en tres videojuegos retro de combate; The Legend of Zelda, Mega Man X y M.U.G.E.N. En estos entornos, el objetivo del agente es derrotar a un oponente reduciendo su barra de vida a cero. Cada juego presenta diferentes niveles de complejidad, lo que permite evaluar la efectividad de la técnica en distintos escenarios.

El algoritmo de aprendizaje por refuerzo utilizado por Bakos y Davoudi fue PPO, esto por ser adecuado para estos escenarios, ya que maneja espacios de acción discretos y ofrece gran estabilidad durante el entrenamiento.

El enfoque de recompensa decaída presentado en el trabajo de Bakos y Davoudi ha demostrado ser una solución eficaz para mitigar la cobardía en agentes de RL entrenados en escenarios de combate. Al ajustar la recompensa o penalización en función del desempeño del agente, esta técnica permite un aprendizaje más eficiente, alentando a los agentes a desarrollar estrategias más efectivas y agresivas.

3.4. Videojuegos escogidos

3.4.1. Punch Out!! para NES

El primero de los videojuegos escogidos para evaluar las estrategias de recompensa es Punch-Out!! (Figura 3.2), un juego de boxeo desarrollado por Nintendo, lanzado originalmente en 1987 para la consola Nintendo Entertainment System (NES). Se juega con el protagonista Little Mac, un joven boxeador que sueña con convertirse en el campeón mundial de boxeo [6]. Este juego fue elegido debido a su enfoque exclusivo en batallas entre dos personajes y a la simplicidad de su jugabilidad.

Para avanzar el jugador se enfrenta con diferentes oponentes, cada uno más difícil que el anterior y con un patrón de ataque único. El jugador debe esquivar, bloquear y contragolpear en el momento oportuno. El jugador no puede avanzar al siguiente oponente si no ha derrotado al actual. Este es el entorno de batalla más sencillo de los tres videojuegos escogidos, el entorno es específicamente una pelea contra uno de los oponentes.

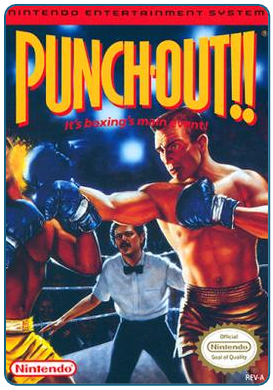


Figura 3.2. Portada del juego Punch Out!! para NES.

La pelea se divide en tres rondas de tres minutos de juego cada una. Si uno de los boxeadores es derribado tres veces en una ronda, ocurre un *knockout* técnico automático (TKO). Si la ronda 3 termina antes de que uno de los dos sea eliminado, es posible ganar por decisión. Cada boxeador posee una barra de vida con un valor de 96 unidades, la barra disminuye cuando el personaje es golpeado, si la barra es drenada, el personaje queda *knockout* (KO), pero esto no necesariamente significa que ha sido derrotado, ya que el personaje puede levantarse con una fracción de su vida total después de un KO si es la primera o segunda vez que pasa dentro de una ronda.

Otro factor importante es el contador de corazones, el cual indica cuántos golpes puede dar Little Mac, este valor disminuye cuando falla un ataque, bloquea un ataque de su oponente, cuando lo golpean, y cuando el oponente bloquea uno de sus ataques. Si los corazones llegan a cero, Little Mac no puede atacar y deberá esquivar ataques enemigos para aumentar los corazones. La batalla termina si uno de los jugadores gana por TKO, si uno de los boxeadores no se levanta antes de 10 segundos en caso de KO, o si los tres rounds terminan por tiempo.

Para esta memoria se utilizaron los escenarios contra los dos primeros oponentes del juego: **Glass Joe** y **Von Kaiser**. Estos contrincantes fueron seleccionados por ser los dos primeros del juego. Cada uno posee un conjunto de ataques con distintos patrones y niveles de daño. Glass Joe realiza ataques como **ganchos** (Figura 3.3a) **y jabs**, que pueden ser esquivados, bloqueados o contraatacados, además de una burla que permite derribarlo con el tiempo adecuado. Por otro lado, Von Kaiser emplea **jabs y uppercuts** (Figura 3.3b), este último siendo un golpe más poderoso que no puede bloquearse. La descripción detallada de estos ataques se encuentra en el Apéndice 102.

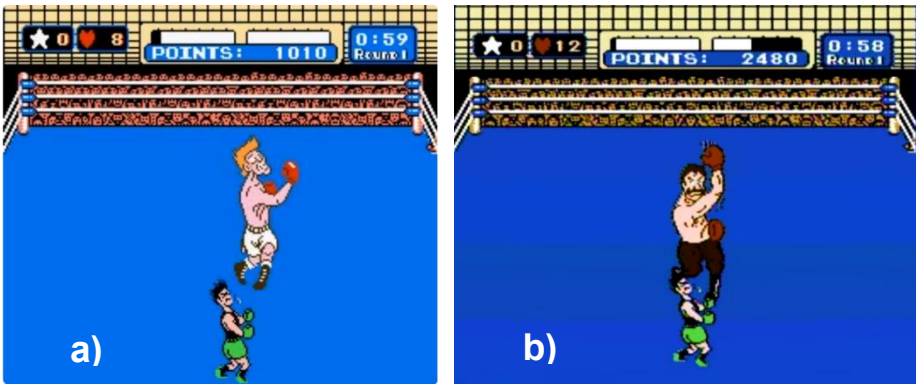


Figura 3.3. a) Gancho de derecha de Glass Joe y b) Uppercut de derecha de Von Kaiser.

3.4.2. The Legend of Zelda para NES

El segundo de los juegos es The Legend of Zelda (Figura 3.4), un juego de acción y aventura desarrollado por Nintendo para NES y lanzado en 1986 en Japón. Es el primer título en la icónica serie de The Legend of Zelda. El juego sigue las aventuras del protagonista Link, quien debe rescatar a la Princesa Zelda y derrotar a Ganon, quien ha robado la Trifuerza de la Sabiduría. Durante el transcurso del juego, Link debe pasar por varias mazmorras que deben ser completados para poder avanzar en la historia. Cada mazmorra tiene su propio conjunto de enemigos, acertijos y oponentes.

En el caso de este juego, el entorno de batalla es una pelea contra el oponente final de una mazmorra [7]. Este juego fue elegido debido a su utilización en [9], ya que se buscó comprobar la eficacia de la propuesta presentada en dicho *paper* en el escenario original y, además, explorar otro escenario dentro del mismo juego.

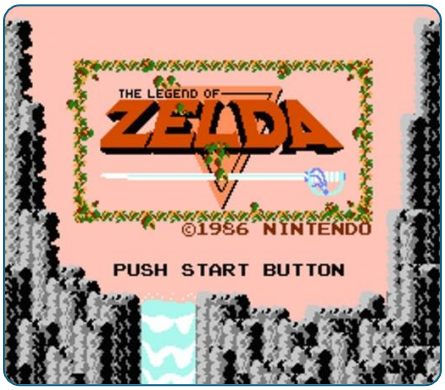


Figura 3.4. Pantalla de inicio del juego The Legend of Zelda.

Durante la pelea, Link debe moverse a través de la habitación y utilizar su espada para golpear al oponente hasta derrotarlo, mientras no usa la espada, automáticamente utiliza su escudo para proteger la parte frontal de su cuerpo, este escudo funciona solo contra ciertos ataques. En el juego, la vida máxima de Link es representada por contenedores de corazón, la cantidad de corazones rojos indica cuánta vida le queda a Link, si tiene su vida

completa, su espada adquiere temporalmente la capacidad de lanzar ataques a distancia. La batalla termina si Link pierde todos los corazones, si Link derrota al oponente final de la mazmorra o si Link sale de la habitación del oponente.

Para esta memoria se utilizaron los escenarios contra dos oponentes del juego: **Aquamentus** y **Manhandla**. Aquamentus fue seleccionado porque su batalla es una de las utilizadas en [9]. Inicialmente, se había considerado incluir a Gleeok como segundo oponente, debido a su mayor nivel de dificultad y a su comportamiento similar a Aquamentus, sin embargo, no fue posible localizar las direcciones de memoria necesarias para su entrenamiento. Por este motivo, se optó por Manhandla, quien presenta un mayor nivel de dificultad y, aunque su comportamiento es significativamente diferente, comparte el hecho de ser derrotado utilizando la espada.

Cada uno de estos oponentes posee un conjunto de ataques que Link debe esquivar o bloquear. Aquamentus ataca con **bolas de fuego** (Figura 3.5a), las cuales pueden bloquearse con el escudo, mientras que Manhandla combina **bolas de fuego** con **embestidas rápidas** (Figura3.5b), aumentando el nivel de dificultad. La descripción detallada de estos ataques se encuentra en el Apéndice 103.

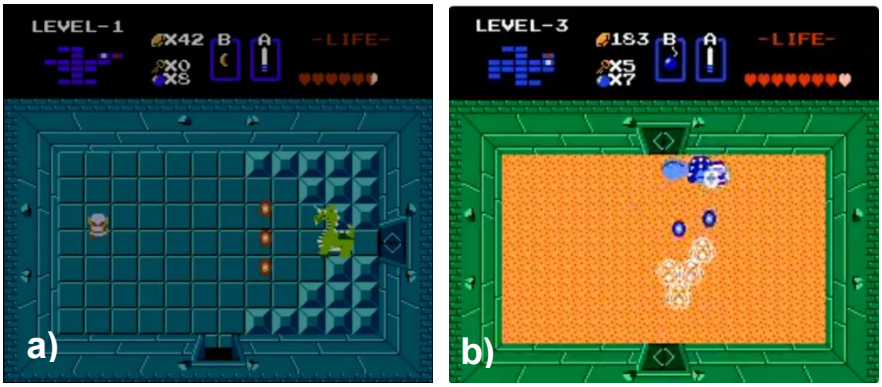


Figura 3.5. a) Bolas de Fuego de Aquamentus y b) Embestida de Manhandla.

3.4.3. Mega Man X para SNES

El tercer y último juego es Mega Man X (Figura 3.6), un videojuego lanzado en 1993 para Super Nintendo Entertainment System (SNES). Este es un título destacado en la serie Mega Man. El juego sigue a Mega Man X (conocido como X), un robot de combate avanzado que se despierta en un futuro distante, donde el mundo está en peligro por un grupo de robots conocidos como los Mavericks, liderados por el malvado Sigma, el objetivo del protagonista es derrotar a cada uno.

Mega Man X presenta varios niveles temáticos, cada uno con su propio oponente al final de ellos. Estas batallas contra oponentes son los entornos de batalla elegidos para este juego, este entorno es el más complejo de los tres juegos escogidos [8]. Este juego, al igual que el anterior, fue elegido debido a su utilización en [9], buscando comprobar la eficacia

de la propuesta presentada en dicho *paper* en el escenario original y, además, explorar otro escenario dentro del mismo juego.

Figura 3.6. Portada del juego Mega Man X para SNES.

Durante la batalla, X deberá usar el arma que tenga equipada, la cual disparará distintos proyectiles dependiendo de su tipo, para golpear al oponente, mientras se mueve y esquiva los ataques enemigos. Tanto X como el oponente poseen una barra de vida de 32 unidades. La pelea termina si X derrota al oponente reduciendo su vida a cero o si la vida de X llega a cero.

Para esta memoria se utilizaron los escenarios contra dos oponentes del juego: **Chill Penguin** y **Spark Mandrill**. Chill Penguin fue seleccionado porque su batalla es una de las utilizadas en [9]. Como segundo oponente, se eligió a Spark Mandrill debido a las similitudes entre ambos escenarios: en ambos casos, la sala es estática y los oponentes utilizan proyectiles o su cuerpo para infligir daño.

Cada uno de estos jefes posee un conjunto de ataques que el agente debe esquivar o contrarrestar. Chill Penguin emplea **bolas de hielo** (Figura 3.7a)**, deslizamientos y tormentas de nieve** que afectan el movimiento del jugador, mientras que Spark Mandrill usa **descargas eléctricas, embestidas** (3.7b) **y ataques desde el techo** para acorralar al agente. La descripción detallada de estos ataques se encuentra en el Apéndice 104.

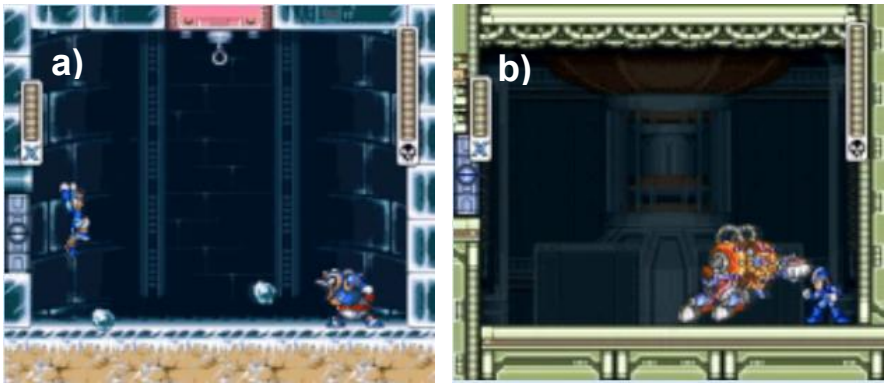


Figura 3.7. a) Shotgun Ice de Chill Penguin y b) Dash Punch de Spark Mandrill.

De esta forma, se han seleccionado tres videojuegos que representan **ambientes de combate diversos y desafiantes** para el entrenamiento de agentes en aprendizaje por refuerzo. Cada uno de ellos presenta diferentes tipos de interacciones, estrategias y dinámicas de combate, lo que permite evaluar el desempeño de agentes en escenarios con mecánicas variadas.

A continuación, se detallará el proceso de desarrollo e implementación utilizado en esta memoria.

4. Desarrollo

En este capítulo se describe el proceso de desarrollo e implementación llevado a cabo en esta memoria. Se presentan las herramientas y configuraciones utilizadas, así como los métodos empleados para recopilar y procesar los datos obtenidos durante los entrenamientos. Además, se detalla la metodología experimental aplicada en cada uno de los videojuegos seleccionados, especificando las condiciones del entorno, los espacios de observación y acción, y los esquemas de recompensas definidos para evaluar el desempeño del agente.

4.1. Ambiente de desarrollo

Para la realización de esta memoria se trabajó en un notebook con el sistema operativo Ubuntu 22.04, una tarjeta de video NVIDIA GeForce RTX 4050 para laptop, un procesador 13th Gen Intel(R) Core(TM) i5-13420H 2.10 GHz y 16 GB de RAM.

Se utilizó la biblioteca para Python 3 llamada Stable-Baselines3 [4] para entrenar los agentes de aprendizaje por refuerzo, ya que ofrece una implementación eficiente y flexible de algoritmos, incluyendo PPO, el cual se utilizó en esta memoria.

Para poder usar los juegos elegidos, se utilizó Stable-Retro, una integración de la biblioteca Stable-Baselines3 y de Gym-Retro, una herramienta que permite entrenar agentes de aprendizaje por refuerzo en entornos de videojuegos retro (títulos desarrollados en generaciones anteriores al cambio de milenio, como juegos arcade y de consolas clásicas como NES, SNES y Sega Genesis). Actualmente, Stable-Retro ofrece más de 1000 integraciones de juegos, cada una con archivos que enumeran ubicaciones de memoria para algunas variables del juego, funciones de recompensa basadas en esas variables, condiciones de finalización de episodios y *savestates* (archivos que almacenan un estado exacto del juego en un momento determinado, permitiendo reanudar la ejecución desde ese punto) al comienzo de algunos niveles [10].

Stable-Retro incluye la herramienta *gym-retro-integration*, que permite integrar otros juegos retro, encontrar ubicaciones de memoria para variables de interés, establecer funciones de recompensa y condiciones de finalización de episodios, y crear savestates del juego. Se utilizó esta herramienta para integrar los juegos The Legend of Zelda y Mega Man X, así como para crear los savestates y definir variables y condiciones de término para estos juegos, además del ya integrado Punch Out!! [11].

Stable-Retro permite utilizar GPU para el entrenamiento y ejecutar un **entorno vectorizado**, lo que posibilita la ejecución de múltiples instancias del mismo entorno en paralelo, mejorando la eficiencia del aprendizaje. Sin embargo, incluso cuando el

entrenamiento se realiza en la GPU, la CPU sigue siendo utilizada para manejar la simulación de los entornos, el preprocesamiento de los datos y la comunicación con la GPU.

Durante este trabajo se usaron **12 instancias paralelas**, una cantidad determinada en función de dos criterios principales. En primer lugar, este valor coincide con el número de hilos de la CPU utilizada, lo que permite aprovechar al máximo los recursos sin afectar la estabilidad del sistema. En segundo lugar, en [9], se usaron 12 entornos paralelos para el entrenamiento en The Legend of Zelda (32 para Mega Man X y 4 para M.U.G.E.N), por lo que se optó por replicar esa cantidad para mantener consistencia metodológica en ese juego. Se probó con un mayor número de instancias, pero el sistema no pudo manejarlas sin afectar la eficiencia del entrenamiento.

Para todos los juegos, se utilizó una **arquitectura de red neuronal convolucional (CNN)** diseñada para manejar entradas de tipo imagen (espacios de observación visuales). Esta política está optimizada para procesar eficientemente imágenes en entornos como videojuegos, donde las observaciones son representaciones visuales del estado del entorno.

Se utilizó la arquitectura predeterminada en PPO (Figura 4.1) para todos los juegos, que consta de **tres capas convolucionales** con funciones de activación ReLU (*Rectified Linear Unit*), seguidas de un **aplanado** (*flatten*) que convierte las salidas de la última capa convolucional en un vector unidimensional. Este vector luego es procesado por una **capa completamente conectada** de **512 neuronas** con función de activación ReLU [12].

La arquitectura de la CNN emplea diferentes configuraciones de filtros, tamaños de kernel y stride en cada capa para permitir una extracción progresiva de características de las imágenes de entrada:

- La **primera capa convolucional** utiliza 32 filtros, un kernel de 8x8 y un stride de 4, lo que permite capturar patrones generales en la imagen y reducir rápidamente su tamaño.
- La **segunda capa** cuenta con 64 filtros, un kernel de 4x4 y un stride de 2, logrando un balance entre la detección de detalles y la eficiencia computacional.
- La **tercera capa** también utiliza 64 filtros, pero con un kernel de 3x3 y un stride de 1, para enfocarse en detectar características más específicas en las últimas etapas del procesamiento visual.

Esta combinación de parámetros permite que la red detecte desde patrones generales hasta detalles finos, optimizando su desempeño al interpretar las observaciones visuales del entorno.

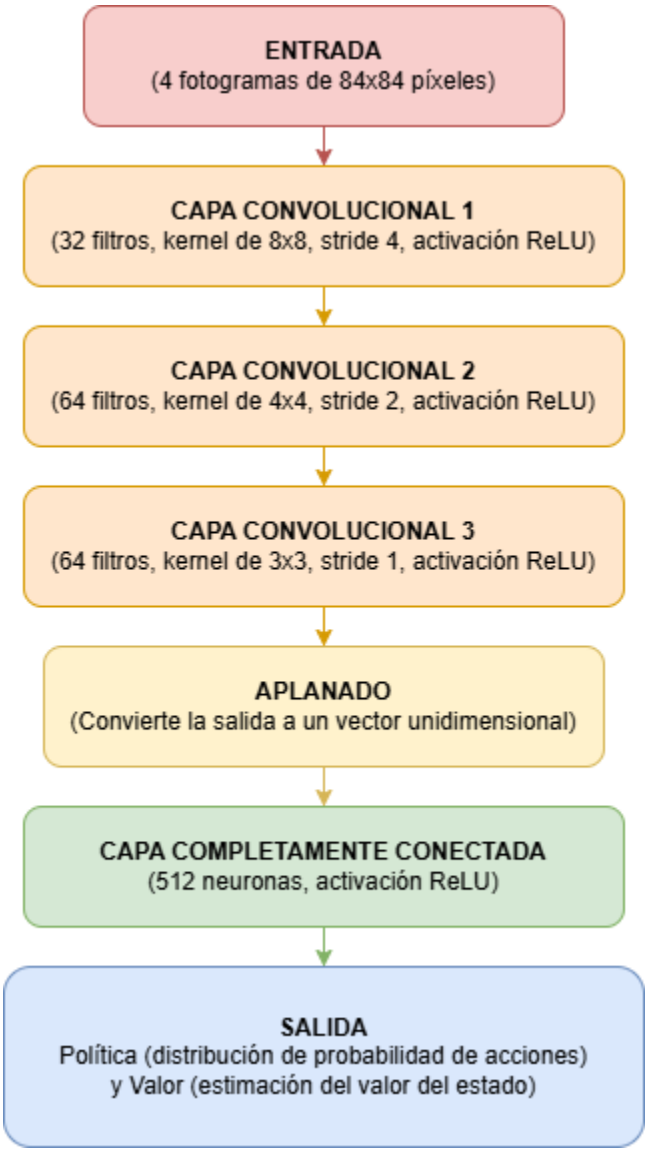


Figura 4.1. Arquitectura de la Red Neuronal Convolucional en PPO: arquitectura predeterminada de Stable-Baselines3 para CnnPolicy.

En el caso de los hiperparámetros utilizados en el entrenamiento, se utilizaron en su mayoría los valores default de esta implementación de PPO, con la excepción del *learning rate, number of steps y batch size.* A continuación, se muestran las fórmulas asociadas a estos hiperparámetros.

El entrenamiento de la red neuronal convolucional (CNN) se realiza mediante el algoritmo de **backpropagation** [16], un método fundamental para el ajuste de los **pesos de la red** representados por el **vector de parámetros θ** (*theta*). Estos pesos definen las conexiones internas de la red neuronal y son los parámetros del modelo que se ajustan durante el proceso de aprendizaje.

El objetivo del entrenamiento es encontrar el conjunto de pesos θ que minimicen la **función de pérdida**, lo que se logra mediante el **descenso de gradiente**. Para ello, se calcula el gradiente del error con respecto a cada peso, permitiendo actualizarlos de acuerdo con el *learning rate*, que controla la magnitud de los cambios en cada iteración.

Durante este proceso, se ajustan diferentes **hiperparámetros**, incluyendo:

- **Learning rate:** Define el tamaño de los pasos al actualizar los pesos θ.
- **Batch size:** Cantidad de muestras procesadas antes de cada actualización de θ.
- **Número de epochs:** Cuántas veces se pasa por todo el conjunto de datos durante el entrenamiento.
- **Gamma (γ):** Factor de descuento para recompensas futuras en el contexto del aprendizaje por refuerzo.
- **Clip range:** Limita las actualizaciones de θ evitando cambios bruscos en la política del agente.
- **Coeficiente de entropía (ent_coef):** Favorece la exploración agregando aleatoriedad a la política.

El ajuste adecuado de estos hiperparámetros permite que el modelo alcance un equilibrio entre estabilidad y capacidad de aprendizaje, maximizando la recompensa acumulada en los entornos de juego utilizados.

Learning Rate (α)

El *learning rate* controla cuánto cambian los parámetros del modelo (θ) durante cada paso de optimización. Afecta la fórmula del gradiente descendente:

$$\theta_{k+1} = \theta_k - \alpha \nabla_\theta L(\theta_k) \tag{9}$$

Donde:

- θ_k son los parámetros actuales del modelo en la iteración k.
- $\nabla_\theta L(\theta_k)$ es el gradiente de la función de pérdida $L(\theta_k)$, la cual se describe en la sección 3.2.
- α es el *learning rate*.

Número de *Steps* (n_steps)

Este hiperparámetro determina cuántos pasos de interacción se recolectan antes de cada actualización de los parámetros del modelo. Un **paso de interacción** (o *timestep*) se refiere a cada ciclo en el que el agente:

1. Observa el estado actual del entorno.
2. Selecciona una acción basada en su política actual.
3. Ejecuta la acción en el entorno.
4. Recibe una recompensa y una nueva observación del estado.

Cada uno de estos ciclos genera un **dato de interacción**, que incluye la observación inicial, la acción tomada, la recompensa obtenida y la nueva observación. En este contexto, los términos pasos de interacción y datos de interacción están relacionados, ya que cada paso produce un dato relevante para el entrenamiento del agente.

La estimación de la ventaja $\hat{A}_t$, descrita en la sección 3.2, se basa en los valores acumulados de recompensas obtenidos durante n_steps.

Batch Size (*B*)

El batch size define cuántos datos de interacción se utilizan para calcular el gradiente en cada iteración. Este valor afecta la estimación del gradiente:

$$\nabla_\theta L(\theta_k) = \frac{1}{B}\sum_{i=1}^{B} \nabla_\theta L_i(\theta) \tag{10}$$

Donde:

- *B* es el *batch size*.

En los tres juegos seleccionados, el espacio de observación está compuesto por una pila de 4 fotogramas consecutivos del juego, los cuales han sido preprocesados para estar en escala de grises y redimensionados a 84x84 píxeles. Durante cada *timestep*, se agrega un nuevo fotograma a la pila y se elimina el más antiguo, lo que permite al agente captar información temporal como la dirección y velocidad de los movimientos en el entorno.

Esta estrategia es especialmente útil porque, a diferencia de las imágenes individuales, la pila de fotogramas proporciona un contexto visual sobre cómo los elementos del entorno se desplazan con el tiempo. Esto permite al agente no solo reconocer el estado actual del juego, sino también anticipar futuros eventos al analizar el movimiento y la trayectoria de los objetos. Por ejemplo, si un oponente se mueve rápidamente hacia el agente, la secuencia de fotogramas permitirá identificar esta tendencia, ayudando al agente a tomar decisiones más proactivas y efectivas.

Esta pila de 4 fotogramas se transforma en un tensor 4D que se utiliza como entrada para la CNN. La forma de este tensor es (*batch_size, n_stack, height, width*), donde:

- *batch_size*: es el número de pilas procesadas simultáneamente.
- *n_stack*: es el número de fotogramas apilados.
- *height, width*: son las dimensiones de cada fotograma.

Para el caso específico de este trabajo, el tensor tiene la forma (64,4,84,84). Este tensor representa el estado visual del juego, proporcionando a la CNN información clave para que el agente pueda interpretar su entorno y tomar decisiones informadas.

A partir de esta entrada, la CNN genera como salida una distribución de probabilidades sobre las acciones posibles y un valor estimado que representa la recompensa futura esperada desde el estado actual.

4.2. Metodología experimental

En [9], los autores utilizaron *The Legend of Zelda*, *Mega Man X* y *M.U.G.E.N.* como entornos de prueba para evaluar la efectividad de las recompensas decaídas en el aprendizaje por refuerzo. A diferencia de ese enfoque, esta memoria seleccionó los juegos *Punch Out!!*, *The Legend of Zelda* y *Mega Man X*, ampliando el espectro de evaluación a un juego con dinámicas de combate diferentes. En cada uno de estos videojuegos se escogieron escenarios específicos basándose en criterios como dificultad, comportamiento de los enemigos y disponibilidad de variables de memoria, además de los escenarios utilizados en [9].

4.2.1. Punch Out!! para NES

Se utilizaron dos escenarios para este juego, la batalla contra el primer contrincante, Glass Joe (Figura 4.2a), y la batalla contra el segundo contrincante, Von Kaiser (Figura 4.2b), siendo el segundo de mayor dificultad que el primero.

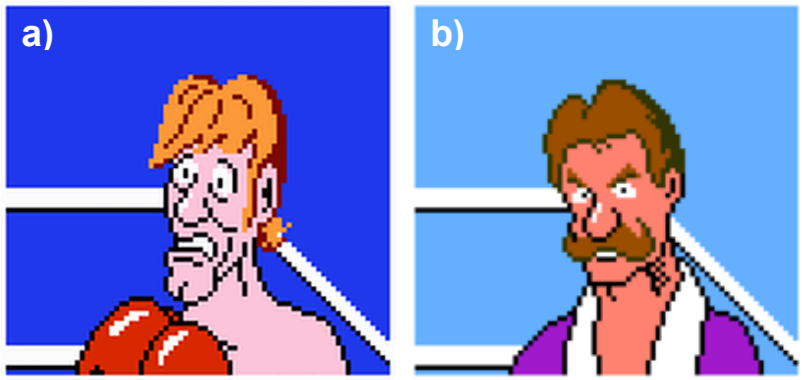


Figura 4.2. Oponentes de Punch Out!!: a) Glass Joe y b) Von Kaiser.

Espacio de Observación

Como se mencionó anteriormente, el espacio de observación para este juego está compuesto por una pila de 4 fotogramas consecutivos del juego, los cuales han sido preprocesados para estar en escala de grises y redimensionados a 84x84 píxeles (Figura 4.3).

Figura 4.3. Fotograma del juego Punch Out!! redimensionado a 84x84 pixeles y en escala de grises.

El agente no recibe de forma explícita información sobre la acción realizada en el paso anterior ni sobre variables como su vida restante. No obstante, la pila de fotogramas le proporciona suficiente información para deducir cómo sus decisiones afectan el entorno. Las recompensas y penalizaciones que se le otorgan al agente dependen de los valores de ciertas variables, pero no se utilizan directamente como entrada en la red convolucional. En cambio, estas influyen en el proceso de ajuste de los pesos de la red a través de los cálculos de ventaja y pérdida, permitiendo al agente mejorar sus decisiones en el futuro.

Por ejemplo, cuando Little Mac (el agente que instancia el modelo entrenado) pierde vida, esto se refleja en la disminución del valor de la variable que representa su vida restante. Aunque la CNN no recibe directamente este valor como entrada, sí procesa la pila de fotogramas que muestra el golpe recibido. La disminución de la variable de vida genera una penalización que afecta el ajuste de los pesos en la CNN, contribuyendo al aprendizaje del agente.

Espacio de Acciones

Para interactuar con el entorno, el agente puede bloquear (hacia arriba o hacia abajo), esquivar (hacia la derecha o hacia la izquierda) y golpear de cuatro maneras diferentes (con el puño derecho o el izquierdo al cuerpo o a la cabeza, ver Apéndice 1).

Condiciones de Término

Las condiciones de término de los episodios de entrenamiento se definieron según el escenario, y se detallan en la tabla 4.1 (las variables empleadas se describen en el Apéndice 2).

Tabla 4.1. Condiciones de término de los episodios para el entrenamiento en el juego Punch Out!!.

Resultado	Condición	Descripción
Victoria contra Glass Joe	*match* es 1	El valor de la variable match es 0 cuando el agente está peleando contra Glass Joe, si el valor cambia a 1, el agente ganó y pasó al siguiente contrincante
Victoria contra Von Kaiser	*match* es 2	El valor de la variable match es 1 cuando el agente está peleando contra Von , si el valor cambia a 2, el agente ganó y pasó al siguiente contrincante
Derrota	*health_mac* es 0 y *is_match* es 1	La combinación de que la vida restante del agente sea 0 y que la pantalla no esté mostrando una batalla, significa que la pantalla actual es Game Over

Esquema de Recompensas

Los autores de [9] no utilizaron este juego, por lo que su trabajo no se utilizó como referencia para la definición de recompensas. Para determinar qué eventos merecían recompensas o penalizaciones, se elaboró una lista de acciones y eventos que contribuyen a la victoria o a la derrota. Inicialmente, se otorgaron recompensas y penalizaciones solo por las acciones más básicas (perder vida, quitarle vida al oponente, perder la pelea, ganar la pelea). Se planeaba agregar más criterios si los resultados del agente no eran satisfactorios, pero esto no fue necesario en este caso.

Se probó con distintos valores de recompensas y penalizaciones, los mejores resultados se obtuvieron con los valores presentados en la tabla 4.2 (las variables empleadas se describen en el Apéndice 2).

Tabla 4.2. Recompensas y penalizaciones para el entrenamiento en el juego Punch Out!!.

Evento	Recompensa/Penalización
Perder vida	Se penaliza con la cantidad de vida perdida
Quitar vida	Se recompensa con la cantidad de vida quitada
Derrota	-100
Victoria	+100

Sin contar la recompensa o penalización final, en el mejor de los casos, la recompensa acumulada sería 96, y en el peor de los casos, la recompensa acumulada sería de -96. Estos valores se derivan de la suma de las recompensas otorgadas por las acciones individuales del agente durante un episodio típico. Cada acción que reduce la vida del oponente otorga una recompensa positiva, mientras que cada pérdida de vida del agente resulta en una penalización negativa.

El valor máximo de 96 refleja el escenario ideal en el que el agente realiza consistentemente acciones positivas, esto es quitarle los 96 puntos de vida total al oponente y además no recibir daño, mientras que el -96 representa el escenario opuesto, donde el agente acumula penalizaciones por cada acción, esto es perder los 96 puntos de vida total y no lograr disminuir ningún punto de vida del oponente.

Debido a esta diferencia entre las recompensas acumuladas durante el juego y el objetivo final de ganar o perder, se definió una recompensa final de -100 en caso de derrota y de 100 en caso de victoria. Esto se hizo para priorizar la importancia de ganar, asegurando que el agente no solo busque maximizar las recompensas intermedias, sino que también se enfoque en el objetivo final del episodio.

Recompensa Final Decaída

Para la versión decaída de la penalización por derrota, se utilizó como parámetro la vida restante del contrincante con el objetivo de mitigar la cobardía del agente. La recompensa final decaída por la derrota se calcula como la recompensa final estática elevada a una potencia proporcional a la vida restante del oponente en relación con su vida total. La fórmula es la siguiente:

- Recompensa por derrota: $-100^{\frac{health\_com}{96}}$

Este enfoque incentiva al agente a enfrentar al contrincante de manera más agresiva, ya que si el agente evita atacar o se comporta de manera cobarde, la vida del oponente se mantendrá alta. Esto se traduce en una **penalización mayor en caso de derrota**, desincentivando comportamientos en los que el agente solo busca **sobrevivir sin enfrentar directamente al oponente**.

