



UNIVERSITAT OBERTA DE CATALUNYA (UOC)

MÁSTER UNIVERSITARIO EN CIENCIA DE DATOS (*DATA SCIENCE*)

TRABAJO FINAL DE MÁSTER

ÁREA: 4

Aprendizaje por refuerzo multiagente en entornos competitivos Atari: Transferencia de conocimiento de Pong a Quadrapong

Autor: Juan Manuel Camacho Lugo

Tutor: Luis Esteve Elfau

Profesor: Ismael Benito Altamirano

Barcelona, 5 de junio de 2025

Créditos/Copyright



Esta obra está sujeta a una licencia de Reconocimiento - NoComercial - SinObraDerivada 3.0 España de [CreativeCommons](#).

FICHA DEL TRABAJO FINAL

Título del trabajo:	Aprendizaje por refuerzo multiagente en entornos competitivos / colaborativos: Transferencia de conocimiento de Pong a Quadrapong.
Nombre del autor:	Juan Manuel Camacho Lugo
Nombre del colaborador/a docente:	Luis Esteve Elfau
Nombre del PRA:	Ismael Benito Altamirano
Fecha de entrega (mm/aaaa):	06/2025
Titulación o programa:	Máster Universitario en Ciencia de Datos
Área del Trabajo Final:	Área 4
Idioma del trabajo:	Español
Palabras clave	Reinforcement learning, Multi-Agent, Knowledge transfer

Abstract

In the context of Multi-Agent Reinforcement Learning ([MARL](#)), the ability of agents to learn from their interactions and adapt to dynamic environments is crucial. This study is based on the paper "Multiagent Cooperation and Competition with Deep Reinforcement Learning", with the aim of expanding its scope.

In this work, agent training is conducted in the classic Atari Pong game, optimizing the learning process to investigate how acquired knowledge can be effectively transferred to a more complex environment (Quadrabpong), where players must coordinate in teams of two. Through the analysis of different reward configurations, this research explores how agents balance competition and cooperation.

The findings of this study contribute to the understanding of how knowledge transfer can enhance multi-agent learning in more complex environments. Additionally, we observe how different agents, depending on the conditions, assess the application of communication and collaboration strategies within teams to achieve their objectives.

Keywords: Reinforcement learning, Multi-Agent, Knowledge transfer

Resumen

En el contexto del Aprendizaje por Refuerzo Multiagente (MARL), la capacidad de los agentes para aprender de sus interacciones y adaptarse a entornos dinámicos es fundamental. Este estudio toma como referencia el paper "Multiagent Cooperation and Competition with Deep Reinforcement Learning", con el propósito de expandir su alcance.

En este trabajo se toma el entrenamiento de agentes en el clásico juego de Atari Pong, optimizando su proceso de aprendizaje, con el objetivo de investigar cómo el conocimiento adquirido, puede transferirse eficazmente a un entorno más complejo (Quadrabpong), donde los jugadores deben coordinarse en equipos de dos. Mediante el análisis de diferentes configuraciones de recompensa, esta investigación explora cómo los agentes equilibran la competencia y la cooperación.

Los resultados de esta investigación contribuyen a la comprensión de cómo la transferencia de conocimiento puede mejorar el aprendizaje multiagente en entornos más complejos. Además, se observa cómo los diferentes agentes, según las condiciones, evalúan la aplicación de estrategias de comunicación y colaboración dentro de los equipos a la hora de alcanzar sus objetivos.

Palabras clave: Aprendizaje por refuerzo, Multiagente, Transferencia de conocimiento

Índice general

Abstract	III
Resumen	IV
Índice	V
Lista de Figuras	VII
Lista de Tablas	VIII
Lista de Ecuaciones	1
1 Introducción	2
1.1 Contexto	2
1.2 Justificación	3
1.3 Motivación	3
1.4 Objetivos	4
1.5 Enfoque y metodología	5
1.6 Sostenibilidad, diversidad y desafíos ético/sociales	6
1.6.1 Sostenibilidad	6
1.6.2 Diversidad, género y derechos humanos	7
1.6.3 Comportamiento ético y responsabilidad social	7
1.7 Planificación	7
1.7.1 FASE 1 (19/02 - 09/03) Introducción.	7
1.7.2 FASE 2 (10/03 - 30/03) Estado del arte.	8
1.7.3 FASE 3 (31/03 - 04/05) Diseño e implementación.	8
1.7.4 FASE 4 (05/05 - 18/05) Presentación de resultados.	8
1.7.5 FASE 5 (19/05 - 25/05) Conclusiones y trabajo futuro.	8
1.7.6 FASE 6 (26/05 - 06/06) Finalización del proyecto.	8
2 Estado del arte	10

2.1	Introducción al Aprendizaje por Refuerzo	10
2.1.1	Proceso	11
2.1.2	Evolución	13
2.1.3	Enfoques y Métodos	15
2.1.4	Aplicaciones	22
2.2	Aprendizaje por refuerzo en entornos multiagente (MARL)	23
2.2.1	Aprendizaje individual vs. aprendizaje multiagente	23
2.2.2	Enfoques y Métodos	24
2.2.3	Retos y Desafíos	25
2.3	Transferencia de conocimiento	26
2.3.1	Transferencia de conocimiento a entornos multiagentes	27
2.3.2	Técnicas de transferencia de conocimiento	28
2.3.3	Retos y Desafíos	29
2.4	Líneas de investigación previas	30
2.5	Tendencias y Trabajo Futuro	31
3	Diseño e implementación	34
3.1	Herramientas de desarrollo y recursos utilizados	34
3.2	Establecer el entorno de Pong	36
3.3	Definir los agentes y sus modelos	37
3.4	Entrenamiento y aprendizaje de los agentes en Pong	39
3.4.1	Estructura del entrenamiento	39
3.4.2	Ciclo de aprendizaje de los agentes	41
3.4.3	Configuración del entrenamiento en Pong	42
3.5	Establecer el entorno de Quadrapong	43
3.6	Transferencia de conocimiento a Quadrapong	45
3.6.1	Técnica empleada	46
3.6.2	Asignación de agentes y equipos	46
3.7	Entrenamiento y aprendizaje de los agentes en Quadrapong.	47
3.7.1	Estructura del entrenamiento desde cero	47
3.7.2	Ciclo de aprendizaje de los agentes en entrenamiento Quadrapong desde cero	48
3.7.3	Estructura del entrenamiento después de transferencia de conocimiento .	48
3.7.4	Ciclo de aprendizaje de los agentes en entrenamiento Quadrapong con Transfer Learning y Fine-tuning	49
3.7.5	Configuración del entrenamiento en Quadrapong	50
3.8	Recolección de datos y métricas	50

3.9	Pruebas y ajustes	51
3.9.1	Pruebas en Pong	51
3.9.2	Pruebas en Quadrapong	52
4	Presentación de resultados	54
4.1	Metodología de evaluación	54
4.2	Análisis de recompensas en Pong	55
4.3	Análisis de impactos de la pelota en Pong	56
4.4	Resumen y conclusiones preliminares en Pong	56
4.5	Comparación entre entrenamiento con y sin transferencia de conocimiento	58
4.5.1	Análisis de recompensas por equipos en Quadrapong	59
4.5.2	Análisis de impactos a la pelota por agentes en Quadrapong	61
4.5.3	Análisis de impactos a la pelota por equipos en Quadrapong	62
4.6	Resumen y conclusiones preliminares en Quadrapong	65
5	Conclusiones y trabajo futuro	66
Glosario		71
Bibliografía		72
A	Detección de objetos con Label Studio y YOLOv8	77
A.1	Etiquetado manual de datos con Label Studio	77
A.2	Entrenamiento de un modelo YOLOv8	78

Índice de figuras

1.1	Entorno Pong y Quadrapong [1] [2]	3
1.2	Diagrama de Gantt con la planificación	9
2.1	Tipos de aprendizajes automáticos	11
2.2	Elementos del aprendizaje por refuerzo	13
2.3	Estructura DQN	17
2.4	Tipos de enfoques para resolver problemas de aprendizaje por refuerzo	22
3.1	Espacio de observación	36

3.2	Red neuronal convolucional	39
3.3	Diagrama del ciclo de entrenamiento de un agente en Pong	42
3.4	Entorno Quadrapong [2]	43
3.5	Diagrama del ciclo de entrenamiento de un agente en Quadrapong con transferencia de conocimiento	50
4.1	Evolución de recompensas por rondas en Pong	55
4.2	Evolución de impactos por rondas en Pong	56
4.3	Impactos y recompensas por ronda agente second_0 en Pong	57
4.4	Impactos y recompensas por ronda agente first_0 en Pong	57
4.5	Impactos durante cinco partidas de cada agente en Pong	57
4.6	victorias durante cinco partidas de cada agente en Pong	57
4.7	Evolución de recompensas por Rondas y Equipos (Sin transferencia de conocimiento)	60
4.8	Evolución de recompensas por Rondas y Equipos (Con transferencia de conocimiento)	60
4.9	Evolución individual de impactos por Rondas (Sin transferencia de conocimiento)	61
4.10	Evolución individual de impactos por Rondas (Con transferencia de conocimiento)	62
4.11	Evolución por equipos de impactos por Rondas (Sin transferencia de conocimiento)	63
4.12	Impactos y recompensas por ronda, Team 1 (Sin transferencia de conocimiento)	63
4.13	Impactos y recompensas por ronda, Team 2 (Sin transferencia de conocimiento)	63
4.14	Evolución por equipo de impactos por Rondas (Con transferencia de conocimiento)	64
4.15	Impactos y recompensas por ronda, Team 1 (Con transferencia de conocimiento)	64
4.16	Impactos y recompensas por ronda, Team 2 (Con transferencia de conocimiento)	64
A.1	Fotogramas del entorno de juego de Pong con las cajas delimitadoras[3]	78
A.2	Fotogramas del entorno de juego de Quadrapong con las cajas delimitadoras[3]	78
A.3	Detección de palas y pelota en Pong	79
A.4	Detección de palas y pelota en Quadrapong	79

Índice de Tablas

2.1	Importancia de las recompensas según valor de descuento	13
-----	---	----

3.1	Comparativa entre los entornos Pong y Quadrapong en PettingZoo.	45
3.2	Asignación de agentes y transferencia de pesos desde Pong a Quadrapong.	47
5.1	Comparativa de huella de carbono entre estrategias de entrenamiento	68

Listado de Ecuaciones

2.1	Función de valor del estado $v(s)$	12
2.2	Función de valor de una acción $q(s, a)$	13
2.3	Ecuación de Bellman (Q-learning)	16
2.4	Ecuación del valor objetivo en DQN	17
2.5	Ecuación de la función de pérdida en DQN	18
2.6	Término de entropía	19
2.7	Clipping	20
2.8	Función de pérdida del crítico	20
2.9	Término de entropía	20
2.10	Función objetivo para el algoritmo PPO	20

Capítulo 1

Introducción

1.1. Contexto

El Aprendizaje por Refuerzo en entornos multiagente ha demostrado ser clave en el desarrollo de sistemas inteligentes para juegos, robótica y vehículos autónomos entre otros. Pong [1] es un entorno ampliamente utilizado en la investigación de Aprendizaje por Refuerzo debido a su simplicidad y carácter competitivo. Sin embargo, la aplicación del conocimiento adquirido en Pong a entornos más complejos como Quadrapong [2] presenta nuevos desafíos, como la coordinación entre agentes y la adaptación a dinámicas de equipo.

En este trabajo se explorará el entrenamiento de agentes en el clásico juego de Atari Pong, y nos centraremos en la optimización del aprendizaje y la transferencia de conocimiento a entornos más complejos. En particular, se estudiará la aplicación de las estrategias aprendidas en Pong al entorno Quadrapong, un juego similar donde los jugadores deben coordinarse en equipos de dos. Podemos apreciar las características básicas de ambos entornos en la [Figura 1.1](#). En el nuevo entorno, se analizará la evolución del comportamiento de los agentes en función de diferentes configuraciones de entrenamientos. En el nuevo entorno contaremos con dos modelos, uno donde los agentes serán entrenados desde cero un cierto número de pasos. Y otro donde utilizaremos los modelos preentrenados en Pong y completaremos el entrenamiento el mismo número de pasos que en el caso anterior para adaptar el conocimiento al nuevo entorno. El estudio tomará como referencia el paper "Multiagent Cooperation and Competition with Deep Reinforcement Learning" [4] y buscará expandirlo con nuevas estrategias.

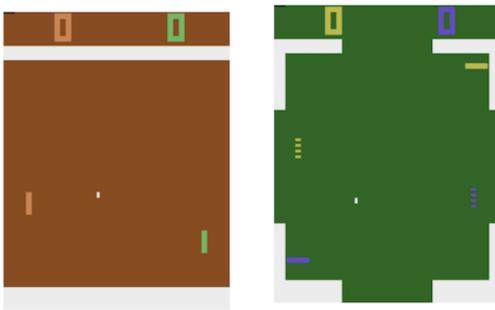


Figura 1.1: Entorno Pong y Quadrapong [1] [2]

1.2. Justificación

Este Trabajo Final de Máster pretende contribuir al Aprendizaje por Refuerzo Multiagente (MARL) estudiando la transferencia de conocimiento desde un entorno simple a otro más complejo. Se explorará cómo los agentes pueden mejorar su rendimiento en Quadrapong basándose en la experiencia previa adquirida en Pong y se analizará la capacidad de los modelos de aprendizaje para adaptarse a nuevos escenarios con un reentrenamiento mínimo. Además, se pretende evaluar el impacto de diferentes configuraciones de recompensa en la cooperación y competitividad entre agentes.

En la mayoría de sistemas reales la interacción entre agentes no se limita únicamente a la competencia directa, sino que involucra colaboración y coordinación. En este sentido, la transición desde Pong a un entorno más complejo como Quadrapong introduce desafíos adicionales, como la necesidad de comunicación entre compañeros de equipo y la toma de decisiones coordinadas.

De esta manera se podría contribuir al desarrollo de sistemas inteligentes que sean capaces de generalizar habilidades adquiridas en un contexto y aplicarlas en otros más complejos. En este ámbito, se podrían encontrar aplicaciones en la planificación de movimientos de robots autónomos, así como facilitar la toma de decisiones de vehículos autónomos o señalización en intersecciones, donde los agentes deben equilibrar competencia y cooperación.

1.3. Motivación

Mi interés en el Aprendizaje por Refuerzo Multiagente (MARL) surge de la creciente importancia de la inteligencia artificial en la resolución de problemas complejos que requieren la interacción entre múltiples agentes. Este tipo de aprendizaje ha demostrado ser una herramienta clave para desarrollar estrategias eficientes en entornos donde sea un requerimiento principal

la cooperación y la competencia.

El desafío de optimizar el aprendizaje de agentes y estudiar su capacidad de adaptación a nuevos entornos ha sido una de las principales motivaciones. La posibilidad de transferir conocimientos adquiridos en un entorno a otro me resulta interesante, debido a que, permitirá reutilizar y generalizar el aprendizaje sin necesidad de un entrenamiento desde cero.

Por otro lado, desde siempre los videojuegos han sido una de mis aficiones, no solo en el aspecto lúdico, sino que también me ha llamado la atención la complejidad de sus mecánicas y la toma de decisiones continuas que requieren. A medida que he ido estudiando el mundo de la inteligencia artificial, he visto como la aplicación del aprendizaje por refuerzo en este campo ha obtenido resultados impresionantes. Por este motivo, me despierta bastante interés estudiar cómo los agentes encuentran estrategias eficientes y la posibilidad de aplicarlas en diferentes circunstancias.

Por último, me gustaría tener la posibilidad de contribuir en este campo y explorar nuevas estrategias de aprendizaje, ya que, representa un reto apasionante que me permitirá ampliar mis conocimientos en aprendizaje profundo, optimización y transferencia de conocimiento.

1.4. Objetivos

Son varios los objetivos que se plantea alcanzar. Los cuales, se pueden diferenciar en dos categorías:

Objetivo principal:

Investigar la transferencia de conocimiento desde Pong a Quadrapong, y evaluar el impacto en la cooperación y competencia de los agentes.

Objetivos secundarios:

1. Entrenar agentes en Pong utilizando Aprendizaje por Refuerzo Multiagente.
2. Aplicar la política aprendida en Pong a Quadrapong y analizar su rendimiento.
3. Explorar diferentes configuraciones de recompensa (competitivas y colaborativas) para mejorar la coordinación entre agentes en Quadrapong.
4. Evaluar la evolución del comportamiento de los agentes en términos de estrategias y comunicación en equipos.
5. Optimizar el proceso de entrenamiento para reducir el consumo computacional manteniendo o mejorando el rendimiento.

1.5. Enfoque y metodología

Se empleará un enfoque experimental y comparativo, basado en simulaciones en entornos de PettingZoo [5] (plataforma estandarizada para entornos multiagente) y análisis cuantitativo del desempeño de los agentes. Se evaluarán distintos algoritmos de Aprendizaje por Refuerzo:

Estrategia de Investigación:

1. Deep Q-Network ([DQN](#)): Redes neuronales similares, entrenadas independientemente. El entrenamiento incluiría actualizar las redes neuronales con las recompensas que se obtienen de las acciones del agente.
2. Proximal Policy Optimization ([PPO](#)): Equilibrar exploración y explotación.
3. Multi-Agent Deep Deterministic Policy Gradient ([MADDPG](#)): Permitir que los agentes colaboren o compitan dependiendo del estado del juego y sus objetivos.

Técnicas de Optimización:

1. **Parameter Sharing:** Múltiples agentes comparten los mismos parámetros de red neuronal.
2. **Transfer Learning / Parameter Transfer:** Aplicar modelos entrenados en un entorno a otro diferente, minimizando la necesidad de entrenamiento desde cero.
3. **Fine-tuning:** Aplicar un proceso de optimización a modelos preentrenados, para facilitar su adaptación a nuevos entornos o mejorar su rendimiento en tareas específicas.

Herramientas:

1. **Entornos PettingZoo:**
 - Pong.
 - Quadrapong.
2. **Librerías / paquetes:**
 - Stable-Baselines3 [6].
 - NumPy ¹.
 - Gymnasium [7].
 - Supersuit [8].

¹Numpy, 2005. <https://numpy.org/es/>

- Pytorch ².
- OpenCV ³.
- TensorBoard ⁴.
- Codecarbon [9].

Posteriormente en el apartado ??, se describirán en detalle las herramientas de "software" y recursos "hardware" empleados en el desarrollo del proyecto.

1.6. Sostenibilidad, diversidad y desafíos ético/sociales

1.6.1. Sostenibilidad

Aunque este proyecto no tiene un impacto directo y grave en la sostenibilidad en términos de contaminación o agotamiento de recursos físicos, sí tiene un impacto indirecto en el consumo de energía y, por tanto, en la huella de carbono.

1. **Consumo de energía:** El entrenamiento de agentes, especialmente cuando se utilizan técnicas como DQN, PPO o MADDPG, puede ser muy demandante en términos de computación. Esto implica el uso de Graphics Processing Unit (**GPU**), para acelerar el proceso de aprendizaje, lo que aumenta el consumo de energía.
2. **Desperdicio y uso de recursos:** Dado que el proyecto se desarrolla en un entorno digital, no genera desechos físicos ni implica consumo directo de materias primas.

Para reducir la huella ecológica, se tomarán una serie de medidas como:

- Reducir el tiempo de entrenamiento en Quadrapong utilizando transfer learning.
- Utilizar GPUs optimizadas para Inteligencia Artificial (**IA**).

Relación con los Objetivos de Desarrollo Sostenible (**ODS**) ⁵:

- **ODS 7:** Energía asequible y no contaminante.
- **ODS 9:** Industria, innovación e infraestructura.
- **ODS 12:** Producción y consumo responsables

²Pytorch documentation, 2002. <https://pytorch.org>

³Opencv, 1999. <https://opencv.org>

⁴Tensorboard, 2015. <https://www.tensorflow.org/tensorboard?hl=es-419>

⁵ODS, 2015. <https://www.un.org/sustainabledevelopment/es/objetivos-de-desarrollo-sostenible/>

- **ODS 13:** Acción por el clima.

Una vez se realicen los entrenamientos, se analizará en detalle la huella de carbono para comprobar si se alcanzan los objetivos de eficiencia energética.

1.6.2. Diversidad, género y derechos humanos

Este proyecto tiene un impacto nulo o mínimo en términos de género, diversidad y derechos humanos, ya que:

1. No maneja datos humanos ni toma decisiones que afecten a personas.
2. No introduce barreras de accesibilidad.
3. No tiene riesgos en seguridad de la información.

1.6.3. Comportamiento ético y responsabilidad social

El impacto ético y social de este proyecto es mínimo en su estado actual, ya que se centra en el entrenamiento de agentes en entornos simulados sin implicaciones en datos privados, decisiones críticas o sesgos sociales. Sin embargo, si se aplicara a otras áreas más sensibles, podrían surgir desafíos éticos relacionados con la privacidad y la seguridad de los datos.

Por lo tanto, el proyecto no representa un riesgo ético o social directo y no se aprecia impacto relacionado con el género, la diversidad y los derechos humanos, aunque posibles aplicación futura fuera del contexto de este trabajo, podrían requerir un análisis más detallado para garantizar su uso responsable.

1.7. Planificación

La planificación del proyecto se basa en la consecución de una serie de hitos, los cuales se han adaptado a las entregas y fechas que a continuación se detallan y que se pueden observar en detalle en la [Figura 1.2](#).