Por ejemplo, si el agente pierde un combate sin haber infligido ningún daño, la vida del contrincante será su valor máximo, lo que resultará en una penalización significativa. En cambio, si el agente reduce la vida del oponente antes de ser derrotado, la penalización será menor, motivándolo a siempre buscar una oportunidad de atacar, incluso en situaciones difíciles.

Para la versión decaída de la recompensa por victoria, se utilizó como parámetro la vida restante del agente con el objetivo de fomentar un comportamiento defensivo. La recompensa final decaída por la victoria se calcula como la recompensa final estática elevada a una potencia proporcional a la vida restante del agente en relación con su vida total. La fórmula es la siguiente:

- Recompensa por victoria: $+100^{\frac{health\_mac}{96}}$

Este enfoque incentiva al agente no solo a buscar la victoria, sino también a **minimizar el daño** recibido durante el combate. Al vincular la vida restante del agente con la recompensa final, se refuerza la idea de que **sobrevivir con más vida es un resultado más deseable**, premiando las estrategias que combinan ataque efectivo con defensa eficiente.

Por ejemplo, si el agente logra ganar un combate con una vida cercana al máximo, recibirá una recompensa significativamente mayor en comparación con una victoria en la que apenas sobreviva. Esto ayuda a evitar que el agente adopte un comportamiento demasiado arriesgado, donde priorice ganar sin importar las consecuencias, y en su lugar, fomenta una actitud más calculadora y defensiva.

Para el entrenamiento de los agentes, se utilizaron los valores de hiperparámetros (learning rate, n_steps, batch size y target KL) propuestos en [9] (ver Apéndice 7). Estos valores fueron seleccionados debido a su eficacia demostrada en [9], donde se utilizaron en entornos similares de videojuegos retro, logrando un buen equilibrio entre estabilidad y velocidad de aprendizaje del agente.

Durante las pruebas preliminares, se evaluaron diferentes configuraciones de *timesteps* para determinar el punto en el que el agente dejaba de mostrar mejoras significativas en su rendimiento. Se observó que, más allá de los 5 millones de *timesteps* en el escenario de Glass Joe y los 10 millones de *timesteps* en el escenario de Von Kaiser, el aprendizaje se estabilizaba, indicando que el agente había alcanzado su máximo rendimiento posible bajo las condiciones actuales.

El máximo de 5000 *timesteps* por episodio fue seleccionado porque representa una cantidad suficiente para que el agente pueda alcanzar una victoria o sufrir una derrota en un solo episodio. Un límite mayor resultaba innecesario, ya que, en las pruebas realizadas, el agente siempre completaba el episodio antes de llegar a este umbral.

Además, se estableció un intervalo de guardado de 1200 episodios, ya que esto permitía capturar el progreso del modelo en momentos clave del entrenamiento, sin sobrecargar el almacenamiento ni interrumpir excesivamente el proceso.

4.2.2. The Legend of Zelda para NES

Se utilizaron dos escenarios para este juego, la batalla contra el oponente final de la primera mazmorra, Aquamentus (Figura 4.4a), y la batalla contra el oponente final de la tercera mazmorra, Manhadla (Figura 4.4b), siendo el segundo mayor en dificultad que el primero. Ambos oponentes pueden atacar de dos formas, tocando al agente con su cuerpo o con las bolas de fuego que lanzan, pero mientras Aquamentus tiene un movimiento lento y limitado a la derecha de la pantalla, Manhandla se mueve más rápido y a través de toda la pantalla, además cuenta con varias “cabezas” desde las cuales puede efectuar ataques.

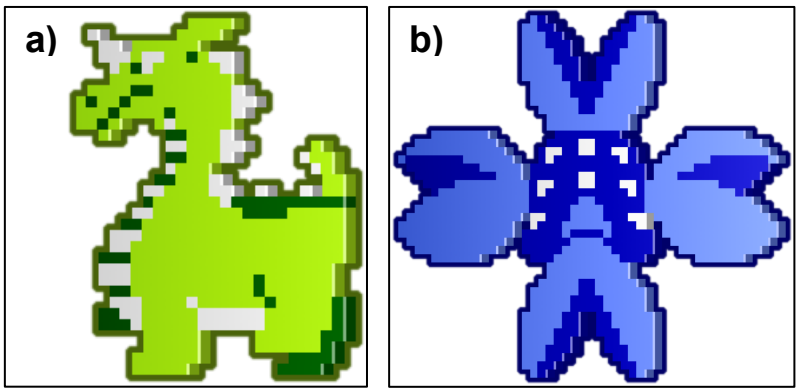


Figura 4.4. Oponentes de The Legend of Zelda: a) Aquamentus y b) Manhandla.

Espacio de Observación

Está compuesto por una pila de 4 fotogramas consecutivos del juego, los cuales han sido preprocesados para estar en escala de grises y redimensionados a 84x84 píxeles (Figura 4.5).

Figura 4.5. Fotograma del juego The Legend of Zelda redimensionado a 84x84 pixeles y en escala de grises.

Al igual que en el juego anterior, el agente no recibe de forma explícita información sobre la acción realizada en el paso anterior ni sobre variables como su vida restante.

Espacio de Acciones

Para interactuar con el entorno, el agente puede moverse en cuatro direcciones (hacia arriba, abajo, izquierda y derecha) y atacar con la espada (a distancia si tiene toda la vida o cuerpo a cuerpo en otro caso) (ver Apéndice 3).

Condiciones de Término

Las condiciones de término de los episodios de entrenamiento se definieron según el escenario, y se detallan en la tabla 4.3 (las variables empleadas se describen en el Apéndice 4).

Tabla 4.3. Condiciones de término de los episodios para el entrenamiento en el juego The Legend of Zelda.

Resultado	Condición	Descripción
Victoria contra Aquamentus	*Enemy_HP* es 0 u 80	Si Aquamentus es derrotado, su vida puede ser 0 (no deja botín al morir) u 80 (deja botín al morir)
Victoria contra Manhandla	*Manhandla_HP_1, 2, 3 y 4* son 0	Si la vida de todas las cabezas de Manhandla llegan a 0, el oponente ha sido derrotado
Derrota contra Aquamentus	*Hearts* es 50 y *Partial_Heart* es 0	En este escenario la vida máxima del agente es 6 corazones, cuando la condición se cumple, el agente tiene 0 corazones
Derrota contra Manhandla	*Hearts* es 70 y *Partial_Heart* es 0	En este escenario la vida máxima del agente es 8 corazones, cuando la condición se cumple, el agente tiene 0 corazones
Abandono de combate	*Screen* no es 5	Si el valor deja de ser 5, el agente está saliendo de la habitación del combate

Esquema de Recompensas

Este juego es uno de los utilizados en [9]. En dicho trabajo, los autores indican que se otorgó una recompensa al agente cuando lograba disminuir la vida del oponente (+1) o se acercaba a él (+0.001), y una penalización cuando perdía vida (0.5 si es golpeado por una bola de fuego y 1 si toca al oponente) o se alejaba del oponente (0.001).

Para esta memoria, inicialmente se emplearon esos mismos criterios para la entrega de recompensas y penalizaciones. Sin embargo, estos no generaron los resultados esperados.

Al observar el desempeño del agente entrenado, se pudo notar que, si bien el agente se acercaba al oponente, no siempre lo afrontaba directamente, lo cual es necesario para infligirle daño. Esto derivó en un comportamiento subóptimo, donde el agente priorizaba estar cerca del oponente sin realizar acciones efectivas para atacar.

Para abordar este problema, se decidió añadir un **nuevo criterio de recompensa y penalización**: el agente recibe una **recompensa** adicional cuando **afronta directamente al oponente** y una **penalización** si **evita el enfrentamiento**. Esta modificación introdujo una diferencia respecto a [9], ya que no solo se evaluó la proximidad del agente al oponente, sino también su disposición activa a atacar. Esta diferencia en la definición de recompensas resultó en un **aumento significativo del rendimiento del agente**, promoviendo un comportamiento más agresivo y efectivo durante los combates.

Se experimentó con distintos valores para las recompensas y penalizaciones, obteniéndose los mejores resultados con los valores presentados en la tabla 4.4 (las variables empleadas se describen en el Apéndice 4).

Tabla 4.4. Recompensas y penalizaciones para el entrenamiento en el juego The Legend of Zelda.

Evento	Recompensa/Penalización
Perder vida	Se penaliza con la cantidad de vida perdida
Quitar vida a Aquamentus	Se recompensa con la cantidad de vida quitada dividida en 4
Quitar vida a Manhandla	Se recompensa con la cantidad de vida quitada dividida en 10
Acercarse al oponente	+0.001 por acercarse, -0.001 por alejarse
Afrontar al oponente	Si está en el mismo eje x o y que el oponente, +0.001 por afrontarlo, -0.001 por no hacerlo
Abandono de combate	-30
Derrota	-30
Victoria	+30

En la batalla contra Aquamentus, la vida máxima del agente es 12, mientras que la del oponente es 60. Si el agente pierde toda su vida, la recompensa total sería de -12. Por otro lado, siguiendo la misma lógica, si el agente logra quitarle toda la vida al oponente, la recompensa sería 60. Para equilibrar estos valores, se optó por dividir entre 4 la cantidad de vida quitada a Aquamentus y utilizar este nuevo valor como recompensa. De esta forma, la recompensa total por derrotar al oponente sería 15, lo que da una importancia similar tanto a perder vida como a infligir daño al enemigo.

En la batalla contra Manhandla, la vida máxima del agente es 16, mientras que la del oponente es 160. Si el agente pierde toda su vida, la recompensa total sería de -16. En cambio, si el agente logra quitarle toda la vida al oponente, la recompensa sería 160. Para estandarizar estos valores, se decidió dividir entre 10 la cantidad de vida quitada a Manhandla y usar este valor como recompensa. Así, la recompensa total por derrotar a este oponente sería 16, manteniendo nuevamente una importancia similar entre perder vida y dañar al enemigo.

Finalmente, se definió una recompensa de 30 al ganar la batalla y de -30 al perder o abandonar el combate, dado que el objetivo principal es ganar, y se quiso resaltar la importancia de este logro.

Recompensa Final Decaída

Para la versión decaída de la penalización por derrota, se utilizó como parámetro la vida del oponente para mitigar la cobardía del agente. La recompensa final decaída por la derrota es la recompensa final estática por derrota elevada a la potencia proporcional a la vida restante del oponente en relación con su vida total. La fórmula es la siguiente:

- Recompensa por derrota contra Aquamentus: $-30^{\frac{Enemy\_HP}{60}}$
- Recompensa por derrota contra Manhandla: $-30^{\frac{Vida\_Manhandla}{160}}$

Para la versión decaída de la recompensa por victoria, se utilizó como parámetro la vida del agente para impulsar un comportamiento defensivo. La recompensa final decaída por la victoria es la recompensa final estática por victoria elevada a la potencia proporcional a la vida restante del oponente en relación con su vida total. La fórmula es la siguiente:

- Victoria contra Aquamentus: $+30^{\frac{Vida\_Link}{12}}$
- Victoria contra Manhandla: $+30^{\frac{Vida\_Link}{16}}$

Para el entrenamiento de los agentes, en un inicio, se utilizaron los valores de hiperparámetros (*learning rate, n steps, batch size* y *target KL*) propuestos en [9], pero los resultados no fueron los esperados para el escenario de Aquamentus, por lo que se utilizó *Optuna*, un *framework* de optimización de hiperparámetros y se utilizaron los valores obtenidos en él (ver Apéndice 8).

Para el caso de Manhandla, también se utilizó Optuna para la optimización de hiperparámetros. Sin embargo, debido a la larga duración del proceso, no se pudo realizar con una gran cantidad de episodios. Como resultado, los valores de hiperparámetros obtenidos no son necesariamente los óptimos. Posteriormente esos hiperpárametros se fueron modificando en base a las métricas de pérdida obtenidas en el entrenamiento, para lograr una configuración de hiperparámetros que permitiese al agente aprender (ver Apéndice 9).

Para cada escenario se entrenó a un agente con la versión de recompensa final estática y otro con la versión decaída. Tanto para el escenario de combate contra Aquamentus como para el contra Manhandla, se entrenó a ambos agentes durante 10 millones de *timesteps*.

La elección de la cantidad de *timesteps* se basó en el análisis de las curvas de aprendizaje durante las pruebas preliminares. En el caso de Aquamentus, se observó que la curva de recompensas promedio crecía rápidamente al inicio del entrenamiento, pero luego se estabilizaba con un crecimiento muy lento, casi imperceptible. Esto indicaba que, aunque el agente seguía mejorando, el rendimiento adicional obtenido con más *timesteps* no justificaría un mayor tiempo de entrenamiento.

En el escenario de Manhandla, la situación era diferente, el crecimiento de la curva de aprendizaje era muy lento pero constante. Aunque se puede ver que el modelo puede mejorar con mayor entrenamiento, se optó por dejarlo en 10 millones de *timesteps* para no exceder el tiempo disponible para el proyecto.

El límite de 5000 *timesteps* por episodio se definió para asegurar que cada episodio tuviera suficiente tiempo para permitir la victoria o derrota del agente, sin prolongar innecesariamente el tiempo de simulación. Además, el modelo se guardó cada 3000 episodios, permitiendo un seguimiento continuo del progreso sin interrumpir excesivamente el entrenamiento.

4.2.3. Mega Man X para SNES

Se utilizaron tres escenarios para este juego, la batalla contra Chill Penguin (Figura 4.6a), la batalla contra Spark Mandrill (Figura 4.6b) usando el arma base y la batalla contra Spark Mandrill usando *shotgun ice*. La batalla contra Spark Mandrill tiene mayor dificultad que la batalla contra Chill Penguin si se utiliza el arma base. En este juego, los niveles se pueden jugar en cualquier orden, sin embargo, para generar los *savestates* utilizados se siguió un orden recomendado: primero se peleó contra Chill Penguin, teniendo disponible solo el arma base, y luego se peleó contra Spark Mandrill, teniendo disponible el arma base y *shotgun ice* (el arma que se gana al derrotar a Chill Penguin), esta arma es la debilidad de Spark Mandrill y, al usarla contra él, la dificultad de ambos escenarios se equipara.

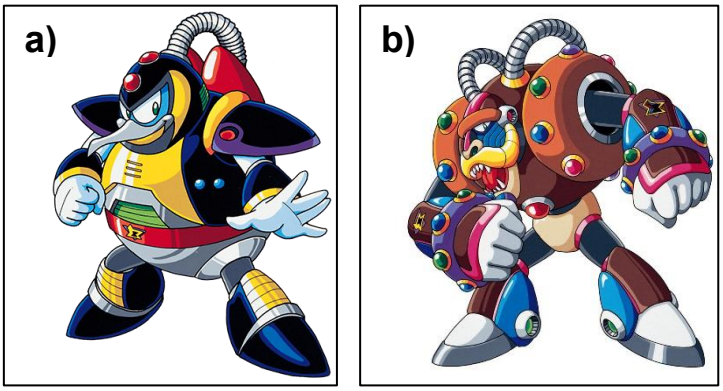


Figura 4.6. Oponentes de Mega Man X: a) Chill Penguin y b) Spark Mandrill.

Espacio de Observación

Está compuesto por una pila de 4 fotogramas consecutivos del juego, los cuales han sido preprocesados para estar en escala de grises y redimensionados a 84x84 píxeles (Figura 4.7).

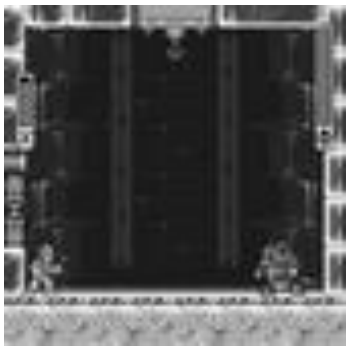


Figura 4.7. Fotograma del juego Mega Man X redimensionado a 84x84 pixeles y en escala de grises.

Al igual que en los juegos anteriores, el agente no recibe de forma explícita información sobre la acción realizada en el paso anterior ni sobre variables como su vida restante.

Espacio de Acciones

Para interactuar con el entorno, el agente puede moverse en el eje x (hacia la izquierda y derecha), saltar, hacer un *dash* y disparar con su arma (ver Apéndice 5).

Condiciones de Término

Las condiciones de término de los episodios de entrenamiento se definieron según el escenario, y se detallan en la tabla 4.5 (las variables empleadas se describen en el Apéndice 6).

Tabla 4.5. Condiciones de término de los episodios para el entrenamiento en el juego Mega Man X.

Resultado	Condición	Descripción
Victoria	*boss* es 0	Si la vida del oponente es 0, este ha sido derrotado
Derrota	*health* es 0	Si la vida del agente es 0, ha perdido

Esquema de Recompensas

Este juego es otro de los utilizados en [9], en él, los autores indican que se otorgó una recompensa al agente cuando logra disminuir la vida del oponente (+1 por cada punto de daño) y una penalización cuando pierde vida (1 por cada punto de daño).

Se utilizaron los mismos eventos y valores para las recompensas y penalizaciones que usaron los autores en [9], pero se probó con distintos valores para la recompensa o penalización final, los mejores resultados se obtuvieron con los valores presentados en la tabla 4.6 (las variables empleadas se describen en el Apéndice 6).

Tabla 4.6. Recompensas y penalizaciones para el entrenamiento en el juego Mega Man X.

Evento	Recompensa/Penalización
Perder vida	Se penaliza con la cantidad de vida perdida
Quitar vida	Se recompensa con la cantidad de vida quitada
Derrota	-50
Victoria	+50

Sin contar la recompensa o penalización final, en el mejor de los casos, la recompensa acumulada sería 32, y en el peor de los casos, la recompensa acumulada sería de -32. Es por esto que se definió la recompensa final como 50 en caso de ganar y -50 en caso de perder, ya que el objetivo principal es ganar y se le quiso dar más importancia a esto.

Recompensa Final Decaída

Para la versión decaída de la penalización por derrota, se utilizó como parámetro la vida del oponente para mitigar la cobardía del agente. La recompensa final decaída por la derrota es la recompensa final estática por derrota elevada a la potencia proporcional a la vida restante del oponente en relación con su vida total. La fórmula es la siguiente:

- Recompensa por derrota: $-50^{\frac{boss}{32}}$

Para la versión decaída de la recompensa por victoria, se utilizó como parámetro la vida del agente para impulsar un comportamiento defensivo. La recompensa final decaída por la victoria es la recompensa final estática por victoria elevada a la potencia proporcional a la vida restante del agente en relación con su vida total. La fórmula es la siguiente:

- Victoria: $+50^{\frac{health}{32}}$

Para el entrenamiento de los agentes, se utilizaron los valores de hiperparámetros (*learning rate, n steps, batch size* y *target KL*) propuestos en [9] (ver Apéndice 7).

Para cada escenario se entrenó a un agente con la versión de recompensa final estática y otro con la versión decaída. En los tres escenarios de combate de Mega Man X, se entrenó a ambos agentes durante 40 millones de *timesteps*. Se definió un máximo de 5000 *timesteps* por episodio y se guardó el modelo cada 6000 episodios.

La decisión de entrenar durante 40 millones de *timesteps* se basó en el análisis de las curvas de aprendizaje obtenidas durante las pruebas preliminares. En los escenarios de Mega Man X, se observó que el crecimiento de las recompensas promedio mostraba un aumento constante, pero a medida que el entrenamiento avanzaba, el progreso se hacía más lento.

A medida que se extendía el entrenamiento, se observó que el agente continuaba mejorando gradualmente. Esto indicaba que el agente aún tenía espacio para aprender, pero las mejoras se volvían cada vez más sutiles.

Debido a las limitaciones de tiempo del proyecto, se optó por 40 millones de *timesteps*, lo cual permitía un equilibrio adecuado entre tiempo de entrenamiento y calidad del aprendizaje del agente.

El límite de 5000 *timesteps* por episodio se mantuvo ya que era suficiente para permitir que el agente completara cada combate sin interrumpir innecesariamente los episodios. Además, se eligió guardar el modelo cada 6000 episodios para capturar el progreso del agente en intervalos representativos, considerando la mayor duración de los entrenamientos.

4.3. Recopilación y tratamiento de datos

Debido al volumen de datos generados y a que múltiples entrenamientos se ejecutaban en paralelo, se optó por implementar un **monitor personalizado** para el seguimiento del rendimiento del agente. Este monitor, basado en VecEnvWrapper, una clase de la biblioteca Stable-Baselines3, permitió capturar y registrar métricas clave de cada instancia en un archivo CSV sin sobrescribir la información existente.

El monitor registraba datos como:

- Recompensas obtenidas en cada episodio.
- Duración de los episodios (en número de *steps*).
- Estados finales de cada entorno, incluyendo la vida restante y el resultado del episodio (victoria o derrota).

Para evitar conflictos al escribir datos desde múltiples entornos paralelos, se utilizó la función de Python *step_wait*, que establece una estructura tipo pila donde cada instancia espera a que la anterior termine de escribir sus datos antes de proceder. Esto aseguró un registro ordenado y sin pérdidas de información, permitiendo un análisis preciso del desempeño del agente en cada episodio.

Además, se implementó una función personalizada para modificar la recompensa acumulada en cada episodio, con el fin de tener un mayor control sobre las recompensas y penalizaciones otorgadas al agente.

El archivo *scenario.json* es una herramienta proporcionada por Gym-Retro que permite definir recompensas y penalizaciones automáticas basadas en variables específicas del entorno, así como establecer condiciones de finalización de episodios. Sin embargo, se optó por no utilizarlo para este propósito, ya que algunas acciones que requerían una recompensa o penalización dependían de más de una variable, lo que hacía más práctico gestionar la modificación de la recompensa directamente en el código.

Dada la cantidad de datos procesados, se utilizaron las bibliotecas Pandas, Matplotlib y NumPy de Python para el tratamiento y visualización de los resultados.

Finalmente, también se implementó una función *callback* para guardar periódicamente el modelo actual y poder comparar su desempeño con el del mismo modelo en distintos momentos del entrenamiento.

5. Experimentos y resultados

Se compararon todos los modelos entrenados con recompensa final estática y decaída, que en adelante serán denominados modelo estático y modelo decaído, respectivamente. Los entrenamientos correspondientes a cada tipo de recompensa se denominarán entrenamiento estático y entrenamiento decaído. A continuación, se presentan los resultados obtenidos en ambos tipos de entrenamiento, realizados en distintos escenarios: dos en Punch-Out!!, dos en The Legend of Zelda y tres en Mega Man X. Por cada escenario se generaron un modelo estático y uno decaído, evaluados en tres aspectos: la eficiencia durante el entrenamiento, el desempeño en el escenario utilizado para entrenar y el rendimiento en un escenario distinto al del entrenamiento.

Los gráficos que comparan el entrenamiento de ambos casos muestran los resultados a lo largo de todos los episodios. Cada valor de un episodio representa el promedio de los episodios de un conjunto de juegos, formado por los 12 entornos paralelizados que comparten el mismo índice (es decir, promedia el N-ésimo juego en todos los entornos). Dado que el entrenamiento se realiza en función de una cantidad de *timesteps* y no de episodios, es posible que los modelos estático y decaído no hayan sido entrenados durante el mismo número de episodios.

Por otro lado, los resultados de los agentes que jugaron utilizando los modelos estático y decaído (en adelante, referidos como “agente estático” y “agente decaído” respectivamente) se presentan en tablas que incluyen el promedio y la desviación estándar de los 100 episodios jugados por cada modelo, así como los resultados de un agente que toma decisiones aleatorias (en adelante, referido como “agente aleatorio”) en cada escenario. Para determinar si la diferencia observada entre los modelos es significativa, se utilizó la prueba t de Student [13], una prueba estadística que determina si existe una diferencia significativa entre las medias de dos grupos de datos, usando prueba de hipótesis. La prueba produce un valor p que indica si la diferencia es significativa; en este caso, se utilizó el umbral convencional de p=0.05. Si el valor p obtenido es inferior a este límite, se concluye que los datos son significativamente diferentes con un 95% de confianza. Este enfoque se aplicó a los resultados de recompensas acumuladas por episodio, cantidad de *timesteps* por episodio y vida restante al final de los episodios.

Para el caso de los porcentajes de victoria, se utilizó la prueba z para proporciones [14] con un nivel de significancia de p = 0.05, con el objetivo de contrastar si las diferencias observadas eran estadísticamente significativas. Adicionalmente, para estimar la incertidumbre asociada a cada proporción, especialmente en casos extremos como proporciones iguales a 0% o 100%, se calcularon intervalos de confianza binomiales al 95% mediante el método de Clopper-Pearson [17].

5.1. Punch Out!! para NES

5.1.1. Entrenamiento con Glass Joe

La Figura 5.1 muestra las recompensas promedio al final de cada episodio. Debido a que el entrenamiento decaído recibe una recompensa final menor, no es posible comparar ambas curvas directamente en cuanto a sus valores, pero sí es posible hacerlo en cuanto a su forma. Esto ocurre en todos los gráficos de este informe que muestran la recompensa acumulada durante el entrenamiento.

En la Figura 5.1 se observa que tanto la curva de aprendizaje del entrenamiento decaído como la del entrenamiento estático crecieron rápidamente al inicio. Sin embargo, mientras que la curva del entrenamiento estático alcanzó un máximo alrededor del episodio 40 y luego comenzó a disminuir, la del entrenamiento decaído mantuvo una inclinación positiva hasta converger con la otra curva aproximadamente luego de 225 episodios.

Esta diferencia puede explicarse por varios factores observados en las métricas de pérdida:

- **Ajustes de la política**: La versión decaída mostró cambios en la política más graduales y sostenidos en el tiempo, mientras que la versión estática realizó ajustes más agresivos al inicio, pero luego redujo su aprendizaje. Esto sugiere que la versión decaída continuó refinando su estrategia, lo que se traduce en una curva de recompensa más estable y en ascenso (ver Apéndice 105).
- **Exploración y explotación**: La versión decaída mostró una exploración más rápida al inicio, seguida de una estabilización, lo que indica que el agente encontró buenas estrategias más temprano. En cambio, la versión estática mantuvo una exploración más prolongada e ineficiente, lo que podría haber causado oscilaciones en su recompensa acumulada (ver Apéndice 106).
- **Estimación de valor**: La función de valor de la versión decaída se estabilizó antes, lo que sugiere que el agente aprendió a evaluar mejor los estados del juego de manera más rápida y precisa. En la versión estática, la estimación del valor fue menos estable, lo que podría haber afectado la consistencia de sus decisiones y explicado la caída en su recompensa acumulada (ver Apéndice 107).

Otro posible factor que contribuye a la disminución en el rendimiento de la versión estática es el fenómeno conocido como "*catastrophic forgetting*". Al no recibir un incentivo dinámico basado en su desempeño, el agente podría haber sobreajustado su política a comportamientos recientes, olvidando estrategias previas efectivas. Esto es especialmente relevante en escenarios donde una pequeña diferencia en las decisiones puede tener un gran impacto en la recompensa final.

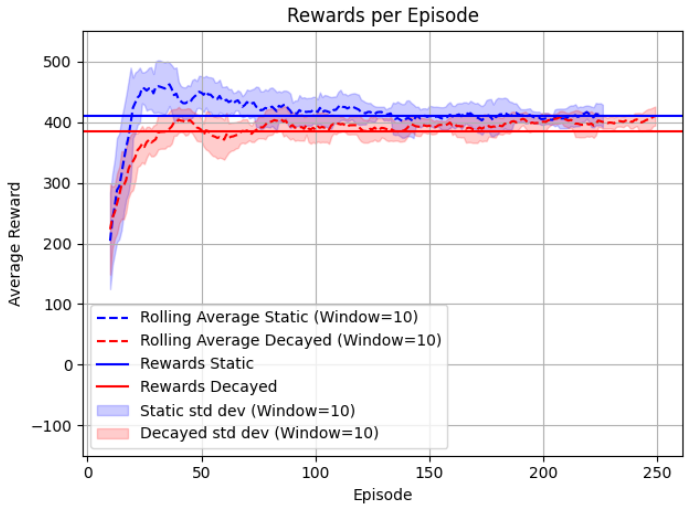


Figura 5.1. Entrenamiento con Glass Joe: Gráfico de recompensa promedio por episodio.

La Figura 5.2 muestra el promedio de *timesteps* que duraron los episodios. Se observa que el agente decaído finaliza consistentemente los episodios en menos *timesteps* durante todo el entrenamiento. Esto se relaciona con su mejor capacidad de exploración y refinamiento estratégico, que le permite identificar rápidamente las mejores acciones sin prolongar innecesariamente los episodios.

Aunque la diferencia entre ambos agentes no alcanza un orden de magnitud, el modelo decaído presenta un promedio general inferior (1711.1 *timesteps* por episodio) en comparación con el modelo estático (1883.2 *timesteps*), lo que representa una reducción aproximada del 9%. Esta diferencia se mantiene estable a lo largo del entrenamiento y se evidencia también en la trayectoria de su media móvil, que permanece sistemáticamente por debajo de la del agente estático.

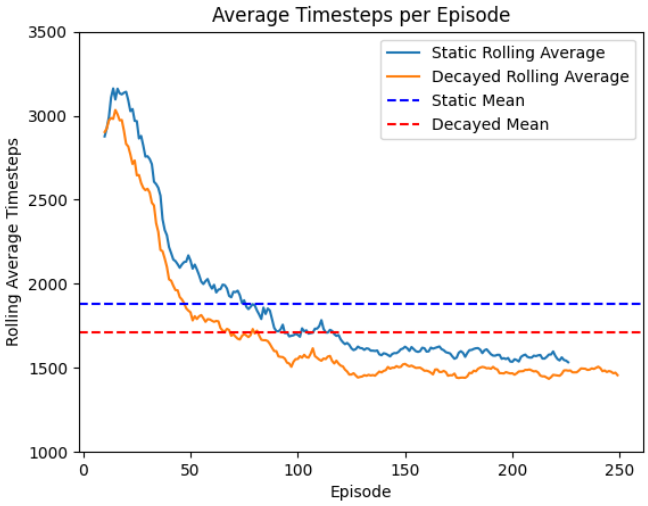


Figura 5.2. Entrenamiento con Glass Joe: Gráfico de timesteps promedio por episodio.

La Figura 5.3 muestra el porcentaje de victorias. Un valor de 1 significa que el 100% de los 12 episodios de las distintas instancias promediadas para cada punto en el eje X fueron ganados en ese episodio. El entrenamiento estático alcanzó buenos resultados más rápido, pero a partir del episodio 50, la curva del entrenamiento decaído logra mantener un 100% de victorias de manera más consistente, mientras que la del entrenamiento estático presenta una leve inestabilidad.

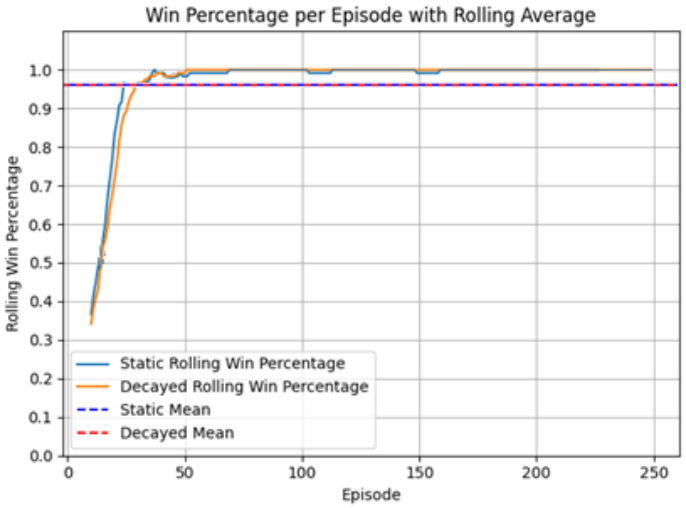


Figura 5.3. Entrenamiento con Glass Joe: Gráfico de porcentaje de victoria promedio por episodio.

Estos resultados indican que, en este escenario, el aprendizaje es más estable y eficiente con el entrenamiento decaído. La tendencia positiva en la recompensa acumulada (Figura 5.1), la menor cantidad de *timesteps* por episodio (Figura 5.2) y la estabilidad en el porcentaje de victorias (Figura 5.3) respaldan la efectividad de este enfoque.

En línea con la propuesta de [9], la recompensa decaída debería fomentar menos "cobardía" y una mayor maestría defensiva. Los gráficos respaldan esta idea: un comportamiento menos temeroso permite finalizar los episodios más rápido, ya que el agente no pierde tiempo en una defensa excesiva. Además, este enfoque facilita un aprendizaje más eficiente al explorar mejores estrategias que la versión estática podría haber evitado por adoptar un comportamiento más conservador. La maestría en defensa también se refleja en la consistencia en el porcentaje de victorias una vez avanzado el entrenamiento.

Con los modelos estático y decaído ya entrenados, se evaluó su desempeño en dos escenarios: la batalla contra Glass Joe, para analizar su rendimiento en el mismo escenario donde fue entrenado, y la batalla contra Von Kaiser, para evaluar si el agente puede generalizar lo aprendido a un entorno que no ha visto antes. Glass Joe representa un desafío más sencillo que Von Kaiser, por lo que este último no solo es un escenario desconocido para el agente, sino también más complejo que aquel en el que fue entrenado.

5.1.1.1. Evaluación contra Glass Joe

A continuación, se muestra el rendimiento de los agentes estático y decaído, además de un agente aleatorio, cuando juegan en el mismo escenario en el que fueron entrenados.

Tabla 5.1.1. Resultados del agente entrenado al enfrentarse a Glass Joe: Promedio de Recompensas Acumuladas, *Timesteps* y Vida Final por Episodio (Desviación Estándar entre paréntesis).

Agente	Recompensa Acumulada	*Timesteps*	Vida Final por Episodio
Aleatorio	-131.45 (78.614677)	6387.31 (1790.664132)	4.03 (18.868203)
Estático	411.83 (29.090567)	1544.56 (234.541822)	92.09 (11.315560)
Decaído	397.83 (58.674195)	1439.73 (240.018618)	94.21 (9.541798)

Dados estos resultados, se puede concluir que el mejor rendimiento, en el ámbito de recompensa acumulada, lo obtuvo el modelo estático, seguido del modelo decaído y, finalmente, el agente aleatorio, cuyo desempeño fue significativamente inferior (ver Apéndice 10).

Se puede concluir que el modelo decaído tardó menos que el modelo estático, y que ambos modelos fueron considerablemente más rápidos que el agente aleatorio (ver Apéndice 11).

No se puede concluir que los modelos estático y decaído tengan una diferencia significativa en el ámbito de vida final por episodio (valor p = 0.1536), pero sí que ambos logran finalizar los episodios con una vida final considerablemente mayor que la del agente aleatorio (ver Apéndice 12).

Porcentaje de victoria

El agente aleatorio ganó un 5% de los juegos, por otro lado, tanto el modelo estático como el decaído triunfaron el 100% de las veces. Dados estos resultados y su interpretación estadística en base a los valores p, dados por la prueba z, no se puede concluir que los porcentajes de victoria de los modelos estático y decaído tengan una diferencia significativa (valor p = 1), pero sí que ambos ganan significativamente más veces que el agente aleatorio.

Además, se calcularon intervalos de confianza binomiales al 95% utilizando el método descrito en [17]. Para el agente aleatorio, el intervalo fue (1.6%, 11.3%), mientras que para ambos modelos entrenados fue (96.3%, 100%).

Análisis cualitativo

Al observar a los agentes jugar, se puede notar que los agentes estático y decaído aprendieron a utilizar únicamente golpes con el puño izquierdo: el *jab* de izquierda y el golpe

al cuerpo con la izquierda. El agente estático (ver Apéndice 13) rara vez esquiva, y no se observó que el agente decaído (ver Apéndice 14) lo hiciera. Esto se debe a que ambos logran derribar al contrincante Glass Joe antes de perder la capacidad de seguir golpeando, y como Glass Joe no tiene ataques que requieran ser esquivados, la esquiva no resulta necesaria. Esta estrategia es efectiva, ya que el *jab* de izquierda contrarresta uno de los golpes del oponente y el golpe al cuerpo contrarresta el otro. Que el agente decaído no esquive se relaciona con terminar los episodios más rápido que el agente estático y demuestra mayor valentía.

En ningún caso los agentes llegan a la segunda ronda, ya que derrotan a su oponente antes. Por otro lado, el agente aleatorio raramente gana (ver Apéndice 15), ya que sus acciones no dependen de lo que hace su enemigo, por lo que usualmente no esquiva cuando lo van a golpear ni golpea donde debe.

5.1.1.2. Evaluación contra Von Kaiser

A continuación, se muestra el rendimiento de los agentes estático y decaído, además de un agente aleatorio, cuando juegan en un escenario que nunca han visto antes. Esta evaluación pone a prueba la capacidad de generalización de los agentes entrenados.

Tabla 5.1.2. Resultados del agente entrenado al enfrentarse a Von Kaiser: Promedio de Recompensas Acumuladas, *Timesteps* y Vida Final por Episodio (Desviación Estándar entre paréntesis).

Agente	Recompensa Acumulada	*Timesteps*	Vida Final por Episodio
Aleatorio	-162.61 (26.349533)	3341.8 (498.894137)	0 (0)
Estático	119.31 (158.422138)	2212.53 (405.801983)	6.41 (11.119437)
Decaído	77.56 (142.845603)	2291.58 (393.769353)	4.08 (9.790485)

Dados estos resultados, no se puede concluir que el modelo estático haya obtenido un mejor rendimiento que el modelo decaído (valor p = 0.0517), pero sí que ambos tuvieron significativamente mejor desempeño que el agente aleatorio (ver Apéndice 16).

No se puede concluir que el modelo estático tardó menos que el modelo decaído (valor p = 0.1637), pero sí que ambos modelos fueron considerablemente más rápidos que el agente aleatorio (ver Apéndice 17).

Tampoco se puede concluir que el modelo estático termine con más vida al final de los episodios que el modelo decaído (valor p = 0.1174); sin embargo, ambos modelos finalizan con más vida que el agente aleatorio (ver Apéndice 18).

Porcentaje de victoria

El agente aleatorio no ganó ningún juego (0%), mientras que el modelo estático obtuvo victorias en el 31% de las partidas y el modelo decaído en el 19%. Dados estos resultados y su interpretación estadística en base a los valores p, dados por la prueba z, no se puede concluir que el modelo estático tenga un mejor porcentaje de victorias en comparación con el modelo decaído (valor p = 0.05); sin embargo, ambos ganan significativamente más veces que el agente aleatorio.

Adicionalmente, se calcularon intervalos de confianza binomiales al 95% utilizando el método descrito en [17]. Para el modelo estático, el intervalo fue (22.0%, 40.9%), para el modelo decaído (11.4%, 28.7%) y para el agente aleatorio (0.0%, 3.6%).

Análisis cualitativo

Al observar a los agentes jugar, se puede ver que en este escenario ambos agentes utilizan principalmente golpes con el puño izquierdo: el *jab* de izquierda y el golpe al cuerpo con la izquierda, aunque en raras ocasiones emplean el puño derecho. Ambos agentes esquivan de vez en cuando, lo cual es necesario, ya que Von Kaiser es más hábil bloqueando golpes. Esto provoca que los corazones de Mac se agoten antes de terminar la partida, lo que obliga a esquivar golpes para recuperar corazones y poder volver a atacar. No se observan diferencias claras entre los comportamientos de los agentes.

Ninguno de los agentes conoce la mejor estrategia para enfrentarse a este oponente, ya que sus movimientos son distintos a los del contrincante con el que entrenaron. Por ejemplo, a diferencia de Glass Joe, no es posible bloquear todos los ataques de Von Kaiser, además de que este oponente es más rápido en su ataque, lo que resulta en que los agentes reciban múltiples golpes. En muchos casos, los agentes llegan a la segunda ronda, ya que no logran derrotar a su oponente durante el tiempo que dura la primera ronda (ver Apéndice 19 y 20). Por otro lado, el agente aleatorio (ver Apéndice 21) se comporta de la misma manera que en el caso anterior, pero al ser este contrincante más difícil, no logra ganar en ningún caso.

5.1.2. Entrenamiento con Von Kaiser

En la Figura 5.4 se pueden observar las recompensas promedio al final de cada episodio. Se observa que tanto la curva de aprendizaje del entrenamiento decaído como la del entrenamiento estático crecieron rápidamente. Sin embargo, al igual que en los entrenamientos contra Glass Joe, mientras el entrenamiento estático alcanzó un máximo y luego comenzó a disminuir, el entrenamiento decaído mantuvo una tendencia positiva hasta encontrarse con la otra curva. Este comportamiento podría explicarse por la falta de exploración que se genera con la estrategia estática, ya que un esquema de recompensas que no varía según el desempeño del agente podría limitar su capacidad para descubrir estrategias óptimas.

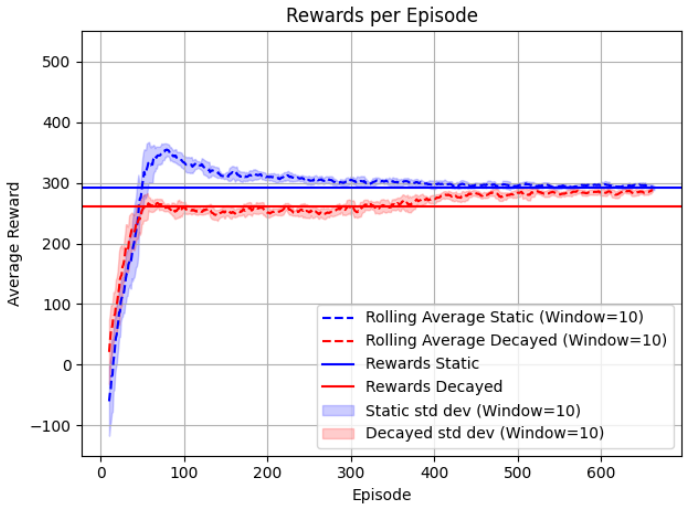


Figura 5.4. Entrenamiento con Von Kaiser: Gráfico de recompensa promedio por episodio.

La Figura 5.5 muestra el promedio de *timesteps* por episodio. En este caso, ambos entrenamientos se comportan de manera similar, presentando curvas con forma y valores promedio prácticamente idénticos: 1271.8 *timesteps* por episodio para el modelo estático y 1267.8 para el decaído, con una diferencia entre promedios menor al 0.3%.

Si bien se aprecia que la curva del entrenamiento decaído alcanza un valor máximo levemente inferior al del entrenamiento estático en los primeros episodios, esta diferencia se reduce rápidamente y ambas curvas convergen a un mismo rango estable.

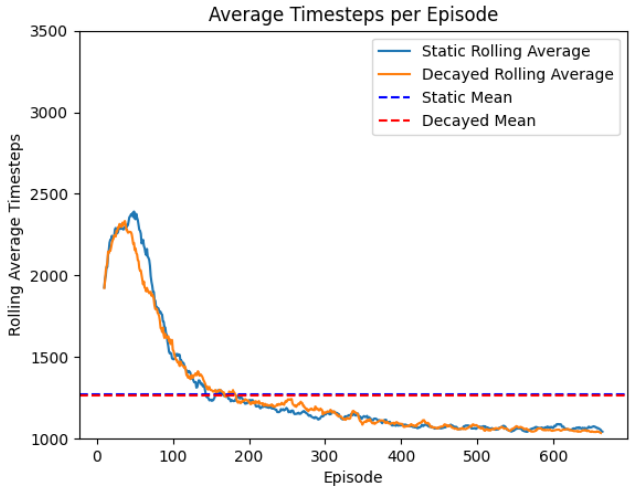


Figura 5.5. Entrenamiento con Von Kaiser: Gráfico de timesteps promedio por episodio.

La Figura 5.6 muestra el porcentaje de victorias. En este caso, el entrenamiento decaído alcanza mejores resultados antes que el entrenamiento estático. Sin embargo, a partir del episodio 200, ambos entrenamientos logran consistentemente un 100% de victorias.

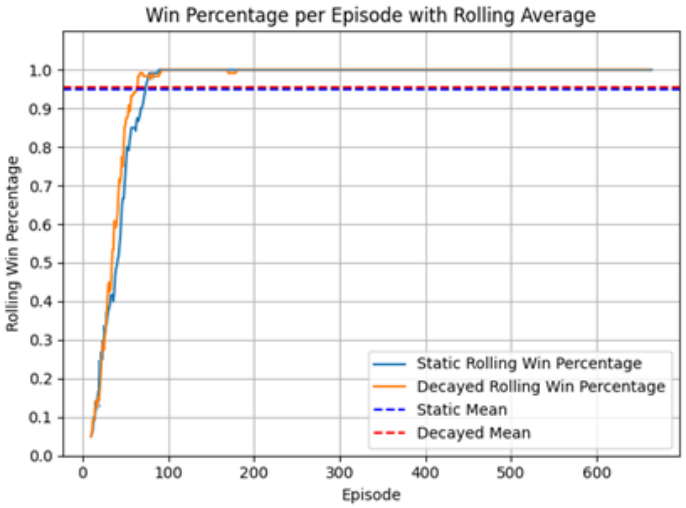


Figura 5.6. Entrenamiento con Von Kaiser: Gráfico porcentaje de victoria promedio por episodio.

Estos gráficos indican que, en este escenario, el aprendizaje es más estable cuando se aplica un entrenamiento decaído. La curva de aprendizaje del entrenamiento decaído es mejor (Figura 5.4) y logra un mejor porcentaje de victoria antes que el entrenamiento estático (Figura 5.6).

Los gráficos muestran un mejor aprendizaje con el entrenamiento decaído, ya que este explora estrategias más efectivas que, debido a tener un comportamiento más conservador, el entrenamiento estático habría evitado.

Con los modelos estático y decaído ya entrenados, se evaluó su desempeño en dos escenarios: la batalla contra Glass Joe y la batalla contra Von Kaiser.

5.1.2.1. Evaluación contra Von Kaiser

A continuación, se muestra el rendimiento de los agentes estático y decaído, además de un agente aleatorio, cuando juegan en el mismo escenario en el que fueron entrenados.

Tabla 5.1.3. Resultados del agente entrenado al enfrentarse a Von Kaiser: Promedio de Recompensas Acumuladas, *Timesteps* y Vida Final por Episodio (Desviación Estándar entre paréntesis).

Agente	Recompensa Acumulada	*Timesteps*	Vida Final por Episodio
Aleatorio	-166.06 (25.155842)	3273.48 (433.426268)	0 (0)
Estático	296.05 (12.441362)	1099.66 1(132.235564)	89.65 (12.153497)
Decaído	296.05 (11.468544)	1061.16 (84.276298)	90.31 (11.906045)

Dados estos resultados, no se puede concluir que el modelo decaído haya tenido un mejor rendimiento que el modelo estático (valor p = 1); sin embargo, ambos obtuvieron un desempeño significativamente mejor que el agente aleatorio (ver Apéndice 22).

Se puede concluir que el modelo decaído tarda menos que el modelo estático; además, ambos modelos son considerablemente más rápidos que el agente aleatorio (ver Apéndice 23).

No se puede concluir que los modelos estático y decaído tengan una diferencia significativa en el ámbito de vida final por episodio (valor p = 0.6985), pero sí que ambos logran finalizar los episodios con una vida final considerablemente mayor que la del agente aleatorio (ver Apéndice 24).

Porcentaje de victoria

El agente aleatorio no ganó ningún juego (0%), por otro lado, tanto el modelo estático como el decaído triunfaron el 100% de las veces. Dados estos resultados y su interpretación estadística en base a los valores p, dados por la prueba z, no se puede concluir que los porcentajes de victoria de los modelos estático y decaído tengan una diferencia significativa (valor p = 1), pero sí que ambos ganan significativamente más veces que el agente aleatorio.

Para complementar este análisis, se calcularon intervalos de confianza binomiales al 95% utilizando el método descrito en [17]. El intervalo para el agente aleatorio fue (0.0%, 3.6%), mientras que para los modelos entrenados fue (96.3%, 100%) en ambos casos.

Análisis cualitativo

Al observar a los agentes jugar, se puede ver que ambos aprendieron a utilizar principalmente golpes con el puño derecho: el gancho de derecha y el golpe al cuerpo con la derecha, aunque en algunas ocasiones usan el puño izquierdo. A veces esquivan cuando Von Kaiser ejecuta el ataque que no puede bloquearse, aunque es raro que Von Kaiser tenga la oportunidad de atacar, ya que generalmente los agentes atacan con la suficiente frecuencia como para no permitirle contraatacar, esto se observa más en el caso del agente decaído, esto se relaciona con su mayor rapidez para terminar los episodios. Esta estrategia resulta efectiva, ya que el golpe al cuerpo con la derecha es útil para contrarrestar uno de los ataques, y el golpe al cuerpo con la izquierda o esquivar es eficaz para contrarrestar el otro. En ningún caso los agentes llegan a la segunda ronda, ya que logran derrotar a su oponente antes de que termine la primera (ver Apéndice 25 y 26).

5.1.2.2. Evaluación contra Glass Joe

A continuación, se muestra el rendimiento de los agentes estático y decaído, además de un agente aleatorio, cuando juegan en un escenario que nunca han visto antes, la batalla

contra Glass Joe. Esta evaluación pone a prueba la capacidad de generalización de los agentes entrenados.

Tabla 5.1.4. Resultados del agente entrenado al enfrentarse a Glass Joe: Promedio de Recompensas Acumuladas, *Timesteps* y Vida Final por Episodio (Desviación Estándar entre paréntesis).

Agente	Recompensa Acumulada	*Timesteps*	Vida Final por Episodio
Aleatorio	-120.55 (79.037886)	6755.65 (1666.326357)	2.28 (13.566930)
Estático	304.83 (146.889214)	3068.67 (916.858845)	36.1 (28.374813)
Decaído	336.65 (107.505477)	2864.26 (812.960573)	38.88 (25.65210)

Dados estos resultados, no se puede concluir que el modelo estático haya obtenido un mejor rendimiento que el modelo decaído (valor p = 0.082), pero sí que ambos tuvieron significativamente mejor desempeño que el agente aleatorio (ver Apéndice 27).

No se puede concluir que el modelo decaído haya tardado menos que el modelo estático (valor p = 0.0969); sin embargo, sí se puede afirmar que ambos modelos fueron considerablemente más rápidos que el agente aleatorio (ver Apéndice 28).

Tampoco se puede concluir que los modelos estático y decaído tengan una diferencia significativa en el ámbito de vida final por episodio (valor p = 0.4682), pero sí que ambos logran finalizar los episodios con una vida final considerablemente mayor que la del agente aleatorio (ver Apéndice 29).

Porcentaje de victoria

El agente aleatorio ganó un 5% de los juegos, el modelo estático un 80% y el decaído un 93% de las veces. Dados estos resultados y su interpretación estadística en base a los valores p, dados por la prueba z, se puede concluir que el modelo decaído tiene un mejor porcentaje de victorias en comparación con el modelo estático; además, ambos ganan significativamente más veces que el agente aleatorio.

Sin embargo, los intervalos de confianza binomiales al 95% calculados mediante el método descrito en [17] fueron (71.2%, 87.6%) para el modelo estático y (85.4%, 97.4%) para el modelo decaído y (1.6%, 11.3%) para el agente aleatorio, lo que evidencia una superposición parcial entre los modelos entrenados. Esta discrepancia entre métodos estadísticos indica que, si bien la prueba z detecta una diferencia significativa, la interpretación basada en intervalos de confianza es más conservadora. Por lo tanto, la superioridad del modelo decaído sobre el estático debe considerarse con cautela.

Análisis cualitativo

Al observar a los agentes jugar, se puede ver que, en este escenario, ambos agentes utilizan ambos puños sin mostrar una clara preferencia por uno en particular. También esquivan más de lo necesario. En este caso, Mac no parece identificar correctamente cuándo golpear arriba o abajo, lo que provoca que Glass Joe bloquee y Mac se quede sin corazones, obligándolo a esquivar para recuperarlos y poder seguir atacando. Sin embargo, también esquivan cuando tienen corazones disponibles, lo cual es innecesario contra este oponente. En muchos casos, los agentes llegan a la segunda e incluso tercera ronda, ya que no siempre logran derrotar a su oponente dentro del tiempo que dura la primera ronda. No se observan diferencias claras entre los comportamientos de ambos agentes (ver Apéndice 30 y 31).

Ambos escenarios utilizados en este juego son bastante similares, aunque el escenario contra Von Kaiser es más difícil que el de Glass Joe. En ambos casos, el contrincante siempre esquivará hacia los lados y se posicionará frente al jugador antes de atacar. Gracias a esta consistencia en el comportamiento del contrincante, los modelos son capaces de generalizar y lograr victorias en el escenario donde no fueron entrenados. El desempeño es mejor cuando el agente fue entrenado en el escenario más difícil y luego evaluado en el más sencillo.

Resumen del Análisis para Punch Out!!

En general, los resultados obtenidos en los escenarios de combate contra Glass Joe y Von Kaiser muestran que los agentes estático y decaído lograron un desempeño superior al agente aleatorio en los cuatro ámbitos analizados: porcentaje de victorias, recompensas acumuladas, cantidad de *timesteps* y vida al final de los episodios.

Se observaron diferencias entre los modelos de recompensa estática y decaída, tanto en el proceso de aprendizaje como en los resultados obtenidos. El modelo de recompensa decaída mostró una tendencia más consistente en el aprendizaje y un mejor equilibrio entre exploración y explotación.

En el escenario de entrenamiento con Glass Joe (el oponente más sencillo), no se evidenció una superioridad clara entre los modelos de recompensa estática y decaída, ya fuera al enfrentarse al mismo escenario o a un escenario distinto. Sin embargo, al entrenar a los agentes contra Von Kaiser (el oponente más difícil), el modelo decaído demostró un mejor desempeño tanto en el mismo escenario de entrenamiento como en un escenario nuevo, indicando una mayor capacidad de generalización.

La capacidad del modelo decaído para mantener un aprendizaje estable y evitar comportamientos subóptimos como la "cobardía" indica que la propuesta de recompensas adaptativas fue efectiva en este contexto.

5.2. The Legend of Zelda para NES

5.2.1. Entrenamiento con Aquamentus

La Figura 5.7 muestra las recompensas promedio al final de cada episodio. Se observa que tanto la curva de aprendizaje del entrenamiento decaído como la del entrenamiento estático poseen una forma muy similar.

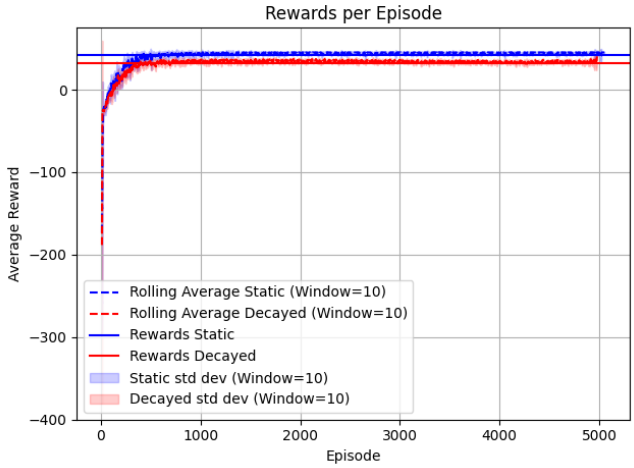


Figura 5.7. Entrenamiento con Aquamentus: Gráfico de recompensa promedio por episodio.

La Figura 5.8 muestra el promedio de *timesteps* que duraron los episodios. En este caso, ambas curvas presentan un comportamiento prácticamente idéntico, con fluctuaciones mínimas y un tramo estable que se mantiene consistente a lo largo del entrenamiento. El modelo estático obtuvo un promedio de 167.1 *timesteps* por episodio, mientras que el modelo decaído alcanzó un promedio de 170.7, con una diferencia de apenas 2.1%.

Si bien se observa que el valor máximo alcanzado por la curva del modelo decaído es ligeramente inferior al del modelo estático durante los primeros episodios, ambas curvas convergen rápidamente a un mismo rango.

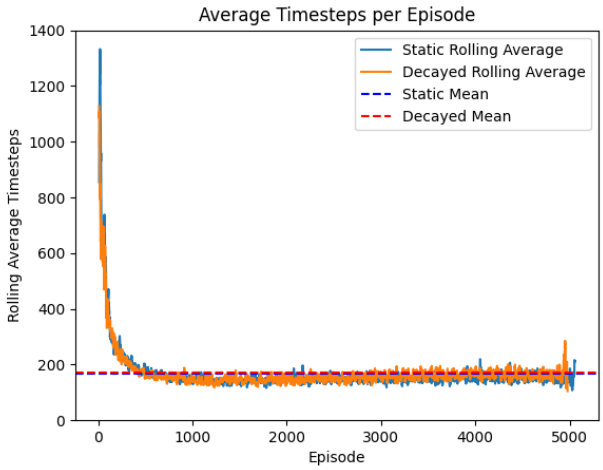


Figura 5.8. Entrenamiento con Aquamentus: Gráfico de timesteps promedio por episodio.

La Figura 5.9 muestra el porcentaje de victorias. Ambas curvas se ven muy similares, se puede ver que, en promedio, el entrenamiento decaído tuvo un porcentaje de victoria mínimamente mayor que el entrenamiento estático.

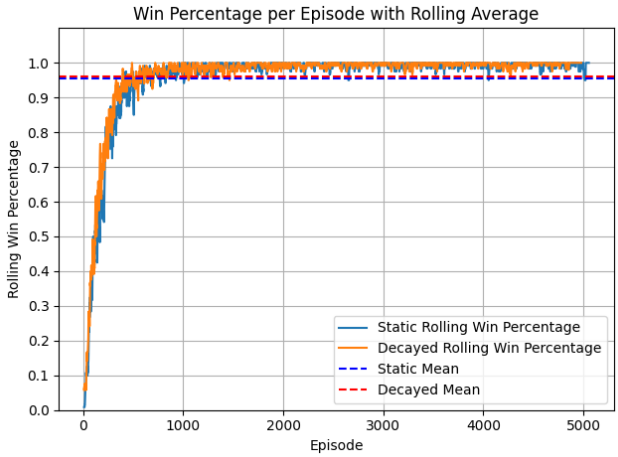


Figura 5.9. Entrenamiento con Aquamentus: Gráfico de porcentaje de victoria promedio por episodio.

Estos gráficos indican que, en este escenario, el aprendizaje no es significativamente mejor cuando se aplica un entrenamiento decaído. No se observa indicación de menor cobardía ni masterización defensiva en el entrenamiento.

Una posible explicación para esta falta de diferencia podría ser que el entorno específico de este escenario no ofrece suficientes oportunidades para que las recompensas decaídas influyan en el comportamiento del agente. Si el combate no presenta una variedad significativa de situaciones críticas o si las acciones óptimas son evidentes desde el inicio, el agente podría alcanzar un comportamiento efectivo rápidamente, sin necesidad de ajustes adicionales en las recompensas por derrota o victoria.

Además, es posible que el nivel de dificultad del oponente no genere el tipo de presión que motive al agente a desarrollar estrategias defensivas complejas o a mostrar una valentía activa. En un escenario más simple, la recompensa estática podría ser suficiente para guiar al agente hacia un buen rendimiento, minimizando el impacto positivo de la recompensa decaída.

Con los modelos estático y decaído ya entrenados, se evaluó su desempeño en dos escenarios: la batalla contra Aquamentus y la batalla contra Manhandla. En este caso, el primer contrincante representa un desafío más sencillo que el segundo.

5.2.1.1. Evaluación contra Aquamentus

A continuación, se muestra el rendimiento de los agentes estático y decaído, además de un agente aleatorio, cuando juegan en el mismo escenario en el que fueron entrenados.

Tabla 5.2.1. Resultados del agente entrenado al enfrentarse a Aquamentus: Promedio de Recompensas Acumuladas, *Timesteps* y Vida Final por Episodio (Desviación Estándar entre paréntesis).

Agente	Recompensa Acumulada	*Timesteps*	Vida Final por Episodio
Aleatorio	-30.4813 (2.290607)	768.49 (2061.561532)	0 (0)
Estático	42.1628 (2.353653)	162.15 (133.655481)	10.51 (1.292246)
Decaído	41.5585 (2.407223)	170.57 (126.616212)	9.82 (1.492515)

Dados estos resultados, no se puede concluir que el modelo decaído tenga un mejor rendimiento que el modelo estático (valor p = 0.0742); sin embargo, ambos modelos superan al agente aleatorio (ver Apéndice 32).

No se puede concluir que el modelo estático haya tardado menos que el modelo decaído (valor p = 0.6479); sin embargo, ambos modelos fueron más rápidos que el agente aleatorio (ver Apéndice 33).

Se puede concluir que el modelo estático termina con más vida al finalizar los episodios que el modelo decaído; además, ambos logran finalizar los episodios con una vida final considerablemente mayor que la del agente aleatorio (ver Apéndice 34).

Porcentaje de victoria

El agente aleatorio no ganó ningún juego (0%), por otro lado, tanto el modelo estático como el decaído triunfaron el 100% de las veces. Dados estos resultados y su interpretación estadística en base a los valores p, dados por la prueba z, no se puede concluir que los porcentajes de victoria de los modelos estático y decaído tengan una diferencia significativa (valor p = 1), pero sí que ambos ganan significativamente más veces que el agente aleatorio.

Como complemento, se calcularon intervalos de confianza binomiales al 95% utilizando el método descrito en [17]. Para el agente aleatorio, el intervalo fue (0.0%, 3.6%), mientras que para ambos modelos entrenados fue (96.3%, 100%).

Análisis cualitativo

Al observar a los agentes jugar, se puede notar que el agente estático tiene la estrategia de avanzar de manera diagonal en dirección a la esquina superior derecha para acercarse a la cabeza de Aquamentus por su lado izquierdo y de esta manera atacar con la espada hasta derrotarlo (ver Apéndice 35). El agente decaído también avanza en dirección diagonal hacia arriba a la derecha, pero en vez de ir hacia la esquina, se posiciona detrás del enemigo, donde no le pueden llegar las bolas de fuego, pero si puede perder vida al entrar en contacto con el oponente, desde esta posición puede golpear al oponente, pero no esquivarlo (ver Apéndice 36). Por otro lado, el agente aleatorio tiene problemas para salir

de la entrada de la habitación, en esa posición no puede atacar al oponente, pero tampoco puede ser atacado, por lo que los episodios terminan por llegar al máximo de tiempo o porque accidentalmente el agente retrocedió a la habitación anterior (ver Apéndice 37).

5.2.1.2. Evaluación contra Manhandla

A continuación, se muestra el rendimiento de los agentes estático y decaído, además de un agente aleatorio, cuando juegan en un escenario que nunca han visto antes, la batalla contra Manhandla. Esta evaluación pone a prueba la capacidad de generalización de los agentes entrenados.

Tabla 5.2.2. Resultados del agente entrenado al enfrentarse a Manhandla: Promedio de Recompensas Acumuladas, *Timesteps* y Vida Final por Episodio (Desviación Estándar entre paréntesis).

Agente	Recompensa Acumulada	*Timesteps*	Vida Final por Episodio
Aleatorio	-16.6224 (33.467936)	1604.92 (435.442067)	1.67 (3.193916)
Estático	-45.1891 (1.545293)	643.08 (67.713910)	0 (0)
Decaído	-45.4022 (1.434730)	633.93 (36.293871)	0 (0)

Dados estos resultados, no se puede concluir que el modelo decaído haya obtenido un mejor rendimiento que el modelo estático (valor p = 0.3134), pero sí que ambos tuvieron significativamente peor desempeño que el agente aleatorio (ver Apéndice 38).

No se puede concluir que el modelo decaído tardó menos que el modelo estático (valor p = 0.2351), pero sí que ambos modelos fueron considerablemente más rápidos que el agente aleatorio (ver Apéndice 39).

Tampoco se puede concluir que existe una diferencia significativa entre el modelo estático y el decaído en el ámbito de vida final por episodio (valor p = 1), pero sí que el agente aleatorio consigue un mejor promedio (ver Apéndice 40).

Porcentaje de victoria

El agente aleatorio ganó un 28% de los juegos, mientras que tanto el modelo estático como el decaído no ganaron ningún juego (0%). Dados estos resultados y su interpretación estadística en base a los valores p, dados por la prueba z, no se puede concluir que el modelo estático y el modelo decaído tengan una diferencia significativa entre sus porcentajes de victoria (valor p = 1); sin embargo, si se puede concluir que el agente aleatorio tiene un mejor porcentaje de victoria.

Se calcularon además intervalos de confianza binomiales al 95% utilizando el método descrito en [17]. Para el agente aleatorio, el intervalo fue (19.0%, 38.2%), mientras que para ambos modelos entrenados fue (0.0%, 3.6%).

Análisis cualitativo

Al observar a los agentes jugar, se puede notar que el agente estático sigue la misma estrategia que en el caso anterior, avanza de manera diagonal en dirección a la esquina superior derecha, pero esto no le sirve contra este oponente, ya que solo le puede pegar cuando el oponente se acerque a él. A pesar de que esto ocurre y logra quitarle vida al oponente, no se preocupa de esquivar las bolas de fuego, por lo que muere antes de derrotarlo (ver Apéndice 41). En este caso, el agente decaído se comporta de la misma manera que el agente estático (ver Apéndice 42). Por otro lado, el agente aleatorio tiene un mejor rendimiento, ya que al igual que el oponente, este se mueve por toda la habitación, teniendo más posibilidades de golpear al oponente, por lo que, a pesar de que no esquive las bolas de fuego, en ocasiones logra derrotarlo antes de que el oponente lo derrote a él (ver Apéndice 43).

5.2.2. Entrenamiento con Manhandla

En la Figura 5.10, se pueden observar las recompensas promedio al final de cada episodio. A diferencia del entrenamiento contra Aquamentus, en el que ambas curvas mostraban un aprendizaje muy similar, en este caso se aprecia una diferencia más clara entre los modelos. La versión con recompensa final decaída presenta una tendencia positiva, pero en general recibe menores recompensas acumuladas en comparación con la versión estática, que muestra valores más altos a lo largo del entrenamiento y una tendencia positiva más pronunciada.

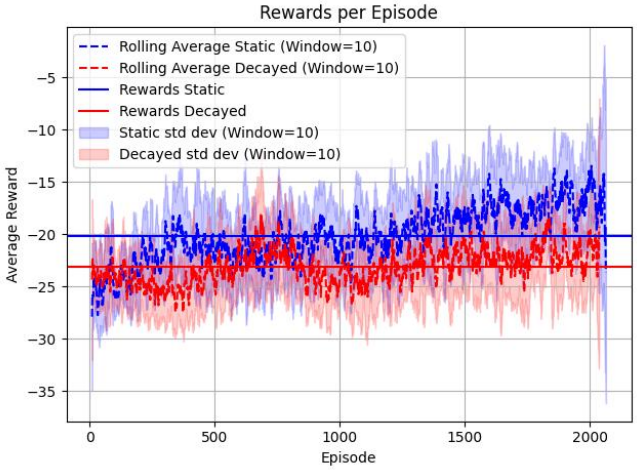


Figura 5.10. Entrenamiento con Manhandla: Gráfico de recompensa promedio por episodio.