1.7.1. FASE 1 (19/02 - 09/03) Introducción.

En esta fase se da contexto al proyecto a la vez que se expone la justificación y motivación que han llevado al desarrollo del mismo. Por otro lado, se marcan una serie de objetivos a cumplir y se presenta la metodología a seguir.

1.7.2. FASE 2 (10/03 - 30/03) Estado del arte.

Se realiza un estudio del aprendizaje por refuerzo multiagente (MARL) y transferencia de aprendizaje, aplicados a videojuegos. Se analizan ciertos algoritmos claves, así como su uso en entornos competitivos y cooperativos. Además, se revisan técnicas de transferencia de aprendizaje para evaluar cómo el conocimiento adquirido en Pong puede aplicarse en Quadrapong.

1.7.3. FASE 3 (31/03 - 04/05) Diseño e implementación.

- Herramientas de desarrollo y recursos utilizados.
- Establecer el entorno de Pong.
- Definir los agentes y sus modelos.
- Entrenamiento y aprendizaje de los agentes en Pong.
- Establecer el entorno de Quadrapong.
- Transferencia de conocimiento a Quadrapong.
- Entrenamiento y aprendizaje de los agentes en Quadrapong.
- Recolección de datos y métricas.
- Pruebas y ajustes.

1.7.4. FASE 4 (05/05 - 18/05) Presentación de resultados.

Llegados a este punto se analizan los resultados y se comprueba si se han alcanzados los objetivos marcados.

1.7.5. FASE 5 (19/05 - 25/05) Conclusiones y trabajo futuro.

Una vez analizados los resultados, estaremos en disposición de exponer las conclusiones más reseñables. A su vez, se establecen unas líneas de trabajo futuro.

1.7.6. FASE 6 (26/05 - 06/06) Finalización del proyecto.

Completadas todas las fases del proyecto, estamos en disposición de finalizar el trabajo con la entrega de la memoria definitiva.

1.7. Planificación

9

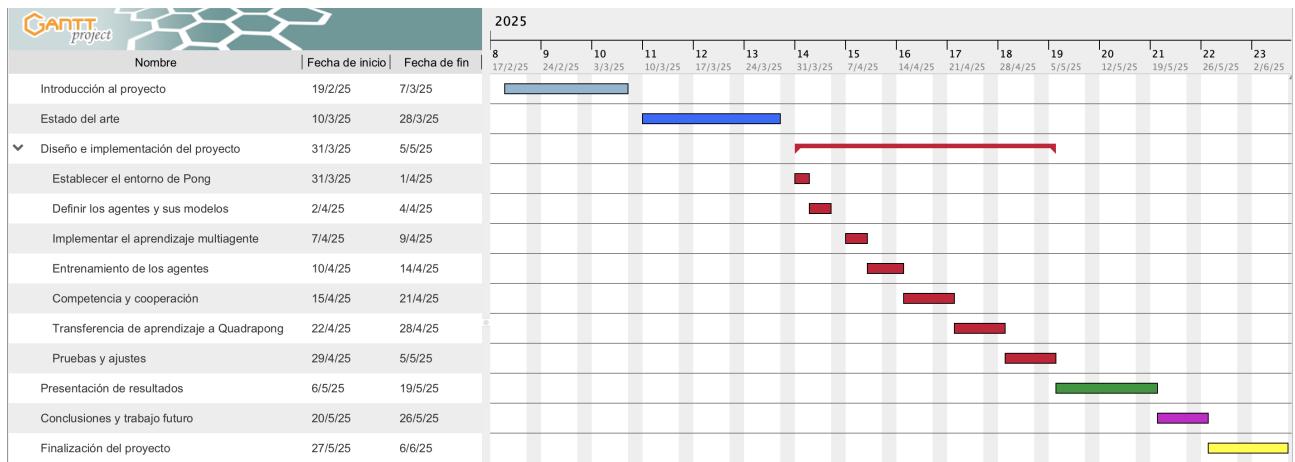


Figura 1.2: Diagrama de Gantt con la planificación

En el [capítulo 2](#), que se presenta a continuación, se detallará el estado del arte. Aquí, se introducirá el concepto de aprendizaje por refuerzo, se mostrarán las líneas de investigación previas, continuaremos con la aplicación del aprendizaje por refuerzo en entornos multiagente, nos centraremos en la transferencia de conocimiento entre entornos y señalaremos posibles tendencias o trabajos futuros. Una vez definido el estado del arte, pasaremos al [capítulo 3](#) donde detallaremos el diseño e implementación del trabajo. En este capítulo, estableceremos el entorno Pong, se definirán los agentes y sus modelos, se implementará el aprendizaje multiagente y se entrenarán los agentes. Después se establecerá el entorno Quadrapong, se entrenará un modelo desde cero sin transferencia de conocimiento y otro realizando la transferencia de conocimiento entre entornos y se realizarán las pruebas y ajustes pertinentes. Para finalizar lo relativo al estudio, en el [capítulo 4](#), se presentarán los resultados obtenidos durante el trabajo. Por último, en el [capítulo 5](#), se expondrán las conclusiones acorde a los resultados obtenidos y se indicarán posibles trabajos futuros.

Todo el código fuente, gráficos, documentación y demás materiales relacionados con este trabajo están disponibles públicamente en el siguiente repositorio de GitHub:

<https://github.com/JCAMLUG/TFM.git>

Este recurso permite consultar, reproducir y ampliar los resultados obtenidos durante el desarrollo del proyecto.

Capítulo 2

Estado del arte

2.1. Introducción al Aprendizaje por Refuerzo

En el campo de la IA la rama encargada del desarrollo de los algoritmos y modelos que permiten a las máquinas aprender, es el aprendizaje automático ó Machine Learning ([ML](#)). Este se lleva a cabo a partir de datos, los cuales permiten identificar patrones, hacer predicciones o tomar decisiones. De esta manera, se mejora el rendimiento en las diferentes tareas y se evita la necesidad de una programación previa. Cómo podemos observar en la [Figura 2.1](#), se pueden destacar varios tipos de aprendizajes automáticos:

- **Aprendizaje Supervisado:** En este caso, el modelo se entrena utilizando un conjunto de datos etiquetados, es decir, se le indica al modelo cuáles son las respuestas correctas. El modelo aprenderá a predecir la etiqueta o salida correcta de datos que no ha visto previamente gracias a los ejemplos que se le han proporcionado.
- **Aprendizaje No Supervisado:** A diferencia del caso anterior, el modelo recibe datos sin etiquetas. El modelo buscará patrones, estructuras o relaciones ocultas dentro de los datos sin la necesidad de que se le proporcionen ejemplos.
- **Aprendizaje por Refuerzo ó Reinforcement Learning ([RL](#)):** En este último tipo de aprendizaje, un agente tras interactuar con el entorno, gracias a recompensas y penalizaciones, aprende a tomar decisiones por medio de ensayo y error.

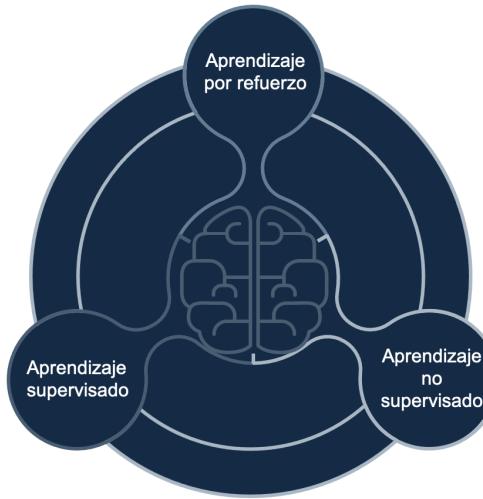


Figura 2.1: Tipos de aprendizajes automáticos

En el desarrollo del trabajo se utilizará RL para conseguir que los agentes aprendan a jugar a Pong.

2.1.1. Proceso

El aprendizaje por refuerzo sigue un proceso donde los agentes interactúan con el entorno por medio de acciones y, en función de ellas, reciben una serie de recompensas o penalizaciones que le indican cuán buena o mala ha sido la acción tomada. El objetivo es maximizar la recompensa acumulada a lo largo del tiempo, de esta manera se aprenderá cuál es la mejor estrategia o política para dicho entorno.

En el proceso se ven involucrados una serie de elementos, los cuales se reflejan en la Figura 2.2 y, que se detallan a continuación:

- **Agente.**

Es el ente que toma decisiones en un entorno determinado con el fin de alcanzar un objetivo y para ello se puede ayudar de diferentes elementos que se definen más adelante.

- **Acción (A_t).**

Corresponde con las decisiones que toma el agente en un determinado estado y que le permitirán evolucionar al siguiente estado. Es la manera que tiene el agente de interactuar con el entorno, repercutiendo en el próximo estado y en la próxima recompensa.

- **Recompensa / penalización.**

Se trata de la retroalimentación que el agente recibe al realizar cada acción. Esta señal escalar (R_t) puede ser positiva y por lo tanto, conllevar una recompensa, o negativa y

generar una penalización. En ambos casos, el agente contará con una información que le permitirá saber si sus acciones son correctas o no, y en consecuencia, aportar una influencia directa en las decisiones futuras del agente.

- **Entorno.**

Es el contexto en el que el agente se desenvuelve e interactúa. A medida que el agente ejecuta sus acciones, el entorno va cambiando su estado y generando una recompensa / penalización que ayudarán al agente a tomar nuevas decisiones.

- **Estado (S_t).**

Representa la situación actual del entorno en cada momento, la cual es observada por el agente, y contiene toda la información relevante del entorno necesaria para la toma de decisiones de este.

Entre los elementos que ayudan a los agentes a tomar decisiones podemos señalar la **política**, función que define el comportamiento del agente a la hora de seleccionar la acciones en cada estado, y puede ser determinista (cuando la acción viene determinada directamente por el estado) o estocástica (cuando pueden existir más de una acción a seleccionar en cada estado). En el segundo caso, se asignan probabilidades a las acciones, lo que ayuda a explorar el entorno. Esto se debe a que, en caso de volver a encontrarnos en un estado ya visitado, el agente puede escoger una acción diferente a la anterior que le permita avanzar a nuevos estados. Otro elemento que puede ayudar en la toma de decisiones del agente es la **función de valor**, que muestra lo óptimo que es un estado (o acción) acorde a las recompensas futuras que pueden obtenerse. Cuando hablamos de función de valor podemos diferenciar entre **función de valor del estado ($v(s)$)** y **función de valor de una acción $q(s,a)$** . La función de valor del estado, definida por la [Ecuación 2.1](#), se encarga de estimar la recompensa que el agente puede obtener en un estado (s), siguiendo una determinada política (π).

$$v(s) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t R_{t+1} \mid S_0 = s \right] \quad (2.1)$$

Ecuación 2.1: Función de valor del estado $v(s)$

Donde $\gamma \in [0,1]$, es el factor de descuento y determina la importancia de las recompensas futuras como podemos ver en la [Tabla 2.1](#). \mathbb{E}_π es el promedio de todas las posibles recompensas futuras que puede recibir el agente siguiendo la política. R_{t+1} la recompensa recibida en el paso $t+1$ tras haber realizado una acción en el paso previo. Todo ello, condicionado a que el agente tome como estado inicial (S_0) al estado (s).

Valor de γ	Interpretación 2	Comportamiento del agente
$\gamma=0$	Solo importa la recompensa inmediata.	Decisiones cortoplacistas
$0<\gamma<1$	Recompensas futuras valen menos cuanto más lejos estén	Balance entre corto y largo plazo
$\gamma=1$	Las recompensas futuras valen lo mismo que la actual	Decisiones a largo plazo

Tabla 2.1: Importancia de las recompensas según valor de descuento

Y la función de valor de una acción, definida por la [Ecuación 2.2](#), estima el valor esperado (recompensa total futura) al tomar una acción (a) en un estado (s) siguiendo la política (π) a partir de ese momento.

$$q(s, a) = \mathbb{E}_\pi \left[\sum_{t=0}^{\infty} \gamma^t R_{t+1} \mid S_0 = s, A_0 = a \right] \quad (2.2)$$

Ecuación 2.2: Función de valor de una acción $q(s, a)$

A diferencia de la función de valor del estado, en este caso todo está condicionado a que el agente se encuentre como estado inicial (S_0) al estado (s) y ejecute como acción inicial (A_0) la acción (a).

Por último, el agente puede contar con un **modelo**, que es una perspectiva del entorno, y le permite predecir el comportamiento de este para actuar en consecuencia. De esta manera, el agente puede predecir el próximo estado o el valor de la próxima recompensa.

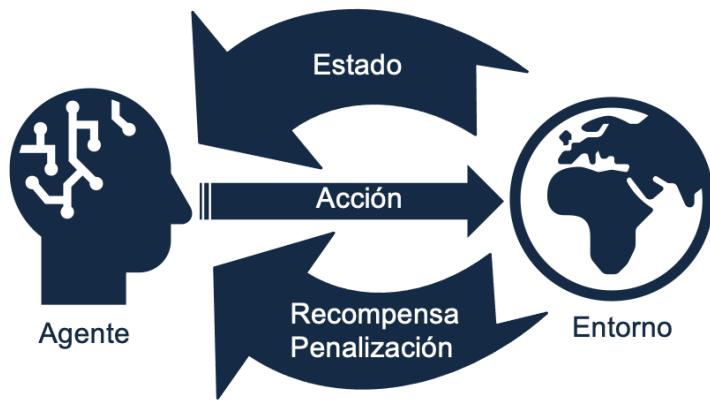


Figura 2.2: Elementos del aprendizaje por refuerzo

2.1.2. Evolución

Como describen Sutton and Barto [10] en su libro *Reinforcement Learning: An Introduction* (2018), el aprendizaje por refuerzo surge de la combinación de tres hilos independientes. El primer hilo, se centra en la psicología del aprendizaje animal, siguiendo el principio de ensayo

y error. Las primeras formulaciones en este campo fueron obra de Edward Thorndike ¹, quién mediante recompensas y castigos, observó la tendencia de los animales a seleccionar acciones. Uno de los primeros en pensar en la posibilidad de aplicar este método de aprendizaje a una computadora fue Alan Turing ², describiendo sistemas capaces de aprender mediante retroalimentación. El segundo hilo, está enfocado en el control óptimo y la programación dinámica, donde destacó la figura de Richard Bellman ³, quién definió una ecuación capaz de relacionar el valor de un estado con los posibles valores de estados futuros (*ecuación de Bellman* [11]), que junto al uso de procesos de decisión de Markov o Márkov Decision Process (MDP) [11], consiguió una base sólida para la toma de decisiones secuenciales. El tercer hilo, relacionado con el aprendizaje por diferencia temporal, se caracteriza por la actualización de predicciones a partir de cambios sucesivos en el tiempo. Siendo Arthur Samuel ⁴ el primero en aplicar un método de aprendizaje que incluía ideas de diferencia temporal al juego de damas. Pero, no fue hasta el desarrollo del Q-learning [12] por Chris Watkins ⁵, cuando se integraron los trabajos previos de los tres hilos para constituir el eje central del aprendizaje por refuerzo. El Q-learning puede definirse, de manera básica, como un método mediante el cual el agente aprende una política que le indica qué acción ejecutar en función del estado en que se encuentra con el objetivo de maximizar las recompensas futuras acumuladas.

La convergencia de estos tres enfoques dio lugar a una nueva generación de algoritmos capaces de aprender de manera autónoma a través de la interacción con el entorno. En este enclave, la combinación del aprendizaje por refuerzo con las redes neuronales profundas, ha dado lugar al aprendizaje por refuerzo profundo o Deep Reinforcement Learning (DRL). Un hito clave fue el desarrollo del algoritmo DQN por DeepMind ⁶ [13][14], que permitió entrenar agentes capaces de superar el rendimiento humano en videojuegos clásicos, utilizando únicamente píxeles como entrada. Este avance demostró la capacidad de los agentes para generalizar a partir de experiencias previas, y marcó el comienzo de una nueva etapa. Desde entonces, se han propuesto múltiples opciones entre las que se podrían señalar, Asynchronous Advantage Actor-Critic (A3C) [15], PPO [16] o Deep Deterministic Policy Gradient (DDPG) [17], cada una con enfoques específicos para entornos continuos, políticas deterministas o estabilidad en el entrenamiento. Estas técnicas han sido fundamentales consolidando el aprendizaje por refuerzo

¹Edward Lee Thorndike (1874-1949), psicólogo y pedagogo estadounidense. https://es.wikipedia.org/wiki/Edward_Thorndike

²Alan Mathison Turing (1912-1954), matemático, lógico, informático teórico, criptógrafo, filósofo y biólogo teórico británico. https://es.wikipedia.org/wiki/Alan_Turing

³Richard Ernest Bellman (1920-1984), matemático aplicado estadounidense. https://es.wikipedia.org/wiki/Richard_Bellman

⁴Arthur L. Samuel (1901-1990), pionero estadounidense en el campo de los juegos informáticos y la inteligencia artificial. https://es.wikipedia.org/wiki/Arthur_L._Samuel

⁵Christopher J.C.H. Watkins, Professor in Computer Science, Royal Holloway. <https://scholar.google.com/citations?user=v8Qhi0wAAAAJ&hl=es>

⁶DeepMind, compañía inglesa de investigación y desarrollo de inteligencia artificial, fundada en 2010 por Demis Hassabis. <https://deepmind.google/>

como una herramienta poderosa para la inteligencia artificial moderna.

2.1.3. Enfoques y Métodos

En la actualidad, el aprendizaje por refuerzo se ha consolidado como un área en constante evolución dentro de la inteligencia artificial. Como se aprecia en la [Figura 2.4](#), dependiendo del enfoque en el que nos basemos (función de valor, política o una combinación de ambos), existen diversos métodos que permiten abordar problemas complejos en entornos dinámicos e inciertos. A continuación, presentamos algunos de esos métodos:

- **Basados en funciones de valor.**

El objetivo es estimar una función que permita al agente tomar decisiones sin necesidad de una política y para ello solo tendrá en cuenta el valor (recompensa esperada si toma una acción determinada en un estado específico) de las acciones. De esta manera, conseguirá una estimación de la recompensa futura esperada desde un estado y la política se obtendrá a partir de estos valores. Los métodos basados en funciones de valor pueden diferenciarse según cuándo y cómo actualizan sus estimaciones:

- **Actualizan la función de valor solo al final de un episodio.**

Este tipo de métodos pertenecen a los Monte Carlo methods ([MC](#)) y requieren una experiencia completa, es decir, la actualización de la función de valor se realiza una vez finalizado el episodio, basándose en el promedio de los retornos observados para cada estado. Cuando un mismo estado es visitado múltiples veces dentro de un mismo episodio, existen dos enfoques posibles:

- **MC de primera visita**

Utiliza el retorno obtenido en la primera vez que se visita un estado en cada episodio para calcular el promedio de retornos observados.

- **MC de cada visita**

Considera todas las visitas al mismo estado en un episodio, utilizando el promedio de los retornos obtenidos en cada una de ellas para calcular el promedio de retornos observados.

Estos métodos son adecuados para problemas episódicos, donde no se requiere una respuesta inmediata después de cada acción y se puede esperar a la finalización del episodio para realizar la actualización del valor.

- **Actualizan la función de valor en cada paso.**

Estos métodos se encuentran dentro de la familia Temporal Difference methods ([TD](#)),

se trata de una combinación de los métodos de Montecarlo y la programación dinámica. A diferencia de los métodos de Montecarlo la actualización de la función de valor se lleva a cabo con estimaciones futuras y no se realiza al completar la experiencia completa. Son muy empleados en entornos en línea o continuos, donde es necesaria una actualización rápida después de cada acción. Entre los más destacados se pueden señalar:

- **DQN**

Los métodos DQN son una extensión del Q-learning, algoritmo de Aprendizaje por Refuerzo basado en valores. Durante el empleo de Q-learning, el agente aprende una función de valor $q(s,a)$ que estima la recompensa esperada acumulada al tomar la acción (a) en el estado (s) y posteriormente seguir una política óptima. La función se actualiza por medio de la ecuación de Bellman que se muestra en la [Ecuación 2.3](#).

$$Q(s_t, a_t) \leftarrow Q(s_t, a_t) + \alpha \left[r_t + \gamma \max_a Q(s_{t+1}, a) - Q(s_t, a_t) \right] \quad (2.3)$$

Ecuación 2.3: Ecuación de Bellman (Q-learning)

donde:

- α es la tasa de aprendizaje,
- γ es el factor de descuento,
- r_t es la recompensa inmediata.

Q-learning cuenta con una estructura relativamente simple, su funcionamiento es adecuado cuando nos encontramos ante espacios de estado discretos y reducidos. Pero no se comporta eficientemente cuando los estados son grandes o continuos, como entornos con observaciones visuales (imágenes).

Para resolver esta limitación, el método DQN utiliza una Convolutional Neural Network ([CNN](#)) o red neuronal convolucional, que aproxima la función $q(s,a;\theta)$, donde θ representa los pesos de la red. En la [Figura 2.3](#) podemos ver la estructura DQN.