La Figura 5.11 muestra el promedio de *timesteps* que duraron los episodios. En este caso, ninguno de los entrenamientos presenta una diferencia considerable entre el inicio y el final del proceso de entrenamiento, y las curvas no siguen una tendencia clara de mejora en la duración de los episodios. El modelo decaído obtuvo un promedio de 399.9 *timesteps* por episodio, mientras que el modelo estático alcanzó un promedio ligeramente superior de 410.5, lo que representa una diferencia de aproximadamente 2.6%. Aunque esta diferencia

no es significativa desde el punto de vista práctico, sí se observa que el entrenamiento con decaimiento presenta menor variabilidad a lo largo del tiempo, con oscilaciones más controladas en la duración de los episodios. Por el contrario, la curva del modelo estático muestra mayor dispersión en los *timesteps*, lo que podría reflejar una política menos estable o mayores fluctuaciones estratégicas durante el entrenamiento.

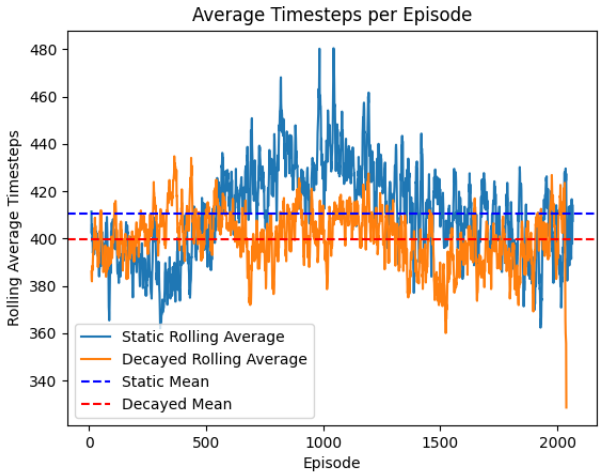


Figura 5.11. Entrenamiento con Manhandla: Gráfico de timesteps promedio por episodio.

La Figura 5.12 muestra el porcentaje de victorias alcanzadas por cada modelo. La versión con recompensa estática logra un porcentaje de victorias más alto en comparación con la versión decaída, manteniendo una tendencia positiva más pronunciada a lo largo del entrenamiento. La diferencia sugiere que, en este escenario, la penalización decaída no contribuye significativamente a mejorar la toma de decisiones del agente, sino que podría estar afectando su capacidad de optimizar estrategias de victoria. Esto puede ser debido a la dificultad del escenario, en este caso una victoria es tan difícil de conseguir, que aunque la victoria no sea “buena”, es importante premiarla para que el agente pueda aprender.

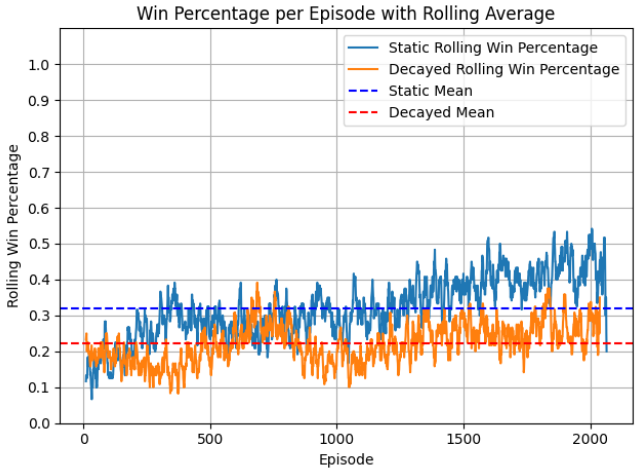


Figura 5.12. Entrenamiento con Manhandla: Gráfico de porcentaje de victoria promedio por episodio.

Estos gráficos indican que, en este caso particular, el aprendizaje es más efectivo cuando se utiliza un esquema de recompensa estático. A diferencia de lo observado en el juego anterior, en este escenario no se observa una clara indicación de menor cobardía ni una mejor adaptación defensiva en el entrenamiento decaído. En cambio, el modelo estático muestra un mejor desempeño en términos de recompensa acumulada y tasa de victorias.

Con los modelos estático y decaído ya entrenados, se evaluó su desempeño en dos escenarios: la batalla contra Manhandla y la batalla contra Aquamentus.

5.2.2.1. Evaluación contra Manhandla

A continuación, se muestra el rendimiento de los agentes estático y decaído, además de un agente aleatorio, cuando juegan en el mismo escenario en el que fueron entrenados.

Tabla 5.2.3. Resultados del agente entrenado al enfrentarse a Manhandla: Promedio de Recompensas Acumuladas, *Timesteps* y Vida Final por Episodio (Desviación Estándar entre paréntesis).

Agente	Recompensa Acumulada	*Timesteps*	Vida Final por Episodio
Aleatorio	-27.7008 (30.378154)	1622.56 (373.550219)	1.25 (2.779838)
Estático	-5.5042 (34.791403)	407.19 (118.915491)	2.37 (3.579539)
Decaído	-12.9666 (34.237116)	385.57 (98.363637)	1.74 (2.890052)

Dados estos resultados, no se puede concluir que el modelo decaído haya obtenido un mejor rendimiento que el modelo estático (valor p = 0.1279), pero sí que ambos superaron al agente aleatorio (ver Apéndice 44).

No se puede concluir que el modelo decaído tardó menos que el modelo estático (valor p = 0.1628), pero sí que ambos modelos fueron considerablemente más rápidos que el agente aleatorio (ver Apéndice 45).

Tampoco se puede concluir que en el ámbito de vida final por episodio exista una diferencia significativa entre los modelos estático y decaído (valor p = 0.1724) ni entre el modelo decaído y el agente aleatorio (valor p = 0.2232), pero sí que el modelo estático superó al agente aleatorio (ver Apéndice 46).

Porcentaje de victoria

El agente aleatorio ganó el 21% de los juegos, el modelo estático el 41% y el modelo decaído el 33%. Dados estos resultados y su interpretación estadística en base a los valores p, dados por la prueba z, no se puede concluir que exista una diferencia significativa entre los porcentajes de victoria de los modelos estático y decaído (valor p = 0.242) ni modelo decaído y agente aleatorio (valor p = 0. 0561), pero sí que el modelo estático superó al agente aleatorio.

Adicionalmente, se calcularon intervalos de confianza binomiales al 95% utilizando el método descrito en [17]. El agente aleatorio presentó un intervalo de (13.6%, 30.3%), el modelo decaído (23.4%, 43.8%) y el modelo estático (31.2%, 51.6%).

Análisis cualitativo

Al observar a los agentes jugar, se puede notar que el agente estático utiliza la estrategia de mantenerse pegado a las paredes, de esta manera no le pueden llegar bolas de fuego por el lado de la pared a la que se encuentra pegado, pero solo puede golpear al oponente cuando se acerca a la pared donde se encuentra, por lo que en muchas ocasiones muere antes de derrotarlo (ver Apéndice 47). En el caso del agente decaído, este se mantiene cerca de las paredes, pero no siempre pegado a ellas, por lo que tiene más posibilidades de golpear al oponente (ver Apéndice 48).

5.2.2.2. Evaluación contra Aquamentus

A continuación, se muestra el rendimiento de los agentes estático y decaído, además de un agente aleatorio, cuando juegan en un escenario que nunca han visto antes, la batalla contra Aquamentus. Esta evaluación pone a prueba la capacidad de generalización de los agentes entrenados.

Tabla 5.2.4. Resultados del agente entrenado al enfrentarse a Aquamentus: Promedio de Recompensas Acumuladas, *Timesteps* y Vida Final por Episodio (Desviación Estándar entre paréntesis).

Agente	Recompensa Acumulada	*Timesteps*	Vida Final por Episodio
Aleatorio	-30.5015 (2.271626)	616.26 (1423.645325)	0 (0)
Estático	-17.0579 (26.693199)	2144.21 (1953.599213)	1.31 (2.914430)
Decaído	-33.5933 (21.392376)	1225.06 (1397.744196)	0.47 (1.956809)

Dados estos resultados, se puede concluir que el modelo estático obtuvo un mejor rendimiento que el modelo decaído y que el agente aleatorio. Por otro lado, no se puede concluir que el modelo decaído haya logrado un mejor rendimiento que el agente aleatorio (valor p = 0.1522) (ver Apéndice 49).

Se puede concluir que el agente aleatorio tardó menos que el modelo decaído, y que este tardó menos que el modelo estático (ver Apéndice 50).

Se puede concluir que el modelo estático termina con mayor vida al final de los episodios que el modelo decaído y que ambos obtuvieron un mejor promedio de vida al final del episodio en comparación con el agente aleatorio (ver Apéndice 51).

Porcentaje de victoria

El agente aleatorio no ganó ningún juego (0%), mientras que el modelo estático ganó el 19% de los juegos y el modelo decaído el 7%. Dados estos resultados y su interpretación estadística en base a los valores p, dados por la prueba z, se puede concluir que el porcentaje de victorias del modelo estático es significativamente superior al del modelo decaído, y que ambos modelos superan al agente aleatorio.

Sin embargo, los intervalos de confianza binomiales al 95% calculados mediante el método descrito en [17] revelaron **ligeras superposiciones**: entre el agente aleatorio (0.0%, 3.6%) y el modelo decaído (2.9%, 14.0%), y entre este último y el modelo estático (11.4%, 28.7%). Esta discrepancia entre los resultados de la prueba z y los intervalos de confianza refleja que las conclusiones de significancia deben interpretarse con precaución. Si bien los resultados sugieren una ventaja del modelo estático sobre los demás, esta no puede considerarse concluyente desde todos los enfoques estadísticos.

Análisis cualitativo

Al observar a los agentes jugar, se puede notar que el agente estático vuelve a utilizar la estrategia de mantenerse pegado a las paredes, de esta forma, cuando se acerca a la pared donde se encuentra el oponente, puede derrotarlo, y usualmente se acerca a esa pared primero (ver Apéndice 52). En el caso del agente decaído, este nuevamente se mantiene cerca de las paredes, prefiriendo acercarse primero a la del enemigo, lo que le permite enfrentarse a él directamente (ver Apéndice 53).

Los escenarios utilizados en este juego son muy diferentes, donde el escenario contra Manhandla es significativamente más difícil que el de Aquamentus. En el caso de Aquamentus, el oponente siempre se moverá lentamente por el eje x de la pantalla hacia un lado y otro dentro de un espacio determinado, mientras que en el caso de Manhandla, el oponente se mueve rápidamente a través de toda la pantalla de manera aleatoria. Debido a esta diferencia en el comportamiento del contrincante, los modelos entrenados en el escenario de Aquamentus no son capaces de generalizar y lograr victorias en el escenario donde no fueron entrenados. Por otro lado, los modelos entrenados en el escenario contra Manhandla si logran generalizar y conseguir victorias en un escenario diferente, pero la tasa de victoria es baja.

5.3. Mega Man X para SNES

5.3.1. Entrenamiento con Chill Penguin

La Figura 5.13 muestra las recompensas promedio al final de cada episodio. Se observa que la curva de aprendizaje del entrenamiento estático crece rápidamente hasta alcanzar

un máximo y mantenerse en ese nivel. En contraste, el entrenamiento decaído exhibe una tendencia positiva continua.

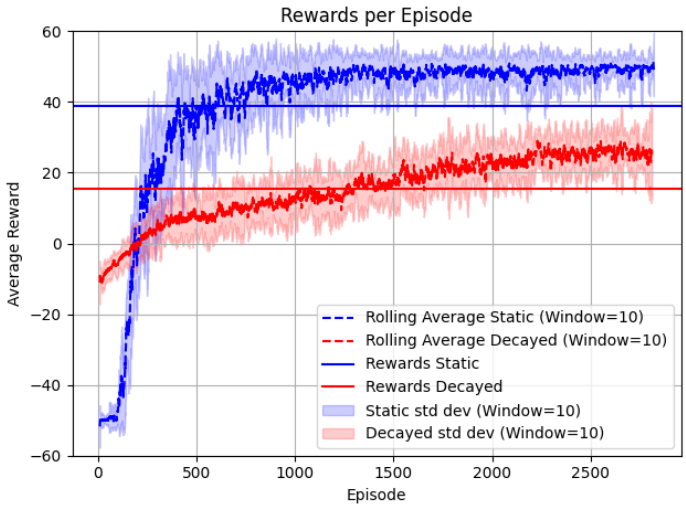


Figura 5.13. Entrenamiento con Chill Penguin: Gráfico de recompensa promedio por episodio.

La Figura 5.14 muestra el promedio de *timesteps* que duraron los episodios. En este caso, ambas curvas presentan un comportamiento muy similar a lo largo de todo el entrenamiento, sin diferencias significativas ni visuales ni en los valores promedio. El modelo estático tuvo un promedio de 1189.5 *timesteps* por episodio, mientras que el modelo decaído alcanzó 1193.9, con una diferencia mínima de apenas 0.37%. Ambas curvas comienzan con una fase de adaptación algo inestable y luego convergen gradualmente hacia un mismo nivel estable, sin que se observe ventaja clara de un modelo sobre el otro. Esta similitud en la duración de los episodios sugiere que, en este caso, ambos modelos desarrollaron políticas de comportamiento equivalentes en términos de eficiencia temporal.

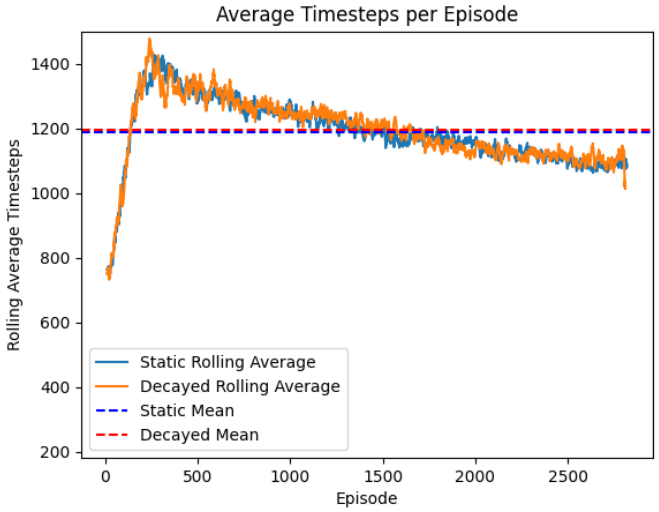


Figura 5.14. Entrenamiento con Chill Penguin: Gráfico de timesteps promedio por episodio.

La Figura 5.15 muestra el porcentaje de victorias. Ambas curvas presentan un comportamiento similar; en este caso, ninguno de los entrenamientos logró alcanzar consistentemente un 100% de victorias al finalizar.

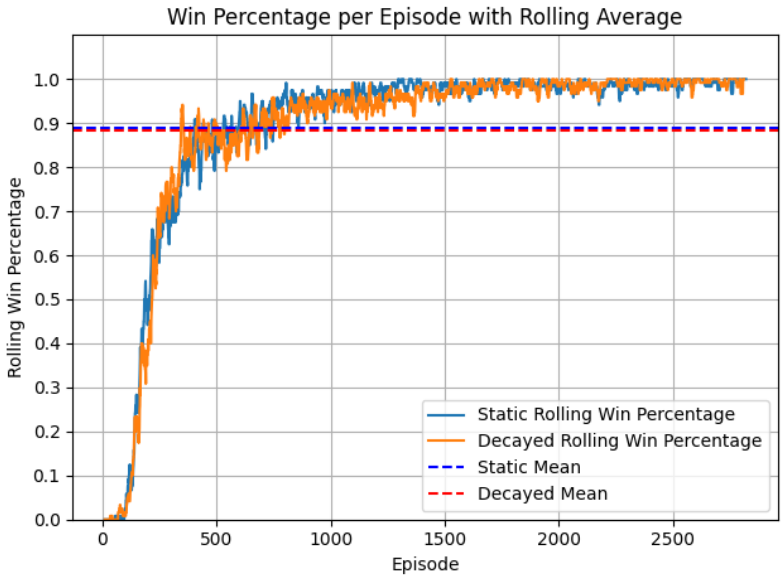


Figura 5.15. Entrenamiento con Chill Penguin: Gráfico de porcentaje de victoria promedio por episodio.

Estos gráficos indican que, en este escenario, el aprendizaje es más estable y eficiente cuando se aplica un entrenamiento decaído. La curva de aprendizaje del entrenamiento decaído es superior (Figura 5.13), lo que sugiere que este entrenamiento está promoviendo un aprendizaje más efectivo y posiblemente ayudando al agente a descubrir nuevas estrategias o comportamientos que le permiten mejorar su rendimiento con el tiempo. Sin embargo, no se observa que este entrenamiento complete los episodios más rápidamente (Figura 5.14) ni que tenga un mayor porcentaje de victorias (Figura 5.15).

Se observa una actitud menos cobarde en la curva del entrenamiento decaído (Figura 5.13), que presenta una curva de aprendizaje más eficiente al explorar mejores estrategias que, por tener un comportamiento más conservador, el entrenamiento estático habría evitado. No se observa un comportamiento que respalde una maestría en defensa.

Con los modelos estático y decaído ya entrenados, se evaluó su desempeño en tres escenarios: la batalla contra Chill Penguin, la batalla contra Spark Mandrill y la batalla contra Spark Mandrill utilizando el arma que representa su debilidad. En este caso, el primer contrincante presenta un desafío más sencillo que el segundo cuando se utiliza el arma inicial del juego, mientras que al usar el arma *Shotgun Ice* contra Spark Mandrill, el escenario se vuelve considerablemente más fácil.

5.3.1.1. Evaluación contra Chill Penguin

A continuación, se muestra el rendimiento de los agentes estático y decaído, además de un agente aleatorio, cuando juegan en el mismo escenario en el que fueron entrenados.

Tabla 5.3.1. Resultados del agente entrenado al enfrentarse a Chill Penguin: Promedio de Recompensas Acumuladas, *Timesteps* y Vida Final por Episodio (Desviación Estándar entre paréntesis).

Agente	Recompensa Acumulada	*Timesteps*	Vida Final por Episodio
Aleatorio	-48.15 (16.767454)	3091.6 (690.021840)	0.05 (0.357071)
Estático	50.07 (9.944099)	1099.02 (128.148194)	24.95 (6.651879)
Decaído	48.11 (21.558244)	1111.41 (146.244391)	25.19 (6.966628)

Dados estos resultados, no se puede concluir que el modelo estático obtuvo un mejor rendimiento que el modelo decaído (valor p = 0.41), pero sí que ambos obtuvieron un mejor rendimiento que el agente aleatorio (ver Apéndice 54).

No se puede concluir que el modelo estático haya tardado menos que el modelo decaído (valor p = 0.5247); sin embargo, sí se puede concluir que ambos modelos tardaron menos que el agente aleatorio (ver Apéndice 55).

Tampoco se puede concluir que exista una diferencia significativa entre los promedios de vida al final del episodio de los modelos estático y decaído (valor p = 0.8035), pero sí que ambos modelos superaron al agente aleatorio (ver Apéndice 56).

Porcentaje de victoria

El agente aleatorio ganó el 2% de sus juegos, mientras que el modelo estático ganó el 100% de sus juegos y el modelo decaído el 98%. Dados estos resultados y su interpretación estadística en base a los valores p, dados por la prueba z, no se puede concluir que exista una diferencia significativa entre los porcentajes de victoria de los modelos estático y decaído (valor p = 0.1556), pero sí que ambos modelos superaron al agente aleatorio.

Como complemento, se calcularon intervalos de confianza binomiales al 95% utilizando el método descrito en [17]. El agente aleatorio tuvo un intervalo de (0.2%, 7.0%), el modelo decaído (93.0%, 99.8%) y el modelo estático (96.3%, 100%).

Análisis cualitativo

Al observar a los agentes jugar, se puede ver que ambos agentes adoptan la misma estrategia: mantenerse alejados del oponente, acercándose a la pared opuesta, y esquivar los proyectiles que lanza el oponente saltando o colgándose de la pared (ver Apéndice 57

y 58). En comparación, el agente aleatorio no intenta esquivar, por lo que raramente logra una victoria (ver Apéndice 59).

5.3.1.2. Evaluación contra Spark Mandrill y arma base

Los siguientes son los resultados cuando los agentes entrenados con el escenario contra Chill Penguin se enfrentan a un escenario que nunca han visto antes, la batalla contra Spark Mandrill utilizando el arma base. Esta evaluación pone a prueba la capacidad de generalización de los agentes entrenados.

Tabla 5.3.2. Resultados del agente entrenado al enfrentarse a Spark Mandrill y arma base: Promedio de Recompensas Acumuladas, *Timesteps* y Vida Final por Episodio (Desviación Estándar entre paréntesis).

Agente	Recompensa Acumulada	*Timesteps*	Vida Final por Episodio
Aleatorio	-50.16 (4.608080)	1843.18 (248.213794)	0 (0)
Estático	-50.21 (4.329654)	531.58 (76.962352)	0 (0)
Decaído	-50.25 (3.811496)	338.75 (49.525221)	0 (0)

Dados estos resultados, no se puede concluir que exista una diferencia significativa entre los promedios de recompensas acumuladas de los modelos estático y decaído (valor p = 0.9448), entre el modelo estático y el agente aleatorio (valor p = 0.9371), ni entre el modelo decaído y el agente aleatorio (valor p = 0.8805) (ver Apéndice 60).

Dados estos resultados, se puede concluir que el modelo decaído tardó menos que el modelo estático, y que este, a su vez, tardó menos que el agente aleatorio (ver Apéndice 61).

Dados estos resultados, no se puede concluir que exista una diferencia significativa entre los promedios de vida al final de cada episodio de los modelos estático, decaído y agente aleatorio (valor p = 1 en todos los casos) (ver Apéndice 62).

Porcentaje de victoria

Ni el agente aleatorio ni los modelos estático y decaído ganaron algún juego (0%). Dados estos resultados y su interpretación estadística en base a los valores p, dados por la prueba z, no se puede concluir que exista una diferencia significativa entre los porcentajes de victoria de los modelos estático, decaído y agente aleatorio (valor p = 1 en todos los casos).

Adicionalmente, se calcularon intervalos de confianza binomiales al 95% utilizando el método descrito en [17]. Para los tres agentes, el intervalo fue idéntico: (0.0%, 3.6%). Esto refleja que, aunque no se observaron victorias en la muestra, no se puede afirmar con certeza que sus tasas reales de victoria sean exactamente cero, solo que probablemente son muy bajas.

Análisis cualitativo

Al observar a los agentes jugar, se puede observar que el agente estático aplica la misma estrategia que aprendió en el escenario con Chill Penguin: intentar alejarse del oponente acercándose a la pared opuesta. Sin embargo, lo hace solo cuando el oponente está en el suelo; cuando Spark Mandrill está colgando del techo, el agente no se aleja de él, lo que permite que el oponente caiga sobre el agente y le cause daño (ver Apéndice 63). Por otro lado, el agente decaído se acerca a la pared izquierda y se mantiene disparando desde esa posición (ver Apéndice 64). Como resultado, este agente muere más rápido que el agente estático, que esquiva más al moverse de una pared a otra. En este caso, es más difícil ganar si el agente solo se mantiene en el suelo, por lo que la versión aleatoria tiene un peor rendimiento en este escenario, en comparación al anterior (ver Apéndice 65).

5.3.1.3. Evaluación contra Spark Mandrill y Shotgun Ice

Los siguientes son los resultados cuando los agentes se enfrentan nuevamente a la batalla contra Spark Mandrill, pero esta vez utilizando el arma Shotgun Ice, la cual es la debilidad de Spark Mandrill, de esta manera, la dificultad del escenario disminuye.

Tabla 5.3.3. Resultados del agente entrenado al enfrentarse a Spark Mandrill y *Shotgun Ice*: Promedio de Recompensas Acumuladas, *Timesteps* y Vida Final por Episodio (Desviación Estándar entre paréntesis).

Agente	Recompensa Acumulada	*Timesteps*	Vida Final por Episodio
Aleatorio	50.18 (8.715939)	925.89 (73.542354)	19.51 (5.785318)
Estático	50.24 (7.606734)	260.3 (23.278960)	21.66 (5.122929)
Decaído	44.21 (31.489139)	242.12 (18.558168)	15.74 (7.465413)

Dados estos resultados, no se puede concluir que exista una diferencia significativa entre los promedios de recompensas acumuladas de los modelos estático y decaído (valor p = 0.0642), entre el modelo estático y el agente aleatorio (valor p = 0.9587), ni entre el modelo decaído y el agente aleatorio (valor p = 0.0692) (ver Apéndice 66).

Se puede concluir que el modelo decaído tardó menos que el modelo estático, y que este, a su vez, tardó menos que el agente aleatorio (ver Apéndice 67).

También se puede concluir que el mejor promedio lo obtuvo el modelo estático, seguido por el agente aleatorio y, finalmente, el modelo decaído (ver Apéndice 68).

Porcentaje de victoria

El agente aleatorio y el modelo estático ganaron el 100% de sus juegos, mientras que el modelo decaído ganó el 98%. Dados estos resultados y su interpretación estadística en base a los valores p, dados por la prueba z, no se puede concluir que exista una diferencia

significativa entre los porcentajes de victoria del modelo estático y el agente aleatorio (valor p = 1), ni entre ellos y el modelo decaído (valor p = 0.1556).

Los intervalos de confianza binomiales al 95% calculados mediante el método descrito en [17] fueron (96.3%, 100%) para el modelo estático y el agente aleatorio, y (93.0%, 99.8%) para el modelo decaído.

Análisis cualitativo

El arma utilizada en este escenario es *Shotgun Ice*, que dispara cristales de hielo que vuelan rápidamente y, al golpear cualquier obstáculo, rebotan y se dividen en cinco proyectiles en dirección de gran angular. Esta arma causa a Spark Mandrill tres veces el daño del arma base y además lo congela por unos momentos. Por esta razón, el agente aleatorio es el que tiene mejor rendimiento, ya que, aunque el agente dispare en cualquier dirección, existe una gran probabilidad de que le acierte al oponente, congelándolo e impidiendo que lo golpee (ver Apéndice 71). El agente estático muestra el mismo comportamiento que se describió en el escenario anterior; sin embargo, al acertar y congelar al oponente, esto le permite seguir causando daño sin que el oponente pueda atacarlo (ver Apéndice 69). En el caso del agente decaído, al igual que en el escenario anterior, se mantiene cerca de la pared izquierda de la sala disparando, a pesar de congelar al oponente y quitarle vida, en algunos casos el oponente llega a la pared izquierda y, al no alejarse, el agente termina perdiendo (ver Apéndice 70).

5.3.2. Entrenamiento con Spark Mandrill y arma base

La Figura 5.16 muestra las recompensas promedio al final de cada episodio. Se observa que la curva de aprendizaje del entrenamiento estático crece rápidamente hasta alcanzar un máximo y mantenerse en ese nivel. En contraste, el entrenamiento decaído exhibe una tendencia positiva continua.

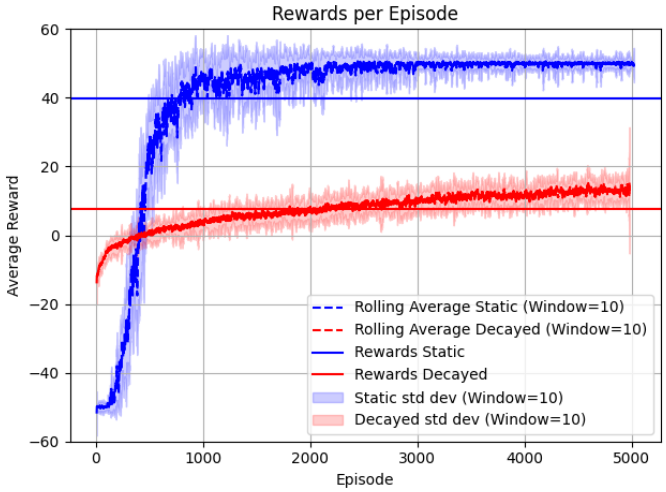


Figura 5.16. Entrenamiento con Spark Mandrill y arma base: Gráfico de recompensa promedio por episodio.

La Figura 5.17 muestra el promedio de *timesteps* que duraron los episodios. Aunque ambas curvas convergen hacia un mismo nivel estable, se observa una diferencia en la forma de evolución a lo largo del entrenamiento: el modelo estático presenta un descenso más abrupto durante las primeras fases, mientras que el modelo decaído desciende de forma más gradual y continua.

En cuanto al desempeño promedio, el modelo estático obtuvo un valor de 664.6 *timesteps* por episodio, mientras que el decaído alcanzó 671.9, con una diferencia relativa de solo 1.1%. Esta diferencia es mínima y no tiene relevancia práctica directa, aunque el patrón más estable del modelo decaído podría indicar una curva de aprendizaje más progresiva y consistente.

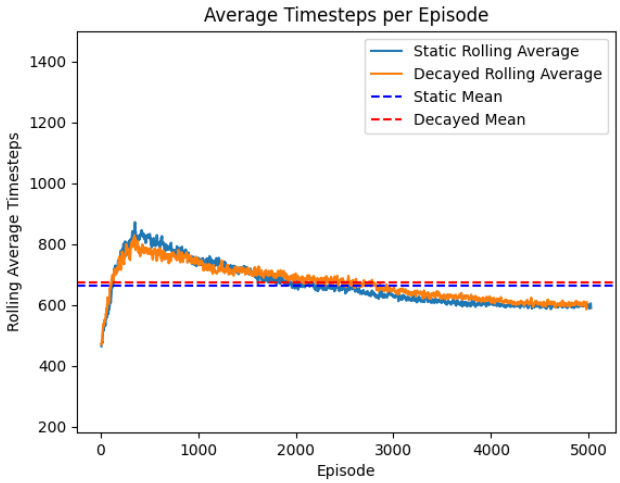


Figura 5.17. Entrenamiento con Spark Mandrill y arma base: Gráfico de timesteps promedio por episodio.

La Figura 5.18 muestra el porcentaje de victorias. Ambas curvas presentan un comportamiento similar; en este caso, ninguno de los entrenamientos logró alcanzar consistentemente un 100% de victorias al finalizar.

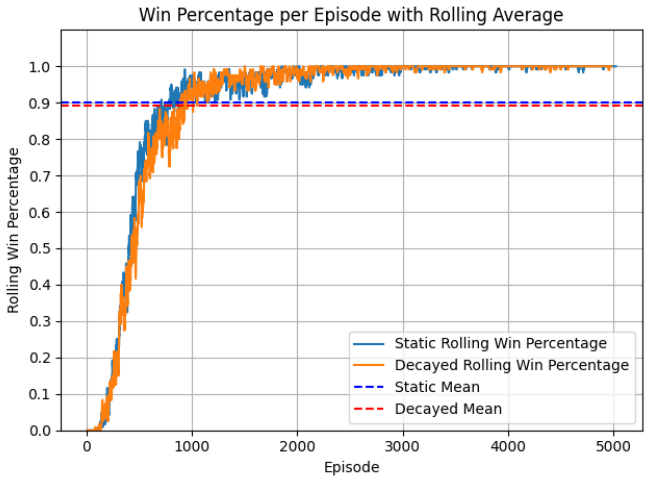


Figura 5.18. Entrenamiento con Spark Mandrill y arma base: Gráfico de porcentaje de victoria promedio por episodio.

Estos gráficos sugieren que, en este escenario, el entrenamiento decaído contribuye a un aprendizaje más consistente y equilibrado. La curva del entrenamiento decaído indica que este enfoque fomenta un desarrollo más sostenido, permitiendo al agente explorar nuevas estrategias o comportamientos para mejorar su rendimiento con el tiempo (Figura 5.16). No obstante, no se observa una ventaja en cuanto a la rapidez con la que se completan los episodios (Figura 5.17) ni en el porcentaje de victorias alcanzado (Figura 5.18).

Además, se aprecia una menor tendencia a la cobardía en el entrenamiento decaído, lo que se ve en una mayor exploración de estrategias que el entrenamiento estático evita por miedo (figura 5.16). Sin embargo, no se identifica un comportamiento que demuestre una masterización defensiva.

5.3.2.1. Evaluación contra Spark Mandrill y arma base

A continuación, se muestra el rendimiento de los agentes estático y decaído, además de un agente aleatorio, cuando juegan en el mismo escenario en el que fueron entrenados.

Tabla 5.3.4. Resultados del agente entrenado al enfrentarse a Spark Mandrill y arma base: Promedio de Recompensas Acumuladas, *Timesteps* y Vida Final por Episodio (Desviación Estándar entre paréntesis).

Agente	Recompensa Acumulada	*Timesteps*	Vida Final por Episodio
Aleatorio	-50.12 (5.310894)	1824.78 (252.230870)	0 (0)
Estático	50.24 (7.458043)	592.69 (41.058908)	17.5 (5.218237)
Decaído	50.23 (7.810064)	607.79 (46.592337)	19.13 (5.102264)

Dados estos resultados, no se puede concluir que exista una diferencia significativa entre los promedios de recompensas acumuladas de los modelos estático y decaído (valor p = 0.9926), pero sí que ambos modelos superan al agente aleatorio (ver Apéndice 72).

Se puede concluir que el modelo estático tardó menos que el modelo decaído, y que este, a su vez, tardó menos que el agente aleatorio (ver Apéndice 73).