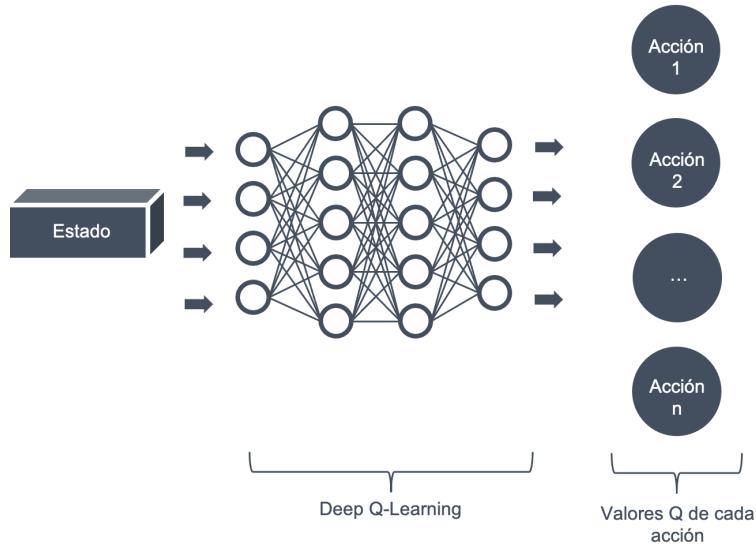


Figura 2.3: Estructura DQN

En DQN se utilizan dos redes neuronales con la misma arquitectura y a diferencia del Q-learning clásico, contamos con:

◊ **Red principal (online network):**

Es la red neuronal con parámetros θ , que se actualiza en cada paso. Esta red es la que predice los valores Q y se utiliza tanto para elegir acciones como para estimar el valor actual de una determinada acción en un estado dado.

◊ **Red objetivo (target network):**

Se introduce una segunda red, definida por la [Ecuación 2.4](#), con parámetros congelados (θ^-), utilizada para calcular el valor objetivo en la pérdida. Esta red se actualiza cada cierto número de pasos con los pesos de la red principal y su objetivo es mejorar la estabilidad.

$$y_t = r_t + \gamma \max_a Q(s_{t+1}, a; \theta^-) \quad (2.4)$$

Ecuación 2.4: Ecuación del valor objetivo en DQN

◊ **Replay Buffer:**

Estructura donde cada agente almacena sus propias experiencias pasadas (transiciones), esto permite romper la correlación entre las transiciones generadas al interactuar con el entorno. Las experiencias consecutivas están altamente correladas, dificultando el entrenamiento efectivo de redes neuronales, siendo necesario contar con datos Independent and Identically Distri-

buted (IID), para una convergencia estable. El Replay Buffer soluciona este problema proporcionando "batches" o lotes de experiencias, ayudando a que los datos estén menos correlados y favoreciendo la estabilidad y eficiencia del aprendizaje.

- ◊ **Función de pérdida:**

Como podemos ver en la [Ecuación 2.5](#), la red se entrena minimizando el error cuadrático entre el valor estimado $q(s_t, a_t; \theta)$ y el valor objetivo (y_t) :

$$\mathcal{L}(\theta) = \mathbb{E}_{(s,a,r,s') \sim \mathcal{D}} [(y_t - Q(s, a; \theta))^2] \quad (2.5)$$

Ecuación 2.5: Ecuación de la función de pérdida en DQN

donde \mathcal{D} es un lote de transiciones extraídas del Replay Buffer.

- ◊ **Exploración ϵ -greedy**

Durante el entrenamiento, se sigue una política que con probabilidad ϵ selecciona una acción aleatoria y con probabilidad $1-\epsilon$, selecciona la acción que maximiza Q . Este valor se reduce gradualmente para pasar de la exploración del entorno a explotación del mismo.

Esto permite dar solución a problemas que cuentan con muchos estados, facilitando al agente la posibilidad de aprender a interactuar gracias a la exploración eficiente del entorno.

- **State-Action-Reward-State-Action (SARSA)**

En este caso, el agente se ve envuelto en un ciclo que le permite ir aprendiendo de los errores que comete, de manera que va actualizando la función de valor según los resultados que obtiene de las acciones que realmente se ejecutan y no de estimaciones. Su empleo es útil cuando necesitamos tener en cuenta el comportamiento real del agente, esto hace que mantenga una linea segura, realista y conservadora.

- **Basados en políticas.**

Este enfoque se centra en aprender una política que maximice la recompensa esperada sin la necesidad de obtener una función de valor. Para ello, el agente realiza la elección de las acciones según la política que sigue y no el valor de dichas acciones. Podemos destacar:

- **REINFORCE**

Su comportamiento se centra en realizar la actualización de la política al final del proceso. El objetivo es ir ajustando la política para que aumente la posibilidad de que se produzcan las acciones que llevan a las mejores recompensas.

- **PPO**

Este algoritmo de aprendizaje por refuerzo, desarrollado en 2017 por John Schulman⁷, entrena la red de política de un agente, o lo que es lo mismo, la función que el agente utiliza para tomar decisiones. Esto evita actualizaciones agresivas de la política, realizando pequeñas actualizaciones por bloques o rondas y no al final del proceso. Esta manera de trabajar garantiza que la política actualizada no sea demasiado diferente a la anterior para que se produzcan mínimas varianzas en el entrenamiento.

La manera de evitar las actualizaciones agresivas es por medio de una función de perdida que se encarga de poner límites a la magnitud del cambio en la política en cada paso del entrenamiento. Esta función, también llamada función objetivo, está definida por una serie de términos, los cuales se describen a continuación, y que aseguran la maximización de la política mientras se mantienen la estabilidad y se favorece la exploración.

- **Ratio de probabilidad ($r_t(\theta)$):**

Como se aprecia en la [Ecuación 2.6](#), se trata del cociente entre la probabilidad que asigna la nueva política π_θ y la política anterior $\pi_{\theta_{\text{old}}}$ a la acción a_t tomada en el estado s_t .

$$r_t(\theta) = \frac{\pi_\theta(a_t|s_t)}{\pi_{\theta_{\text{old}}}(a_t|s_t)} \quad (2.6)$$

Ecuación 2.6: Término de entropía

Este ratio indica cuánto ha cambiado la política en cuanto a la probabilidad de la acción ejecutada.

- **Estimación de la ventaja (\hat{A}_t):**

Representa cuánto mejor (o peor) es la acción a_t en el estado s_t comparada con la expectativa general para ese estado. Se encarga de guiar la actualización, acciones con ventaja positiva deberían aumentar su probabilidad, y con ventaja negativa disminuirla.

- **Clipping (recorte)**

El recorte como se aprecia en la [Ecuación 2.7](#), limita el ratio para evitar que su valor se desvíe demasiado de 1, es decir, para prevenir cambios grandes en la política.

⁷John Schulman, Doctor en Ciencias de la Computación en la Universidad de California Berkeley. <http://joschu.net>

$$\text{clip}(r_t(\theta), 1 - \epsilon, 1 + \epsilon) \quad (2.7)$$

Ecuación 2.7: Clipping

Donde ϵ es un valor pequeño.

- **Función de pérdida del crítico ($L^{VF}(\phi)$)**

El crítico, que estima la función de valor $V_\phi(s_t)$, se entrena minimizando el error cuadrático medio entre su predicción y el retorno observado R_t como muestra la [Ecuación 2.8](#).

$$L^{VF}(\phi) = \mathbb{E}_t[(V_\phi(s_t) - R_t)^2] \quad (2.8)$$

Ecuación 2.8: Función de pérdida del crítico

- **Término de entropía ($S[\pi_\theta](s_t)$)**

Está definido por la [Ecuación 2.9](#) y su objetivo es fomentar la exploración, manteniendo la política suficientemente aleatoria y evitando que el agente se estanque.

$$S[\pi_\theta](s_t) = - \sum_a \pi_\theta(a|s_t) \log \pi_\theta(a|s_t) \quad (2.9)$$

Ecuación 2.9: Término de entropía

La función objetivo combina todos estos términos como se muestran en la [Ecuación 2.10](#)

$$L(\theta, \phi) = \mathbb{E}_t[\min(r_t(\theta)\hat{A}_t, \text{clip}(r_t(\theta), 1 - \epsilon, 1 + \epsilon)\hat{A}_t) - c_1(V_\phi(s_t) - R_t)^2 + c_2 S[\pi_\theta](s_t)] \quad (2.10)$$

Ecuación 2.10: Función objetivo para el algoritmo PPO

Donde:

- El primer término es el objetivo de la política con recorte para estabilidad.
- El segundo término es la pérdida del crítico para mejorar la función de valor.
- El tercero es el término de entropía para mantener exploración.

- Los coeficientes c_1 y c_2 balancean la importancia de cada término durante el entrenamiento.

La función objetivo está diseñada para maximizar el rendimiento mientras previene actualizaciones inestables, y fomenta la exploración mediante la entropía. Estas características hacen de PPO un algoritmo robusto y flexible, adecuado para problemas complejos, incluyendo entornos multiagente y tareas con espacios de acción continuos.

- **DDPG**

En este caso, la política se va actualizando en función de las experiencias pasadas, es decir, en función de los resultados obtenidos en cada paso. Son muy eficaces en tareas que requieran un control continuo.

- **Actor-Critic (AC).**

Por último, nos quedaría la combinación de los dos enfoques anteriores, donde se empleará la función de valor y la política para que el agente seleccione la acción que sea mejor en cada estado. Para ello, utiliza dos redes neuronales una para el actor (basada en la política y decidirá que acción se toma en cada estado) y otra para el crítico (basada en la función de valor y evaluará lo buena o mala que ha sido la acción seleccionada para ese estado). Existen ciertas variaciones que señalamos a continuación:

- **Advantage Actor-Critic (A2C).**

En este caso, el crítico al evaluar la acción dada, no sólo indicará lo buena o mala que es, sino que también mostrará cuanto mejor puede ser, esto genera buena estabilidad.

- **Asynchronous Advantage Actor-Critic (A3C).**

Se realiza el mismo procedimiento que en A2C pero se entrena en paralelo a muchos agentes que interactúan con copias independientes del entorno, lo que permite explorar el entorno de manera más efectiva y eficiente.

- **Soft Actor-Critic (SAC).**

Se incorpora la medida de entropía (indica para una política cuán impredecible o aleatoria es) a la máxima recompensa con el objetivo de favorecer la estabilidad y la exploración.

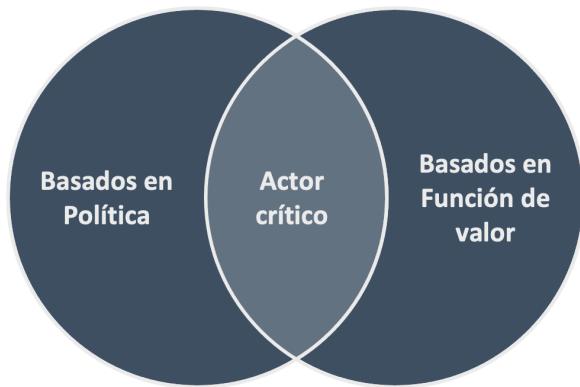


Figura 2.4: Tipos de enfoques para resolver problemas de aprendizaje por refuerzo

2.1.4. Aplicaciones

Como ya se ha mencionado anteriormente, el aprendizaje por refuerzo cuenta con un gran potencial a la hora de solventar problemas de toma de decisiones en entornos dinámicos. Existen numerosas áreas donde su aplicación permite que un agente tras interactuar con el entorno pueda encontrar comportamientos óptimos que le garanticen alcanzar sus objetivos.

1. Robótica.

El uso del aprendizaje por refuerzo ayuda a que los robots aprendan a realizar tareas complejas o a desplazarse. Gracias a ello, tendrán la habilidad de realizar repartos de productos o asistir a personas con limitación de capacidades.

2. Vehículos autónomos.

En este sector, el RL está contribuyendo a la hora de eludir obstáculos y con ello reducir la posibilidad de accidentes, facilitar la navegación para evitar situaciones de tráfico difíciles y optimizar el consumo de combustible haciendo más eficiente los desplazamientos.

3. Finanzas.

A nivel financiero es importante reducir los riesgos al mínimo, aquí el aprendizaje por refuerzo permite evaluar dichos riesgos con eficiencia para mitigarlos. Su facilidad de adaptación ayuda a tomar mejores decisiones en inversiones o gestionando carteras.

4. Sistemas de recomendación.

Algo a lo que sin duda ya estamos bastante acostumbrados hoy día, no son pocas las plataformas que analizando el comportamiento de sus usuarios, van definiendo continuamente las recomendaciones de productos, contenido u otros servicios.

5. Salud y medicina personalizada.

Cuando hablamos de la aplicación en el campo sanitario, la capacidad con la que contamos nos permitirá personalizar tratamientos, conseguir mejores diagnósticos o conseguir nuevos medicamentos. Todo esto repercute en un sistema sanitario eficaz y más eficiente, proporcionando mejor atención y por consiguiente, evitando la perdida de vidas.

2.2. Aprendizaje por refuerzo en entornos multiagente (MARL)

El aprendizaje por refuerzo multiagente tiene como objetivo estudiar cómo se comportan varios agentes mientras comparten el mismo entorno. Cada agente puede contar con metas individuales o colectivas, lo que puede afectar a las dinámicas en la toma de decisiones surgiendo la aparición de escenarios donde prime la competitividad o sea necesaria la cooperación. Este nuevo contexto gana relevancia cuando barajamos posibles aplicaciones como el control de tráfico, robótica colaborativa o gestión de recursos.

2.2.1. Aprendizaje individual vs. aprendizaje multiagente

Como hemos visto anteriormente, en el aprendizaje por refuerzo individual un solo agente aprende en un entorno estacionario, el cual, responde con cierta predictibilidad a las acciones del agente. Si ahora ese mismo entorno lo compartimos con más agentes, la respuesta que reciba cada uno no dependerá sólo del entorno, puesto que, las acciones de todos los agentes están interrelacionadas. Esto implicará ciertos aspectos relevantes:

- Como las respuestas, además de depender del entorno, también lo harán de las acciones del resto de agentes, el entorno dejará de ser estacionario y pasará a ser no estacionario desde la perspectiva de cada agente. Esto puede ocasionar que las reglas, recompensas o dinámicas varíen, como consecuencia de que los agentes también están aprendiendo.
- Es necesario considerar que el escenario puede tomar diferentes aspectos, nos podemos encontrar en uno competitivo, donde un agente solo consigue su objetivo a expensas de otros agentes y por lo tanto, las recompensas serán totalmente opuestas, es decir, si un agente recibe una recompensa el resto será penalizado. Podemos encontrarnos en un escenario de cooperación, donde todos los agentes obtendrán las mismas recompensas. Y también podría darse el caso en el que nos situásemos en un escenario mixto, donde contemos tanto con elementos de competencia como de cooperación, y dependiendo de la situación, obtengamos recompensas opuestas o iguales.

- En este nuevo enfoque, como se debe tener en cuenta lo que hará cada uno de los agentes, el objetivo de aprendizaje no es siempre maximizar una recompensa individual. Aquí, lo ideal es encontrar un equilibrio de los agentes, es decir, una situación estable en la que ningún agente tenga incentivo para cambiar su estrategia. De esta manera, aprenderán a convivir y coordinarse con otros buscando situaciones estables que beneficien al conjunto o aprenderán a competir contra otros buscando ventajas que beneficien a sí mismos.

2.2.2. Enfoques y Métodos

En el aprendizaje por refuerzo multiagente condiciona la elección del enfoque y la técnica de entrenamiento más adecuada. A continuación se describen algunos métodos relevantes para este trabajo, agrupados según el tipo de escenario.

- **Métodos para entornos competitivos.**

En los entornos competitivos, donde los agentes tienen objetivos contrapuestos es fundamental desarrollar estrategias robustas que maximicen la recompensa de un agente en detrimento de los demás.

- **Self-Play**

Permite a un agente mejorar progresivamente al jugar contra versiones anteriores de sí mismo. Es especialmente útil cuando no se dispone de un oponente externo o cuando se desea generar oponentes con niveles de habilidad crecientes.

- **Minimax-Q**

El agente considera el peor escenario posible (estrategia óptima del oponente) y ajusta su comportamiento para maximizar su rendimiento bajo esa condición.

- **Métodos para entornos cooperativos.**

En entornos cooperativos, todos los agentes comparten el mismo objetivo y recompensa. La clave está en la coordinación y el aprendizaje conjunto.

- **PPO**

Permite aprender políticas estables en entornos complejos. En MARL, puede utilizarse de forma descentralizada o coordinada entre agentes que comparten políticas o recompensas.

- **QMIX**

Descompone la función de valor conjunta en funciones de valor individuales de los agentes, asegurando que las políticas individuales contribuyen al objetivo global. Utilizará una red neuronal para combinar los valores individuales de forma coherente.

- **Métodos para entornos mixtos.**

En escenarios mixtos, los agentes deben aprender cuándo cooperar y cuándo competir. Aquí se requiere un equilibrio entre adaptabilidad y estabilidad, ya que las recompensas pueden estar parcialmente alineadas.

- **MADDPG**

Cada agente aprende su propia política, pero el entrenamiento se realiza de manera centralizada teniendo en cuenta las acciones de los demás. Esto permite modelar comportamientos diferenciados y adaptativos en entornos con interacciones mixtas.

- **Win or Learn Fast Policy Hill Climbing (WoLF-PHC)**

Adapta la tasa de aprendizaje del agente dependiendo de si está “ganando” o “perdiendo”, favoreciendo la exploración en momentos de bajo rendimiento y la estabilidad cuando se están obteniendo buenos resultados.

Es importante tener en cuenta que en los escenarios competitivos se priorizan métodos que permitan a los agentes ser robustos frente a oponentes que intentan minimizar su rendimiento. En cambio, en los entornos cooperativos, el objetivo principal es lograr una coordinación eficiente entre agentes para maximizar el desempeño colectivo. Por último, los entornos mixtos requieren métodos que sean capaces de adaptarse a situaciones cambiantes, manteniendo al mismo tiempo cierta estabilidad, ya que los agentes deben aprender a colaborar o competir según el contexto.

2.2.3. Retos y Desafíos

Cuando trabajamos con un aprendizaje por refuerzo multiagente tenemos que enfrentarnos a una serie de desafíos, más allá de los que nos encontrábamos con un solo agente. A continuación destacamos los más relevantes:

- **Escalabilidad.**

Al aumentar el número de agentes en un entorno, crece el tamaño del espacio de estados y acciones haciéndolo más complejo, esto complica la actualización de la función de valor y de la política, y por consiguiente, dificulta un aprendizaje eficiente y una toma de decisiones óptimas.

- **No estacionariedad.**

En un entorno multiagente, los diferentes agentes están aprendiendo al mismo tiempo en el mismo entorno. Esto alterará constantemente la dinámica desde la perspectiva de cada agente, complicando la toma de decisiones de cada uno.

- **Competencia.**

Encontrarnos en un escenario competitivo puede derivar en comportamientos inestables, los agentes pueden centrarse en maximizar sus propias recompensas y no atender al perjuicio que están causando al rendimiento global del sistema.

- **Coordinación.**

En escenarios cooperativos, es necesario que los agentes tomen decisiones compatibles para lograr un comportamiento conjunto eficiente. Si no hay coordinación, puede que incluso tomando buenas decisiones por separado, el equipo no logre el objetivo o lo haga de forma ineficiente.

- **Selección de equilibrios.**

Lograr un equilibrio entre competencia y cooperación resulta fundamental en el contexto del aprendizaje multiagente. En entornos mixtos, donde los intereses de los agentes están parcialmente alineados, es necesario implementar mecanismos que promuevan la colaboración cuando sea beneficiosa, sin comprometer la eficacia individual de cada agente.

- **Robustez.**

Puede darse el caso que se asuman observaciones perfectas del entorno y de las acciones de los demás, lo cual es poco realista cuando lo aplicamos al mundo real.

- **Diseño del objetivo de aprendizaje.**

Definir objetivos que equilibren estabilidad y adaptabilidad continua sin una solución clara que permita dicho equilibrio.

- **Rendimiento.**

En escenarios prácticos, es deseable garantizar un buen rendimiento y además, un comportamiento aceptable durante el proceso de aprendizaje.

Abordar estos desafíos exige combinar distintos tipos de herramientas, algunas que permiten manejar la incertidumbre, otras que simplifican las tareas, otras que permiten aproximar soluciones cuando los cálculos exactos no son viables, y también aprovechar todo el conocimiento previo que se tenga sobre el entorno o la tarea.

2.3. Transferencia de conocimiento

Cuando hacemos referencia a la transferencia de conocimiento entre entornos, nos referimos a la capacidad de reutilizar información aprendida en un entorno relativamente simple, para

mejorar el rendimiento o acelerar el aprendizaje en otro entorno más complejo. Su uso puede aportar múltiples beneficios como la reducción del tiempo de entrenamiento, la mejora en la exploración del entorno, aplicación en entornos donde es difícil encontrar información como es el mundo real o fomentar la generalización que le ayude a evitar el sobreajuste.

2.3.1. Transferencia de conocimiento a entornos multiagentes

Como ya hemos mencionado anteriormente, la transición de entornos de un solo agente a entornos multiagente introduce varios retos. Uno de los principales es que pasaremos de entornos estacionarios a no estacionarios, ya que los agentes modifican constantemente la dinámica del entorno. Por lo tanto, ya sea con un enfoque colaborativo o competitivo, adaptarnos a un entorno donde contaremos con un número mayor de agentes conllevará un costo en todos los aspectos. Pero, para evitar dicho coste, tendremos que tener en cuenta qué tipo de transferencia vamos a realizar para saber qué técnicas de transferencia de conocimiento será mejor aplicar.

- Si la transferencia es entre entornos, se transferirá el conocimiento del entorno origen hacia otro entorno objetivo diferente. En este caso, se modificarán las dinámicas, el número de agentes y la estructura espacial del entorno.
- Por otro lado, si la transferencia es entre agentes, se transferirá el conocimiento aprendido por un agente a otros agentes, pero manteniendo el mismo entorno.
- También se puede dar el caso que un mismo agente aplique conocimiento previamente adquirido en un contexto a un nuevo contexto, pero asumiendo un rol diferente.
- Otra situación que puede surgir es que todos los agentes compartan la misma arquitectura y roles, por lo que la transferencia sería directa.
- O por el contrario, que los agentes tengan roles u observaciones distintas, por lo que la transferencia debe adaptarse.

Teniendo en cuenta el contexto del presente trabajo, podríamos necesitar una transferencia entre entornos, puesto que el entorno Pong es diferente al entorno Quadrapong. Además, también podríamos necesitar una transferencia entre agentes, ya que se transferiría el conocimiento de los agentes entrenados en el entorno Pong a los agentes que conforman los diferentes equipos en el entorno Quadrapong. Pero, podríamos entender que los agentes tengan observaciones o roles distintos. Lo que se daría en el entorno Quadrapong, para los agentes que se encuentren en la parte superior e inferior de la pantalla, el desplazamiento será a izquierda y derecha a diferencia del aprendido en el entorno Pong que es arriba y abajo.

Por lo tanto, sería de mucha utilidad aplicar la transferencia de conocimiento a este trabajo, porque gracias a ella, los agentes no necesitarán aprender desde cero cómo interactuar con la pelota, reduciendo coste temporal y computacional. Y pueden centrarse en aprender cómo coordinarse con su compañero de equipo o cómo adaptarse al nuevo entorno, y/o a nuevos roles u oponentes.

2.3.2. Técnicas de transferencia de conocimiento

Existen múltiples técnicas para realizar la transferencia de conocimiento entre entornos, el uso de estas dependerá del tipo de transferencia que vayamos a realizar (entre entornos, entre agentes, ...), a continuación presentamos algunos de ellas.

■ **Parameter Sharing.**

Útil especialmente cuando los agentes cuentan con el mismo espacio de acciones y observaciones. En nuestro caso, los agentes de Quadrapong comparten inicialmente los mismos parámetros, durante el entrenamiento se introducirán mecanismos de especialización para que cada agente aprenda su rol, esto repercutirá en que:

- Reduce el número de parámetros.
- Facilita la coordinación.
- Mejora la eficiencia de entrenamiento.

■ **Policy Distillation.**

Enfoque en el que un agente experto actúa como profesor de los agentes inexpertos, los cuales aprenderán imitando. Tendremos agentes expertos en jugar a Pong y los agentes de Quadrapong aprenderán a imitarlos inicialmente realizándose los ajustes necesarios progresivamente.

■ **Transfer Learning / Parameter Transfer.**

Esta técnica reutilizará los parámetros aprendidos (pesos) en una red neuronal de origen (Pong) para acelerar el aprendizaje de nuevas tareas en entornos más complejos (Quadrapong). Los parámetros podrán mantenerse fijos (congelados) o continuar ajustándose mediante entrenamiento adicional (Fine-tuning).

■ **Fine-tuning.**

Después de transferir los pesos de una red entrenada en Pong, se realiza entrenamiento adicional en Quadrapong para adaptarse a la nueva tarea sin partir de cero, en el caso de Quadrapong para los agentes situados en la parte superior e inferior, adaptación de su dirección de movimiento.

- **Meta-learning.**

El objetivo es que el agente aprenda una política base que pueda adaptarse rápidamente a nuevas tareas con pocos datos. En MARL, puede ayudar a que los agentes adapten sus estrategias al comportamiento de otros agentes, incluso si estos cambian.

- **Feature Extraction Transfer.** Es ideal para cambios de perspectivas o rol, consiste en reutilizar capas intermedias de una red entrenada para extraer características relevantes (pelota, palas, ...), y conectar nuevas capas de decisión adaptadas al nuevo entorno (nuevas posiciones de los agentes).

2.3.3. Retos y Desafíos

Es importante considerar que para que una transferencia de conocimiento se lleve a cabo de manera óptima, sería conveniente que se cumpliesen una serie de condiciones. Por un lado, sería interesante que las tareas de aprendizaje sean lo más similares posibles en términos de objetivos, dinámica o estructura. Por otro lado, que las observaciones, las transiciones o recompensas entre el entorno de origen y el de destino no difieran excesivamente. Y también, que sea posible aplicar un modelo comparable o compatible en ambos entornos, de modo que las condiciones aprendidas en el entorno original puedan reutilizarse de forma efectiva en el nuevo entorno.

Entre los principales desafíos que presenta la transferencia de conocimiento podemos señalar:

- **Cambio en la dimensionalidad del espacio de observaciones y acciones.**

Contar con más agentes implica que las observaciones incorporen información de todos ellos. Esto puede requerir una reestructuración del modelo original.

- **Naturaleza no estacionaria del entorno.**

Como ya se ha mencionado, el comportamiento de múltiples agentes afecta directamente a la dinámica del entorno. Por ello, el conocimiento aprendido en un entorno estacionario puede no ser estable en un entorno no estacionario.

- **Desalineamiento de objetivos.**

La función de recompensa en el entorno de origen puede ser muy diferente a la del entorno objetivo, lo que obliga a adaptarla, al igual que le ocurrirá a las estrategias aprendidas.

- **Transferencia negativa.**

En algunos casos, el conocimiento transferido puede ser contraproducente si no se adapta adecuadamente al nuevo entorno, especialmente si las dinámicas estratégicas cambian drásticamente. Esto puede ralentizar el aprendizaje o impedir que se adopten nuevas conductas más eficientes.

A pesar de estos desafíos, un uso adecuado de las técnicas de transferencia que aseguren el cumplimiento de las condiciones señaladas anteriormente, puede acelerar significativamente el aprendizaje en entornos complejos, facilitar la generalización y permitir una exploración más eficiente del entorno, algo crucial dado que el espacio de búsqueda es exponencialmente mayor.

2.4. Líneas de investigación previas

En este trabajo se ha tomado como eje una serie de estudios centrados en el aprendizaje por refuerzo y su extensión a entornos multiagente, con un enfoque especial en la cooperación, competencia y transferencia de conocimiento entre entornos. A continuación, se resumen las principales líneas de investigación consultadas y su aporte al diseño del proyecto.

1. El artículo **Multiagent Cooperation and Competition with Deep Reinforcement Learning** [4] constituye la base fundamental del proyecto, en él se exploran las dinámicas de cooperación y competencia entre dos agentes autónomos, entrenados por medio de DQN en el entorno del videojuego Pong. Se ha empleado una metodología basada en la modificación del esquema de recompensas para inducir comportamientos cooperativos o competitivos. Esto ha servido de inspiración directa para los apartados iniciales, [apartado 3.2](#) y [apartado 3.3](#), donde se establece el entorno de Pong, se definen los agentes y sus modelos.
2. Por otro lado, el trabajo **Playing Atari Ball Games with Hierarchical Reinforcement Learning** [13], explora el uso de instrucciones de manuales de usuario o de expertos para acelerar el aprendizaje y mejorar el rendimiento en entornos como Pong. Este enfoque destaca el valor de las representaciones de alto nivel y del conocimiento previo, lo cual se relaciona estrechamente con la parte final de la implementación, [apartado 3.6](#) dedicado a la transferencia de conocimiento desde un entorno previamente dominado (Pong) a otro más complejo (Quadraball).
3. La comparativa presentada en **Investigation of independent reinforcement learning algorithms in multi-agent environments** [18], proporciona un análisis de algoritmos independientes en entornos cooperativos, competitivos y mixtos. Este trabajo se alinea con la metodología del proyecto, donde se utilizará inicialmente agentes que aprenden de manera independiente a explorar sus limitaciones y ventajas. Esta tarea será llevada a cabo en la parte de entrenamiento y aprendizaje [apartado 3.4](#), y posteriormente se introducirán mecanismos de interacción para observar cómo evolucionan los comportamientos adaptativos.

4. Llegado el momento de considerar varios agentes interactuando, el artículo **A Comprehensive Survey of Multiagent Reinforcement Learning** [19], ofrece una visión exhaustiva del aprendizaje por refuerzo multiagente, permitiendo situar el presente trabajo en el contexto más amplio de este campo. Incluye los objetivos que persiguen los agentes (estabilidad del aprendizaje o capacidad de adaptación al comportamiento de otros agentes) o los diferentes tipos de entornos (colaborativos, competitivos o mixtos).
5. Asimismo, el método propuesto en **Actor-Mimic: Deep Multitask and Transfer Reinforcement Learning** [20], introduce una método capaz de aprender múltiples tareas de forma simultánea y transferir conocimiento a nuevas tareas sin supervisión experta. Este enfoque apoya la transición entre Pong y Quadrapong, abriendo la posibilidad de acelerar el aprendizaje en entornos nuevos.
6. Finalmente, en cuanto a la transferencia entre dominios, los artículos **Visual Transfer Between Atari Games Using Competitive Reinforcement Learning** [21] y **Knowledge Transfer in Deep Reinforcement Learning via an RL-Specific GAN-Based Correspondence Function** [22], abordan distintas estrategias para reutilizar políticas aprendidas en un entorno inicial a un entorno nuevo. Estos trabajos refuerzan la viabilidad y relevancia de la fase de transferencia a Quadrapong, explorando cómo los modelos preentrenados pueden ser ajustados o adaptados, con el objetivo de reducir el tiempo de entrenamiento y mejorar la generalización.

En conjunto, estos estudios permiten enmarcar el trabajo final de máster entre el aprendizaje multiagente y la transferencia de conocimiento a entornos más complejos, estableciendo una base tanto teórica como práctica para explorar la evolución de comportamientos colaborativos y competitivos, así como la reutilización eficiente de conocimientos aprendidos a nuevos contextos de juego.

2.5. Tendencias y Trabajo Futuro

La transferencia de conocimiento entre entornos de aprendizaje por refuerzo, especialmente en escenarios multiagente, está adquiriendo un papel clave en el desarrollo de sistemas inteligentes más generalizables, adaptativos y escalables. A partir de este proyecto, se pueden identificar diversas aplicaciones directas en problemas reales, tendencias emergentes y líneas de trabajo futuro.

- **Aplicaciones reales.**

Los avances en entornos como PettingZoo permiten desarrollar, probar y analizar dinámicas multiagente con gran control y bajo coste. Aun así, un paso natural es la transferencia

hacia entornos reales, lo que implica desafíos añadidos. Las técnicas desarrolladas en este trabajo, como la reutilización del conocimiento adquirido, son aplicables en ámbitos como:

- **Robótica colaborativa**, donde múltiples robots deben coordinarse para mover objetos, ensamblar piezas, dar asistencia o explorar entornos. En el siguiente artículo **Multi-agent Collaborative Perception for Robotic Fleet: A Systematic Review** [23], se presentan los hallazgos de varios artículos de investigación sobre percepción colaborativa en flotas multirobóticas.
 - **Sistemas de vehículos autónomos**, donde los coches deben interactuar con el entorno u otros vehículos en tiempo real, para hacer eficientes sus desplazamientos. Atendiendo al artículo **Predictive Active Steering Control for Autonomous Vehicle Systems** [24], se muestra un enfoque de control predictivo para controlar un sistema de dirección delantera en un vehículo autónomo.
 - **Logística multiagente** (almacenamiento, reparto, ...), donde drones o vehículos comparten recursos, rutas y objetivos. Según el artículo **Evaluating city logistics measures using a multi-agent model** [25], podemos encontrar una metodología que mediante un modelo multiagente permite evaluar medidas logísticas urbanas considerando el comportamiento de diversos actores involucrados en el transporte urbano de mercancías.
- **Tendencias emergentes.**
- Las tendencias actuales en transferencia de conocimiento se orientan hacia una mayor generalización y adaptación flexible a contextos nuevos o cambiantes. Entre ellas destacan:
- **Meta-aprendizaje**, donde se desarrollan agentes capaces de adaptarse rápidamente a nuevos roles, reglas o compañeros de equipo. En el artículo **Meta-Learning in Neural Networks: A Survey** [26], se describe el panorama actual del meta-aprendizaje.
 - **Aprendizaje curricular**, en el que los agentes se entrena de forma progresiva en tareas de dificultad creciente, facilitando así la obtención de habilidades transferibles. Podemos observar en el artículo **Evolutionary Population Curriculum for Scaling Multi-Agent Reinforcement Learning** [27], como el aprendizaje curricular escala el aprendizaje por refuerzo multiagente incrementando progresivamente la población de agentes durante el entrenamiento.
- **Líneas de trabajo futuro.**

Basadas en los resultados y el enfoque del presente trabajo, unas posibles líneas de investigación futuras serían extender la transferencia a:

- **Incorporación de mecanismos de atención y comunicación**, explorar cómo la comunicación explícita entre agentes mejora la transferencia y coordinación. Un agente observa como el resto cumple su cometido y en caso de pérdida de rumbo de alguno, lo redirige para que continúe alineado con el objetivo.
- **Escenarios con asignación dinámica de roles**, un agente mantiene un comportamiento hasta un determinado momento en el que será necesario que lo cambie para que se mantenga la evolución hacia el objetivo.
- **Aprendizaje basado en objetivos múltiples**, investigar cómo transferir políticas cuando hay múltiples objetivos que pueden entrar en conflicto. Un agente continua su camino hacia conseguir las metas mientras le da cobertura o protección a otro agente.

En definitiva, la transferencia de conocimiento a entornos multiagente representa un paso fundamental hacia el desarrollo de sistemas más robustos, colaborativos y eficientes. A medida que los algoritmos de aprendizaje por refuerzo evolucionan, se espera que estas técnicas se apliquen cada vez más en situaciones reales, donde la capacidad de aprender de forma continua, adaptarse a cambios y cooperar con otros agentes será crucial. Este trabajo constituye un punto de partida para futuras investigaciones centradas en la mejora de técnicas de transferencia de conocimiento a entornos multiagente complejos y con aplicaciones en entornos reales.

Capítulo 3

Diseño e implementación

Como ya se ha mencionado, en este trabajo se realiza el entrenamiento de agentes en el clásico juego de Atari Pong, en concreto se selecciona el entorno pong_v3 proporcionado por PettingZoo. Posteriormente, se ha realizado la transferencia del conocimiento adquirido al entorno quadrapong_v4 de Atari, también proporcionado por PettingZoo, un entorno más complejo donde aumenta el número de agentes participantes.

3.1. Herramientas de desarrollo y recursos utilizados

Para el desarrollo de este proyecto se han empleado diversas herramientas de "software" y "hardware" que han permitido implementar, entrenar y evaluar los agentes de manera eficiente:

- **Herramientas de software**

- **Python:**

Lenguaje de programación de alto nivel ampliamente utilizado en proyectos de Inteligencia Artificial y Aprendizaje por Refuerzo. Se ha empleado la versión 3.10.16, aprovechando su compatibilidad con las principales librerías de Deep Learning y entornos multiagente.

- **PettingZoo:**

Librería estándar para entornos de Aprendizaje por Refuerzo Multiagente. Proporciona interfaces consistentes y entornos como pong_v3 y quadrapong_v5 utilizados en este proyecto.

- **SuperSuit:**

Colección de "wrappers" para el preprocesamiento de entornos en PettingZoo, incluyendo reducción de color, redimensionamiento de imágenes y apilado de frames, esenciales para preparar los datos de entrada a las redes neuronales.

- **PyTorch:**

Librería de aprendizaje profundo utilizada para implementar las redes neuronales de los agentes. Permite la construcción de arquitecturas convolucionales personalizadas y el entrenamiento eficiente sobre "hardware" acelerado.

- **NumPy:**

Librería para operaciones numéricas, utilizada tanto para gestionar los datos de experiencia de los agentes como para el almacenamiento y análisis de métricas de entrenamiento.

- **Matplotlib y Pandas:**

Herramientas auxiliares para la representación gráfica de resultados (recompensas, impactos, victorias) y la gestión de datos para su análisis posterior.

- **TensorBoard:**

Se ha utilizado TensorBoard para una visualización más estructurada de métricas durante el entrenamiento.

- **Recursos de hardware**

El proyecto se ha desarrollado utilizando un equipo MacBook Pro M4 con las siguientes especificaciones:

- **Central Processing Unit (CPU):** 14 núcleos
- **GPU:** 20 núcleos
- **Neural Engine:** 16 núcleos
- **Memoria unificada:** 24GB
- **Almacenamiento:** 1 TB Solid State Drive ([SSD](#))

El uso del motor Metal Performance Shaders ([MPS](#)) de Apple ha permitido aprovechar la aceleración por GPU disponible en el MacBook para entrenar los modelos de aprendizaje profundo de manera más eficiente, especialmente en la etapa de entrenamiento de agentes en entornos de alta carga computacional como Pong y Quadrapong.

Estas herramientas de "software" y recursos "hardware" han sido seleccionados cuidadosamente para garantizar la compatibilidad, la eficiencia de entrenamiento y la correcta implementación de los métodos de Aprendizaje por Refuerzo Multiagente que constituyen la base del presente proyecto.

3.2. Establecer el entorno de Pong

Pong es un videojuego clásico de Atari en el que dos jugadores controlan unas paletas en lados opuesto de la pantalla, el objetivo de ambos es devolver una pelota evitando que esta sobrepase su paleta. Es un juego de carácter competitivo donde el vencedor es aquel que consigue sobrepasar la paleta rival en mayor número de ocasiones, siendo 21 la máxima puntuación alcanzable.

Las características principales que presenta este entorno son:

- **Espacio de acciones**

Se cuenta con un espacio de acción discreto con 6 acciones posibles.

- **0:** no hacer nada.
- **1:** disparar / "sacar o servir" la pelota.
- **2:** mover la paleta hacia la derecha (según su perspectiva).
- **3:** mover la paleta hacia la izquierda (según su perspectiva).
- **4:** mover la paleta hacia la derecha mientras se saca la pelota.
- **5:** mover la paleta hacia la izquierda mientras se saca la pelota.

- **Espacio de observaciones**

Cada agente recibe como observación una imagen RGB¹ del entorno cuyas dimensiones son (210, 160, 3). Esto significa que la imagen cuentan con una altura de 210 píxeles, un ancho de 160 píxeles y está representada por los canales de color (rojo, verde, azul) como se aprecia en la Figura 3.1.

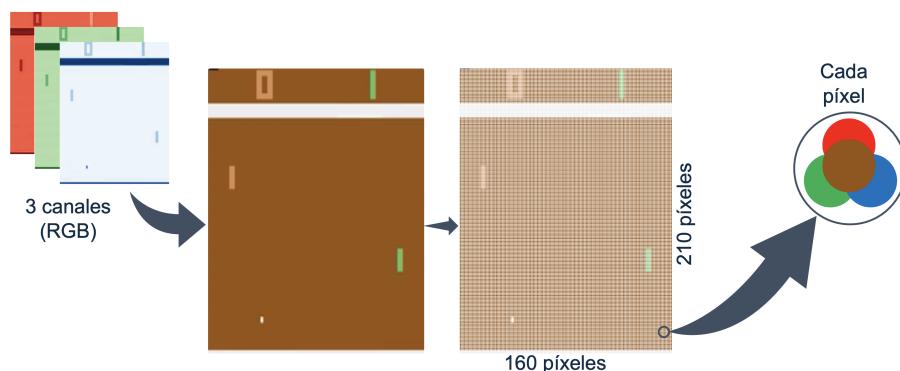


Figura 3.1: Espacio de observación

¹Red, Green, Blue (RGB). <https://es.wikipedia.org/wiki/RGB>

- **Agentes**

Son dos los agentes presentes en el entorno, y están definidos de manera simétrica con el mismo espacio de observaciones y acciones. Lo único que los diferencia es la posición en el entorno, (first_0) controla la paleta del jugador de la izquierda y (second_0) controla la paleta del jugador de la derecha.

- **Recompensas**

Son el principal mecanismo para que los agentes aprendan y se reparten asignando +1 para el agente que anota y -1 para el que recibe el punto. Los saques se cronometran y si el jugador no saca en los 2 segundos siguientes a recibir la pelota, recibe -1 punto.

Para facilitar el entrenamiento de los agentes, Supersuit proporciona para los entornos de PettingZoo "wrappers" o capas intermedias. Estos "wrappers" aplicados a un entorno de entrenamiento permiten transformar las observaciones para evitar problemas con el tamaño de la resolución, prescindir de información redundante y proporcionan la información necesaria sobre movimiento y velocidad. En este trabajo se aplican los siguientes "wrappers":

- **Reducción de color (color_reduction_v0)**². Se convierte la imagen RGB en escala de grises, el color no es relevante y por ello, se reduce de 3 canales a 1.
- **Redimensión (resize_v1)**³. Se redimensiona la imagen a 84x84 píxeles, este tamaño nos ahorra costes computacionales.
- **Apilamiento (frame_stack_v1)**⁴. Apila los últimos 4 frames consecutivos, esto nos añade información temporal, algo inexistente en imágenes fijas.

Con este tratamiento podemos capturar además del estado actual, el movimiento de los objetos en pantalla. Este preprocesamiento es esencial para mejorar la eficiencia y estabilidad del entrenamiento.

3.3. Definir los agentes y sus modelos

Como ya hemos mencionado, contamos con dos agentes denominados first_0 y second_0, que se entrenan de manera independiente utilizando el algoritmo PPO. Cada agente dispone de un modelo propio basado en una política neuronal convolucional (CNN, convolutional neural network) para procesar las observaciones visuales del entorno y decidir sus acciones. Esta

²color_reduction_v0. https://pettingzoo.farama.org/api/wrappers/supersuit_wrappers/#color_reduction_v0

³resize_v1. https://pettingzoo.farama.org/api/wrappers/supersuit_wrappers/#resize_v1

⁴frame_stack_v1. https://pettingzoo.farama.org/api/wrappers/supersuit_wrappers/#frame_stack_v1

separación asegura que cada agente aprende exclusivamente a partir de su experiencia local, sin compartir directamente los parámetros con el adversario.