También se puede concluir que el mejor promedio lo obtuvo el modelo decaído, seguido por el modelo estático y, finalmente, el agente aleatorio (ver Apéndice 74).

Porcentaje de victoria

El agente aleatorio no ganó ningún juego (0%), mientras que tanto el modelo estático como el decaído lograron una tasa de victoria del 100%. Dados estos resultados y su interpretación estadística en base a los valores p, dados por la prueba z, no se puede concluir que exista una diferencia significativa entre los porcentajes de victoria de los modelos estático y decaído (valor p = 1); sin embargo, sí que ambos superan al agente aleatorio.

Los intervalos de confianza binomiales al 95% calculados mediante el método descrito en [17] fueron (0.0%, 3.6%) para el agente aleatorio y (96.3%, 100%) para ambos modelos entrenados.

Análisis cualitativo

Al observar a los agentes se puede ver que ambos adoptan la misma estrategia: esquivar al oponente. Cuando el oponente está colgado del techo, el agente se mueve para evitar que le caiga encima; cuando el oponente está en el suelo, se aleja lo suficiente para evitar el contacto. Dado que este oponente pasa gran parte de la pelea colgado, los agentes aprendieron a utilizar las paredes para dispararle mientras está suspendido (ver Apéndice 75 y 76).

5.3.2.2. Evaluación contra Spark Mandrill y Shotgun Ice

Los siguientes son los resultados cuando los agentes se enfrentan nuevamente al escenario donde se entrenaron, pero esta vez utilizando el arma *Shotgun Ice*, la cual es la debilidad del oponente al cual se enfrentan, de esta manera, la dificultad del escenario disminuye.

Tabla 5.3.5. Resultados del agente entrenado al enfrentarse a Spark Mandrill y *Shotgun Ice*: Promedio de Recompensas Acumuladas, *Timesteps* y Vida Final por Episodio (Desviación Estándar entre paréntesis).

Agente	Recompensa Acumulada	*Timesteps*	Vida Final por Episodio
Aleatorio	49.27 (13.580762)	937.64 (126.659506)	19.35 (6.099795)
Estático	50.27 (5.159176)	242.4 (22.064904)	26.56 (3.156961)
Decaído	50.21 (6.776865)	241.63 (19.880470)	25.92 (4.168165)

Dados estos resultados, no se puede concluir que exista una diferencia significativa entre los promedios de recompensas acumuladas de los modelos estático y decaído (valor p = 0.9439), entre el modelo estático y el agente aleatorio (valor p = 0.492), ni entre el modelo decaído y el agente aleatorio (valor p = 0.5364) (ver Apéndice 77).

No se puede concluir que el modelo decaído haya tardado menos que el modelo estático (valor p = 0.7957); sin embargo, ambos modelos fueron más rápidos que el agente aleatorio (ver Apéndice 78).

Tampoco se puede concluir que exista una diferencia significativa entre los promedios de vida al final de cada episodio de los modelos estático y decaído (valor p = 0.2224); sin embargo, ambos superan al agente aleatorio (ver Apéndice 79).

Porcentaje de victoria

El agente aleatorio ganó un 99% de sus juegos, mientras que tanto el modelo estático como el decaído lograron una tasa de victoria del 100%. Dados estos resultados y su interpretación estadística en base a los valores p, dados por la prueba z, no se puede concluir que exista una diferencia significativa entre los porcentajes de victoria de los modelos estático y decaído (valor p = 1), ni entre los modelos entrenados y el agente aleatorio (valor p = 0.3173).

Esta conclusión fue respaldada por los intervalos de confianza binomiales al 95% calculados mediante el método descrito en [17]. El agente aleatorio presentó un intervalo de (94.6%, 99.9%), mientras que ambos modelos entrenados tuvieron un intervalo de (96.3%, 100%).

Análisis cualitativo

En este caso, los agentes se enfrentan al mismo oponente contra el que entrenaron, pero con la diferencia de que el arma que utilizan permite quitarle más vida y congelarlo para facilitar la batalla. Es por esto que los resultados obtenidos son incluso mejores que en el escenario anterior, aunque utilizan la misma estrategia (ver Apéndice 80 y 81).

5.3.2.3. Evaluación contra Chill Penguin

Los siguientes son los resultados cuando los agentes entrenados con el escenario contra Spark Mandrill y el arma base se enfrentan a un escenario que nunca han visto antes, la batalla contra Chill Penguin. Esta evaluación pone a prueba la capacidad de generalización de los agentes entrenados.

Tabla 5.3.6. Resultados del agente entrenado al enfrentarse a Chill Penguin: Promedio de Recompensas Acumuladas, *Timesteps* y Vida Final por Episodio (Desviación Estándar entre paréntesis).

Agente	Recompensa Acumulada	*Timesteps*	Vida Final por Episodio
Aleatorio	-48.1 (17.752465)	3081.84 (778.153760)	0.06 (0.506360)
Estático	-50.2 (5.314132)	849.71 (209.499227)	0 (0)
Decaído	-50.16 (5.258745)	889.1 (220.516009)	0 (0)

Dados estos resultados, no se puede concluir que exista una diferencia significativa entre los promedios de recompensas acumuladas de los modelos estático y decaído (valor p = 0.9574), entre el modelo estático y el agente aleatorio (valor p = 0.2585), ni entre el modelo decaído y el agente aleatorio (valor p = 0.2672) (ver Apéndice 82).

No se puede concluir que el modelo estático haya tardado menos que el modelo decaído (valor p = 0.1968); sin embargo, ambos modelos fueron más rápidos que el agente aleatorio (ver Apéndice 83).

Tampoco se puede concluir que exista una diferencia significativa entre los promedios de vida al final de cada episodio de los modelos estático y decaído (valor p = 1), entre el modelo estático y el agente aleatorio (valor p = 0.2375), ni entre el modelo decaído y el agente aleatorio (valor p = 0.2375) (ver Apéndice 84).

Porcentaje de victoria

El agente aleatorio ganó un 2% de sus juegos, mientras que tanto el modelo estático como el modelo decaído no obtuvieron ninguna victoria (0%). Dados estos resultados y su interpretación estadística en base a los valores p, dados por la prueba z, no se puede concluir que exista una diferencia significativa entre los porcentajes de victoria de los modelos estático y decaído (valor p = 1), ni entre cada uno de estos y el agente aleatorio (valor p = 0.1556).

Se calcularon además intervalos de confianza binomiales al 95% utilizando el método descrito en [17]. El agente aleatorio presentó un intervalo de (0.2%, 7.0%), mientras que los modelos entrenados tuvieron un intervalo de (0.0%, 3.6%).

Análisis cualitativo

En ambos casos, los agentes utilizan la misma estrategia que aprendieron contra Spark Mandrill, pero esta estrategia no resulta efectiva contra este oponente. Para evitar ser congelado por el aliento de nieve, es necesario mantenerse alejado del oponente. Además, dado que este oponente pasa la mayor parte del tiempo en el suelo, escalar las paredes ayuda a esquivar sus ataques, pero no permite causarle daño (ver Apéndice 85 y 86).

5.3.3. Entrenamiento con Spark Mandrill y *Shotgun Ice*

La Figura 5.19 muestra las recompensas promedio al final de cada episodio. Se observa que la curva de aprendizaje del entrenamiento estático inicia muy cercano a su máximo y llega rápidamente a este, manteniéndose en ese nivel. En contraste, el entrenamiento decaído exhibe una tendencia positiva continua hasta converger con la curva del entrenamiento estático.

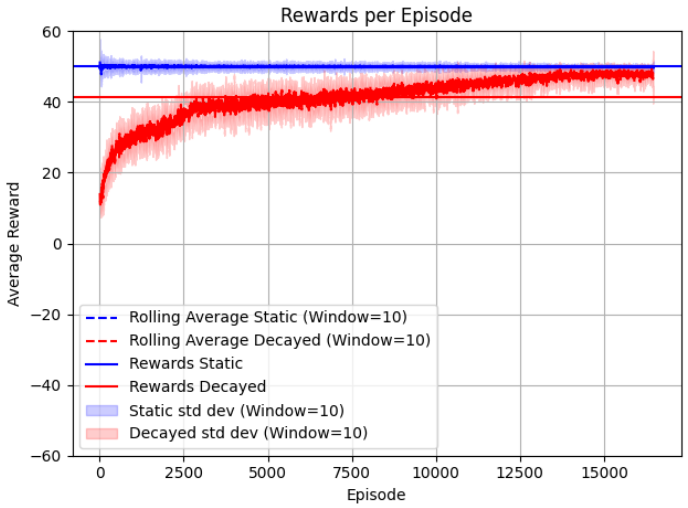


Figura 5.19. Entrenamiento con Spark Mandrill y Shotgun Ice: Gráfico de recompensa promedio por episodio.

La Figura 5.20 muestra el promedio de *timesteps* que duraron los episodios. Ambas curvas presentan un comportamiento prácticamente idéntico a lo largo del entrenamiento, con diferencias mínimas tanto en forma como en valores promedio.

A diferencia de otros entrenamientos con oponentes de este juego, los episodios en este caso comienzan con una duración considerablemente más baja y continúan disminuyendo ligeramente hasta estabilizarse en torno a los 200 *timesteps*. El modelo estático obtuvo un promedio de 202.5, mientras que el modelo decaído registró 202.8, con una diferencia menor al 0.15%, sin implicancia práctica. Esta convergencia tan precisa sugiere que ambos modelos aprendieron a ejecutar episodios de forma muy eficiente desde etapas tempranas del entrenamiento, y mantuvieron ese comportamiento de manera estable.

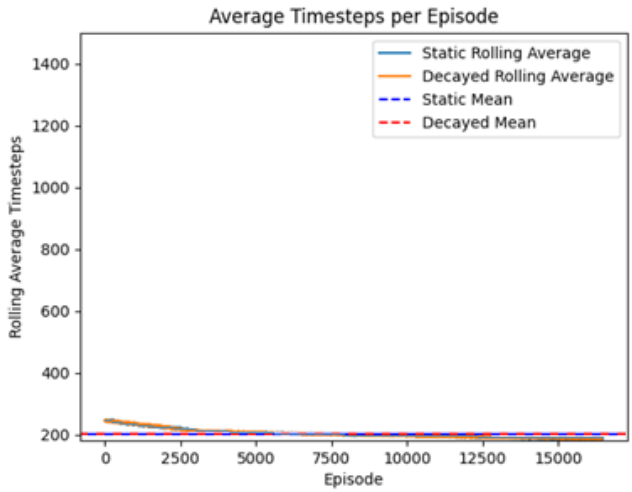


Figura 5.20. Entrenamiento con Spark Mandrill y Shotgun Ice: Gráfico de timesteps promedio por episodio.

La Figura 5.21 muestra el porcentaje de victorias. Ambas curvas presentan un comportamiento similar: comienzan cerca del 100% de victorias y, alrededor del episodio 2000, alcanzan de manera consistente ese porcentaje.

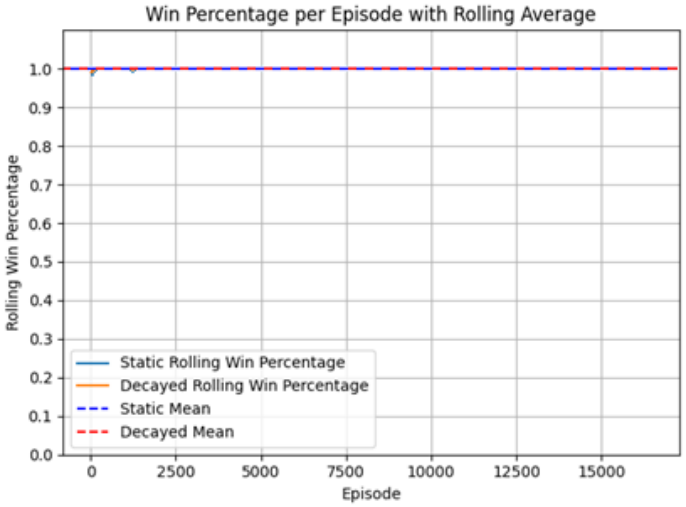


Figura 5.21. Entrenamiento con Spark Mandrill y Shotgun Ice: Gráfico de porcentaje de victoria promedio por episodio.

La diferencia entre las curvas se observa únicamente en el gráfico de recompensa acumulada por episodio (Figura 5.19). Esto se debe a que, al inicio, en ambos entrenamientos los agentes ganan la batalla contra el contrincante casi un 100% de las veces y reciben la recompensa final por ganar; sin embargo, el entrenamiento decaído obtiene una recompensa considerablemente menor, ya que el agente gana sin haber masterizado las estrategias defensivas, es decir, logra vencer, pero recibiendo daño. A lo largo del entrenamiento, se puede observar una mejora en la defensa del agente, ya que va obteniendo una recompensa mayor al finalizar los episodios con el paso del tiempo.

5.3.3.1. Evaluación contra Spark Mandrill y Shotgun Ice

A continuación, se muestra el rendimiento de los agentes estático y decaído, además de un agente aleatorio, cuando juegan en el mismo escenario en el que fueron entrenados.

Tabla 5.3.7. Resultados del agente entrenado al enfrentarse a Spark Mandrill y *Shotgun Ice*: Promedio de Recompensas Acumuladas, *Timesteps* y Vida Final por Episodio (Desviación Estándar entre paréntesis).

Agente	Recompensa Acumulada	*Timesteps*	Vida Final por Episodio
Aleatorio	49.19 (15.249062)	924.07 (75.317628)	19.79 (5.745076)
Estático	50.32 (3.725265)	186.94 (11.527203)	31.47 (1.260595)
Decaído	50.32 (3.649329)	187.94 (12.610963)	31.52 (1.389100)

Dados estos resultados, no se puede concluir que exista una diferencia significativa entre los promedios de recompensas acumuladas de los modelos estático y decaído (valor p = 1), entre el modelo estático y el agente aleatorio (valor p = 0.4725), ni entre el modelo decaído y el agente aleatorio (valor p = 0.472) (ver Apéndice 87).

No se puede concluir que el modelo estático haya tardado menos que el modelo decaído (valor p = 0.559); sin embargo, ambos modelos fueron más rápidos que el agente aleatorio (ver Apéndice 88).

Tampoco se puede concluir que exista una diferencia significativa entre los promedios de vida al final de cada episodio de los modelos estático y decaído (valor p = 0.7901); sin embargo, ambos modelos consiguieron un mejor rendimiento que el agente aleatorio (ver Apéndice 89).

Porcentaje de victoria

El agente aleatorio ganó un 99% de sus juegos, mientras que tanto el modelo estático como el decaído lograron una tasa de victoria del 100%. Dados estos resultados y su interpretación estadística en base a los valores p, dados por la prueba z, no se puede concluir que exista una diferencia significativa entre los porcentajes de victoria de los modelos estático y decaído (valor p = 1), entre cada uno de ellos y el agente aleatorio (valor p = 0.3173).

Para complementar el análisis, se calcularon intervalos de confianza binomiales al 95% utilizando el método descrito en [17]. El agente aleatorio presentó un intervalo de (94.6%, 99.9%), mientras que los modelos entrenados mostraron un intervalo de (96.3%, 100%).

Análisis cualitativo

Al observar a los agentes se puede ver que ambos adoptan la misma estrategia, esto es mantenerse al lado izquierdo de la pantalla y disparar al oponente para congelarlo, impidiendo que salte o avance hacia el agente (ver Apéndice 90 y 91).

5.3.3.2. Evaluación contra Spark Mandrill y arma base

Los siguientes son los resultados cuando los agentes se enfrentan nuevamente al escenario donde se entrenaron, pero esta vez utilizando el arma base, lo cual incrementa bastante la dificultad.

Tabla 5.3.8. Resultados del agente entrenado al enfrentarse a Spark Mandrill y arma base: Promedio de Recompensas Acumuladas, *Timesteps* y Vida Final por Episodio (Desviación Estándar entre paréntesis).

Agente	Recompensa Acumulada	*Timesteps*	Vida Final por Episodio
Aleatorio	-50.19 (5.039236)	1848.77 (248.399793)	0 (0)
Estático	-50.19 (6.281234)	617.91 (159.212317)	0 (0)
Decaído	-50.15 (6.243997)	515.37 (109.301478)	0 (0)

Dados estos resultados, no se puede concluir que exista una diferencia significativa entre los promedios de recompensas acumuladas de los modelos estático y decaído (valor p = 0.964), entre el modelo estático y el agente aleatorio (valor p = 1), ni entre el modelo decaído y el agente aleatorio (valor p = 0.9603) (ver Apéndice 92).

Se puede concluir que el modelo decaído terminó los episodios más rápido, seguido por el modelo estático y, finalmente, el agente aleatorio (ver Apéndice 93).

No se puede concluir que exista una diferencia significativa entre los promedios de vida al final de cada episodio de los modelos estático, decaído y agente aleatorio (valor p = 1 en todos los casos) (ver Apéndice 94).

Porcentaje de victoria

Ni el agente aleatorio ni los modelos estático y decaído ganaron algún juego (0%). Dados estos resultados y su interpretación estadística en base a los valores p, dados por la prueba z, no se puede concluir que exista una diferencia significativa entre los porcentajes de victoria de los modelos estático, decaído y agente aleatorio (valor p = 1 en todos los casos).

Adicionalmente, se calcularon intervalos de confianza binomiales al 95% utilizando el método descrito en [17]. Para los tres agentes, el intervalo fue idéntico: (0.0%, 3.6%). Esto refleja que, aunque no se observaron victorias en la muestra, **no se puede afirmar que sus tasas reales de victoria sean exactamente cero**, solo que probablemente son muy bajas.

Análisis cualitativo

En este caso, se puede ver que los agentes estático y decaído se comportan de manera similar, ambos se mueven de un lado a otro disparándole al oponente, pero debido a que no fue necesario utilizar las paredes en el escenario donde entrenaron, nunca aprendieron a usarlas para dispararle al oponente, algo necesario en este caso (ver Apéndice 95 y 96).

5.3.3.3. Evaluación contra Chill Penguin

Los siguientes son los resultados cuando los agentes entrenados con el escenario contra Spark Mandrill y *Shotgun Ice* se enfrentan a un escenario que nunca han visto antes, la batalla contra Chill Penguin. Esta evaluación pone a prueba la capacidad de generalización de los agentes entrenados.

Tabla 5.3.9. Resultados del agente entrenado al enfrentarse a Chill Penguin: Promedio de Recompensas Acumuladas, *Timesteps* y Vida Final por Episodio (Desviación Estándar entre paréntesis).

Agente	Recompensa Acumulada	*Timesteps*	Vida Final por Episodio
Aleatorio	-43.18 (29.715444)	3175.6 (694.134439)	0.22 (0.855336)
Estático	-50.23 (6.472797)	705.23 (167.028731)	0 (0)
Decaído	-50.22 (4.693783)	768.85 (224.504983)	0 (0)

Dados estos resultados, no se puede concluir que exista una diferencia significativa entre los promedios de recompensa acumulada por episodio de los modelos estático y decaído (valor p = 0.99); sin embargo, ambos modelos consiguieron un peor rendimiento que el agente aleatorio (ver Apéndice 97).

Se puede concluir que el modelo estático terminó los episodios más rápido, seguido por el modelo decaído y, finalmente, el agente aleatorio (ver Apéndice 98).

No se puede concluir que exista una diferencia significativa entre los promedios de vida al final del episodio de los modelos estático y decaído (valor p = 1); sin embargo, ambos modelos consiguieron un peor rendimiento que el agente aleatorio (ver Apéndice 99).

Porcentaje de victoria

El agente aleatorio ganó un 7% de sus juegos, mientras que tanto el modelo estático como el modelo decaído no obtuvieron ninguna victoria (0%). Dados estos resultados y su interpretación estadística en base a los valores p, dados por la prueba z, no se puede concluir que exista una diferencia significativa entre los porcentajes de victoria de los modelos estático y decaído (valor p = 1); sin embargo, sí que ambos modelos consiguieron un peor rendimiento que el agente aleatorio.

No obstante, los intervalos de confianza binomiales al 95% calculados mediante el método descrito en [17] fueron (0.0%, 3.6%) para los modelos entrenados y (2.9%, 14.0%) para el agente aleatorio, mostrando una ligera superposición. Esta diferencia entre métodos refleja que, aunque la prueba z indica significancia, la interpretación basada en intervalos de confianza es más conservadora. Por lo tanto, la conclusión de superioridad del agente aleatorio debe considerarse sugerente, pero no concluyente desde todos los enfoques estadísticos.

Análisis cualitativo

En este caso, tanto el agente estático como el decaído se comportan de manera similar, a diferencia del escenario anterior, sí utilizan las paredes en ocasiones, pero no aprendieron a esquivar los ataques del enemigo, dejándose congelar en múltiples ocasiones y no esquivando las estatuas de hielo (ver Apéndice 100 y 101).

5.4. Resultados generales

Las tablas 5.4.1, 5.4.2, 5.4.3 y 5.4.4 muestran cuál fue el o los modelos que demostraron un mejor rendimiento en las categorías recompensas acumuladas por episodio, cantidad de *timesteps* por episodio, vida al final del episodio, porcentaje de victoria y general (el o los modelos que más se repiten en las otras categorías).

Tabla 5.4.1. Mejor modelo por categoría en juegos contra oponentes de Punch Out!!

		Entrenamiento con Glass Joe	Entrenamiento con Von Kaiser
Juegos contra Glass Joe (fácil)	**Recompensas acumuladas**	Estático	Estático y Decaído
	Cantidad de *timesteps*	Decaído	Estático y Decaído
	Vida final	Estático y Decaído	Estático y Decaído
	Porcentaje de victoria	Estático y Decaído	Decaído
	General	Estático y Decaído	Decaído
Juegos contra Von Kaiser (difícil)	**Recompensas acumuladas**	Estático y Decaído	Estático y Decaído
	Cantidad de *timesteps*	Estático y Decaído	Decaído
	Vida final	Estático y Decaído	Estático y Decaído
	Porcentaje de victoria	Estático y Decaído	Estático y Decaído
	General	Estático y Decaído	Decaído

Tabla 5.4.2. Mejor modelo por categoría en juegos contra oponentes de The Legend of Zelda.

		Entrenamiento con Aquamentus	Entrenamiento con Manhandla
Juegos contra Aquamentus (fácil)	**Recompensas acumuladas**	Estático y Decaído	Estático
	Cantidad de *timesteps*	Estático y Decaído	Estático*
	Vida final	Estático	Estático
	Porcentaje de victoria	Estático y Decaído	Estático
	General	Estático	Estático
Juegos contra Manhandla (difícil)	**Recompensas acumuladas**	Aleatorio	Estático y Decaído
	Cantidad de *timesteps*	Aleatorio**	Estático y Decaído
	Vida final	Aleatorio	Estático y Decaído
	Porcentaje de victoria	Aleatorio	Estático y Decaído
	General	Aleatorio	Estático y Decaído

*La tabla 5.2.4 muestra que la versión estática tardó más que las versiones decaída y aleatoria. En este caso el agente aleatorio se demora poco debido a que pierde rápidamente, por otro lado, en el caso del modelo decaído, las derrotas rápidas disminuyen el promedio del tiempo. Por ello, el mejor desempeño fue el de la versión estática, ya que, aunque es el que tiene un mayor promedio de tiempo, esto es por su mejor porcentaje de victoria.

**La tabla 5.2.2 muestra que la versión aleatoria tardó más que las versiones estática y decaída. Esto se debe a que las versiones estática y decaída pierden rápidamente contra Manhandla. Por esta razón, el mejor desempeño fue el de la versión aleatoria, ya que esta logró ganar en algunas ocasiones.

Tabla 5.4.3. Mejor modelo por categoría en juegos contra oponentes de Mega Man X.

		Entrenamiento con Chill Penguin	Entrenamiento con Spark Mandrill (arma base)	Entrenamiento con Spark Mandrill (*Shotgun Ice*)
Juegos contra Chill Penguin (fácil)	**Recompensas acumuladas**	Estático y Decaído	Ninguno	Aleatorio
	Cantidad de *timesteps*	Estático y Decaído	Estático y Decaído	Aleatorio*
	Vida final	Estático y Decaído	Ninguno	Aleatorio
	Porcentaje de victoria	Estático y Decaído	Ninguno	Aleatorio
	General	Estático y Decaído	Ninguno	Aleatorio
Juegos contra Spark Mandrill (arma base) (difícil)	**Recompensas acumuladas**	Ninguno	Estático y Decaído	Ninguno
	Cantidad de *timesteps*	Ninguno**	Estático	Ninguno***
	Vida final	Ninguno	Decaído	Ninguno
	Porcentaje de victoria	Ninguno	Estático y Decaído	Ninguno
	General	Ninguno	Estático y Decaído	Ninguno

*La tabla 5.3.9 indica que la versión aleatoria tardó más que las versiones estática y decaída. Esto se debe a que las versiones estática y decaída pierden rápidamente contra Chill Penguin. Por lo tanto, el mejor desempeño fue el de la versión aleatoria, ya que logró ganar en algunas ocasiones.

**La tabla 5.3.2 muestra que la versión aleatoria tardó más que las versiones estática y decaída, pero esto es debido a que las versiones estática y decaída pierden rápidamente ante Spark Mandrill. Aunque la versión aleatoria consigue sobrevivir más tiempo, no logra una mayor recompensa acumulada ni ganar en ninguna ocasión.

***La tabla 5.3.8 muestra que la versión decaída terminó los episodios más rápido, seguida por la versión estática y, finalmente, la versión aleatoria. Sin embargo, ninguna versión logra ganar, y aunque algunas logran sobrevivir más tiempo, no consiguen una mayor recompensa acumulada.

Tabla 5.4.4. Mejor modelo por categoría en juegos contra oponentes de Mega Man X (continuación).

		Entrenamiento con Chill Penguin	Entrenamiento con Spark Mandrill (arma base)	Entrenamiento con Spark Mandrill (*Shotgun Ice*)
Juegos contra Spark Mandrill (*Shotgun Ice*) (fácil)	**Recompensas acumuladas**	Ninguno	Ninguno	Ninguno
	Cantidad de *timesteps*	Decaído	Estático y Decaído	Estático y Decaído
	Vida final	Estático	Estático y Decaído	Estático y Decaído
	Porcentaje de victoria	Estático y Aleatorio	Ninguno	Ninguno
	General	Estático	Estático y Decaído	Estático y Decaído

5.4.1. Análisis de resultados para ***Punch Out!!***

5.4.1.1. Entrenamiento con Glass Joe

Analizando los resultados obtenidos para Punch Out!! entrenando al agente con el escenario con Glass Joe (el enemigo fácil), podemos observar que:

- El mejor entrenamiento se observa al aplicar una recompensa decaída, lo que resulta en menor cobardía y mayor maestría en el comportamiento defensivo.
- Al usar los modelos en el mismo ambiente donde fueron entrenados, no se aprecia una clara superioridad entre ellos, aunque el modelo decaído muestra mayor valentía al completar los episodios en menos tiempo.

- En un ambiente distinto y con mayor dificultad, no se observan diferencias en el rendimiento de los modelos. Ambos logran ganar ocasionalmente, pero no alcanzan un buen porcentaje de victorias.

5.4.1.2. Entrenamiento con Von Kaiser

Analizando los resultados obtenidos para Punch Out!! entrenando al agente con el escenario con Von Kaiser (el enemigo difícil), podemos observar que:

- La recompensa decaída resulta en el mejor entrenamiento, observándose menor cobardía.
- Cuando se utilizan los modelos para jugar en el mismo ambiente donde se entrenaron, se puede ver una leve superioridad del modelo con recompensa final decaída, este modelo demuestra una mayor valentía, ya que demora menos en terminar los episodios.
- Cuando se utilizan los modelos para jugar en un ambiente diferente al que se entrenaron y con menor dificultad, se puede ver una leve superioridad del modelo con recompensa final decaída, este modelo logra un mejor porcentaje de victoria. Ambos modelos logran ganar en este nuevo escenario consiguiendo un buen porcentaje de victoria.

5.4.2. Análisis de resultados para ***The Legend of Zelda***

5.4.2.1. Entrenamiento con Aquamentus

Analizando los resultados obtenidos para The Legend of Zelda entrenando al agente con el escenario con Aquamentus (el enemigo fácil), podemos observar que:

- No se observan diferencias en el entrenamiento cuando se aplica una recompensa decaída frente a una estática.
- Al utilizar los modelos en el mismo ambiente donde fueron entrenados, se observa una leve superioridad del modelo con recompensa final estática, ya que este alcanza un mayor promedio de vida al final de los episodios, lo cual indica una mejor maestría en el comportamiento defensivo.
- Cuando se utilizan los modelos para jugar en un ambiente diferente al que se entrenaron y con mayor dificultad, no se puede observar diferencias en el rendimiento de los modelos. Ambos modelos no logran ganar en alguna ocasión y presentan un peor rendimiento que el agente aleatorio.

5.4.2.2. Entrenamiento con Manhandla

Analizando los resultados obtenidos para The Legend of Zelda entrenando al agente con el escenario con Manhandla (el enemigo difícil), podemos observar que:

- El mejor entrenamiento se observa al aplicar una recompensa estática.
- Cuando se utilizan los modelos para jugar en el mismo ambiente donde se entrenaron, no se puede afirmar que exista una superioridad de un modelo sobre el otro.
- Cuando se utilizan los modelos para jugar en un ambiente diferente al que se entrenaron y con menor dificultad, se puede ver una superioridad del modelo estático, este modelo logra un mejor rendimiento en todos los ámbitos analizados. Ambos modelos logran ganar en ocasiones en este nuevo escenario, pero no consiguen un buen porcentaje de victoria.

5.4.3. Análisis de resultados para ***Mega Man X***

5.4.3.1. Entrenamiento con Chill Penguin

Analizando los resultados obtenidos para Mega Man X entrenando al agente con el escenario con Chill Penguin (el enemigo fácil), podemos observar que:

- Hay un mejor rendimiento cuando se aplica una recompensa final decaída, ya que se observa menor cobardía.
- Cuando se utilizan los modelos para jugar en el mismo ambiente donde se entrenaron, no se puede ver una superioridad de un modelo sobre el otro. Ambos modelos consiguen un porcentaje de victoria muy alto y muy por encima al de un agente aleatorio.
- Cuando se utilizan los modelos para jugar en un ambiente diferente al que se entrenaron y con mayor dificultad, no se puede observar diferencias en el rendimiento de los modelos. Ambos modelos no logran ganar y presentan un rendimiento similar al de un agente aleatorio.
- Cuando se utilizan los modelos para jugar en un ambiente diferente al que se entrenaron y con dificultad similar, se puede ver una leve superioridad del modelo con recompensa final estática, este modelo logra un mayor promedio de vida al final de los episodios. Ambos modelos logran ganar en este nuevo escenario consiguiendo un buen porcentaje de victoria en ambos casos, pero cabe destacar que ese rendimiento es similar al de un agente aleatorio.

5.4.3.2. Entrenamiento con Spark Mandrill y arma base

Analizando los resultados obtenidos para Mega Man X entrenando al agente con el escenario con Spark Mandrill y arma base (el enemigo difícil), podemos observar que:

- Hay un mejor rendimiento cuando se aplica una recompensa final decaída, ya que se observa menor cobardía.
- Cuando se utilizan los modelos para jugar en el mismo ambiente donde se entrenaron no se puede ver una superioridad de un modelo sobre el otro, pero ambos modelos

muestran un rendimiento muy superior al de un agente aleatorio. Ambos modelos consiguen un porcentaje de victoria muy alto y muy por encima al de un agente aleatorio.
- Cuando se utilizan los modelos para jugar contra el mismo oponente con el que se entrenaron, pero con un arma que disminuye la dificultad, no se puede ver una superioridad de un modelo sobre el otro. Ambos modelos consiguen un porcentaje de victoria muy alto, pero cabe destacar que ese rendimiento es similar al de un agente aleatorio.
- Cuando se utilizan los modelos para jugar en un ambiente diferente al que se entrenaron y con menor dificultad, no se puede ver una superioridad de un modelo sobre el otro. Ambos modelos no logran ganar y presentan un rendimiento similar al de un agente aleatorio.

5.4.3.3. Entrenamiento con Spark Mandrill y Shotgun Ice

Analizando los resultados obtenidos para Mega Man X entrenando al agente con el escenario con Spark Mandrill y *Shotgun Ice* (el enemigo fácil), podemos observar que:

- No se observan diferencias en el entrenamiento cuando se aplica una recompensa decaída frente a una estática, a excepción del gráfico de la Figura 5.22, donde se observan las diferencias en recompensa acumulada promedio generadas por la diferencia en la recompensa final. Este gráfico muestra la masterización de la estrategia defensiva por parte del modelo con recompensa final decaída.
- Cuando se utilizan los modelos para jugar en el mismo ambiente donde se entrenaron no se puede ver una superioridad de un modelo sobre el otro. Ambos modelos consiguen un porcentaje de victoria muy alto, pero cabe destacar que ese rendimiento es similar al de un agente aleatorio.
- Cuando se utilizan los modelos para jugar contra el mismo oponente con el que se entrenaron, pero con un arma que aumenta la dificultad, no se puede ver una superioridad de un modelo sobre el otro. Ningún modelo ni el agente aleatorio logran ganar.
- Cuando se utilizan los modelos para jugar en un ambiente diferente al que se entrenaron y con menor dificultad, no se puede ver una superioridad de un modelo sobre el otro, pero sí que ambos consiguen un peor rendimiento que el de un agente aleatorio. Ambos modelos no logran ganar y presentan un rendimiento inferior al de un agente aleatorio.