Arquitectura de la política neuronal (CnnPolicy)

La política de cada agente está implementada mediante una red neuronal convolucional profunda que podemos apreciar en la Figura 3.2, diseñada para extraer características relevantes de las imágenes procesadas del entorno. El flujo típico de la red es:

- **Entrada:**

Cuatro frames consecutivos en escala de grises de tamaño 84x84 píxeles, apilados para capturar información temporal y dinámica.

- **Bloques convolucionales:**

Varias capas convolucionales con activación RELU⁵ o unidad lineal rectificada. Se trata de una función que ayuda a que la red aprenda representaciones más complejas, esto le permite detectar patrones visuales complejos, tales como movimientos y trayectorias de la pelota y las palas.

- **Capas totalmente conectadas:**

Transforman la representación convolucional en un vector de características denso.

- **Salida:**

Una capa que genera la distribución de probabilidades sobre las acciones posibles (6 acciones discretas en Pong), modelando la política estocástica $\pi_\theta(a|s)$.

El uso de una política estocástica permite al agente explorar el espacio de acciones durante el entrenamiento y mejorar la robustez frente a entornos dinámicos.

⁵Rectified Linear Unit (RELU). [https://es.wikipedia.org/wiki/Rectificador_\(redes_neuronales\)](https://es.wikipedia.org/wiki/Rectificador_(redes_neuronales))

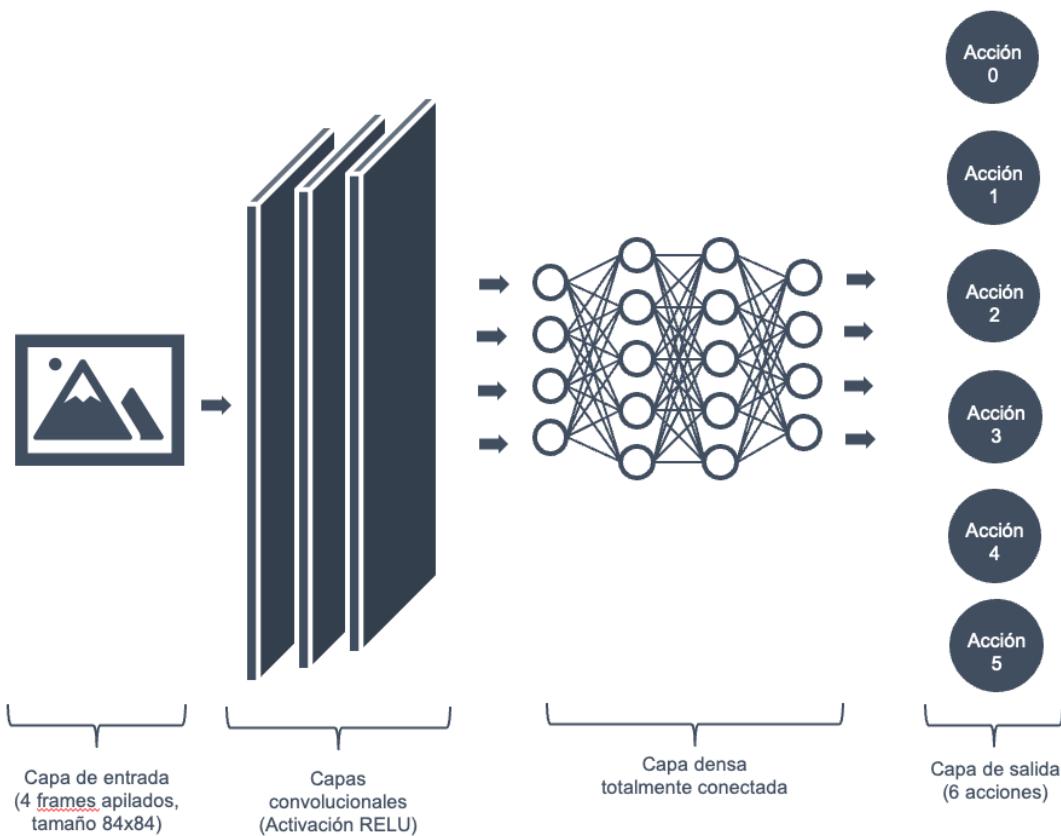


Figura 3.2: Red neuronal convolucional

3.4. Entrenamiento y aprendizaje de los agentes en Pong

Como ya se ha mencionado, el entrenamiento de los agentes en este proyecto sigue un enfoque basado en aprendizaje por refuerzo profundo en entornos multiagente, utilizando la arquitectura PPO. Se ha implementado para cada agente de forma independiente, pero se utiliza la API paralela de PettingZoo, que permite que ambos agentes interactúen simultáneamente con el entorno.

3.4.1. Estructura del entrenamiento

Para entrenar cada agente, se utiliza la librería Stable Baselines3, que proporciona una implementación optimizada de PPO⁶ con soporte para políticas basadas en CNN. El entorno se vectoriza con DummyVecEnv⁷ y se adapta para que las imágenes sean correctamente interpre-

⁶ Implementación optimizada de PPO. <https://stable-baselines3.readthedocs.io/en/master/modules/ppo.html>

⁷ DummyVecEnv. https://stable-baselines3.readthedocs.io/en/master/guide/vec_envs.html

tadas mediante `VecTransposeImage`⁸.

El entrenamiento ha sido diseñado para ser incremental, se ha decidido separar el entrenamiento en rondas para tener un mayor control sobre este, permitiéndonos guardar "checkpoints" periódicos. Esto nos proporciona la garantía de poder pausar y reanudar el entrenamiento sin perdida de información, y así poder realizar los ajustes que sean necesarios, evitando la repetición de entrenamientos innecesarios. También nos ha facilitado la evaluación o monitoreo en fases intermedias, obteniendo una visualización del entrenamiento que nos permita detectar problemas. Además, al trabajar con imágenes el consumo de recursos es mucho mayor, las imágenes ocupan bastante espacio en memoria y si entrenamos durante millones de pasos sin parar, la memoria puede saturarse poco a poco. El empleo de rondas de entrenamiento facilita la liberación de recursos, evitando que la memoria se sobrecargue. El proceso de entrenamiento se ha organizado en bloques de 1.000 rondas, las cuales cuentan con 5.000 pasos de interacción con el entorno. Se han alcanzando un total de 5.000 rondas de entrenamiento, lo que equivale a 25.000.000 de pasos de interacciones con el entorno.

El entrenamiento se realiza bajo un enfoque simétrico y competitivo, donde ambos agentes alternan su entrenamiento enfrentándose al modelo actualizado del adversario (self-play). En cada ronda, primero se entrena al agente `first_0` en un entorno donde el agente `second_0` actúa con su política fija actual (modelo PPO previamente entrenado) en caso de existir o aleatorio si no hubiese modelo previo. Posteriormente, el agente `second_0` se entrena enfrentándose al modelo actualizado de `first_0`. Este enfoque simétrico permite que ambos agentes evolucionen de manera competitiva y equilibrada, adaptándose continuamente a las tácticas del adversario.

Características importantes del enfoque PPO utilizado:

- **Política estocástica:**

Utilizando modelos basados en PPO se aprende directamente una política probabilística que permite balancear exploración y explotación.

- **Actualización estable:**

El algoritmo limita las actualizaciones agresivas de la política mediante su función objetivo con "clipping", lo que mejora la estabilidad del entrenamiento.

- **Capacidad multiagente:**

Gracias a los "wrappers" y a la estructura de entrenamiento alternado, los agentes aprenden a adaptarse dinámicamente a un oponente en constante evolución.

⁸VecTransposeImage. https://stable-baselines3.readthedocs.io/en/master/_modules/stable_baselines3/common/vec_env/vec_transpose.html

- **Preprocesamiento visual:**

Se aplica reducción de color, redimensionamiento y apilamiento de frames para facilitar el aprendizaje eficiente de la política.

Durante el entrenamiento, cada agente actualiza su política para maximizar la función objetivo de PPO, que incluye mecanismos para garantizar actualizaciones estables y controladas. El agente aprende a mapear observaciones visuales a acciones mediante una política estocástica, balanceando exploración y explotación.

Esta organización ha facilitado controlar el progreso del aprendizaje de manera segmentada, facilitar evaluaciones intermedias y detectar posibles problemas de convergencia o estabilidad.

3.4.2. Ciclo de aprendizaje de los agentes

Durante el entrenamiento con PPO, cada agente sigue un ciclo de aprendizaje iterativo y autónomo basado en la interacción continua con el entorno, que podemos apreciar en la [Figura 3.3](#) y que describen las siguientes etapas:

1. **Observación del estado:**

El agente recibe como entrada una representación visual procesada del entorno, consistente en un conjunto de frames apilados que capturan la dinámica temporal.

2. **Selección de acción:**

La política estocástica del agente, genera una distribución de probabilidad sobre las acciones posibles. La acción a ejecutar se muestrea de esta distribución, permitiendo un balance entre exploración y explotación.

3. **Interacción con el entorno:**

Tras ejecutar la acción seleccionada, el entorno devuelve la nueva observación, la recompensa obtenida, y señales que indican si el episodio ha terminado o fue truncado.

4. **Almacenamiento de la experiencia:**

PPO utiliza las experiencias recientes recolectadas durante la interacción secuencial con el entorno, almacenándolas en un buffer temporal para ser usadas en la actualización posterior.

5. **Actualización de la política:**

Tras recopilar un conjunto suficiente de transiciones (experiencias), el agente actualiza los parámetros de su política y función de valor optimizando la función objetivo de PPO.

6. Repetición de ciclo:

El proceso se repite iterativamente, permitiendo que el agente mejore progresivamente su política, ajustándose tanto a las dinámicas del entorno como al comportamiento del adversario.

Este ciclo garantiza un entrenamiento estable y eficiente, aprovechando las ventajas de PPO para adaptar las políticas en entornos visuales complejos y multiagente.



Figura 3.3: Diagrama del ciclo de entrenamiento de un agente en Pong

3.4.3. Configuración del entrenamiento en Pong

Inicialmente, los agentes han sido entrenados en un entorno estrictamente competitivo replicando el escenario clásico de Pong. No existe comunicación o coordinación entre agentes y se aplica la siguiente función de recompensa:

- Si un agente consigue anotar un punto (la pelota sobrepasa al rival), recibe +1.
- Si un agente falla (la pelota le sobrepasa), recibe -1.
- En caso de "stalling" (no servir tras dos segundos), el agente inactivo recibe -1.

Esta función de recompensa refuerza estrategias puramente competitivas, orientadas a maximizar la ventaja sobre el oponente.

3.5. Establecer el entorno de Quadrapong

Quadrapong es un videojuego que extiende el juego Pong a cuatro agentes, organizados en dos equipos de dos jugadores cada uno como se aprecia en la [Figura 3.4](#). Este entorno exige mayor coordinación, ya que cada equipo debe controlar dos paletas, pero en esta ocasión además de los lados opuestos de la pantalla, también hay que defender la parte superior e inferior de la misma. El objetivo, al igual que en Pong, es que ambos equipos devuelvan una pelota evitando que esta sobrepase las dos paletas que conforman su equipo. También es un juego de carácter competitivo donde el equipo vencedor es aquel que consigue sobrepasar las paletas rivales en mayor número de ocasiones, siendo 21 la máxima puntuación al igual que en Pong.



Figura 3.4: Entorno Quadrapong [2]

Las características principales que presenta este entorno son:

- **Espacio de acciones**

Se cuenta con un espacio de acción discreto con 6 acciones posibles.

- **0:** no hacer nada.
- **1:** disparar / "sacar o servir" la pelota.
- **2:** mover la paleta hacia arriba en la pantalla.
- **3:** mover la paleta hacia la derecha en la pantalla.
- **4:** mover la paleta hacia la izquierda en la pantalla.
- **5:** mover la paleta hacia abajo en la pantalla.

Al contar con espacios de acciones diferentes entre Pong y Quadrapong, es conveniente hacer un mapeo de acciones para que los agentes eviten aprender comportamientos

incoherentes y no se vea afectada negativamente la transferencia de conocimiento. Es decir, transferir las acciones "Move up", "Move down" de Pong, a las acciones "Move up", "Move down", "Move left", "Move right" de Quadrapong, según la posición del jugador en pantalla.

- **Espacio de observaciones**

Cada agente también recibe como observación una imagen RGB del entorno con las mismas dimensiones (210, 160, 3).

- **Agentes**

En este entorno contamos con 4 agentes, definidos de manera simétrica con los mismos espacios de observaciones y de acciones. Lo único que los diferencia es la posición en el entorno, (first_0) controla la paleta del jugador de la izquierda, (second_0) controla la paleta del jugador de la derecha, (third_0) controla la paleta del jugador superior y (fourth_0) controla la paleta del jugador inferior.

- **Recompensas**

Al contar con una dinámica de equipos, anotar un punto le da al equipo +1 de recompensa y a al oponente -1 de recompensa. Los saques son cronometrados y si el jugador no saca dentro de los 2 segundos posteriores a recibir la pelota, su equipo recibe -1 punto.

Al ser el espacio de observación de cada agente una imagen de 210x160x3 píxeles (RGB) del área de juego, como ocurre en el entorno Pong, para estandarizar las observaciones y hacerlas compatibles con la arquitectura de los agentes de Pong, se aplican los mismos "wrappers" de procesamiento:

- **Reducción de color (color_reduction_v0).**
- **Redimensión (resize_v1).**
- **Apilamiento (frame_stack_v1).**

Este procesamiento permite mantener la consistencia con los modelos de Pong y facilita la transferencia de conocimiento.

La configuración inicial para entrenamiento en Quadrapong sigue el siguiente orden:

- Se crean 4 agentes PPO independientes (uno por jugador).
- Se transfiere conocimiento desde los modelos de Pong entrenados previamente.
- Se estructuran equipos de dos jugadores para estudio de comportamiento.

- Se configuran los mismos tipo de entrenamiento que en Pong para asegurar la comparabilidad inicial.

El entorno Quadrapong amplía el desafío de Pong al introducir múltiples agentes simultáneos, interacciones más complejas y nuevas dinámicas emergentes. Gracias al uso de "wrappers" consistentes y una API paralela, se facilita la transferencia de conocimiento y la continuación del entrenamiento de los agentes. En la [Tabla 3.1](#) se resaltan las diferencias clave entre los dos entornos utilizados en el trabajo.

Característica	Pong (pong_v3)	Quadrapong (quadrapong_v5)
Número de agentes	2 (uno frente a otro)	4 (uno en cada lado de la pantalla)
Equipos	No (competencia individual)	Sí (equipos de 2 jugadores)
Espacio de acciones	6 acciones (discretas)	6 acciones (discretas)
Tipo de observación	Imagen 210x160x3 (RGB)	Imagen 210x160x3 (RGB)
Procesamiento de observaciones	Resize (84x84), FrameStack (4)	Resize (84x84), FrameStack (4)
Dinámica de juego	1 pelota, 2 líneas de defensa	1 pelota, 4 líneas de defensa
Interacciones	Simple: uno contra uno	Complejas: múltiples rebotes
Función de recompensa	+1 al anotar, -1 al encajar, -1 si tarda en sacar	+1 al anotar, -1 al encajar, -1 si tarda en sacar
Dificultad estratégica	Media (predicción y reacción)	Alta (coordinación y anticipación)
Entorno en PettingZoo	pong_v3	quadrapong_v5
Tipo de API	ParallelEnv (acciones simultáneas)	ParallelEnv (acciones simultáneas)

Tabla 3.1: Comparativa entre los entornos Pong y Quadrapong en PettingZoo.

3.6. Transferencia de conocimiento a Quadrapong

Con el objetivo de acelerar el proceso de aprendizaje en entornos de mayor complejidad, se ha implementado una estrategia de transferencia de conocimiento desde los agentes entrenados en el entorno Pong hacia el entorno Quadrapong.

Teniendo en cuenta que el entorno Quadrapong comparte reglas y dinámicas físicas similares a Pong (mismo tipo de pelota, rebote, paletas,...), y que los agentes deben realizar acciones parecidas como mover las paletas y anticipar trayectorias. Se considera que el conocimiento aprendido (detectar la pelota, reaccionar, devolver, ...) es reutilizable y relevante.

Este proceso busca demostrar que se puede aprovechar la experiencia previa aprendida en una tarea más simple, para acelerar el aprendizaje en la nueva tarea, mejorar el rendimiento inicial y comparar la eficiencia frente a agentes que parten desde cero.

Para ello, se utilizan los pesos aprendidos por los agentes en el entorno de Pong (origen), y se aplican como punto de partida en el nuevo entorno Quadrapong (destino), lo que nos permite:

- No reiniciar el aprendizaje desde cero.
- Reutilizar lo aprendido en una tarea similar.
- Continuar el entrenamiento en una nueva tarea más compleja.

3.6.1. Técnica empleada

Una vez alcanzado un comportamiento adecuado de los agentes entrenados en el entorno origen, se aplica la técnica de "Transfer Learning / Parameter Transfer" para transferir los pesos de las redes neuronales previamente entrenadas al entorno destino. En concreto, se reutilizan los pesos del entrenamiento de los agentes de Pong, siendo asignados a los nuevos agentes de Quadrapong.

A partir de esta base, los agentes continúan su proceso de entrenamiento en el nuevo entorno por medio de "Fine-tuning", esta transferencia no implica compartir parámetros durante el entrenamiento a diferencia de "Parameter Sharing", sino que actúa como una inicialización de los agentes en un entorno más complejo.

El conocimiento aprendido (por ejemplo, seguir la pelota, devolverla con precisión, ...) se reutiliza de forma eficiente en el nuevo entorno, acelerando el aprendizaje y permitiendo estudiar cómo se comportan distintos estilos en el nuevo escenario. Esta estrategia evita la necesidad de entrenar las redes desde cero, proporcionando a los agentes una base de conocimiento previo que acelera su adaptación a la nueva dinámica de juego.

3.6.2. Asignación de agentes y equipos

Con el objetivo de estructurar de manera lógica la transferencia de conocimiento, como se puede apreciar en la [Tabla 3.2](#). Teniendo en cuenta que la dinámica de juego se basa en "Batalla por equipos de dos jugadores", donde cada jugador controla una pala y defiende una zona de puntuación. Sin embargo, al ser un juego de equipo, dos de las cuatro zonas de puntuación pertenecen al mismo equipo. Por lo tanto, cada equipo debe coordinarse para llevar la pelota desde sus zonas de puntuación hacia la del oponente. La distribución de equipos será la siguiente, Equipo 1 (first_0 y third_0) y Equipo 2 (second_0 y fourth_0). Se ha realizado una asignación específica de los pesos entrenados en Pong a los diferentes agentes de Quadrapong. La estrategia ha consistido en que los miembros de cada equipo de Quadrapong comparten los pesos aprendidos por un mismo agente de Pong.

- Los pesos del agente first_0 se transfieren a los agentes del equipo 1 (first_0, third_0). Este equipo ocupa la parte izquierda y superior de la pantalla.
- Los pesos del agente second_0 se transfieren a los agentes del equipo 2 (second_0, fourth_0). Este equipo ocupa la parte derecha e inferior de la pantalla.

Equipo	Agentes	Pesos transferidos desde
Equipo 1	first_0 + third_0	first_0 (Pong)
Equipo 2	second_0 + fourth_0	second_0 (Pong)

Tabla 3.2: Asignación de agentes y transferencia de pesos desde Pong a Quadrapong.

La motivación para asignar el conocimiento de esta manera se debe, a la idea de mantener una serie de principios:

- **Consistencia de estilos de juego**

Al compartir pesos, los agentes de un mismo equipo parten de una base estratégica común, lo que favorece la aparición de comportamientos coordinados.

- **Comparabilidad**

Permite comparar el desempeño de dos equipos con estilos inicialmente diferenciados (por ejemplo, competitivos vs colaborativos si los agentes de Pong son entrenados bajo distintas modalidades).

- **Facilidad de análisis**

Resulta sencillo hacer un seguimiento de la evolución de cada equipo en función del tipo de conocimiento transferido, identificando patrones en su comportamiento.

- **Adaptabilidad espacial**

Al asignar un agente a posiciones laterales (first_0, second_0) y otro a posiciones verticales (third_0, fourth_0), se maximiza la cobertura de todo el campo de juego, permitiendo observar cómo los agentes adaptan su estrategia a diferentes ubicaciones físicas en el entorno.

3.7. Entrenamiento y aprendizaje de los agentes en Quadrapong.

Este apartado describe los procesos y estructuras de entrenamiento implementados para los agentes en el entorno de Quadrapong, incluyendo el entrenamiento desde cero y el entrenamiento con Parameter Transfer y Fine-tuning. Se explica también el ciclo de aprendizaje de los agentes para ambos casos.

3.7.1. Estructura del entrenamiento desde cero

Se ha seguido la misma línea de entrenamiento que se ha empleado para el entrenamiento de Pong pero adaptada a un entorno con 4 agentes. Se vuelve a utilizar la librería Stable Baselines3

con su implementación optimizada de PPO que cuenta con soporte para políticas basadas en CNN. Se ha vuelto a vectorizar el entorno con DummyVecEnv y se han adaptado las imágenes para que VecTransposeImage las interprete correctamente. En este caso se han alcanzando un total de 1.000 rondas de entrenamiento, lo que equivale a 5.000.000 de pasos de interacciones con el entorno.

3.7.2. Ciclo de aprendizaje de los agentes en entrenamiento Quadrapong desde cero

Durante el entrenamiento, al igual que se realizaba en Pong, cada agente sigue un ciclo de aprendizaje iterativo y autónomo, descrito en la [Figura 3.3](#) y que presenta las etapas siguientes:

- Observación del estado
- Selección de acción
- Interacción con el entorno
- Almacenamiento de la experiencia
- Actualización de la política
- Repetición de ciclo

3.7.3. Estructura del entrenamiento después de transferencia de conocimiento

En este caso el entrenamiento se inicia con la carga de los modelos preentrenados, estos modelos ya cuentan con un entrenamiento de 4000 rondas (20.000.000 de pasos de interacción). Se cargan los pesos de los modelos preentrenados en Pong en la primera ronda de entrenamiento en las capas convolucionales de los nuevos modelos para Quadrapong, adaptándolos mediante un mapeo de acciones para alinear con el espacio de acciones de Quadrapong. A partir de ahí, se sigue la misma línea de entrenamiento utilizando la librería Stable Baselines3, DummyVecEnv y

VecTransposeImage. En este caso se alcanzarán un total de 4.000 rondas de preentrenamiento, lo que equivale a 20.000.000 de pasos (Parameter Transfer) y además realizaremos 1.000 rondas más de entrenamiento (Fine-tuning), llegando a un total de 25.000.000 de pasos de interacción con el entorno.

3.7.4. Ciclo de aprendizaje de los agentes en entrenamiento Quadrapong con Transfer Learning y Fine-tuning

Si observamos la [Figura 3.5](#), el ciclo de aprendizaje durante cada ronda de entrenamiento se diferencia de los anteriores tan solo, en la carga de los pesos de los modelos preentrenados, constando con las siguientes etapas:

- **Carga modelos preentrenados**

En la ronda 1, los modelos para cada agente se inicializan cargando los pesos de los modelos preentrenados de Pong para las capas convolucionales mediante transferencia de conocimiento, adaptando la política al nuevo entorno (mapeo de acciones).

- **Observación del estado**

- **Selección de acción**

- **Interacción con el entorno**

- **Almacenamiento de la experiencia**

- **Actualización de la política**

- **Repetición de ciclo**

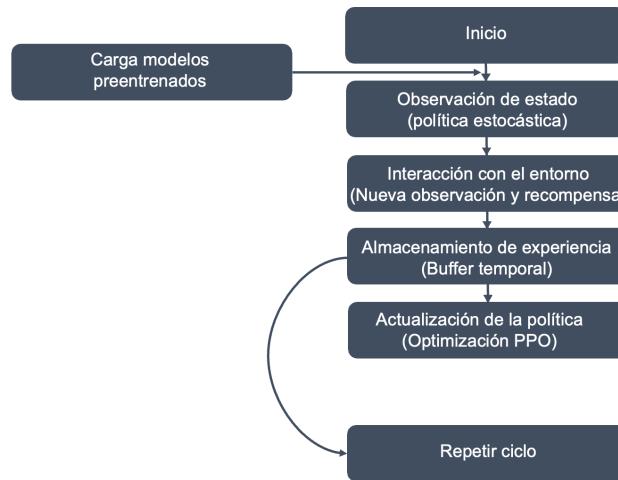


Figura 3.5: Diagrama del ciclo de entrenamiento de un agente en Quadrapong con transferencia de conocimiento

3.7.5. Configuración del entrenamiento en Quadrapong

En Quadrapong, seguimos manteniendo un entrenamiento en un entorno estrictamente competitivo para los agentes. No existe comunicación o coordinación entre agentes y se aplica la siguiente función de recompensa:

- Anotar un punto le da al equipo +1 punto de recompensa y al equipo oponente -1 punto de recompensa.
- Si cualquiera de los agentes de un mismo equipo falla (la pelota le sobrepasa), el equipo recibe -1 punto.
- En caso de "stalling" (no servir tras dos segundos), equipo recibe -1 punto.

Esta función de recompensa refuerza estrategias puramente competitivas, orientadas a maximizar la ventaja sobre el oponente.

3.8. Recolección de datos y métricas

Durante los entrenamientos, se registran una serie de métricas de interés que nos permiten analizar la evolución de la política de los agentes, la calidad del aprendizaje y poder realizar comparaciones entre agentes.

Las diferentes métricas que se registran por cada agente en los diferentes entornos son:

- **Pérdidas promedio:**

Se registran la pérdida para evaluar la convergencia y estabilidad del entrenamiento. Estas

métricas permiten detectar posibles problemas como sobreajuste, falta de convergencia o políticas demasiado deterministas.

- **Recompensa acumulada:**

Se registra la suma de recompensas obtenidas por cada agente para evaluar su rendimiento a lo largo del entrenamiento.

- **Impactos por agente:**

Los impactos de la pelota en las palas de cada agente se monitorean como una métrica específica que refleja la interacción directa con el entorno y la habilidad para mantener el control del juego. En el [Apéndice A](#), se detalla como ha sido el proceso que nos ha permitido obtener los impactos de la pelota en cada pala.

- **Checkpoint y registros para análisis:**

Cada cierto número de rondas, se realizan guardados de checkpoint para preservar el progreso.

- **Modelos y estado de entrenamiento:**

Los pesos de los modelos de ambos agentes se guardan periódicamente tras cada ronda, junto con un archivo de estado que permite reanudar el entrenamiento sin pérdida de información.

Esta recopilación de métricas garantiza un seguimiento detallado y profundo del aprendizaje, facilitando tanto la evaluación del desempeño como la identificación temprana de posibles problemas durante el entrenamiento.

3.9. Pruebas y ajustes

Con el objetivo de validar el aprendizaje de los agentes y optimizar los hiperparámetros de entrenamiento, se han llevado a cabo diversas pruebas tanto en el entorno Pong como en Quadrapong, complementadas con ajustes basados en los resultados obtenidos.

La evaluación periódica no solo ha permitido medir el desempeño cuantitativo, sino también identificar comportamientos cualitativos propios de entornos multiagentes.

3.9.1. Pruebas en Pong

Durante y después del entrenamiento en el entorno Pong, se han desarrollado e implementado varios scripts específicos de evaluación:

- **Visualizar enfrentamientos**

Se ejecutan partidas entre agentes first_0 y second_0 cada 100 rondas de entrenamiento. Esto permite realizar observaciones cualitativas sobre la evolución de estrategias y patrones de juego (por ejemplo, ataques agresivos o defensas conservadoras). También se llevan a cabo una serie de partidas consecutivas (5 partidas), cada 500 rondas, para además de analizar la evolución del aprendizaje hasta ese momento, cual de los agentes ha alcanzado la mejor política.

- **Medición de recompensas promedio por agente**

Se registra la recompensa acumulada por cada agente en partidas de evaluación. Esta métrica es esencial para cuantificar la capacidad competitiva individual y detectar tendencias de mejora o estancamiento a lo largo del tiempo.

- **Medición de impactos con la pelota de cada pala**

Se registran los impactos acumulados por cada agente con la pelota en partidas de evaluación. Esta métrica nos genera una perspectiva de la evolución del aprendizaje, puesto que, a mayor número de impacto, mayor es la interacción de cada agente con el entorno.

- **Registro de métricas de entrenamiento**

Todas las métricas relevantes (recompensa promedio, pérdida, impactos en palas, ...) son almacenadas en archivos .csv para su posterior análisis y representación gráfica.

Estas pruebas han permitido evaluar no solo el rendimiento numérico, sino también la evolución del comportamiento estratégico de los agentes.

3.9.2. Pruebas en Quadrapong

La evaluación en Quadrapong introduce un nivel adicional de complejidad, al tratarse de un entorno 2vs2 donde la coordinación y la competencia entre equipos juegan un papel importante. Para este escenario, se han desarrollado las siguientes pruebas:

- **Visualización de enfrentamientos entre equipos**

Se generan partidas visibles entre los dos equipos (Equipo 1 y Equipo 2) cada 100 rondas de entrenamiento. Esta observación, facilita la identificación de patrones de colaboración, fallos de coordinación o estrategias.

- **Medición de recompensas individuales y por equipo**

Además de registrar las recompensas de cada agente, se calculan las recompensas por equipo. Esto permite analizar la dinámica de colaboración dentro de un mismo equipo, y las diferencias entre los dos equipos.

■ Comparación entre entrenamiento con y sin transferencia de conocimiento

Se realizan evaluaciones cruzadas entre agentes que han recibido transferencia de parámetros desde Pong y agentes inicializados aleatoriamente. El objetivo es medir la diferencia en términos de:

- Velocidad de convergencia.
- Rendimiento final.
- Robustez.

■ Análisis de comportamientos emergentes

Se examinan comportamientos como:

- **Colaboración espontánea:** estrategias cooperativas sin comunicación previa.
- **Competencia interna:** agentes que perjudican a sus propios compañeros.
- **Descoordinación:** fallos de sincronización en movimientos defensivos u ofensivos.

Estas pruebas permiten entender mejor cómo influye el aprendizaje previo y su naturaleza en la dinámica multiagente compleja.

Las pruebas y ajustes implementados han permitido no solo confirmar el correcto aprendizaje de los agentes, sino también optimizar el proceso de entrenamiento de manera dinámica. Gracias a ello, se han obtenido modelos más sólidos y comportamientos adaptados a la complejidad de los distintos entornos multiagente.

Capítulo 4

Presentación de resultados

En este capítulo se presentan los resultados obtenidos tras el entrenamiento de los agentes utilizando el algoritmo Proximal Policy Optimization (PPO) en el entorno competitivo de Pong. El proceso de entrenamiento comprendió un total de 5000 rondas, equivalentes a aproximadamente 25 millones de pasos de interacción con el entorno.

4.1. Metodología de evaluación

Para evaluar la evolución y desempeño de los agentes, se realizaron partidas enfrentando a los dos agentes entrenados, en condiciones controladas. La evaluación se llevó a cabo cada 100 rondas de entrenamiento para observar la progresión de las políticas a lo largo del tiempo. En cada evaluación se registraron:

- La recompensa obtenida por cada agente durante la partida, que refleja su capacidad para ganar puntos y controlar el juego.
- El número de impactos de la pelota contra la pala de cada agente, métrica que permite analizar la interacción directa y la capacidad defensiva/ofensiva de los agentes.

También se ha realizado cada 500 rondas de entrenamiento, evaluación durante un total de 5 partidas en Pong, con el objetivo de ver qué capacidad tiene cada agente de imponerse a su rival, y de qué manera lo hace, para ello hemos registrado:

- El número de victorias conseguidas durante las cinco partidas, así comprobamos cuál de los agentes está alcanzando una política más efectiva.
- El número de impactos totales de cada agente con la pelota, esto nos facilita el análisis de la relación existente entre los impactos y las victorias conseguidas.

4.2. Análisis de recompensas en Pong

La Figura 4.1 muestra la evolución de las recompensas obtenidas por ambos agentes a lo largo del entrenamiento. El eje X representa el número de ronda y el eje Y representa el valor de la recompensa acumulada por cada agente en dicha ronda.

Durante la fase inicial del entrenamiento (aproximadamente entre las rondas 0 y 1500), se observa un rendimiento superior por parte del agente second_0. Este agente alcanza recompensas elevadas y relativamente estables, lo que sugiere un aprendizaje temprano efectivo. En cambio, el agente first_0 presenta una alta variabilidad en sus resultados, incluyendo múltiples rondas con recompensas negativas pronunciadas, lo que indica que su política aún se encuentra en fase de exploración.

A partir de la ronda 1500 se produce un cambio en la tendencia. El agente first_0 mejora considerablemente su rendimiento, estabilizando su comportamiento en torno a recompensas positivas. Esta evolución sugiere una adaptación exitosa a las dinámicas del entorno y, posiblemente, a las estrategias del agente oponente. En contraposición, el rendimiento del agente second_0 comienza a decaer progresivamente, alcanzando valores negativos en varias rondas.

En la fase final del entrenamiento (entre las rondas 3000 y 5000), ambos agentes muestran una alternancia en las rondas, con recompensas que fluctúan de forma más equilibrada. Esta oscilación podría reflejar un entorno más competitivo, en el que las políticas aprendidas por los agentes tienden a contrarrestarse mutuamente.

En conjunto, la gráfica revela un proceso de aprendizaje desigual entre los dos agentes, donde el agente first_0 evidencia una progresión sostenida y una mejora tardía, mientras que el agente second_0, tras un inicio prometedor, parece no mantener dicha evolución en fases posteriores del entrenamiento.

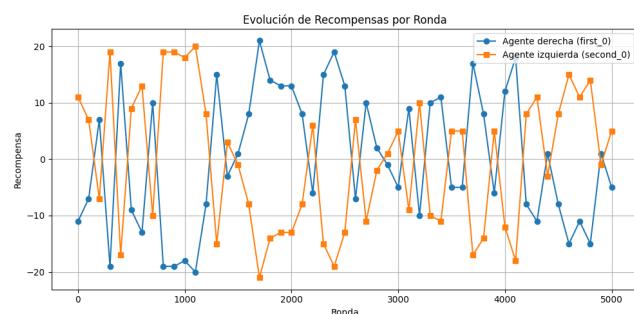


Figura 4.1: Evolución de recompensas por rondas en Pong

4.3. Análisis de impactos de la pelota en Pong

La Figura 4.2 presenta el número de impactos de la pelota con las palas de cada agente por ronda.

El conteo de impactos de la pelota contra las palas de los agentes ofrece una perspectiva complementaria al desempeño basado en recompensas. Un mayor número de impactos sugiere una mayor capacidad para controlar la pelota y mantener el intercambio durante el juego.

Ambos agentes incrementan su actividad de golpeo con el tiempo, lo cual es consistente con el proceso de aprendizaje. Sin embargo, en el caso del agente second_0, a pesar de mantener un número alto de impactos, sus recompensas disminuyen. Esto sugiere que, aunque logra devolver la pelota, no consigue transformar esta actividad en resultados efectivos. El agente first_0 muestra una relación más coherente entre impactos y mejora de recompensas.

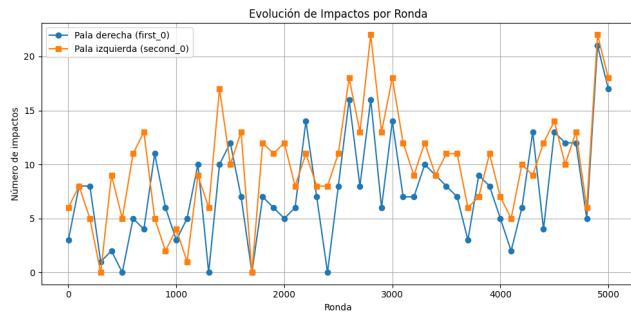


Figura 4.2: Evolución de impactos por rondas en Pong

4.4. Resumen y conclusiones preliminares en Pong

La Figura 4.3 y Figura 4.4 presentan la evolución conjunta de impactos y recompensas por ronda para cada agente. Esta visualización permite identificar correlaciones entre el rendimiento efectivo y la actividad de juego.

En las primeras rondas, se observa una separación entre actividad y recompensa, el agente first_0 golpea con relativa frecuencia pero recibe penalizaciones, indicando errores estratégicos. A partir de la ronda 1000, se consolida una relación positiva, los impactos aumentan y las recompensas se tornan positivas, sugiriendo que first_0 ha aprendido no solo a devolver la pelota, sino a hacerlo de forma táctica y eficaz. Por el contrario, el agente second_0 mantiene un patrón inverso al anterior, a pesar de mantener un alto nivel de impactos durante la mayor parte del entrenamiento, las recompensas tienden a disminuir.

Este fenómeno revela un desajuste entre actividad y eficacia. Aunque second_0 logra mantener la pelota en juego, no consigue capitalizar esos intercambios en forma de puntos, victorias

o recompensas positivas. Esto podría deberse a errores estratégicos recurrentes.

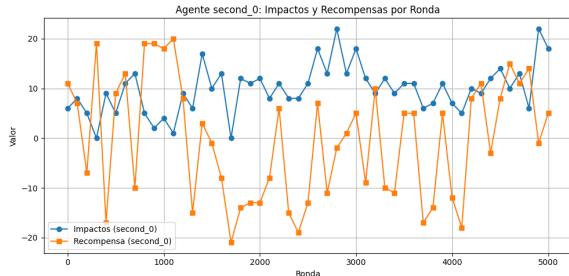


Figura 4.3: Impactos y recompensas por ronda agente second_0 en Pong

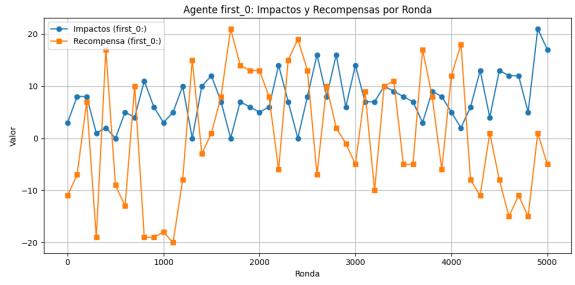


Figura 4.4: Impactos y recompensas por ronda agente first_0 en Pong

Por otro lado, en la [Figura 4.5](#), se muestra la evolución del número total de impactos con la pelota registrados por cada agente después de ejecutar cinco partidas de evaluación cada 500 rondas de entrenamiento. A partir de la ronda 1000, se observa una mejora progresiva en la capacidad de mantener la pelota en juego, alcanzando un pico importante en la ronda 3000. Desde entonces, ambos agentes mantienen un número de impactos relativamente estable entre 58 y 70 impactos, lo que indica que han alcanzado un nivel de aprendizaje consolidado. Las oscilaciones menores podrían atribuirse a ajustes tácticos o pequeñas variaciones de comportamiento tras cada ciclo de entrenamiento.

En general, la diferencia entre ambos agentes es pequeña a partir de la ronda 1500, lo que refleja un nivel de desempeño técnico muy similar en términos de impactos.

Estos resultados indican que, si bien ambos agentes han aprendido a reaccionar frente al entorno, el agente second_0 ha logrado una mayor consistencia en su comportamiento ante la capacidad de golpear la pelota. No obstante, es importante destacar que un mayor número de impactos no implica necesariamente un mayor éxito en el juego.

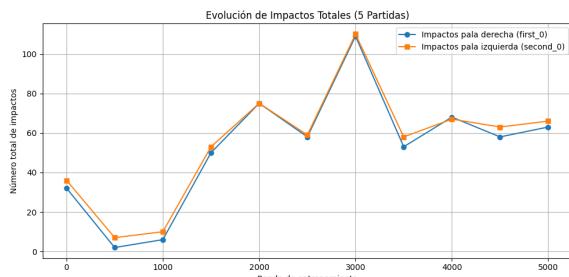


Figura 4.5: Impactos durante cinco partidas de cada agente en Pong

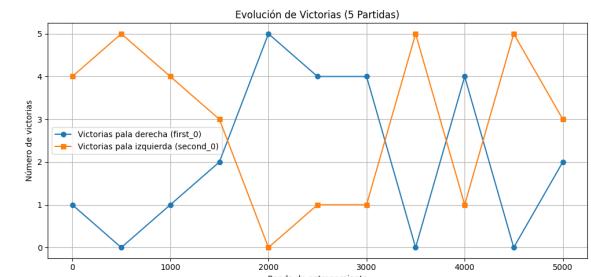


Figura 4.6: victorias durante cinco partidas de cada agente en Pong

Si ahora observamos la [Figura 4.6](#), comprobamos que en las primeras rondas, el agente second_0 muestra un claro dominio, llegando a ganar la mayoría de partidas en múltiples

ocasiones. Esto sugiere que desarrolló inicialmente una política efectiva tanto en defensa como en ataque. Sin embargo, a partir de la ronda 1500, comienza a perder eficacia. Aunque su número de impactos sigue aumentando en las rondas posteriores, el número de victorias cae drásticamente, alcanzando un mínimo de 0 victorias en la ronda 2000 y apenas 1 victoria en las rondas 2500 y 3000. Esto indica que, a pesar de mantener una interacción elevada con la pelota, su política deja de ser competitiva, impacta más, pero no gana.

En contraste, el agente derecho (first_0) empieza con un rendimiento limitado, tanto en impactos como en victorias. Sin embargo, a partir de la ronda 1500 se observa una mejora continuada, aumenta su número de impactos y logra convertir esa mejora en resultados efectivos, obteniendo 5 victorias en la ronda 2000 y consolidando su rendimiento con 4 victorias en las rondas 2500 y 3000. Esto evidencia una evolución positiva en su política, que no solo incrementa su capacidad de mantener la pelota en juego, sino también su capacidad de capitalizar esas interacciones para ganar.

A partir de ahí (ronda 3000), se desarrolla un patrón de alternancia competitiva, los agentes parecen adaptarse el uno al otro, y se van turnando en la victoria total o parcial. Este comportamiento sugiere adaptación o evolución, donde las políticas aprendidas no son estáticas, sino que responden activamente a la estrategia del oponente.

Este análisis pone de manifiesto que el número de impactos por sí solo no refleja completamente el aprendizaje alcanzado por un agente. Es crucial analizar conjuntamente las métricas de actividad y resultado. En este caso, el agente derecho ha demostrado una mejora estratégica clara en la segunda mitad del entrenamiento, mientras que el agente izquierdo ha mantenido una política activa pero ineficaz en términos competitivos.