6. Conclusiones

En esta memoria se implementaron y evaluaron distintas estrategias de recompensas en entornos de videojuegos retro utilizando el algoritmo **Proximal Policy Optimization (PPO)**. Se trabajó con tres juegos: **Punch Out!!**, **The Legend of Zelda** y **Mega Man X**, seleccionados por sus escenarios de combate y su capacidad para evaluar el desempeño del agente bajo diferentes condiciones.

El objetivo principal fue analizar cómo las estrategias de recompensas afectan el **desempeño** y el **comportamiento** del agente en aprendizaje por refuerzo. Para ello, se definieron dos esquemas de recompensas: **estáticas** y **decaídas**, donde las primeras dependían únicamente del resultado del combate, mientras que las segundas se ajustaban en función de variables como la vida restante del agente o del oponente.

A continuación, se relacionan los resultados obtenidos con los objetivos planteados:

1. Los resultados obtenidos demuestran que la estrategia de recompensa decaída tiene un impacto positivo en el desempeño del agente en la mayoría de los casos. En comparación con la recompensa estática, la recompensa decaída mostró un rendimiento similar o superior, mitigó comportamientos subóptimos, como la "cobardía" en ciertos casos, y generalmente fomentó un aprendizaje más eficiente y proactivo, cumpliendo así con el **objetivo de evaluar el desempeño del agente en diferentes escenarios**.

2. Se observó que los agentes entrenados con recompensas decaídas lograron un equilibrio adecuado entre exploración y explotación, lo que favoreció decisiones más estratégicas y un aprendizaje más eficiente en los juegos Punch Out!! y Mega Man X. Esto se alinea con el objetivo de **analizar la influencia de las recompensas en el aprendizaje y la calidad de las decisiones del agente**.

3. Los agentes entrenados mostraron una buena estabilidad y consistencia en su desempeño. Al comparar los modelos con las estrategias estática y decaída, no se observó una diferencia significativa, lo cual podría deberse a que la valentía y la masterización defensiva pueden interferir entre sí. Esto cumple con el objetivo de **investigar la estabilidad en entornos dinámicos**.

4. La elección del algoritmo PPO resultó adecuada para los escenarios planteados, ya que permitió entrenar agentes de manera eficiente en entornos complejos y de alta dimensionalidad. Además, se realizó un estudio exhaustivo del método PPO y su implementación en PyTorch de Stable Baselines3, cumpliendo así con el objetivo de comprenderlo a profundidad.

5. Finalmente, en relación con el objetivo de **determinar qué tipos de recompensas son más efectivos en diferentes ambientes de juego**, se concluye que las recompensas decaídas, adaptadas según parámetros del estado final, son usualmente igual o más efectivas que las recompensas estáticas para fomentar comportamientos estratégicos y valientes en agentes de aprendizaje por refuerzo. No se observó una gran ventaja de la estrategia decaída frente a la estática, pero los resultados son suficientes para preferir su utilización.

7. Trabajo futuro

A pesar de los avances logrados en esta memoria, existen múltiples áreas que podrían explorarse en investigaciones futuras para ampliar y profundizar en los resultados obtenidos:

1. Ampliación de entornos y videojuegos: Aunque el enfoque se centró en tres videojuegos específicos con escenarios de combate (Punch Out!!, The Legend of Zelda, y Mega Man X), sería interesante evaluar la estrategia de recompensas propuesta en una mayor variedad de videojuegos o, dentro de los mismos videojuegos, contra otros oponentes no explorados en esta memoria.

2. Exploración de algoritmos alternativos: En este trabajo se utilizó el algoritmo Proximal Policy Optimization (PPO) debido a su estabilidad y eficiencia. Sin embargo, otro algoritmo como Deep Q-Networks (DQN) podría ofrecer enfoques complementarios o mejoras en escenarios específicos.

3. Evaluación en entornos multiagente: Utilizar ambientes de combate donde 2 o más agentes interactúan simultáneamente, ya sea como aliados o contrincantes, permitiría analizar cómo las estrategias de recompensas influyen en la cooperación, competencia y aprendizaje conjunto.

4. Masterización ofensiva: Durante esta memoria se trabajó con la masterización defensiva, tratando de que la versión decaída se preocupara de ganar recibiendo el menor daño posible, otra opción es la masterización ofensiva, donde el agente debe ganar lo más rápido posible, esto se logra utilizando una recompensa final decaída que dependa del tiempo en caso de victoria.

8. Bibliografía

[1] GeeksforGeeks, "What is reinforcement learning?" [En línea]. Disponible en: https://www.geeksforgeeks.org/what-is-reinforcement-learning/

[2] Amazon Web Services, "What is reinforcement learning?" [En línea]. Disponible en: https://aws.amazon.com/what-is/reinforcement-learning/

[3] Hugging Face, "Deep Reinforcement Learning with PPO," [En línea]. Disponible en: https://huggingface.co/blog/deep-rl-ppo

[4] Stable-Baselines3, "Stable-Baselines3 documentation," [En línea]. Disponible en: https://stable-baselines3.readthedocs.io/en/master/

[5] Stable-Baselines3, "Proximal Policy Optimization (PPO) — Stable-Baselines3 documentation," [En línea]. Disponible en: https://stable-baselines3.readthedocs.io/en/master/modules/ppo.html

[6] Punch-Out!! Wiki, "Punch-Out!! (NES)," [En línea]. Disponible en: https://punchout.fandom.com/wiki/Punch-Out!!_(NES)

[7] Zelda Wiki, "The Legend of Zelda," [En línea]. Disponible en: https://zelda.fandom.com/wiki/The_Legend_of_Zelda

[8] Mega Man Wiki, "Mega Man X (video game)," [En línea]. Disponible en: https://megaman.fandom.com/wiki/Mega_Man_X_(video_game)

[9] S. Bakos and H. Davoudi, "Mitigating Cowardice for Reinforcement Learning Agents in Combat Scenarios," 2022 IEEE Conference on Games (CoG), Beijing, China, 2022, pp. 377-384

[10] Farama Foundation, "Stable-Retro," [En línea]. Disponible en: https://github.com/Farama-Foundation/stable-retro

[11] Retro documentation, "Integration — Retro 0.8.0 documentation," [En línea]. Disponible en: https://retro.readthedocs.io/en/latest/integration.html

[12] IBM, "What are convolutional neural networks?" [En línea]. Disponible en: https://www.ibm.com/topics/convolutional-neural-networks

[13] GraphPad, "T-test calculator," [En línea]. Disponible en: https://www.graphpad.com/quickcalcs/ttest1/

[14] Social Science Statistics, "Z-test calculator," [En línea]. Disponible en: https://www.socscistatistics.com/tests/ztest/

[15] GeeksforGeeks, “What is Markov Decision Process?” [En línea]. Disponible en: https://www.geeksforgeeks.org/markov-decision-process/

[16] D. E. Rumelhart, G. E. Hinton, and R. J. Williams, *"Learning representations by back-propagating errors"*, Nature, vol. 323, no. 6088, pp. 533-536, 1986. doi: 10.1038/323533a0.

[17] C. J. Clopper and E. S. Pearson, "The use of confidence or fiducial limits illustrated in the case of the binomial," *Biometrika*, vol. 26, no. 4, pp. 404–413, 1934.

9. Apéndice

1. Espacio de acciones para el agente en el juego Punch Out!!.

Botón	Acción
Botón A	Golpe al cuerpo derecho
Botón B	Golpe al cuerpo izquierdo
Botón A + D-Pad Arriba	Gancho Derecho
Botón B + D-Pad Arriba	Jab Izquierdo
D-Pad Derecho	Esquivar hacia la derecha
D-Pad Izquierdo	Esquivar hacia la izquierda
D-Pad Abajo	Bloquear abajo
D-Pad Arriba	Bloquear arriba
Ninguno	Volver al centro/Bloquear

2. Variables utilizadas para el entrenamiento en el juego Punch Out!!.

Variable	Valor	Descripción
health_com	0 a 96	La vida actual del contrincante
health_mac	0 a 96	La vida actual del agente
is_match	1 o 255	Si el valor es 255, la pantalla está mostrando una batalla, el valor es 1 en cualquier otro caso
match	0 a 13	La batalla actual, el valor 0 es la batalla contra Glass Joe y el 1 es la batalla contra Von Kaiser

3. Espacio de acciones para el agente en el juego The Legend of Zelda.

Botón	Acción
D-Pad arriba	Mover al agente hacia arriba
D-Pad abajo	Mover al agente hacia abajo
D-Pad derecha	Mover al agente hacia la derecha
D-Pad izquierda	Mover al agente hacia la izquierda
Botón A	Usar la espada
Ninguno	No hacer nada

4. Variables utilizadas para el entrenamiento en el juego The Legend of Zelda.

Variable	Valor	Descripción
Enemy_HP	0 a 60	La vida actual de Aquamentus
Manhandla_HP_1	0 a 40	La vida actual de la cabeza 1 de Manhandla
Manhandla_HP_2	0 a 40	La vida actual de la cabeza 2 de Manhandla
Manhandla_HP_3	0 a 40	La vida actual de la cabeza 3 de Manhandla
Manhandla_HP_4	0 a 40	La vida actual de la cabeza 4 de Manhandla
Hearts	Dos números hexadecimales	El primer número + 1 indica la cantidad máxima de corazones que el agente puede tener, el segundo número + 1 indica cuántos corazones posee en la actualidad
Partial_Heart	0 a 255	El último corazón que el agente posee puede estar vacío (valor 0), a la mitad (valor entre 1 y 127) o lleno (valor entre 128 y 255)
Boss_X_Pos	0 a 255	La coordenada x de la posición del oponente en la pantalla
Boss_Y_Pos	0 a 223	La coordenada y de la posición del oponente en la pantalla
Link_X_Pos	0 a 255	La coordenada x de la posición del agente en la pantalla
Link_Y_Pos	0 a 223	La coordenada y de la posición del agente en la pantalla
Link_Dir	1, 2, 4 u 8	Hacia donde mira el agente: 1 es este, 2 es oeste, 4 es sur y 8 es norte
screen	Número hexadecimal	Si el valor es 5, la pantalla se encuentra en modo “normal”

Como la vida del agente se compone por dos variables, se definió su vida total como el doble de los corazones que posee, por ejemplo, si posee 4 corazones y medio, su vida total es 9. Por otro lado, la vida total de Manhandla es igual a la suma de la vida de sus 4 cabezas.

5. Espacio de acciones para el agente en el juego Mega Man X.

Botón	Acción
Y	Disparar
B	Saltar
A	Dash
A+B	Salto Dash
D-Pad derecha	Moverse a X a la derecha
D-Pad izquierda	Moverse a X a la izquierda
Ninguno	No hacer nada

6. Variables utilizadas para el entrenamiento en el juego Mega Man X.

Variable	Valor	Descripción
boss	0 a 32	La vida actual del oponente
health	0 a 32	La vida actual de Mega Man X (el agente)

7. Valores de hiperparámetros utilizados en los juegos Punch Out!! y Mega Man X:

- *Learning Rate* = 1e-4
- *N Steps* = 2048
- *Batch Size* = 64
- *Target KL* = 0.03

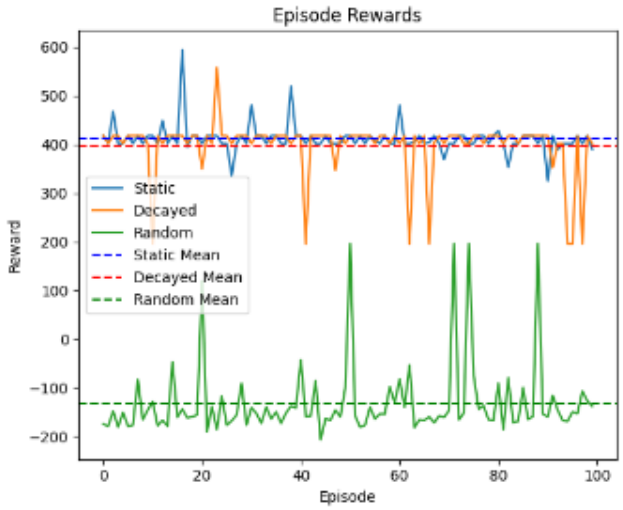
8. Valores de hiperparámetros utilizados en el juego The Legend of Zelda para el escenario con Aquamentus:

- *Learning Rate* = 1e-7
- *N Steps* = 2048
- *Batch Size* = 64
- *Target KL* = 0.03

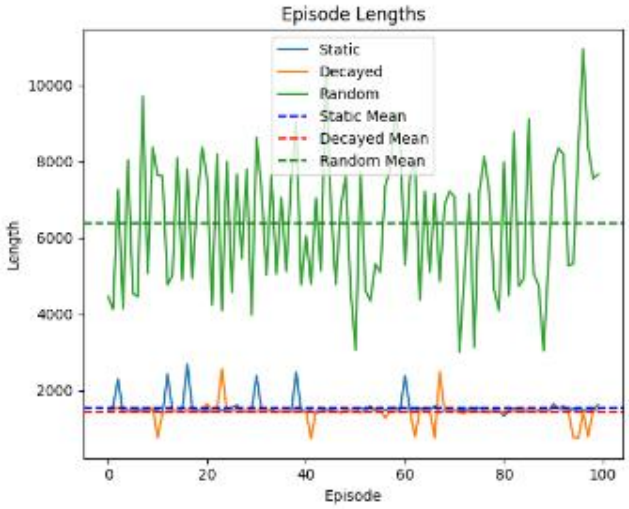
9. Valores de hiperparámetros utilizados en el juego The Legend of Zelda para el escenario con Manhandla:

- *Learning Rate* = 2.5e-4
- *N Steps* = 4096
- *Batch Size* = 64
- *N Epochs* = *7*
- *Clip Range* = *0.02*
- *Target KL* = 0.03

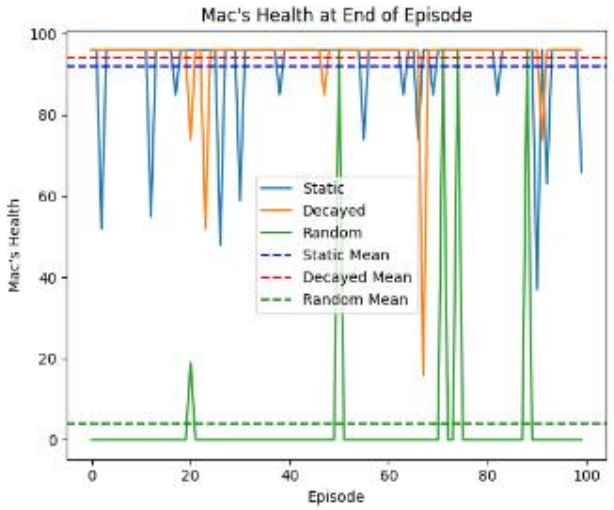
10. Gráfico de recompensas acumuladas por episodio en 100 juegos contra Glass Joe con modelos entrenados con Glass Joe, modelo estático (azul) vs modelo decaído (naranjo) vs agente aleatorio (verde).



11. Gráfico de cantidad de *timesteps* por episodio en 100 juegos contra Glass Joe con modelos entrenados con Glass Joe, modelo estático (azul) vs modelo decaído (naranjo) vs agente aleatorio (verde).



12. Gráfico de vida restante al final del episodio en 100 juegos contra Glass Joe con modelos entrenados con Glass Joe, modelo estático (azul) vs modelo decaído (naranjo) vs agente aleatorio (verde).

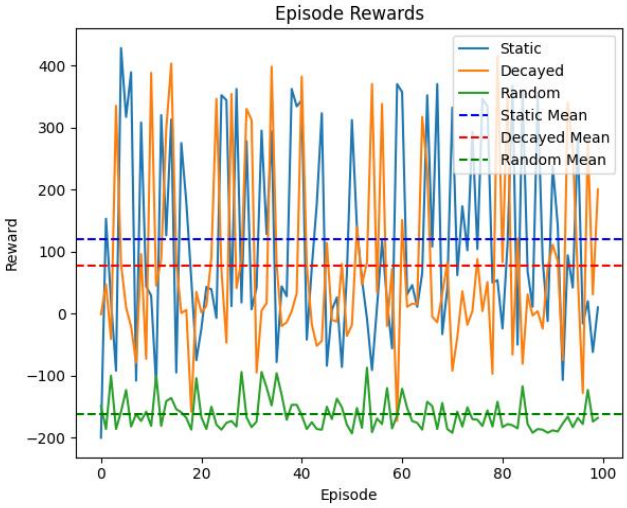


13. Link a video de juegos contra Glass Joe, modelo estático entrenado con Glass Joe: https://drive.google.com/file/d/1Swf9NhU9NjBITQ5LhIZdWCH1Y7UmjJvx/view?usp=sharing

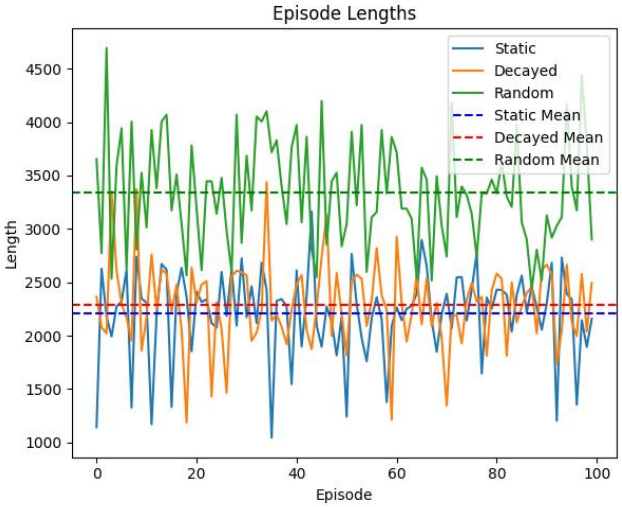
14. Link a video de juegos contra Glass Joe, modelo decaído entrenado con Glass Joe: https://drive.google.com/file/d/1YW4iKFbnRlzTgMGtERpBQ8hGEdRFY9ig/view?usp=sharing

15. Link a video de juegos contra Glass Joe, agente aleatorio: https://drive.google.com/file/d/1WplH8EiMdFr34rZdj9584PmUwbeQADZl/view?usp=sharing

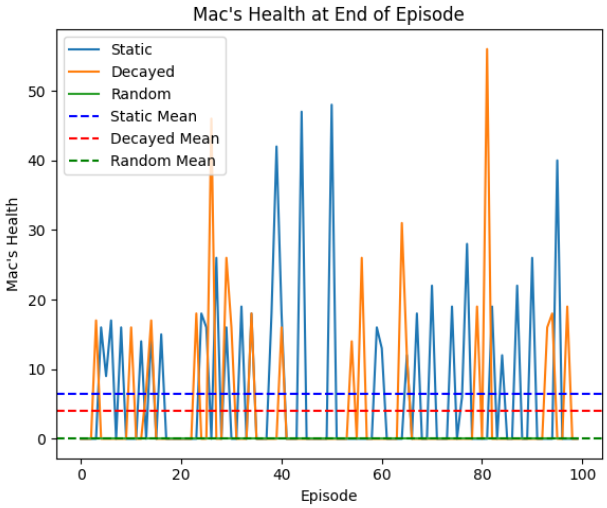
16. Gráfico de recompensas acumuladas por episodio en 100 juegos contra Von Kaiser con modelos entrenados con Glass Joe, modelo estático (azul) vs modelo decaído (naranjo) vs agente aleatorio (verde).



17. Gráfico de cantidad de *timesteps* por episodio en 100 juegos contra Von Kaiser con modelos entrenados con Glass Joe, modelo estático (azul) vs modelo decaído (naranjo) vs agente aleatorio (verde).



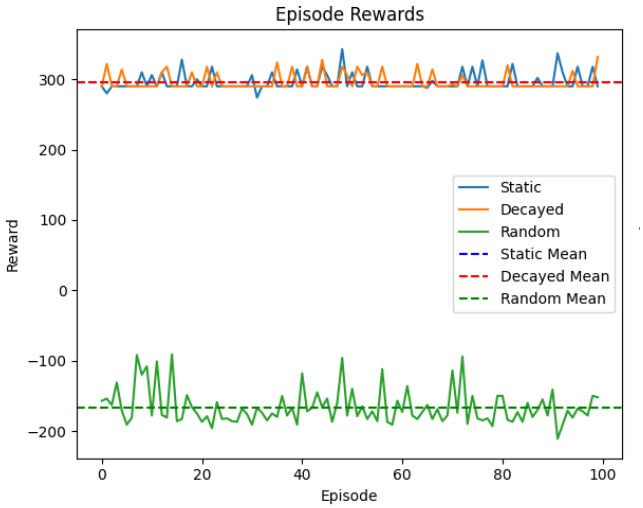
18. Gráfico de vida restante al final del episodio en 100 juegos contra Von Kaiser con modelos entrenados con Glass Joe, modelo estático (azul) vs modelo decaído (naranjo) vs agente aleatorio (verde).



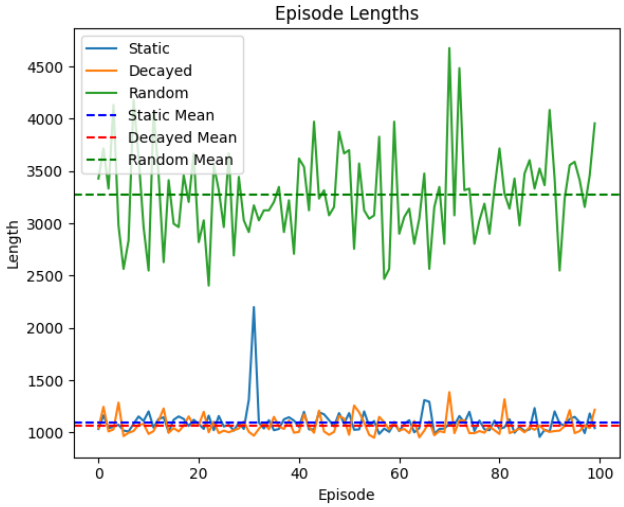
19. Link a video de juegos contra Von Kaiser, modelo estático entrenado con Glass Joe: https://drive.google.com/file/d/1nb07fb3E8jg4vSWXuQVAynWBwxiC00U4/view?usp=sharing

20. Link a video de juegos contra Von Kaiser, modelo decaído entrenado con Glass Joe: https://drive.google.com/file/d/1fWwc5rpK7UR973J02896yILAuwPJd80J/view?usp=sharing
21. Link a video de juegos contra Von Kaiser, agente aleatorio: https://drive.google.com/file/d/1Sjor7d44DPZ0_Znm6Cs0tUSPmfYi8vqk/view?usp=sharing

22. Gráfico de recompensas acumuladas por episodio en 100 juegos contra Von Kaiser con modelos entrenados con Von Kaiser, modelo estático (azul) vs modelo decaído (naranjo) vs agente aleatorio (verde).



23. Gráfico de cantidad de *timesteps* por episodio en 100 juegos contra Von Kaiser con modelos entrenados con Von Kaiser, modelo estático (azul) vs modelo decaído (naranjo) vs agente aleatorio (verde).

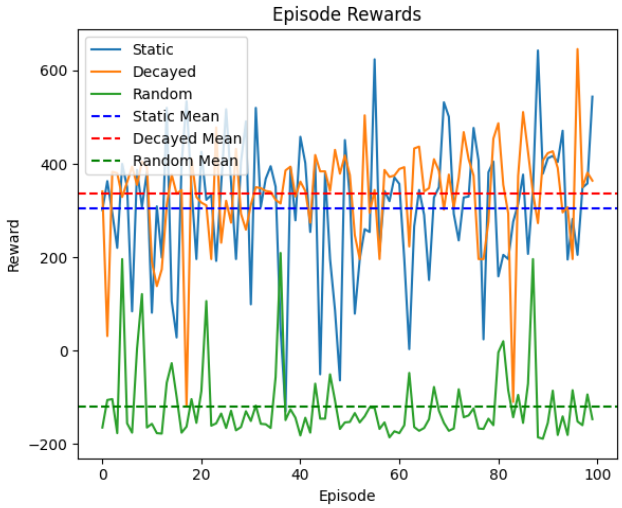


24. Gráfico de vida restante al final del episodio en 100 juegos contra Von Kaiser con modelos entrenados con Von Kaiser, modelo estático (azul) vs modelo decaído (naranjo) vs agente aleatorio (verde).

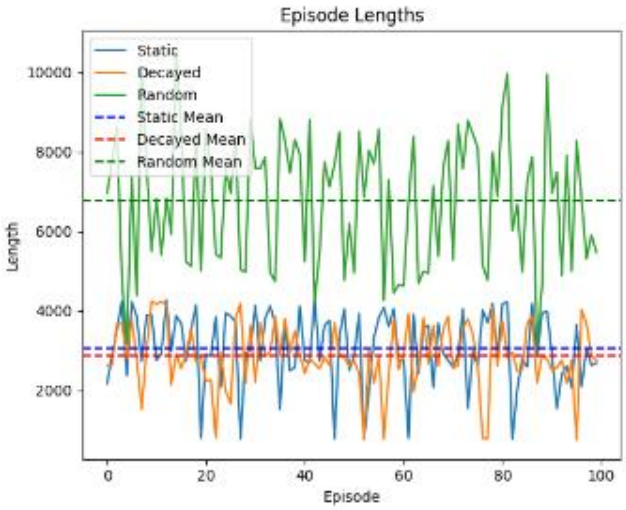
25. Link a video de juegos contra Von Kaiser, modelo estático entrenado con Von Kaiser: https://drive.google.com/file/d/1W2MUFNME6avvZK9H3hfZ0SYzMumtlU0L/view?usp=sharing

26. Link a video de juegos contra Von Kaiser, modelo decaído entrenado con Von Kaiser: https://drive.google.com/file/d/1xTqLsrAQr-5Dwk0uyCiaBKvBnk4OLDgA/view?usp=sharing

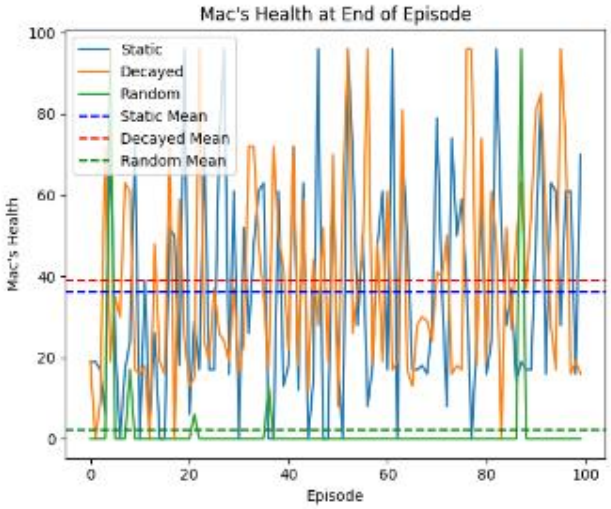
27. Gráfico de recompensas acumuladas por episodio en 100 juegos contra Glass Joe con modelos entrenados con Von Kaiser, modelo estático (azul) vs modelo decaído (naranjo) vs agente aleatorio (verde).



28. Gráfico de cantidad de *timesteps* por episodio en 100 juegos contra Glass Joe con modelos entrenados con Von Kaiser, modelo estático (azul) vs modelo decaído (naranjo) vs agente aleatorio (verde).



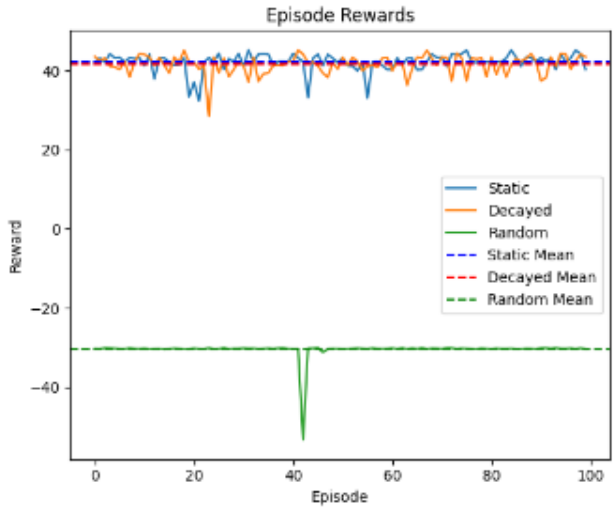
29. Gráfico de vida restante al final del episodio en 100 juegos contra Glass Joe con modelos entrenados con Von Kaiser, modelo estático (azul) vs modelo decaído (naranjo) vs agente aleatorio (verde).



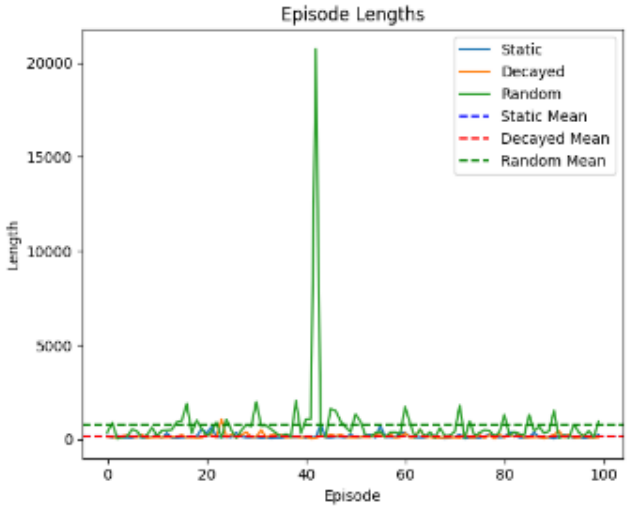
30. Link a video de juegos contra Glass Joe, modelo estático entrenado con Von Kaiser: https://drive.google.com/file/d/1j9R9i62hSH4zrs3ogOgYnOfduPy-sFzq/view?usp=sharing

31. Link a video de juegos contra Glass Joe, modelo decaído entrenado con Von Kaiser: https://drive.google.com/file/d/1mHnAwneo0Dd8qcBGGCJfWF1J5eVUtFIB/view?usp=sharing

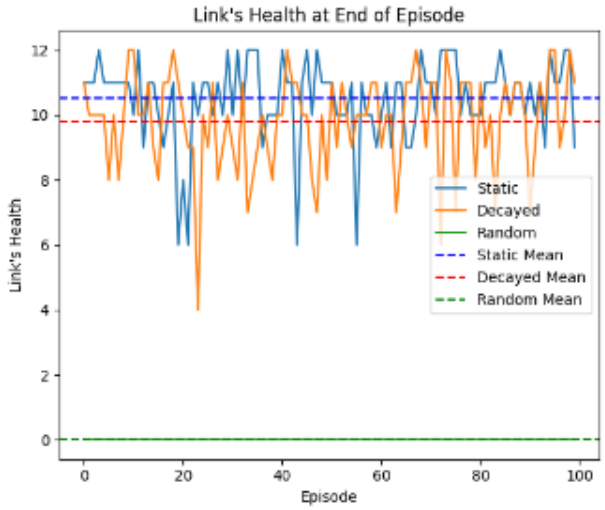
32. Gráfico de recompensas acumuladas por episodio en 100 juegos contra Aquamentus con modelos entrenados con Aquamentus, modelo estático (azul) vs modelo decaído (naranjo) vs agente aleatorio (verde).



33. Gráfico de cantidad de *timesteps* por episodio en 100 juegos contra Aquamentus con modelos entrenados con Aquamentus, modelo estático (azul) vs modelo decaído (naranjo) vs agente aleatorio (verde).



34. Gráfico de vida restante al final del episodio en 100 juegos contra Aquamentus con modelos entrenados con Aquamentus, modelo estático (azul) vs modelo decaído (naranjo) vs agente aleatorio (verde).