Los resultados obtenidos demuestran que el entrenamiento basado en PPO permite desarrollar políticas efectivas para competir en el entorno de Pong. La combinación de análisis de recompensas y métricas específicas como los impactos o victorias ofrece una visión integral del aprendizaje y la evolución estratégica de los agentes.

4.5. Comparación entre entrenamiento con y sin transferencia de conocimiento

Se realiza un análisis comparativo entre dos estrategias de entrenamiento en Quadrapong, una que utiliza transferencia de conocimiento desde los modelos entrenados en Pong y otra que entrena los agentes desde cero en Quadrapong sin preentrenamiento.

4.5. Comparación entre entrenamiento con y sin transferencia de conocimiento 59

4.5.1. Análisis de recompensas por equipos en Quadrapong

En esta sección se examinan las recompensas obtenidas por cada equipo en el entorno Quadrapong atendiendo a si es un modelo con Parameter Transfer y Fine-tuning o es un modelo entrenado desde cero. Se evalúa la evolución del rendimiento a lo largo de las rondas, destacando las diferencias y similitudes en su aprendizaje y adaptación a la nueva dinámica de juego, que involucra cuatro agentes repartidos en equipos de dos.

- **Modelo entrenado desde cero.**

Como se aprecia en la [Figura 4.7](#), en las primeras rondas, ambos equipos oscilan en torno a valores cercanos a cero, lo que es coherente con un comportamiento inicial aleatorio propio de las primeras fases de aprendizaje.

Sin embargo, a partir de la ronda 600, se observa inestabilidad en el rendimiento del equipo 1, las recompensas caen drásticamente a valores negativos muy altos. Este patrón se repite de forma intermitente cada ciertas rondas. Tras un análisis detallado de los episodios de evaluación correspondientes, se identificaron patrones de comportamientos poco funcionales que explican este fenómeno:

- **Intercambios repetitivos entre agentes rivales sin movimiento:**

Dos agentes posicionados frente a frente mantienen un intercambio constante de impactos con la pelota sin moverse ni alterar la trayectoria, generando un bucle de actividad inútil desde el punto de vista estratégico.

- **Rebotes prolongados contra las paredes:**

En este caso, la pelota quedaba atrapada entre un agente y una pared, provocando un patrón repetitivo de rebotes sin que se produjera ningún tipo de progresión o interacción significativa.

Estos bucles terminaban únicamente en dos situaciones: cuando un tercer agente intervenía y rompía el patrón, o cuando el episodio era truncado por alcanzar el límite de pasos estimado.

Este tipo de comportamiento explica los descensos significativos en la recompensa, especialmente en el caso del equipo 1. Pese a que estos agentes registraban un alto número de impactos, su calidad de juego es deficiente desde una perspectiva estratégica, lo cual se refleja en la falta de correspondencia entre impactos y recompensa.

En cambio, el equipo 2 mantiene un comportamiento más estable y consigue recompensas positivas más consistentes, mostrando una capacidad de adaptación más sólida a lo largo del tiempo.

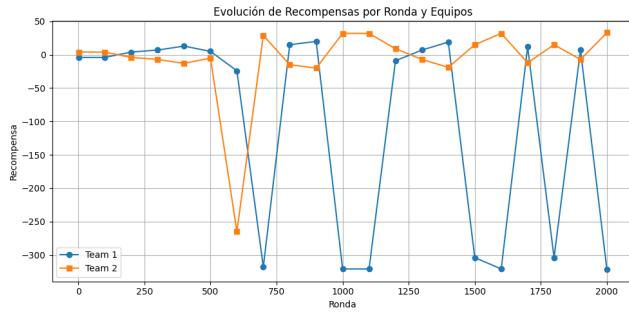


Figura 4.7: Evolución de recompensas por Rondas y Equipos (Sin transferencia de conocimiento)

- **Modelo con Transfer Learning y Fine-tuning.**

En la [Figura 4.8](#), podemos observar que el equipo 1 (con conocimiento transferido del agente first_0 en Pong), aunque presenta una ligera inestabilidad en rondas intermedias, mantiene un rendimiento predominantemente positivo o cerca del equilibrio, mostrando una capacidad para aplicar eficazmente el conocimiento previo adquirido.

En contraste, el equipo 2 (con conocimiento transferido del agente second_0 en Pong) evidencia inestabilidad y valores negativos más frecuentes, lo que refleja una menor transferencia efectiva. Este comportamiento podría estar vinculado al rendimiento ya más limitado que el agente second_0 mostró previamente en el entorno Pong.

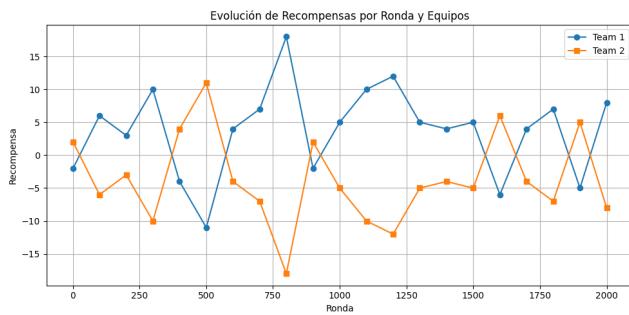


Figura 4.8: Evolución de recompensas por Rondas y Equipos (Con transferencia de conocimiento)

El modelo con transferencia de conocimiento y fine-tuning, muestra una evolución más estable de las recompensas por equipos, manteniendo un comportamiento competitivo, equilibrado y sostenido. Esto indica una mejora sustancial del rendimiento, especialmente en el equipo que recibió la mejor política desde Pong. En contraste, el modelo entrenado desde cero presenta una mayor volatilidad, con fluctuaciones pronunciadas y períodos amplios de recompensas negativas, reflejando un proceso de aprendizaje más lento y errático.

4.5.2. Análisis de impactos a la pelota por agentes en Quadrapong

En este apartado se estudia la frecuencia y distribución de los impactos de la pelota en las palas de cada agente, información clave para comprender la interacción física y el control del juego en Quadrapong.

- **Modelo entrenado desde cero.**

En la [Figura 4.9](#), se observa una distribución heterogénea y con picos irregulares, lo que sugiere poca coordinación y control sobre la pelota. Algunos agentes muestran períodos con muy pocos impactos, indicando que podrían tener dificultades para participar activamente en el juego, mientras que otros tienen episodios aislados de alta actividad. Aunque, parte de estos valores elevados se explican por la aparición de comportamientos no deseados, como los mencionados anteriormente. Los intercambios repetitivos entre agentes enfrentados sin movimiento ni cambio de trayectoria, provocan un gran número de impactos sin progreso real en el juego. Esto genera un aumento artificial de la métrica de impactos, especialmente en las palas más involucradas en esas posiciones.

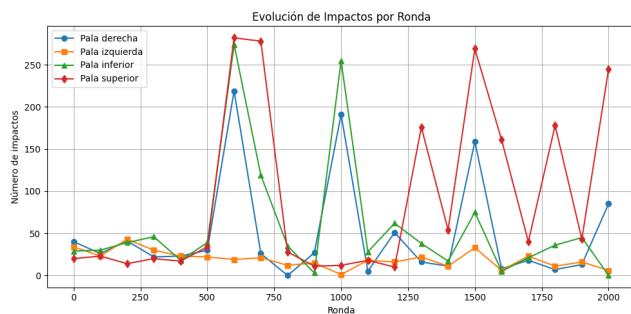


Figura 4.9: Evolución individual de impactos por Rondas (Sin transferencia de conocimiento)

- **Modelo con Transfer Learning y Fine-tuning.**

Analizando la [Figura 4.10](#), se observan diferencias sustanciales en la cantidad de impactos registrados por cada pala, lo que pone de manifiesto distintos niveles de participación y eficacia en la interacción con la pelota.

En términos generales, las palas inferior (third_0) y superior (fourth_0) muestran una actividad más estable y equilibrada a lo largo de las distintas rondas, lo cual sugiere un comportamiento consistente en su implicación durante el juego.

Por otro lado, la pala izquierda (second_0) presenta picos de actividad elevados en ciertas rondas, sin que ello se traduzca en una mejora proporcional de las recompensas acumuladas, lo cual indica una participación alta pero limitada. Este comportamiento refuerza

la hipótesis de que el conocimiento transferido desde el agente original second_0 no se traduce de forma efectiva en estrategias exitosas dentro del nuevo entorno.

En contraste, la pala derecha (first_0) mantiene un número de impactos más moderado, pero con una evolución constante a lo largo del tiempo. Esta estabilidad puede interpretarse como una indicación de una política más eficiente y menos dependiente del intercambio continuo con la pelota, posiblemente como resultado de la política previamente aprendida en Pong por el agente first_0.

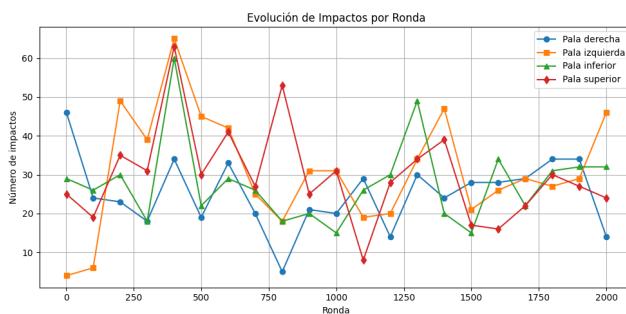


Figura 4.10: Evolución individual de impactos por Rondas (Con transferencia de conocimiento)

Los agentes del modelo con transferencia, mantienen más control y equilibrio sobre la pelota, con impactos distribuidos con cierta uniformidad y una participación activa continua durante las rondas. Esta regularidad sugiere que el conocimiento previo permite una interacción más efectiva y estratégica, lo que en conjunto mejora el desempeño global de los equipos.

4.5.3. Análisis de impactos a la pelota por equipos en Quadrapong

En esta ocasión, se estudia la frecuencia y distribución de los impactos de la pelota en las palas a nivel de equipos, facilitando el estudio de la evolución de las estrategias de los equipos durante el juego en Quadrapong.

- **Modelo entrenado desde cero.**

Observando la [Figura 4.11](#), se aprecia un comportamiento poco estable, con grandes oscilaciones y sin una clara tendencia de mejora en el control de la pelota. No obstante, al considerar los comportamientos no deseados observados, es evidente que estos picos no siempre reflejan una mejora en el rendimiento. Los elevados conteos de impactos están relacionados con interacciones repetitivas en bucle entre palas del equipo rival, más que con una verdadera capacidad de control y estrategia. Así, aunque el número total de impactos puede sugerir inicialmente una mayor implicación de un equipo en el juego, estos valores deben interpretarse con cautela.

4.5. Comparación entre entrenamiento con y sin transferencia de conocimiento 63

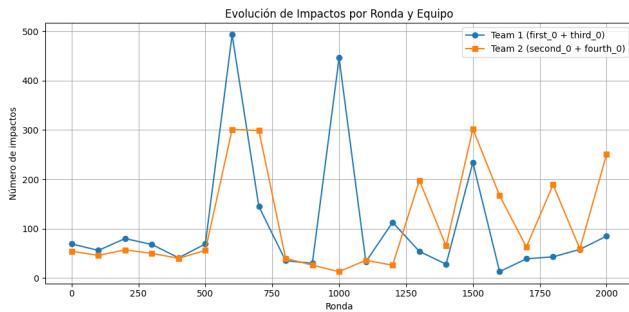


Figura 4.11: Evolución por equipos de impactos por Rondas (Sin transferencia de conocimiento)

En la [Figura 4.12](#) se observa que los episodios con más impactos no siempre corresponden a mejores recompensas. De hecho, se identifican varias rondas con elevada actividad pero recompensas extremadamente negativas. Como se ha descrito anteriormente, esto se debe a que los agentes entran en bucles de comportamiento ineficiente, en los que golpean la pelota reiteradamente sin intención ni estrategia, acumulando impactos que no generan progreso.

La [Figura 4.13](#), muestra una relación más estable entre ambas métricas para el equipo 2. Cuando aumentan sus impactos, la recompensa tiende también a mejorar, lo que sugiere una política más eficiente y efectiva, centrada no solo en mantener la pelota en juego, sino en hacerlo de manera ventajosa.

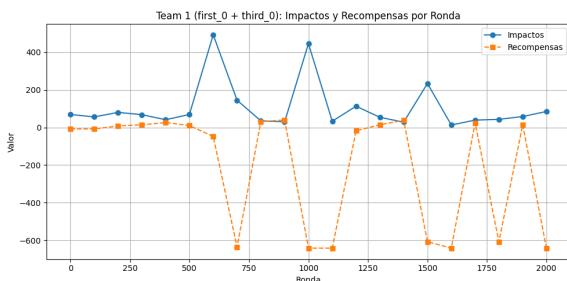


Figura 4.12: Impactos y recompensas por ronda, Team 1 (Sin transferencia de conocimiento)

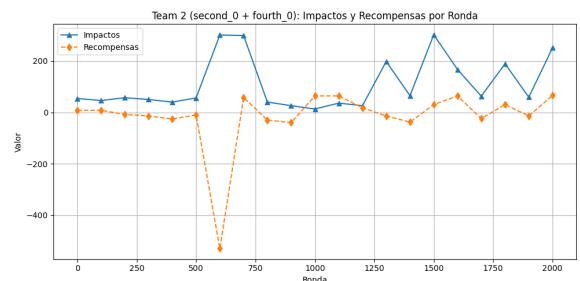


Figura 4.13: Impactos y recompensas por ronda, Team 2 (Sin transferencia de conocimiento)

- **Modelo con Transfer Learning y Fine-tuning.**

La [Figura 4.14](#) describe que en términos generales, el equipo 2 supera en impactos al equipo 1 en la mayoría de las rondas. No obstante, este mayor volumen de actividad no se refleja en mejores recompensas, lo que refuerza la hipótesis de que el equipo 2 realiza acciones menos efectivas, pese a su alto nivel de interacción con la pelota.

En cambio, el equipo 1, con menos impactos, muestra un mejor aprovechamiento de las oportunidades de juego, transformando sus impactos en situaciones ventajosas, lo que refleja una política de mayor calidad heredada del agente first_0 original.

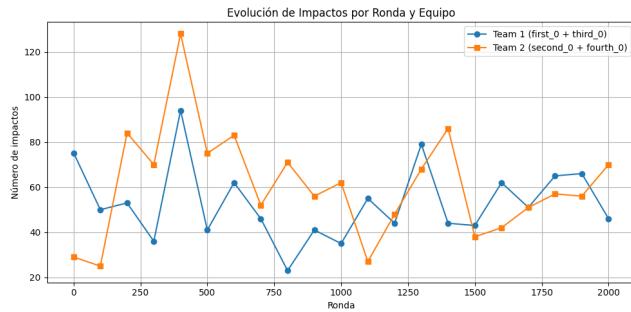


Figura 4.14: Evolución por equipo de impactos por Rondas (Con transferencia de conocimiento)

En la [Figura 4.15](#) y la [Figura 4.16](#), se puede observar una correlación clara entre el número de impactos y las recompensas. En varias rondas, tanto los impactos como las recompensas se elevan, lo que indica que el equipo es capaz de mantener intercambios de calidad y concluirlos con éxito.

La política heredada del agente first_0 parece facilitar una coordinación eficiente entre los miembros del equipo. A pesar de ligeras caídas de rendimiento, el patrón general es positivo, con una clara conversión de actividad en recompensa.

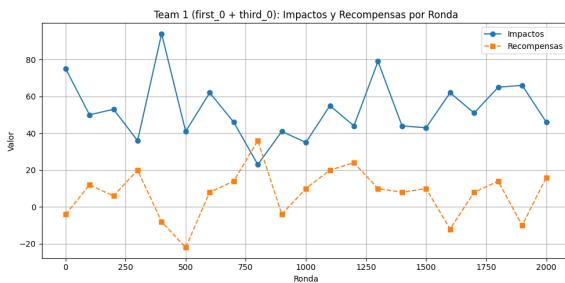


Figura 4.15: Impactos y recompensas por ronda, Team 1 (Con transferencia de conocimiento)

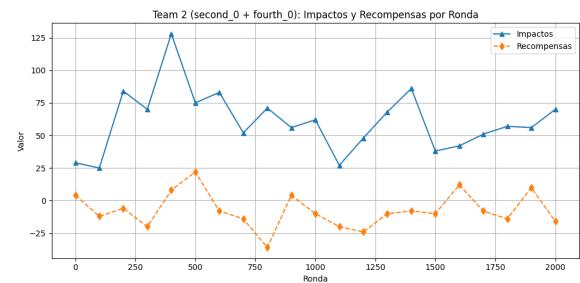


Figura 4.16: Impactos y recompensas por ronda, Team 2 (Con transferencia de conocimiento)

Para el equipo 2, se evidencia una desconexión entre impactos y recompensas. Se identifican algunas rondas donde a pesar de un elevado número de impactos, la recompensa no alcanza valores máximos. Esto podría indicar situaciones en las que la presión ofensiva no se traduce directamente en puntos, posiblemente debido a errores en la estrategia o fallos de coordinación, lo que sugiere ineficiencia en la ejecución de la política aprendida.

Este comportamiento podría derivarse del rendimiento ya mostrado por el agente second_0 en Pong, donde a pesar de mantener altas cifras de impactos, no se traducían en recompensas positivas.

A nivel de equipo, el modelo con transferencia evidencia una progresión más estable y creciente en el número total de impactos, lo que refleja un mejor dominio y capacidad de respuesta conjunta. A diferencia, el modelo sin transferencia muestra oscilaciones significativas y falta de una tendencia clara de mejora sostenida, señalando deficiencias en el aprendizaje a consecuencia de comportamientos no deseado, considerándose necesario un entrenamiento más prolongado.

4.6. Resumen y conclusiones preliminares en Quadrapong

En este trabajo se han comparado dos enfoques para entrenar agentes en el entorno Quadrapong, un modelo entrenado desde cero y otro basado en transferencia de conocimiento y fine-tuning desde modelos previamente entrenados en Pong.

En conjunto, estos análisis confirman que la transferencia de conocimiento ha permitido reducir las anomalías en el número de impactos y ha favorecido interacciones más eficientes, reflejando un entorno de entrenamiento mejor regulado. Esto sugiere una mejora clara en el proceso de aprendizaje colectivo de los equipos.

La transferencia de conocimiento facilita un uso más eficiente de las interacciones con la pelota, donde la actividad tiene mayor valor táctico y se traduce en recompensas. Esto indica que el conocimiento previo ayuda a priorizar la calidad del juego sobre la cantidad de acción.

Incluso una política menos eficaz en Pong puede servir como punto de partida útil. La transferencia mitiga los errores graves de comportamiento, da estabilidad al aprendizaje temprano y genera una mejor relación entre esfuerzo y resultado, aunque con menor eficacia que en el equipo con mejor política transferida.

Capítulo 5

Conclusiones y trabajo futuro

■ Conclusiones del trabajo

El presente proyecto ha demostrado la viabilidad y ventajas del uso de técnicas de transferencia de conocimiento en el entrenamiento de agentes inteligentes para entornos multiagente complejos como Quadrapong. La combinación de modelos preentrenados en entornos más simples con procesos de fine-tuning ha permitido una mejora significativa en la eficiencia y efectividad del aprendizaje, evidenciada en métricas cuantitativas de rendimiento y control del juego.

Los resultados muestran que el modelo con transferencia de conocimiento presenta una mejora más rápida y estable en términos de recompensas acumuladas, impactos a la pelota tanto a nivel individual como por equipos, evidenciando una mayor capacidad de adaptación y coordinación en el entorno multiagente. Por el contrario, el entrenamiento desde cero se caracteriza por una mayor variabilidad y una evolución más lenta, con dificultades notables para lograr un control efectivo a causa de comportamientos no deseado.

Estas diferencias demuestran que la transferencia de conocimiento es una estrategia eficaz para acelerar el aprendizaje y mejorar la robustez de los agentes en entornos complejos y con múltiples agentes, favoreciendo la adquisición de habilidades tácticas y colaborativas.

■ Evaluación crítica del grado de logro de los objetivos iniciales

Se han alcanzado el objetivo principal, que consistía en investigar la transferencia de conocimiento desde Pong a Quadrapong, evaluando el impacto en la competencia de los agentes. Los resultados obtenidos permiten concluir que la transferencia mejora significativamente la velocidad y calidad del aprendizaje en entornos multiagente complejos.

Sin embargo, han quedado pendiente algún objetivo secundario, la exploración de diferentes configuraciones de recompensa (competitivas y colaborativas) podrían potenciar una mejor coordinación y colaboración entre agentes en Quadrapong.

Además, hubiera sido valioso profundizar en la experimentación con arquitecturas heterogéneas, entrenando inicialmente un agente en Pong con un modelo específico (por ejemplo, DQN o PPO) y otro agente con un modelo distinto. Posteriormente, durante la transferencia de conocimiento a Quadrapong, se podrían haber asignado estos modelos diferentes a equipos distintos o incluso mezclarlos dentro de un mismo equipo. Este planteamiento permitiría analizar de forma más detallada cómo la diversidad de políticas y algoritmos afecta la dinámica competitiva y cooperativa, y qué combinaciones resultan más efectivas en entornos multiagente.

Estos enfoques abren nuevas líneas de investigación que pueden contribuir a una comprensión más profunda del aprendizaje multiagente y la transferencia de conocimiento en entornos complejos.