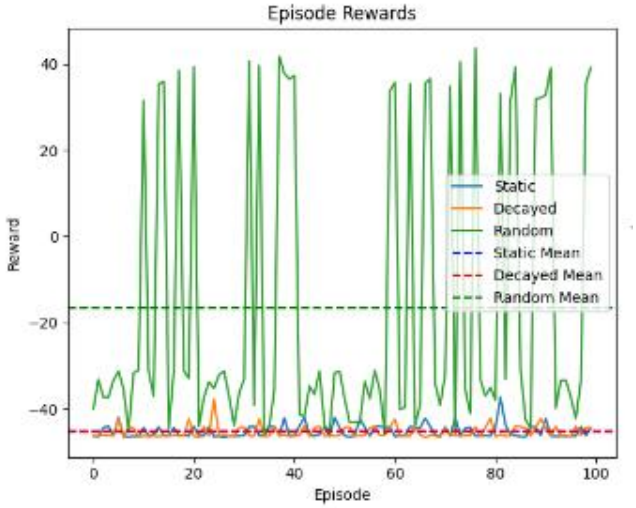


35. Link a video de juegos contra Aquamentus, modelo estático entrenado con Aquamentus: https://drive.google.com/file/d/1uC106fOuYsZCjjU57YqGMM_2hiAsA34p/view?usp=sharing

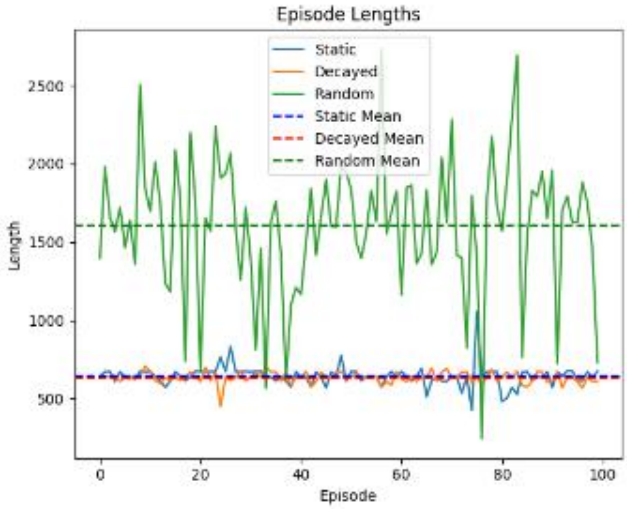
36. Link a video de juegos contra Aquamentus, modelo decaído entrenado con Aquamentus: https://drive.google.com/file/d/1FKn3lp8zi3FIqzeeDW1zccDeqSLsKj5p/view?usp=sharing

37. Link a video de juegos contra Aquamentus, agente aleatorio: https://drive.google.com/file/d/17JJArUlZ6KfcOwcx4oZDxQGcHkzOPF9l/view?usp=sharing

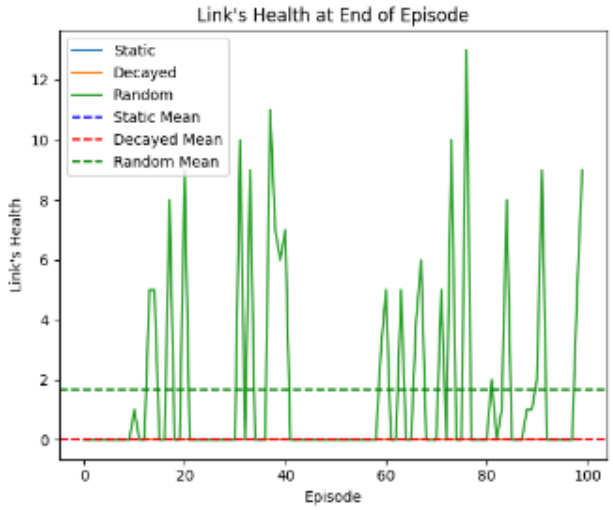
38. Gráfico de recompensas acumuladas por episodio en 100 juegos contra Manhandla con modelos entrenados con Aquamentus, modelo estático (azul) vs modelo decaído (naranjo) vs agente aleatorio (verde).



39. Gráfico de cantidad de *timesteps* por episodio en 100 juegos contra Manhandla con modelos entrenados con Aquamentus, modelo estático (azul) vs modelo decaído (naranjo) vs agente aleatorio (verde).



40. Gráfico de vida restante al final del episodio en 100 juegos contra Manhandla con modelos entrenados con Aquamentus, modelo estático (azul) vs modelo decaído (naranjo) vs agente aleatorio (verde).

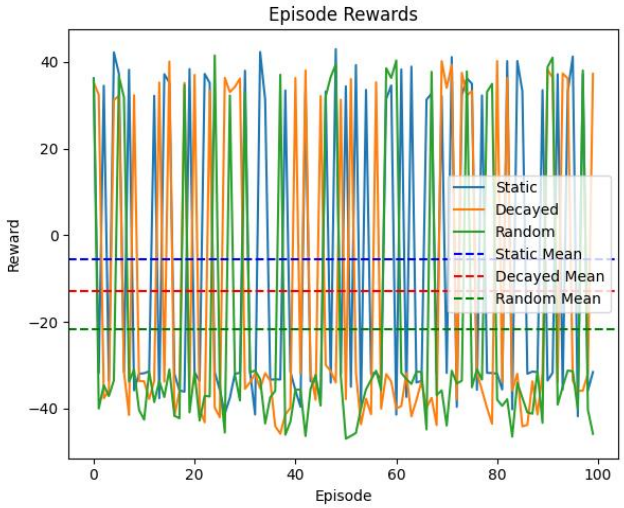


41. Link a video de juegos contra Manhandla, modelo estático entrenado con Aquamentus: https://drive.google.com/file/d/1rvJlwS6Y28yjehoBDD43HhHL8MaJk69k/view?usp=sharing

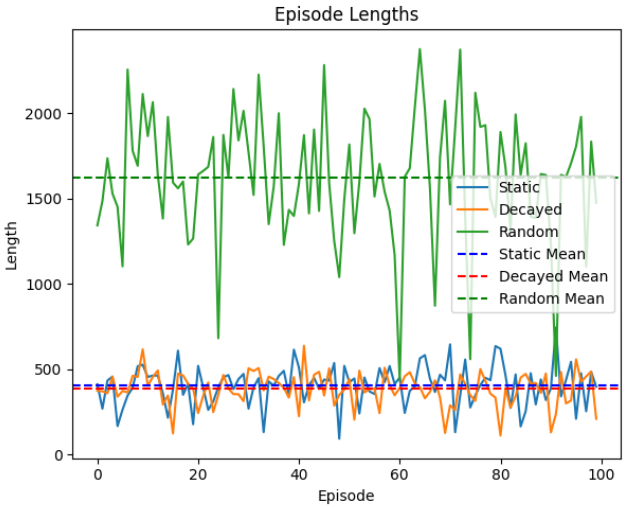
42. Link a video de juegos contra Manhandla, modelo decaído entrenado con Aquamentus: https://drive.google.com/file/d/1VgkbFkzY0u6MClpfxaLH_tsmJUS99zWC/view?usp=sharing

43. Link a video de juegos contra Manhandla, agente aleatorio: https://drive.google.com/file/d/1wz3LsnCCN9C72Nwl8fTTZf3wGjDlbyzL/view?usp=sharing

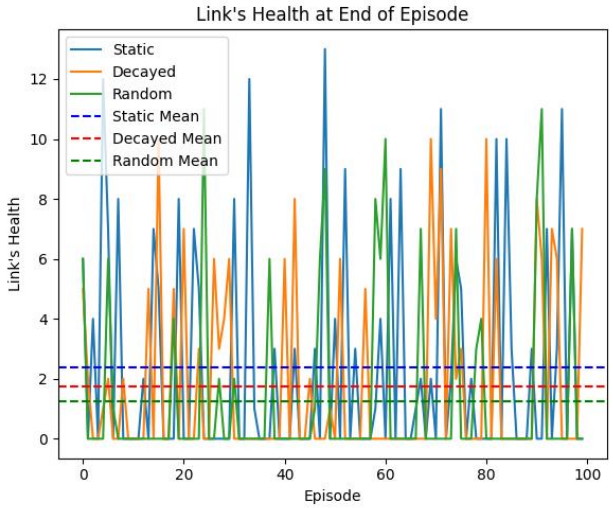
44. Gráfico de recompensas acumuladas por episodio en 100 juegos contra Manhandla con modelos entrenados con Manhandla, modelo estático (azul) vs modelo decaído (naranjo) vs agente aleatorio (verde).



45. Gráfico de cantidad de *timesteps* por episodio en 100 juegos contra Manhandla con modelos entrenados con Manhandla, modelo estático (azul) vs modelo decaído (naranjo) vs agente aleatorio (verde).

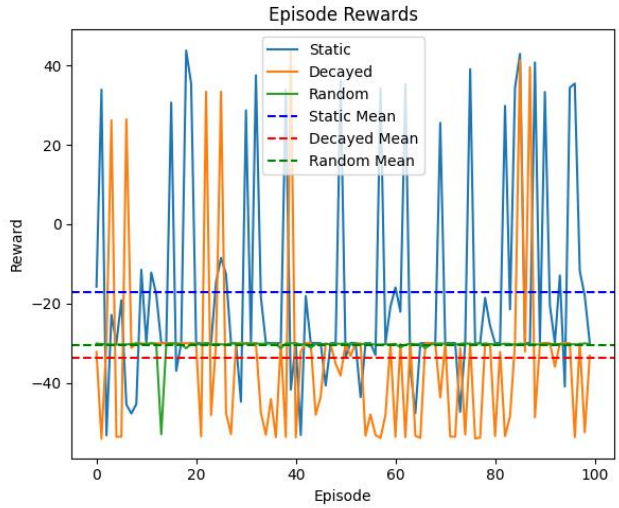


46. Gráfico de vida restante al final del episodio en 100 juegos contra Manhandla con modelos entrenados con Manhandla, modelo estático (azul) vs modelo decaído (naranjo) vs agente aleatorio (verde).

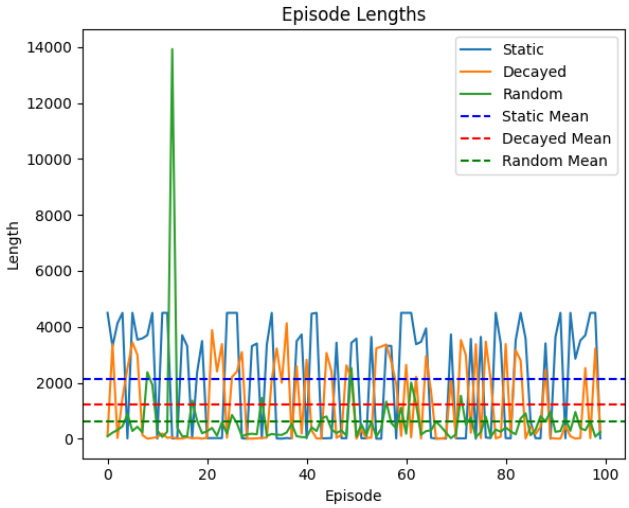


47. Link a video de juegos contra Manhandla, modelo estático entrenado con Manhandla: https://drive.google.com/file/d/1g8sYs019VSFGvAJfKWRUwbN9rZCnfQre/view?usp=sharing

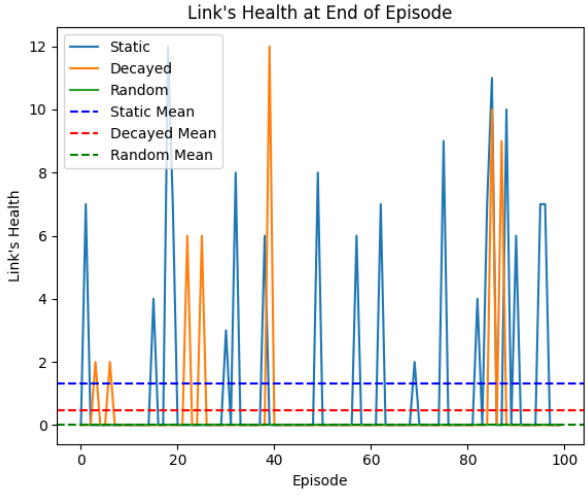
48. Link a video de juegos contra Manhandla, modelo decaído entrenado con Manhandla: https://drive.google.com/file/d/1PCVzD1m87dnNd98gmc9PtS3-xUmdZuXB/view?usp=sharing
49. Gráfico de recompensas acumuladas por episodio en 100 juegos contra Aquamentus con modelos entrenados con Manhandla, modelo estático (azul) vs modelo decaído (naranjo) vs agente aleatorio (verde).



50. Gráfico de cantidad de *timesteps* por episodio en 100 juegos contra Aquamentus con modelos entrenados con Manhandla, modelo estático (azul) vs modelo decaído (naranjo) vs agente aleatorio (verde).



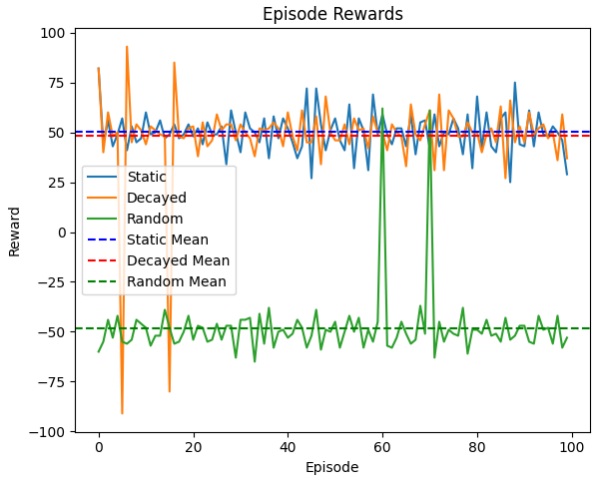
51. Gráfico de vida restante al final del episodio en 100 juegos contra Aquamentus con modelos entrenados con Manhandla, modelo estático (azul) vs modelo decaído (naranjo) vs agente aleatorio (verde).



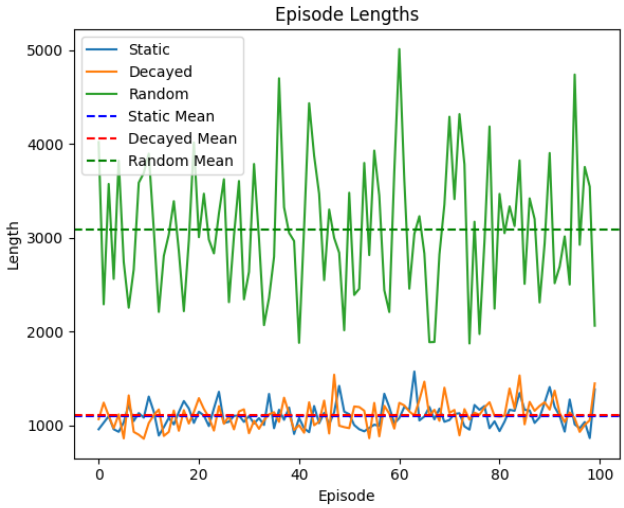
52. Link a video de juegos contra Aquamentus, modelo estático entrenado con Manhandla: https://drive.google.com/file/d/1yvI8Btibnlt-KaQQf2QytBfG7EjXodkp/view?usp=sharing

53. Link a video de juegos contra Aquamentus, modelo decaído entrenado con Manhandla: https://drive.google.com/file/d/1K_WEX5ocRc6S75bDe6GTe0tI3HhUc951/view?usp=sharing

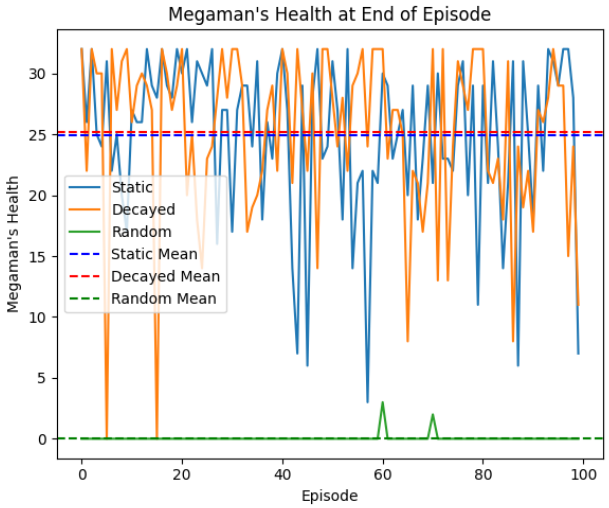
54. Gráfico de recompensas acumuladas por episodio en 100 juegos contra Chill Penguin con modelos entrenados con Chill Penguin, modelo estático (azul) vs modelo decaído (naranjo) vs agente aleatorio (verde).



55. Gráfico de cantidad de *timesteps* por episodio en 100 juegos contra Chill Penguin con modelos entrenados con Chill Penguin, modelo estático (azul) vs modelo decaído (naranjo) vs agente aleatorio (verde).



56. Gráfico de vida restante al final del episodio en 100 juegos contra Chill Penguin con modelos entrenados con Chill Penguin, modelo estático (azul) vs modelo decaído (naranjo) vs agente aleatorio (verde).

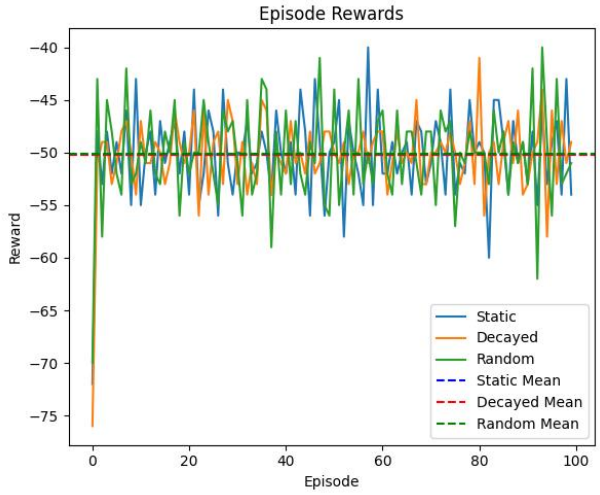


57. Link a video de juegos contra Chill Penguin, modelo estático entrenado con Chill Penguin: https://drive.google.com/file/d/1_fLP5J9wMRVCRP0yw82kDKciqjN0wURG/view?usp=sharing

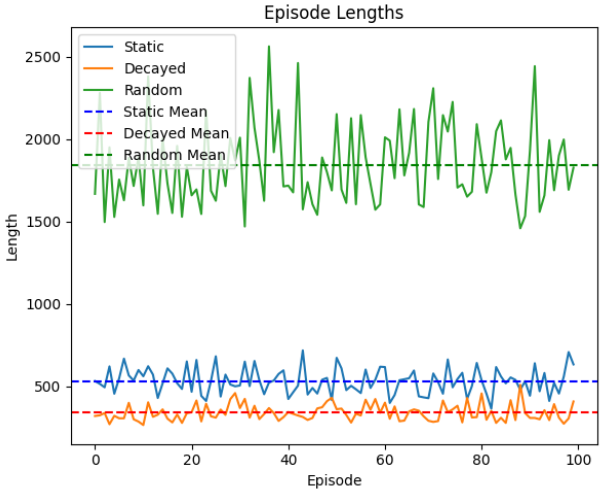
58. Link a video de juegos contra Chill Penguin, modelo decaído entrenado con Chill Penguin: https://drive.google.com/file/d/1d_P4Dn1EvjlP1UvfKOXq5Zn_QvqATrf7/view?usp=sharing

59. Link a video de juegos contra Chill Penguin, agente aleatorio: https://drive.google.com/file/d/1Ha0q02z4OFsWR6ZcUaFe-dvyUffAr13q/view?usp=sharing

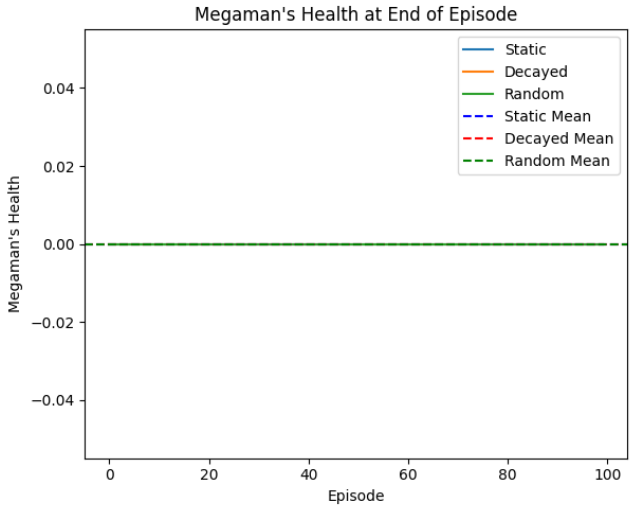
60. Gráfico de recompensas acumuladas por episodio en 100 juegos contra Spark Mandrill y arma base con modelos entrenados con Chill Penguin, modelo estático (azul) vs modelo decaído (naranjo) vs agente aleatorio (verde).



61. Gráfico de cantidad de *timesteps* por episodio en 100 juegos contra Spark Mandrill y arma base con modelos entrenados con Chill Penguin, modelo estático (azul) vs modelo decaído (naranjo) vs agente aleatorio (verde).



62. Gráfico de vida restante al final del episodio en 100 juegos contra Spark Mandrill y arma base con modelos entrenados con Chill Penguin, modelo estático (azul) vs modelo decaído (naranjo) vs agente aleatorio (verde).

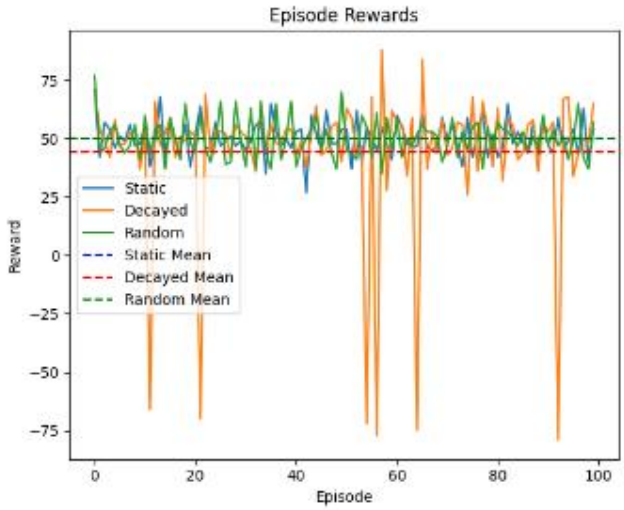


63. Link a video de juegos contra Spark Mandrill y arma base, modelo estático entrenado con Chill Penguin: https://drive.google.com/file/d/1fow49Jjx9I7sKl9O22Me23fqlbNS1gV5/view?usp=sharing

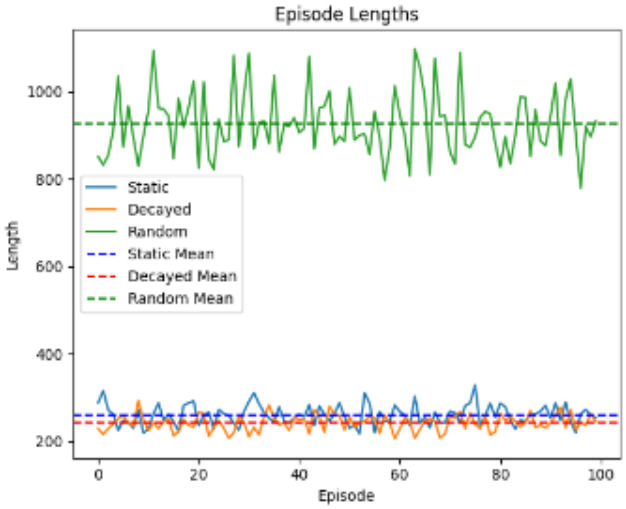
64. Link a video de juegos contra Spark Mandrill y arma base, modelo decaído entrenado con Chill Penguin: https://drive.google.com/file/d/1QbAth_WBSSfiGoCZ_HC87MfHzeaOmNb8/view?usp=sharing

65. Link a video de juegos contra Spark Mandrill y arma base, agente aleatorio: https://drive.google.com/file/d/1dhKQLeZ1THMGLALLaPYIYg3jPVLrQvsZ/view?usp=sharing

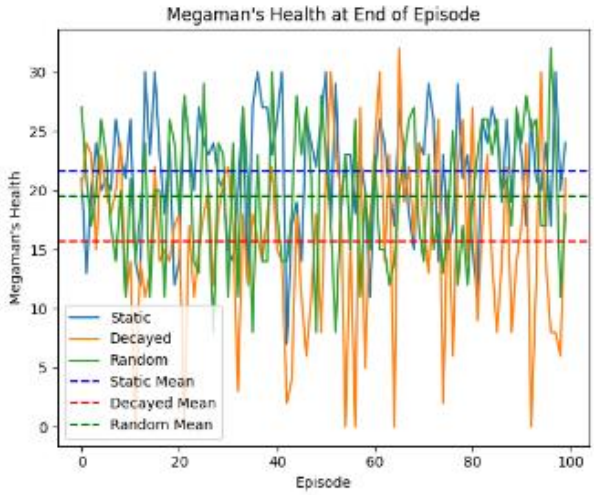
66. Gráfico de recompensas acumuladas por episodio en 100 juegos contra Spark Mandrill y *Shotgun Ice* con modelos entrenados con Chill Penguin, modelo estático (azul) vs modelo decaído (naranjo) vs agente aleatorio (verde).



67. Gráfico de cantidad de *timesteps* por episodio en 100 juegos contra Spark Mandrill y *Shotgun Ice* con modelos entrenados con Chill Penguin, modelo estático (azul) vs modelo decaído (naranjo) vs agente aleatorio (verde).



68. Gráfico de vida restante al final del episodio en 100 juegos contra Spark Mandrill y *Shotgun Ice* con modelos entrenados con Chill Penguin, modelo estático (azul) vs modelo decaído (naranjo) vs agente aleatorio (verde).

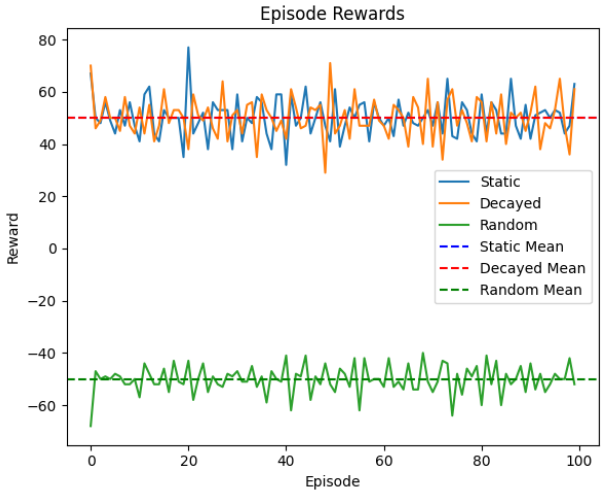


69. Link a video de juegos contra Spark Mandrill y *Shotgun Ice*, modelo estático entrenado con Chill Penguin: https://drive.google.com/file/d/1t19fuxahpitjnKupc4x34W-kb0Iix-Jh/view?usp=sharing

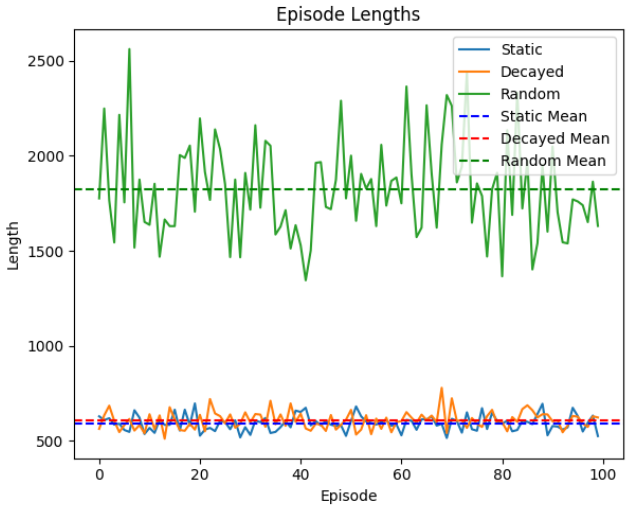
70. Link a video de juegos contra Spark Mandrill y *Shotgun Ice*, modelo decaído entrenado con Chill Penguin: https://drive.google.com/file/d/1s7jgMhjwbA3sI9OpVdFMx7EacYGmc1D3/view?usp=sharing

71. Link a video de juegos contra Spark Mandrill y *Shotgun Ice*, agente aleatorio: https://drive.google.com/file/d/1qoMU-N8aY17Am-khdS59a95mn0FMbsk1/view?usp=sharing

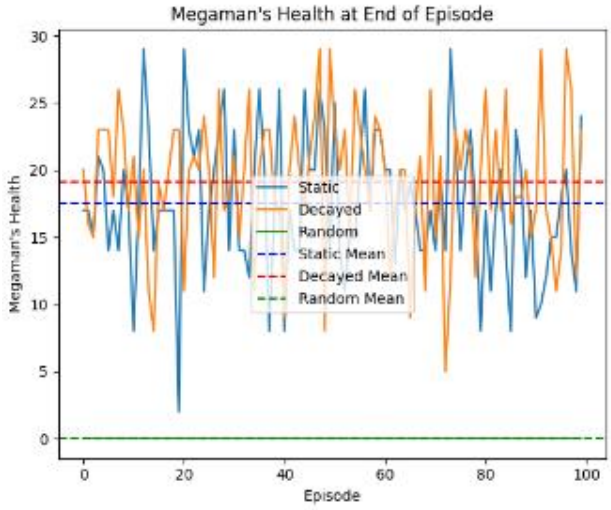
72. Gráfico de recompensas acumuladas por episodio en 100 juegos contra Spark Mandrill y arma base con modelos entrenados con Spark Mandrill y arma base, modelo estático (azul) vs modelo decaído (naranjo) vs agente aleatorio (verde).



73. Gráfico de cantidad de *timesteps* por episodio en 100 juegos contra Spark Mandrill y arma base con modelos entrenados con Spark Mandrill y arma base, modelo estático (azul) vs modelo decaído (naranjo) vs agente aleatorio (verde).



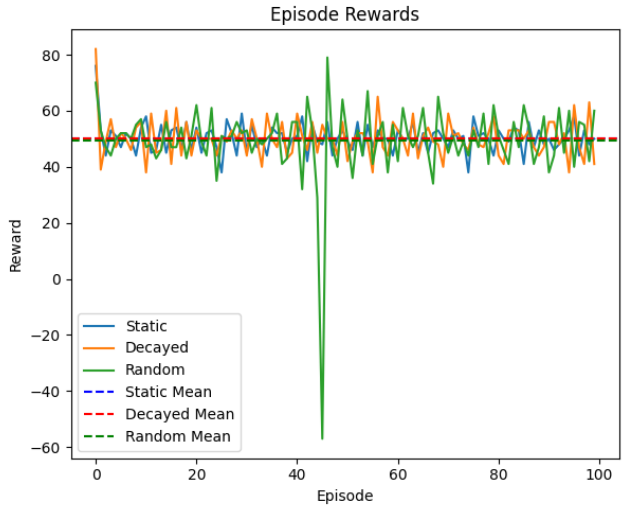
74. Gráfico de vida restante al final del episodio en 100 juegos contra Spark Mandrill y arma base con modelos entrenados con Spark Mandrill y arma base, modelo estático (azul) vs modelo decaído (naranjo) vs agente aleatorio (verde).



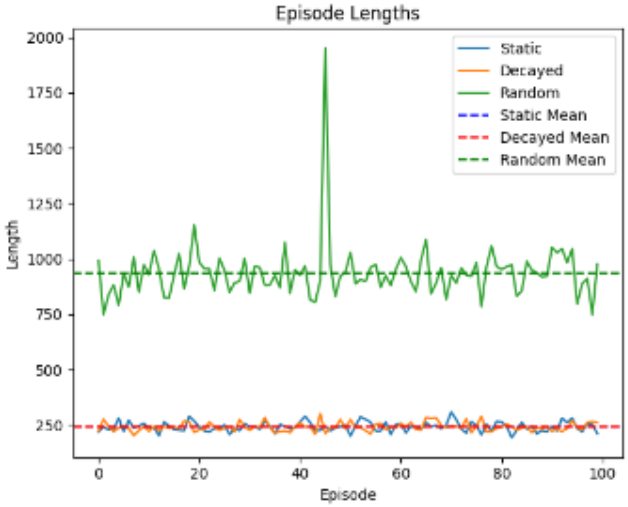
75. Link a video de juegos contra Spark Mandrill y arma base, modelo estático entrenado con Spark Mandrill y arma base: https://drive.google.com/file/d/13H0Lp8bepERThBa3fVdemAI7O6CU5EIe/view?usp=sharing

76. Link a video de juegos contra Spark Mandrill y arma base, modelo decaído entrenado con Spark Mandrill y arma base: https://drive.google.com/file/d/1q05E6hA-zl4uMjTiwgdJXZxgyAxH9L2H/view?usp=sharing

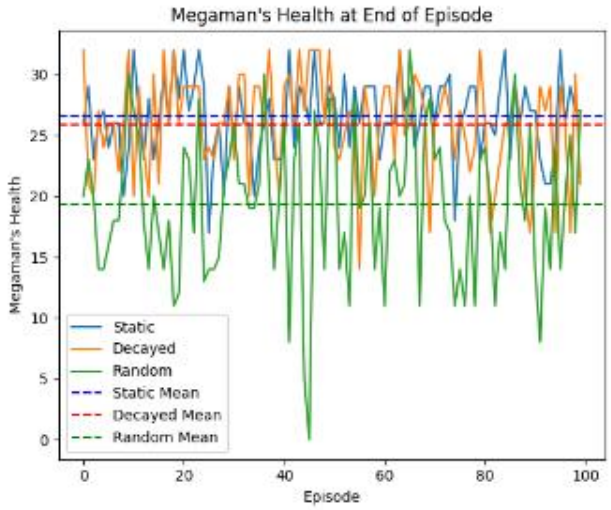
77. Gráfico de recompensas acumuladas por episodio en 100 juegos contra Spark Mandrill y *Shotgun Ice* con modelos entrenados con Spark Mandrill y arma base, modelo estático (azul) vs modelo decaído (naranjo) vs agente aleatorio (verde).



78. Gráfico de cantidad de *timesteps* por episodio en 100 juegos contra Spark Mandrill y *Shotgun Ice* con modelos entrenados con Spark Mandrill y arma base, modelo estático (azul) vs modelo decaído (naranjo) vs agente aleatorio (verde).



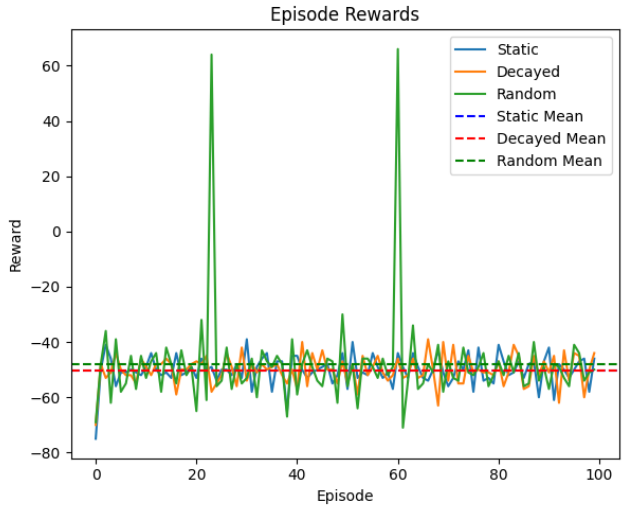
79. Gráfico de vida restante al final del episodio en 100 juegos contra Spark Mandrill y *Shotgun Ice* con modelos entrenados con Spark Mandrill y arma base, modelo estático (azul) vs modelo decaído (naranjo) vs agente aleatorio (verde).