■ Evaluación crítica de la planificación y metodología

El desarrollo de este trabajo ha implicado un reto significativo. Manejar múltiples modelos y etapas de entrenamiento totalmente dependientes, no ha sido fácil. Inicialmente, el enfoque propuesto consistía en entrenar agentes con DQN en Pong como base para luego transferir el conocimiento a Quadrapong. Sin embargo, durante la fase de experimentación se encontraron numerosas dificultades con DQN, incluyendo problemas de estabilidad y convergencia. Esta situación, prolongó excesivamente el tiempo requerido para obtener resultados, los cuales, tras decenas de millones de pasos de entrenamiento seguían sin ser satisfactorios.

Estas limitaciones llevaron a la decisión de adoptar otra alternativa, se decidió utilizar Proximal Policy Optimization (PPO) como algoritmo principal. Este algoritmo presenta ventajas tales como, una mayor estabilidad en el entrenamiento, mejor manejo de espacios de acciones continuos y discretos, así como, una convergencia más rápida y robusta en entornos complejos. Las cuales, se consideraron adecuadas para obtener resultados satisfactorios en el presente trabajo, teniendo en cuenta los escenarios multiagentes utilizados.

El entrenamiento implica, primero obtener modelos competentes en Pong con PPO para poder efectuar la transferencia de conocimiento a Quadrapong. En paralelo, es necesario entrenar modelos desde cero en Quadrapong para poder comparar y validar el impacto de la transferencia y el fine-tuning. Esta doble vía y en ocasiones triple, multiplicó el número de modelos a entrenar, añadiendo una complejidad considerable a la planificación y ejecución.

En conjunto, estas dificultades reflejan los desafíos prácticos propios de este tipo de trabajo con entornos multiagente que conllevan la implementación de estrategias de transferencia de aprendizaje. La planificación y metodología empleadas fueron flexibles para adaptarse

a estos obstáculos, pero resaltan la importancia de una gestión cuidadosa del tiempo y recursos.

■ **Desafíos de sostenibilidad, diversidad y ético-sociales vinculados al proyecto**

Durante el desarrollo del presente trabajo, se ha prestado especial atención a los aspectos de sostenibilidad, diversidad y responsabilidad ética y social. Estos factores, aunque en principio presentan una influencia mínima debido al enfoque simulado del entorno, no deben ser ignorados dadas las implicaciones indirectas que pueden tener, especialmente en el consumo energético.

• **Sostenibilidad**

Durante el desarrollo del proyecto, se ha evaluado la huella de carbono generada en dos estrategias distintas de entrenamiento de agentes multiagente. El objetivo era comparar la eficiencia energética entre:

1. **Enfoque con transferencia de conocimiento**

- Entrenamiento previo en el entorno Pong con 2 agentes durante 5000 rondas, seguido de
- Entrenamiento en Quadrapong con 4 agentes durante 2000 rondas, utilizando los pesos aprendidos en Pong.

2. **Enfoque sin transferencia de conocimiento**

- Entrenamiento desde cero directamente en Quadrapong con 4 agentes durante 2000 rondas.

Resultados de la medición y estimación de huella de carbono:

Estrategia	Entorno(s)	Rondas totales	Huella CO ₂ (kg)	Potencia CPU estimada (W)
Con transferencia	Pong + Quadrapong	5000 + 2000	8,7810 ⁻⁸	60,24
Sin transferencia	Solo Quadrapong	2000	3,9110 ⁻⁸	26,83

Tabla 5.1: Comparativa de huella de carbono entre estrategias de entrenamiento

Si estudiamos la [Tabla 5.1](#), a primera vista el enfoque sin transferencia parece más eficiente por generar una huella menor. Sin embargo, esta impresión es engañosa si se analiza el número de agentes entrenados y la escalabilidad del entorno:

- El entrenamiento en Pong involucra solo 2 agentes, mientras que Quadrapong entrena simultáneamente 4 agentes.
- La transferencia permite reutilizar conocimiento previo, reduciendo significativamente el número de rondas necesarias en el entorno más complejo.

- En términos de huella de carbono por agente y por ronda efectiva, el enfoque con transferencia resulta más favorable.

Aunque la métrica por ronda parece menor en el enfoque sin transferencia, este requiere muchas más rondas para alcanzar el mismo resultado. Además, la correlación entre el entrenamiento previo y el entorno complejo posterior permite ahorrar tiempo de desarrollo, y acelerar el proceso en futuros proyectos mediante reuso de modelos. El análisis demuestra que combinar un entrenamiento previo optimizado en un entorno más simple (Pong) con una fase breve pero estratégica en un entorno más complejo (Quadrabpong) permite entrenar múltiples agentes de forma más inteligente y sostenible. Esto respalda el uso del Transfer Learning / Parameter Transfer, como técnica que no solo mejora la eficiencia de aprendizaje, sino que también contribuye a la reducción de la huella ecológica.

• **Diversidad y derechos humanos**

Este proyecto no involucra datos sensibles ni interactúa con usuarios humanos, por lo que el impacto en términos de género, diversidad y derechos humanos es nulo. Tampoco se han identificado barreras de accesibilidad ni riesgos de discriminación o sesgo, lo que permite confirmar que se trata de un entorno inclusivo en su concepción técnica.

• **Ética y responsabilidad social**

Desde una perspectiva ética, el proyecto no genera decisiones sobre personas, ni procesa datos privados o información crítica. Su desarrollo se ha limitado al entrenamiento en un entorno artificial, lo que minimiza los riesgos relacionados con la privacidad, la seguridad o el uso indebido de la tecnología.

No obstante, se reconoce que, en caso de aplicar este tipo de inteligencia artificial en contextos sensibles (como vehículos autónomos, salud o finanzas), sería imprescindible realizar un análisis más profundo y riguroso de los aspectos éticos involucrados para asegurar un uso justo y seguro.

■ **Trabajo futuro**

Para continuar y ampliar los resultados obtenidos en este proyecto, se plantean varias líneas de trabajo que podrían mejorar la efectividad y la aplicabilidad de agentes preentrenados en entornos simples, a entornos multiagentes complejos.

En primer lugar, ampliar el número de pasos o rondas de entrenamiento podría permitir que los agentes alcancen un nivel de conocimiento más sólido y estable, especialmente en entornos con alta complejidad y dinámica cambiante como Quadrabpong. Un mayor tiempo de entrenamiento facilitaría la consolidación de estrategias más robustas y generalizables.

Además, se plantea la exploración de diferentes configuraciones de función de recompensa. Ajustar y diseñar recompensas alternativas podría favorecer comportamientos más óptimos, incentivando no solo la competitividad sino también la colaboración o la especialización dentro del equipo (atacante / defensor), según el contexto del entorno.

Otra línea interesante es investigar el uso de modelos o algoritmos diferentes para cada agente dentro del mismo entorno. Esta heterogeneidad permitiría estudiar cómo agentes con distintas capacidades y enfoques de aprendizaje interactúan y se adaptan en escenarios competitivos o cooperativos, aportando conocimiento sobre la diversidad de estrategias y su impacto en la dinámica grupal.

Finalmente, se sugiere extender la aplicación de estos métodos a entornos fuera del dominio de los videojuegos, pero que comparten características similares en cuanto a la estructura del espacio de acción y la interacción multiagente. Esto abriría la puerta a casos prácticos en robótica, logística, sistemas de tráfico y otros ámbitos donde el aprendizaje coordinado y competitivo entre agentes autónomos resulta clave.

Estas líneas futuras ofrecen un marco prometedor para profundizar en el aprendizaje por refuerzo multiagente, ampliando su alcance y eficacia en aplicaciones reales.

Glosario

IA Inteligencia Artificial

MARL Multi-Agent Reinforcement Learning

DQN Deep Q-Network

PPO Proximal Policy Optimization

MADDPG Multi-Agent Deep Deterministic Policy Gradient

GPU Graphics Processing Unit

ODS Objetivos de Desarrollo Sostenible

ML Machine Learning

RL Reinforcement Learning

MDP Márkov Decision Process

DRL Deep Reinforcement Learning

DDPG Deep Deterministic Policy Gradient

SARSA State-Action-Reward-State-Action

MC Monte Carlo methods

TD Temporal Difference methods

AC Actor-Critic

A2C Advantage Actor-Critic

A3C Asynchronous Advantage Actor-Critic

SAC Soft Actor-Critic

WoLF-PHC Win or Learn Fast Policy Hill Climbing

RGB Red, Green, Blue

CNN Convolutional Neural Network

RELU Rectified Linear Unit

MPS Metal Performance Shaders

CPU Central Processing Unit

SSD Solid State Drive

IID Independent and Identically Distributed

Bibliografía

- [1] PettingZoo - Atari Environment - Pong, 2024. URL <https://pettingzoo.farama.org/environments/atari/pong/>. Último acceso: 1 de marzo de 2025.
- [2] PettingZoo - Atari Environment - Quadrapong, 2024. URL <https://pettingzoo.farama.org/environments/atari/quadrapong/>. Último acceso: 1 de marzo de 2025.
- [3] Label Studio. Label studio: Data labeling platform, 2024. URL <https://labelstud.io>. Último acceso: 22 de mayo de 2025.
- [4] Tampuu A, Matiisen T, Kodelja D, Kuzovkin I, Korjus K, Aru J, et al. Multiagent cooperation and competition with deep reinforcement learning. *PLoS ONE*, 12(4):e0172395, 2017. URL <https://doi.org/10.1371/journal.pone.0172395>. Último acceso: 1 de marzo de 2025.
- [5] PettingZoo - An API standard for multi-agent reinforcement learning., 2024. URL <https://pettingzoo.farama.org/index.html#>. Último acceso: 1 de marzo de 2025.
- [6] Stable-Baselines3 Docs - Reliable Reinforcement Learning Implementations, 2019. URL <https://stable-baselines3.readthedocs.io/en/master/>. Último acceso: 1 de marzo de 2025.
- [7] Mark Towers, Ariel Kwiatkowski, Jordan Terry, John U Balis, Gianluca De Cola, Tristan Deleu, Manuel Goulão, Andreas Kallinteris, Markus Krimmel, Arjun KG, et al. Gymnasium: A standard interface for reinforcement learning environments. *arXiv preprint arXiv:2407.17032*, 2024. Último acceso: 29 de marzo de 2025.
- [8] J. K Terry, Benjamin Black, and Ananth Hari. Supersuit: Simple microwrappers for reinforcement learning environments. *arXiv preprint arXiv:2008.08932*, 2020. Último acceso: 29 de marzo de 2025.
- [9] Benoit Courty, Victor Schmidt, Sasha Luccioni, Goyal-Kamal, MarionCoutarel, Boris Feld, Jérémie Lecourt, LiamConnell, Amine Saboni, Inimaz, supatomic, Mathilde Lé-

- val, Luis Blanche, Alexis Cruveiller, ouminasara, Franklin Zhao, Aditya Joshi, Alexis Bogroff, Hugues de Lavoreille, Niko Laskaris, Edoardo Abati, Douglas Blank, Ziyao Wang, Armin Catovic, Marc Alencon, Michał Stęchły, Christian Bauer, Lucas Otávio N. de Araújo, JPW, and MinervaBooks. mlco2/codecarbon: v2.4.1, May 2024. URL <https://doi.org/10.5281/zenodo.11171501>.
- [10] Richard S. Sutton and Andrew G. Barto. *Reinforcement Learning: An Introduction*. MIT Press, Cambridge, MA, 2 edition, 2018. URL <http://incompleteideas.net/book/the-book-2nd.html>. Último acceso: 14 de marzo de 2025.
 - [11] Richard Bellman. *Dynamic Programming*. Princeton University Press, Princeton, NJ, 1957.
 - [12] Christopher J.C.H. Watkins. *Learning from Delayed Rewards*. Ph.d. thesis, University of Cambridge, Cambridge, England, 1989. URL <https://www.cs.rhul.ac.uk/~chrisw/thesis.html>.
 - [13] Hua Huang and Adrian Barbu. Playing atari ball games with hierarchical reinforcement learning, 2019. URL <https://arxiv.org/abs/1909.12465>. Último acceso: 16 de marzo de 2025.
 - [14] V. Mnih, K. Kavukcuoglu, D. Silver, and other. Human-level control through deep reinforcement learning. *Nature*, 518:529–533, 2015. URL <https://doi.org/10.1038/nature14236>. Último acceso: 29 de marzo de 2025.
 - [15] Volodymyr Mnih, Adria Puigdomenech Badia, Mehdi Mirza, Alex Graves, Timothy Lillicrap, Tim Harley, David Silver, and Koray Kavukcuoglu. Asynchronous methods for deep reinforcement learning. In Maria Florina Balcan and Kilian Q. Weinberger, editors, *Proceedings of The 33rd International Conference on Machine Learning*, volume 48 of *Proceedings of Machine Learning Research*, pages 1928–1937, New York, New York, USA, 20–22 Jun 2016. PMLR. URL <https://proceedings.mlr.press/v48/mnih16.html>. Último acceso: 29 de marzo de 2025.
 - [16] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017. URL <https://arxiv.org/abs/1707.06347>. Último acceso: 16 de marzo de 2025.
 - [17] Timothy P. Lillicrap, Jonathan J. Hunt, Alexander Pritzel, Nicolas Heess, Tom Erez, Yuval Tassa, David Silver, and Daan Wierstra. Continuous control with deep reinforcement learning, 2019. URL <https://arxiv.org/abs/1509.02971>. Último acceso: 16 de marzo de 2025.

- [18] Ken Ming Lee, Sriram Ganapathi Subramanian, and Mark Crowley. Investigation of independent reinforcement learning algorithms in multi-agent environments. *Frontiers in Artificial Intelligence*, 5, 2022. ISSN 2624-8212. doi: 10.3389/frai.2022.805823. URL <https://www.frontiersin.org/journals/artificial-intelligence/articles/10.3389/frai.2022.805823>. Último acceso: 16 de marzo de 2025.
- [19] Lucian Busoniu, Robert Babuska, and Bart De Schutter. A comprehensive survey of multiagent reinforcement learning. *IEEE Transactions on Systems, Man, and Cybernetics, Part C (Applications and Reviews)*, 38(2):156–172, 2008. URL <https://doi.org/10.1109/TSMCC.2007.913919>. Último acceso: 19 de marzo de 2025.
- [20] Emilio Parisotto, Lei Jimmy Ba, and Ruslan Salakhutdinov. Actor-mimic: Deep multitask and transfer reinforcement learning. In Yoshua Bengio and Yann LeCun, editors, *4th International Conference on Learning Representations, ICLR 2016, San Juan, Puerto Rico, May 2-4, 2016, Conference Track Proceedings*, 2016. URL <http://arxiv.org/abs/1511.06342>. Último acceso: 19 de marzo de 2025.
- [21] Akshita Mittel and Purna Sowmya Munukutla. Visual transfer between atari games using competitive reinforcement learning. In *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 499–501, 2019. URL <https://doi.org/10.1109/CVPRW.2019.00071>. Último acceso: 19 de marzo de 2025.
- [22] Marko Ruman and Tatiana V. Guy. Knowledge transfer in deep reinforcement learning via an rl-specific gan-based correspondence function. *IEEE Access*, 12:177204–177218, 2024. URL <https://doi.org/10.1109/ACCESS.2024.3497589>. Último acceso: 19 de marzo de 2025.
- [23] Apoorv Singh, Gaurav Raut, and Alka Choudhary. Multi-agent collaborative perception for robotic fleet: A systematic review, 2024. URL <https://doi.org/10.48550/arXiv.2405.15777>. Último acceso: 29 de marzo de 2025.
- [24] Paolo Falcone, Francesco Borrelli, Jahan Asgari, Hongtei Eric Tseng, and Davor Hrovat. Predictive active steering control for autonomous vehicle systems. *IEEE Transactions on Control Systems Technology*, 15(3):566–580, 2007. doi: 10.1109/TCST.2007.894653. URL <https://ieeexplore.ieee.org/abstract/document/4162483>. Último acceso: 29 de marzo de 2025.
- [25] Dai Tamagawa, Eiichi Taniguchi, and Tadashi Yamada. Evaluating city logistics measures using a multi-agent model. *Procedia - Social and Behavioral Sciences*, 2(3):6002–6012, 2010. ISSN 1877-0428. doi: <https://doi.org/10.1016/j.sbspro.2010.04.014>. URL <https://doi.org/10.1016/j.sbspro.2010.04.014>.

- //www.sciencedirect.com/science/article/pii/S1877042810010670. The Sixth International Conference on City Logistics.
- [26] Timothy Hospedales, Antreas Antoniou, Paul Micaelli, and Amos Storkey. Meta-learning in neural networks: A survey. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(9):5149–5169, 2022. doi: 10.1109/TPAMI.2021.3079209. URL <https://ieeexplore.ieee.org/abstract/document/9428530>. Último acceso: 29 de marzo de 2025.
- [27] Qian Long, Zihan Zhou, Abhibav Gupta, Fei Fang, Yi Wu, and Xiaolong Wang. Evolutionary population curriculum for scaling multi-agent reinforcement learning, 2020. URL <https://doi.org/10.48550/arXiv.2003.10423>. Último acceso: 29 de marzo de 2025.

Apéndice A

Detección de objetos con Label Studio y YOLOv8

Para detectar y rastrear los objetos relevantes en el entorno del juego Pong, como las palas y la pelota, se utilizó un enfoque basado en visión por computador apoyado en técnicas de aprendizaje profundo supervisado. Este proceso constó de dos fases principales:

- **Etiquetado manual de datos con Label Studio¹**
- **Entrenamiento de un modelo YOLOv8²**

A.1. Etiquetado manual de datos con Label Studio

El primer paso fue la recopilación y anotación de los datos visuales. Para ello, se extrajeron los fotogramas del entorno de juego (frames) y se utilizaron como imágenes base para la anotación. El etiquetado se realizó utilizando Label Studio, una herramienta de código abierto diseñada para facilitar la creación de datasets anotados.

Se definieron cinco clases de objetos que se utilizaron en los diferentes entornos Pong y Quadrapong:

- **ball (pelota)**
- **paddle_left (pala izquierda)**

¹Label Studio. <https://labelstud.io>

²Explore Ultralytics YOLOv8. <https://yolov8.com>

- **paddle_right** (pala derecha)
- **paddle_upper** (pala superior), en el caso de Quadrapong
- **paddle_lower** (pala inferior), en el caso de Quadrapong

Como se aprecia en la [Figura A.1](#) y la [Figura A.2](#), a través de la interfaz gráfica de Label Studio, se dibujaron manualmente las cajas delimitadoras sobre cada objeto en los distintos frames. Una vez completado el proceso de anotación, se exportaron los datos en formato YOLOv8, generando un conjunto de archivos .txt, uno por imagen, donde cada línea representa un objeto detectado mediante su clase y su caja normalizada (x_center, y_center, width, height).

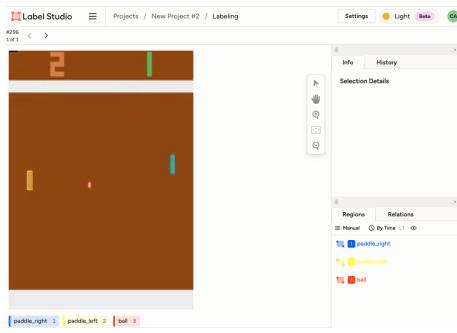


Figura A.1: Fotogramas del entorno de juego de Pong con las cajas delimitadoras[3]

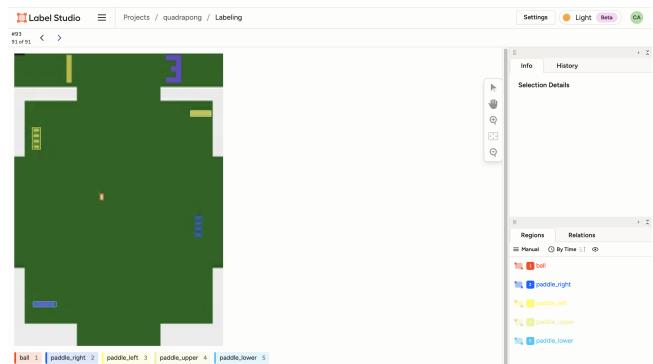


Figura A.2: Fotogramas del entorno de juego de Quadrapong con las cajas delimitadoras[3]

A.2. Entrenamiento de un modelo YOLOv8

Con los datos anotados, se procedió al entrenamiento de un modelo de detección utilizando la arquitectura YOLOv8 de Ultralytics³, conocida por su eficiencia en tareas de detección en tiempo real. Para ello, se organizó el dataset en carpetas de entrenamiento (train) y validación (val), tanto para las imágenes como para sus respectivas anotaciones.

El entrenamiento se realizó con el siguiente comando:

```
yolo task=detect mode=train model=yolov8n.pt data=dataset/data.yaml epochs=50 imgsz=416
Donde:
```

- **yolov8n.pt**

Es la versión ligera del modelo (YOLOv8n).

³Ultralytics. <https://www.ultralytics.com/es>

- **data.yaml**

Define la ruta del dataset y las clases.

- **epochs=50**

Especifica el número de épocas de entrenamiento.

- **imgsz=416**

Define la resolución de las imágenes de entrada.

En la [Figura A.3](#) y la [Figura A.4](#), se puede comprobar como tras el entrenamiento, el modelo aprendió a detectar las palas y la pelota con una precisión aceptable. Estos modelos se utilizaron posteriormente para detectar objetos frame a frame durante partidas reales de Pong y Quadrapong, lo que permitió la recolección automática de métricas como los impactos en las palas.



Figura A.3: Detección de palas y pelota en Pong



Figura A.4: Detección de palas y pelota en Quadrapong