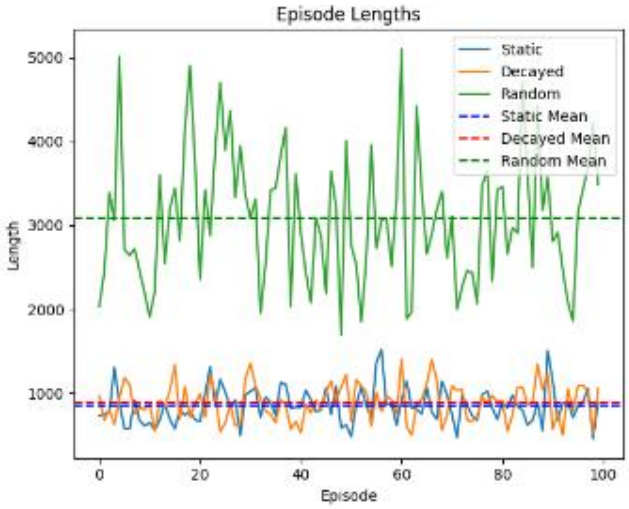
80. Link a video de juegos contra Spark Mandrill y *Shotgun Ice*, modelo estático entrenado con Spark Mandrill y arma base: https://drive.google.com/file/d/1CA4rpTkpzHM7GfAiIail-gSxLZXqqYM3/view?usp=sharing

81. Link a video de juegos contra Spark Mandrill y *Shotgun Ice*, modelo decaído entrenado con Spark Mandrill y arma base: https://drive.google.com/file/d/1hOS1Ud9kv7hcV9JcVGvDRqOLKpayTIH3/view?usp=sharing

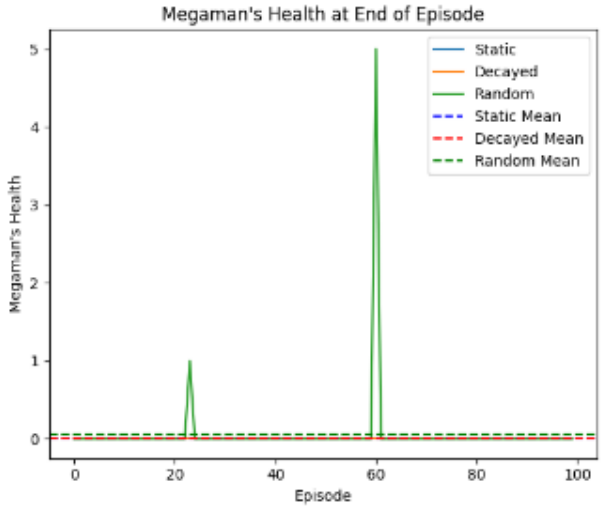
82. Gráfico de recompensas acumuladas por episodio en 100 juegos contra Chill Penguin con modelos entrenados con Spark Mandrill y arma base, modelo estático (azul) vs modelo decaído (naranjo) vs agente aleatorio (verde).



83. Gráfico de cantidad de *timesteps* por episodio en 100 juegos contra Chill Penguin con modelos entrenados con Spark Mandrill y arma base, modelo estático (azul) vs modelo decaído (naranjo) vs agente aleatorio (verde).



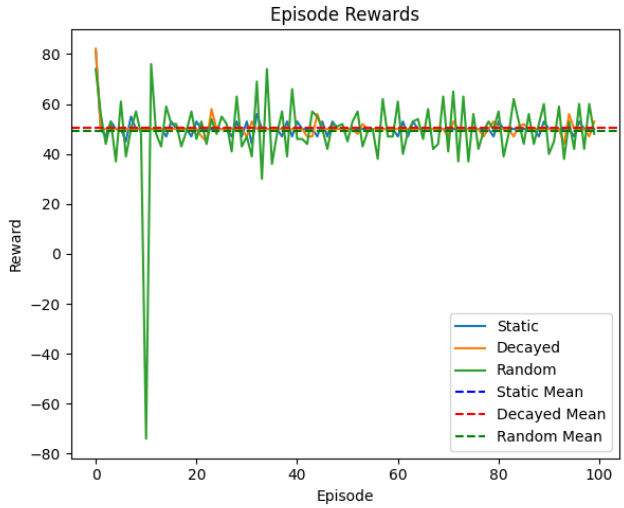
84. Gráfico de vida restante al final del episodio en 100 juegos contra Chill Penguin con modelos entrenados con Spark Mandrill y arma base, modelo estático (azul) vs modelo decaído (naranjo) vs agente aleatorio (verde).



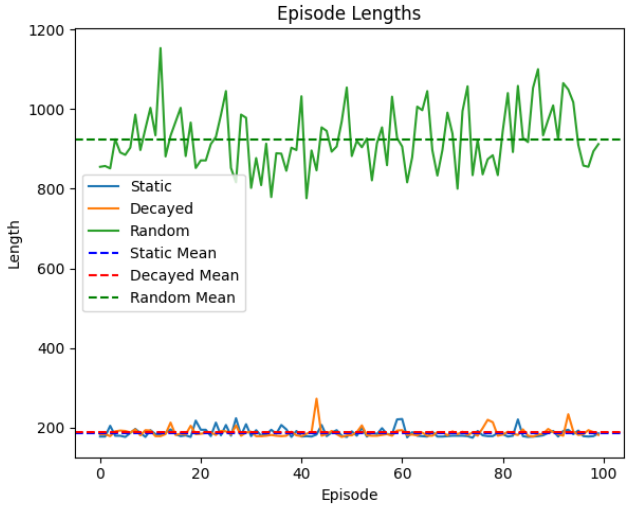
85. Link a video de juegos contra Chill Penguin, modelo estático entrenado con Spark Mandrill y arma base: https://drive.google.com/file/d/1XSX6ShhefeDgm62P6FhehSHzH0lodfW6/view?usp=sharing

86. Link a video de juegos contra Chill Penguin, modelo decaído entrenado con Spark Mandrill y arma base: https://drive.google.com/file/d/1KE1G4by5-NfIcbQ8XoGVCJUmzYihmgFy/view?usp=sharing

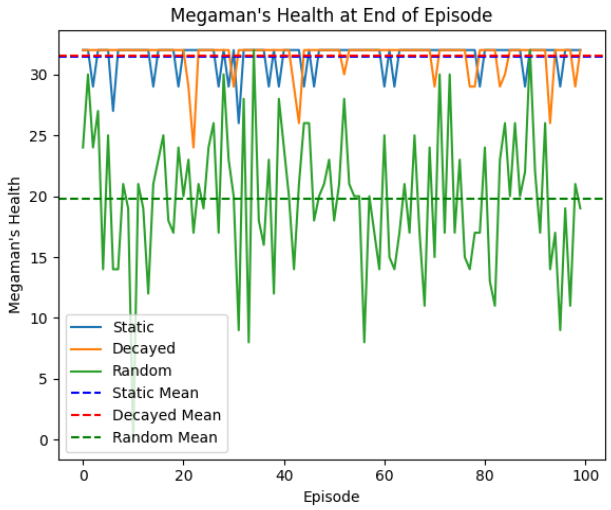
87. Gráfico de recompensas acumuladas por episodio en 100 juegos contra Spark Mandrill y *Shotgun Ice* con modelos entrenados con Spark Mandrill y *Shotgun Ice*, modelo estático (azul) vs modelo decaído (naranjo) vs agente aleatorio (verde).



88. Gráfico de cantidad de *timesteps* por episodio en 100 juegos contra Spark Mandrill y *Shotgun Ice* con modelos entrenados con Spark Mandrill y *Shotgun Ice*, modelo estático (azul) vs modelo decaído (naranjo) vs agente aleatorio (verde).



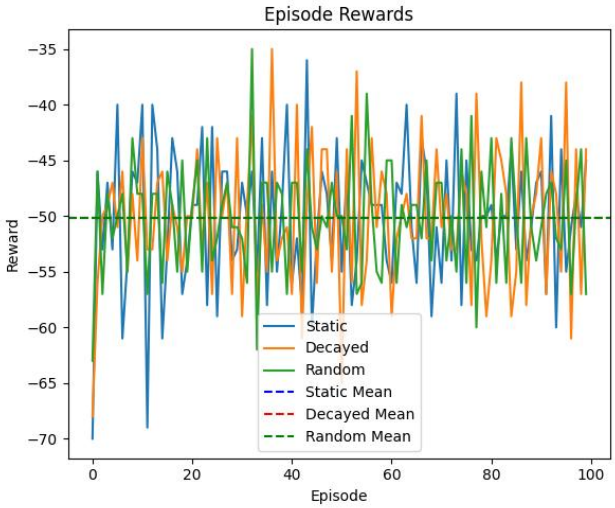
89. Gráfico de vida restante al final del episodio en 100 juegos contra Spark Mandrill y *Shotgun Ice* con modelos entrenados con Spark Mandrill y *Shotgun Ice*, modelo estático (azul) vs modelo decaído (naranjo) vs agente aleatorio (verde).



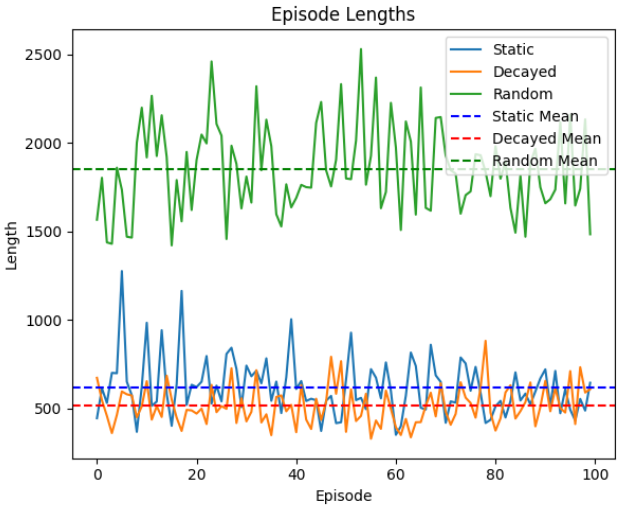
90. Link a video de juegos contra Spark Mandrill y *Shotgun Ice*, modelo estático entrenado con Spark Mandrill y *Shotgun Ice*: https://drive.google.com/file/d/11mlQI9HtETDmowsGE9A-pulEQ7ujU6yN/view?usp=sharing

91. Link a video de juegos contra Spark Mandrill y *Shotgun Ice*, modelo decaído entrenado con Spark Mandrill y *Shotgun Ice*: https://drive.google.com/file/d/1dj_Xs2h023KK0hRYrAM-25qVphumpAtj/view?usp=sharing

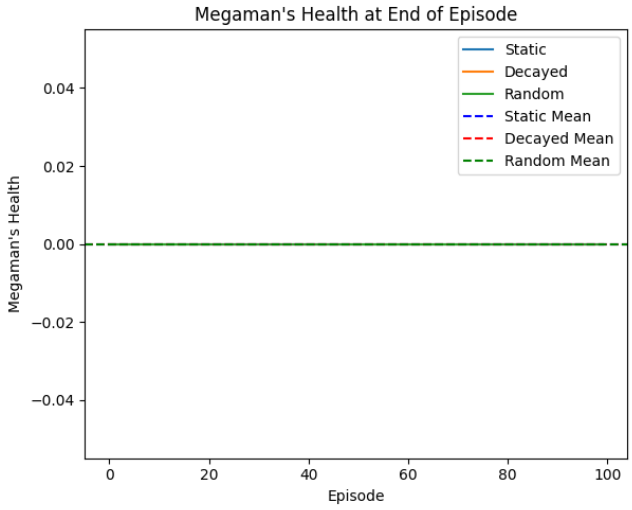
92. Gráfico de recompensas acumuladas por episodio en 100 juegos contra Spark Mandrill y arma base con modelos entrenados con Spark Mandrill y *Shotgun Ice*, modelo estático (azul) vs modelo decaído (naranjo) vs agente aleatorio (verde).



93. Gráfico de cantidad de *timesteps* por episodio en 100 juegos contra Spark Mandrill y arma base con modelos entrenados con Spark Mandrill y *Shotgun Ice*, modelo estático (azul) vs modelo decaído (naranjo) vs agente aleatorio (verde).



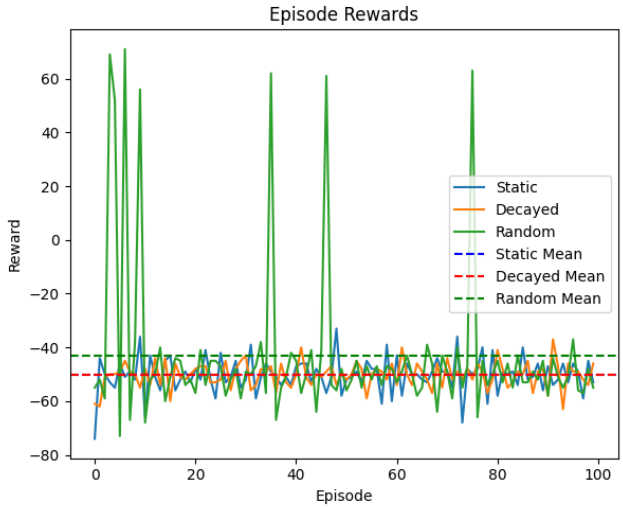
94. Gráfico de vida restante al final del episodio en 100 juegos contra Spark Mandrill y arma base con modelos entrenados con Spark Mandrill y *Shotgun Ice*, modelo estático (azul) vs modelo decaído (naranjo) vs agente aleatorio (verde).



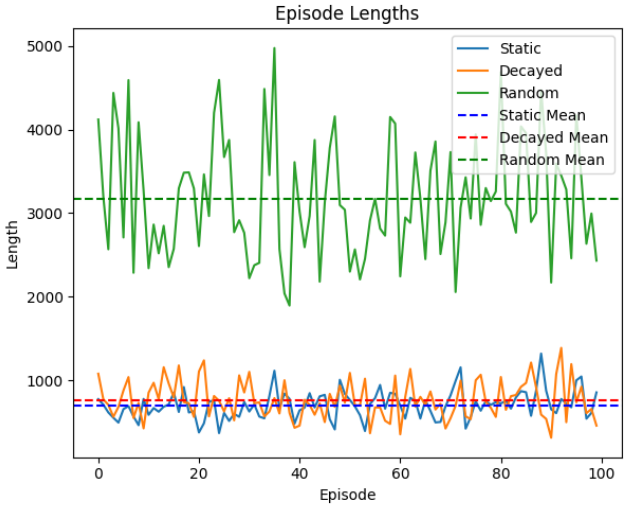
95. Link a video de juegos contra Spark Mandrill y arma base, modelo estático entrenado con Spark Mandrill y *Shotgun Ice*: https://drive.google.com/file/d/1F3k7Z1p1_wbqvA7DZy3B84eJirvf2ndO/view?usp=sharing

96. Link a video de juegos contra Spark Mandrill y arma base, modelo decaído entrenado con Spark Mandrill y *Shotgun Ice*: https://drive.google.com/file/d/1UfctzHgUTAwLb7YS0buHi0i3NKx1zMPI/view?usp=sharing

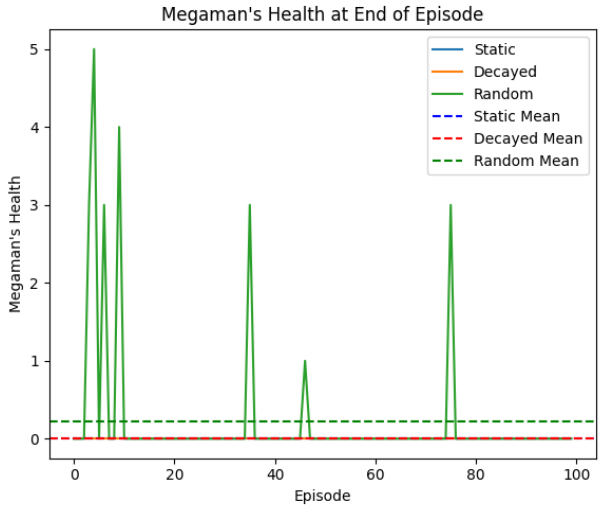
97. Gráfico de recompensas acumuladas por episodio en 100 juegos contra Chill Penguin con modelos entrenados con Spark Mandrill y *Shotgun Ice*, modelo estático (azul) vs modelo decaído (naranjo) vs agente aleatorio (verde).



98. Gráfico de cantidad de *timesteps* por episodio en 100 juegos contra Chill Penguin con modelos entrenados con Spark Mandrill y *Shotgun Ice*, modelo estático (azul) vs modelo decaído (naranjo) vs agente aleatorio (verde).



99. Gráfico de vida restante al final del episodio en 100 juegos contra Chill Penguin base con modelos entrenados con Spark Mandrill y *Shotgun Ice*, modelo estático (azul) vs modelo decaído (naranjo) vs agente aleatorio (verde).

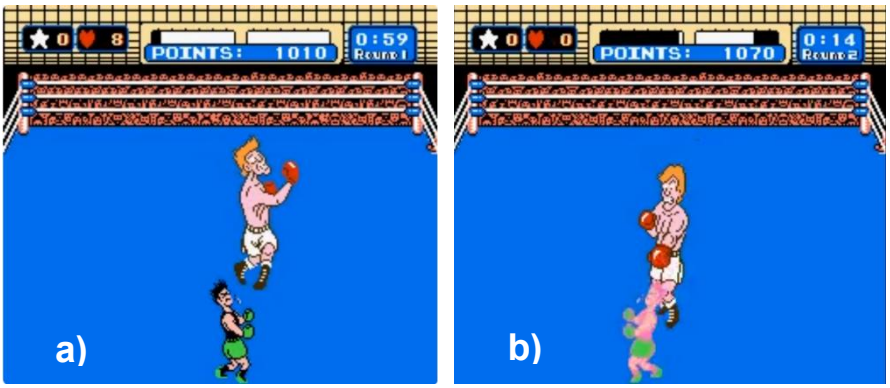


100. Link a video de juegos contra Chill Penguin, modelo estático entrenado con Spark Mandrill y *Shotgun Ice*: https://drive.google.com/file/d/1uFtaCkCxH169paggaeErkU30SqHDEJgO/view?usp=sharing

101. Link a video de juegos contra Chill Penguin, modelo decaído entrenado con Spark Mandrill y *Shotgun Ice*: https://drive.google.com/file/d/1O0ZX6S5wIU3r-t6E7zjGOUUmV6_fFlJo/view?usp=sharing

102. Detalle de los ataques de los oponentes de Punch Out!!:

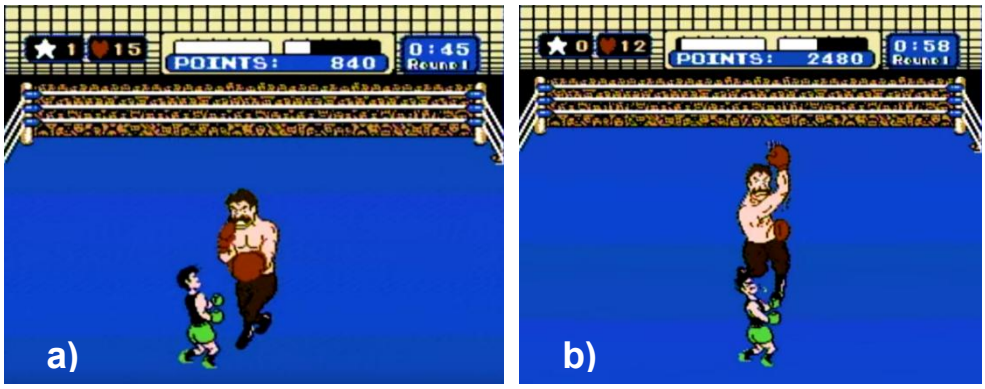
- Ataques de Glass Joe:
  - Gancho de derecha: Se prepara a la izquierda del agente antes de lanzarlo. Puede ser esquivado, bloqueado o contraatacado con un *jab* de izquierda. Le hace 11 puntos de daño a la salud de Mac.
  - *Jab* de izquierda: Glass Joe abre la boca, luego da un paso atrás antes de lanzar un *jab*. Puede ser esquivado, bloqueado o contraatacado con un golpe al cuerpo. Le hace 11 puntos de daño a la salud de Mac.
  - Puño de burla: Da un paso hacia la parte trasera del ring y le hace burla a Mac, antes de regresar al centro y lanzar un gancho. Se puede contrarrestar para derribarlo o noquearlo, dependiendo del tiempo. Le hace 11 puntos de daño a la salud de Mac.



Ataques de Glass Joe: a) Gancho de derecha y b) Jab de izquierda.

Ataques de Glass Joe: Puño de burla.

- Ataques de Von Kaiser:
  - *Jab* de izquierda: Von Kaiser sacude la cabeza antes de lanzar un *jab* de izquierda. Puede ser esquivado, bloqueado o contraatacado con un golpe al cuerpo. Le hace 9 puntos de daño a la salud de Mac.
  - *Uppercut* de derecha: Se agacha a la izquierda de Mac antes de lanzar un devastador *uppercut*. Se puede esquivar en cualquier dirección, pero no se puede bloquear. También se puede contrarrestar con un golpe al cuerpo con el puño izquierdo. Le hace 14 puntos de daño a la salud de Mac.



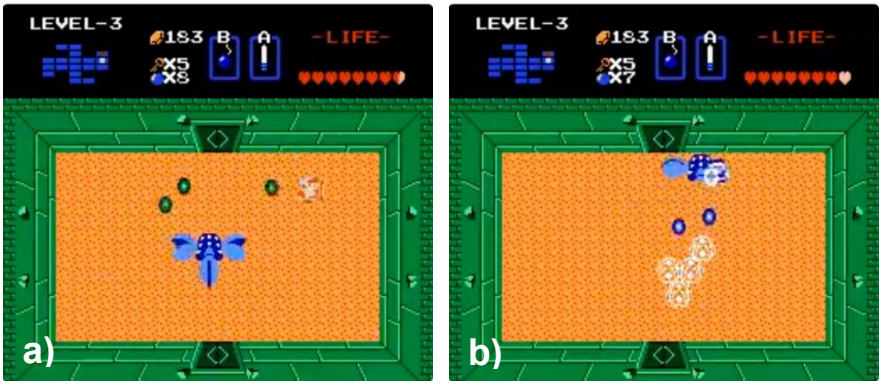
Ataques de Von Kaiser: a) Jab de izquierda y b) Uppercut de derecha.

103. Detalle de los ataques de los oponentes de The Legend of Zelda:

- Ataques de Aquamentus:
  - Bolas de Fuego: Aquamentus camina lentamente hacia adelante y atrás (hacia la izquierda y derecha en la pantalla) y periódicamente dispara un grupo de tres bolas de fuego desde su boca. Link puede bloquear las bolas de fuego con su escudo. Si Aquamentus toca a Link, le quita un cuarto de corazón, y si lo golpea con una bola de fuego, le quita medio corazón.

Ataques de Aquamentus: Bolas de Fuego.

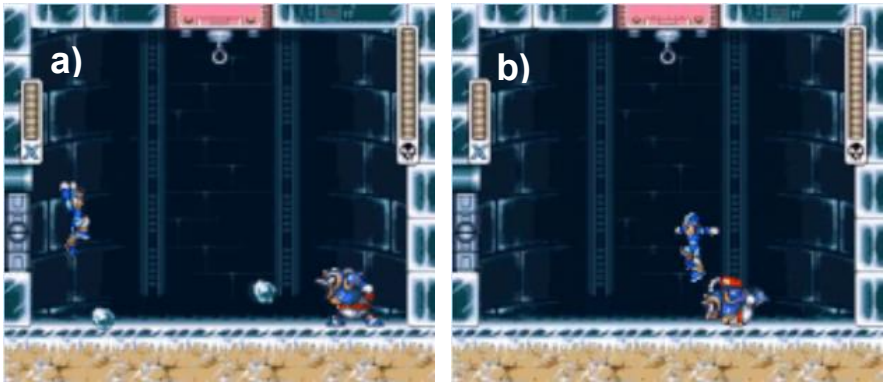
- Ataques de Manhandla:
  - Bolas de Fuego: Manhandla se mueve rápidamente por toda la habitación, disparando bolas de fuego periódicamente. Si una bola de fuego golpea a Link, le quita medio corazón.
  - Embestida: Manhandla embiste en dirección a Link. Si lo toca, le quita medio corazón.



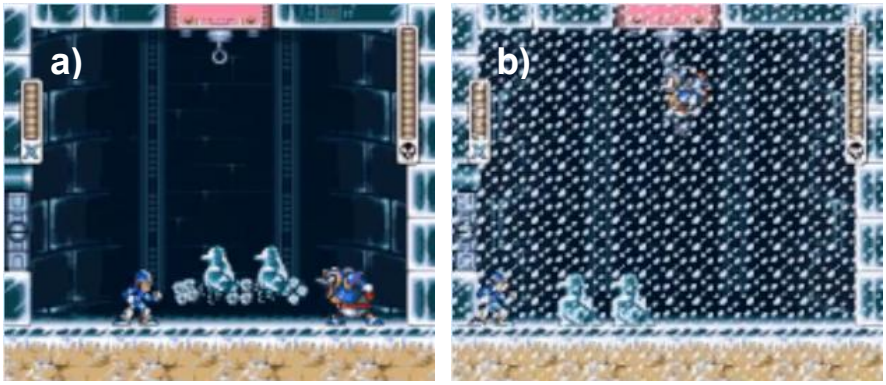
Ataques de Manhandla: a) Bolas de Fuego y b) Embestida.

104. Detalle de los ataques de los oponentes de Mega Man X:

- Ataques de Chill Penguin:
  - *Shotgun Ice*: Chill Penguin escupe cuatro bolas de hielo. Algunas viajan directamente por el aire, mientras que otras se deslizan por el suelo. Generan dos puntos de daño al entrar en contacto con X.
  - *Sliding*: Chill Penguin se desliza sobre su vientre, rebotando en las paredes al entrar en contacto. Si entra en contacto con X, le genera 6 puntos de daño.
  - Estatua de Hielo: Chill Penguin sopla nieve, creando dos esculturas de hielo con forma de pingüino que causan 4 puntos de daño por contacto. X puede destruirlas con ataques. Si X es atrapado por el aliento de nieve, queda congelado.
  - Tormenta de Nieve: Chill Penguin salta para jalar el gancho en el techo, lo que provoca una tormenta de nieve. La tormenta de nieve empuja a X y a cualquier escultura de hielo, las cuales se rompen al chocar con las paredes.

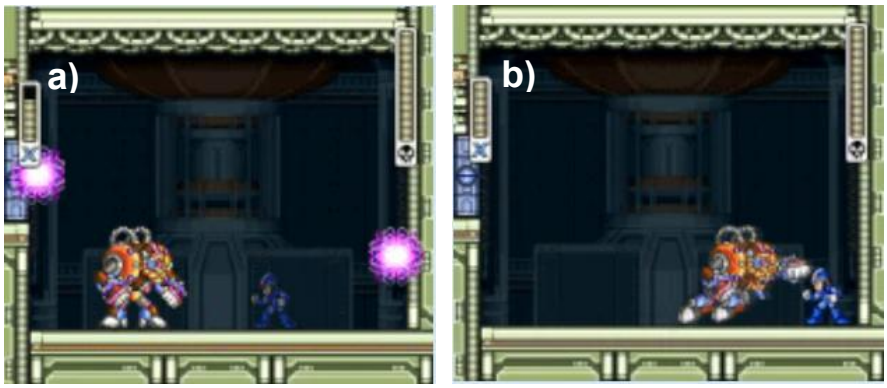


Ataques de Chill Penguin: a) Shotgun Ice y b) Sliding.



Ataques de Chill Penguin: a) Estatua de Hielo y b) Tormenta de Nieve.

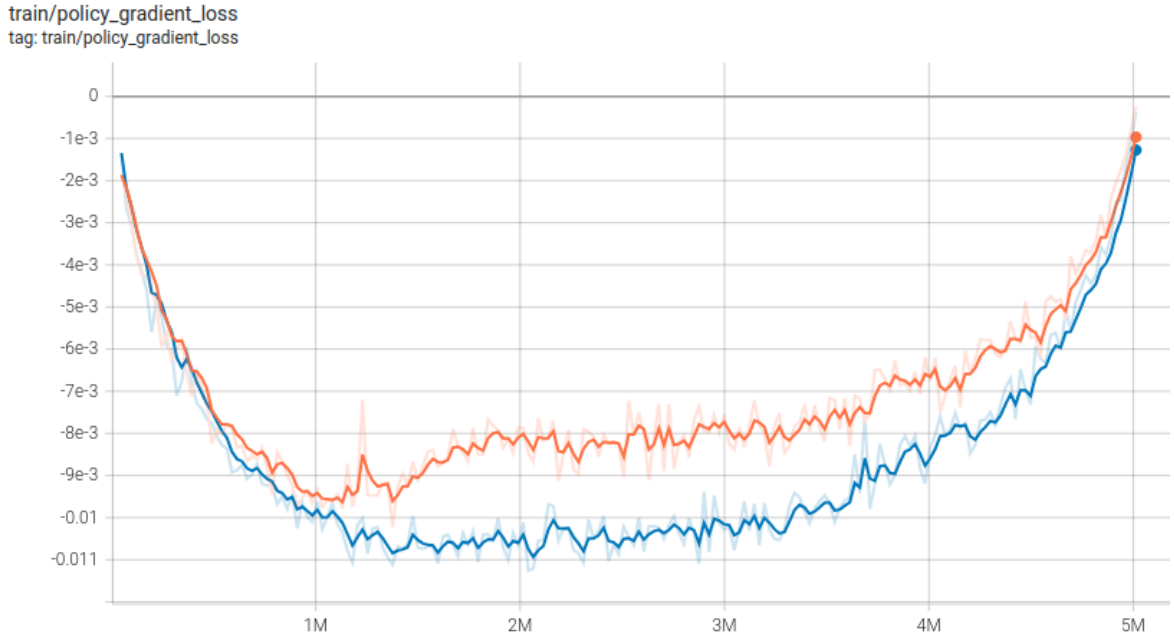
- Ataques de Spark Mandrill:
    - *Electric Spark*: Spark Mandrill golpea el suelo, emitiendo dos bolas de electricidad, una por cada lado, que trepan por las paredes hasta desaparecer en el techo. Generan 4 puntos de daño al entrar en contacto con X.
    - *Dash Punch*: Spark Mandrill se lanza a través de la habitación con el puño extendido. Sacude las paredes al entrar en contacto, interrumpiendo y previniendo que X se sostenga de las paredes. Si X está cerca del oponente, puede realizar un puñetazo de pie en su lugar. Si el oponente golpea a X, le hará 6 puntos de daño.
    - Agarrarse al Techo: Spark Mandrill salta a las tuberías en el techo, balanceándose a través de ellas. Se mueve por encima del jugador, intentando caer sobre él, si lo logra, le causa 6 puntos de daño.



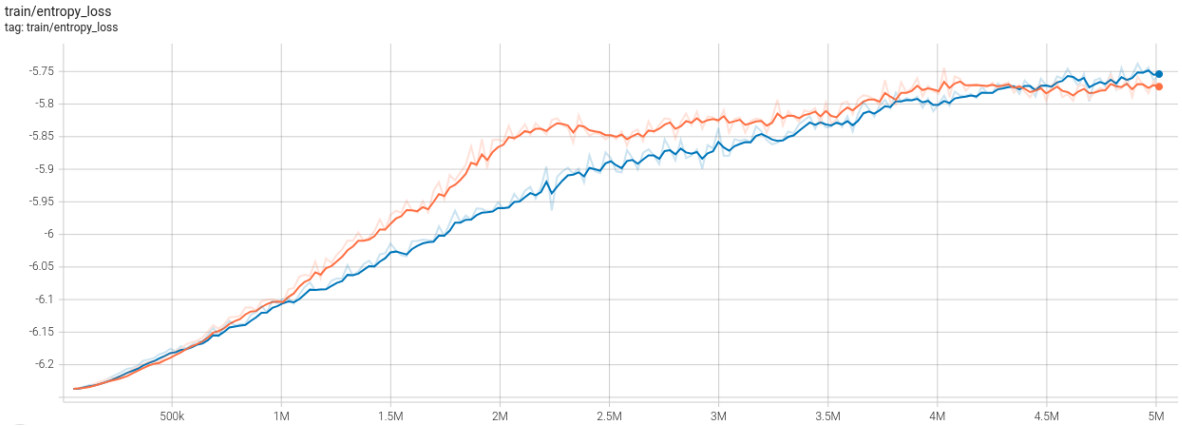
Ataques de Spark Mandrill: a) Electric Spark y b) Dash Punch.

Ataques de Spark Mandrill: Agarrarse del Techo.

105. Gráfico de la Pérdida del Gradiente de Política en el entrenamiento contra Glass Joe, modelo estático (azul) vs modelo decaído (naranjo).



106. Gráfico de la Pérdida de la Entropía en el entrenamiento contra Glass Joe, modelo estático (azul) vs modelo decaído (naranjo).



107. Gráfico de la Pérdida del Valor en el entrenamiento contra Glass Joe, modelo estático (azul) vs modelo decaído (naranjo).