Representación Relacional para la Transferencia de Conocimiento en Aprendizaje por Refuerzo

Por:

LMA. Armando Martínez Ruiz

Tesis sometida como requisito parcial
para obtener el grado de:

Maestría en Ciencias Computacionales

por el:

Instituto Nacional de Astrofísica, Óptica y Electrónica
Abril, 2024

Supervisada por:

Dr. Eduardo Morales Manzanares
Dr. Enrique Sucar Succar
Investigadores Titulares del INAOE
©Coordinación de Ciencias Computacionales

Agradecimientos

Quiero expresar mi profundo agradecimiento a mi esposa, Fabiola, cuyo apoyo incondicional ha sido fundamental en cada paso de este camino académico. Sin su constante respaldo, este logro no habría sido posible.

A mis padres y hermana, les agradezco de corazón por su incansable apoyo y por acompañarme en cada meta que me he propuesto alcanzar. Su amor y aliento han sido mi mayor motivación

Quiero reconocer y agradecer sinceramente al Dr. Eduardo Morales y al Dr. Enrique Sucar por su invaluable orientación, apoyo, motivación y ejemplo durante mi trayectoria académica. Sus enseñanzas han dejado una marca indeleble en mi formación profesional, y por ello les estaré eternamente agradecido.

A mi amiga Yareli, quien compartió conmigo cada etapa de este proceso académico. Su constante presencia y apoyo fueron fundamentales para superar las dificultades y seguir adelante. Su amistad ha sido un pilar en este viaje y ha significado mucho para mí.

Por último, no puedo dejar de expresar mi gratitud al Instituto Nacional de Astrofísica, Óptica y Electrónica, así como al Consejo Nacional de Humanidades, Ciencias y Tecnologías, por su generoso respaldo y apoyo durante el desarrollo de mis estudios de maestría.

Resumen

Esta tesis introduce un enfoque del aprendizaje por refuerzo que aprovecha el poder de la representación relacional para acelerar el proceso de aprendizaje y mejorar la transferencia de conocimientos a dominios relacionados. El concepto central gira en torno a la idea de que, abstrayendo ciertos atributos del entorno, es posible reducir significativamente la cantidad de datos y el tiempo necesarios para que el agente de aprendizaje alcance un rendimiento competente.

La tesis profundiza en los retos fundamentales del aprendizaje por refuerzo, que abarcan los considerables recursos de datos y tiempo necesarios para un aprendizaje eficaz y la limitada capacidad del agente para generalizar sus conocimientos a problemas nuevos y relacionados. Se introduce un enfoque novedoso que emplea la representación relacional, facilitada por un sistema de detección de objetos, para abordar estos retos permitiendo una interpretación más abstracta del entorno a través de las relaciones entre el agente y los objetos del ambiente. Para ilustrar la eficacia de este enfoque, se utiliza la biblioteca Arcade Learning Environment para modelar juegos Atari como ambientes de aprendizaje por refuerzo. Se definen relaciones que representan estados dentro de estos ambientes y se adapta el algoritmo Q-Learning para incorporar representaciones relacionales al proceso de toma de decisiones del agente. Los resultados obtenidos demuestran que este enfoque ofrece un rendimiento comparable al de los métodos existentes, como el algoritmo deep Q-Network. La tesis concluye discutiendo las aplicaciones potenciales de este enfoque más allá de los juegos Atari y destacando las posibilidades de integrar la representación relacional en la construcción de modelos causales para el aprendizaje por refuerzo, ofreciendo un ambiente propicio para investigaciones futuras y progresos en esta área.

Palabras clave: representación relacional, aprendizaje por refuerzo, transferencia de aprendizaje, detección de objetos, q-learning.

Abstract

This thesis introduces an approach to reinforcement learning that harnesses the power of relational representation to expedite the learning process and enhance knowledge transfer to related domains. The core concept revolves around the notion that by abstracting certain environmental attributes, it is possible to significantly reduce the amount of data and time required for the learning agent to achieve proficiency.

The thesis delves into the fundamental challenges of reinforcement learning, which encompass the substantial data and time resources required for effective learning and the agent's limited capacity to generalize knowledge to novel, related problems. It introduces a novel approach that employs relational representation, facilitated by an object detection system, to address these challenges by enabling a more abstract interpretation of the environment through relationships between the agent and environmental objects. To illustrate this approach's effectiveness, the Arcade Learning Environment library is used to model Atari games as reinforcement learning environments. Relationships representing states within these environments are defined, and the Q-Learning algorithm is adapted to incorporate relational representations into the agent's decision-making process. The results obtained demonstrate that this approach yields performance comparable to existing methods, like the deep Q-Network algorithm. The thesis concludes by discussing potential applications of this approach beyond Atari games and highlighting the possibilities of integrating relational representation into the construction of causal models for reinforcement learning, providing fertile ground for future research and advancement in the field.

Índice general

Índice de figuras

Índice de tablas

Capítulo 1

Introducción

Este estudio se centra en la aplicación de representaciones relacionales en el ámbito del aprendizaje por refuerzo, con la finalidad de enriquecer la capacidad de transferencia de conocimiento de los agentes en tareas relacionadas, especialmente en el contexto de los videojuegos. El aprendizaje por refuerzo se enfrenta a desafíos notables, entre ellos la necesidad de recopilar grandes volúmenes de datos y el problema de generalizar conocimiento hacia problemas nuevos pero relacionados. En esta tesis, se profundiza en la forma en que las representaciones relacionales pueden abstraer características del entorno y capturar las complejas interacciones entre los objetos presentes en el mismo. Además, se explora la capacidad de transferencia de conocimiento a una tarea nueva utilizando experiencia previa en una tarea similar.

Una faceta destacada de esta investigación implica la propuesta de un enfoque que permita a los agentes aprender y transferir conocimiento de manera efectiva en tareas relacionadas, particularmente en ambientes basados en videojuegos. Al enfocarse en la mejora de la capacidad de los agentes para adaptarse a nuevos desafíos y aplicar el conocimiento previamente adquirido en situaciones similares, este estudio busca avanzar en el campo en constante evolución del aprendizaje por refuerzo. Además, este estudio tiene como propósito fundamental impulsar el avance de la inteligencia artificial y el aprendizaje computacional en el ámbito de los videojuegos, así como en otras áreas que requieran adaptabilidad y transferencia de conocimiento.

1.1 — Motivación

El aprendizaje computacional es una de las principales áreas de la inteligencia artificial que busca mejorar el rendimiento de los programas de forma automática a través de la experiencia. En particular, el aprendizaje computacional aborda y modela los procesos de aprendizaje en sus diversas manifestaciones. El aprendizaje por refuerzo es un paradigma dentro del aprendizaje computacional que se enfoca en cómo un agente puede maximizar una recompensa al interactuar con su entorno. En cada iteración, el agente recibe información sobre su estado actual y selecciona una acción. Esta acción puede modificar el estado, y el agente recibe una señal de recompensa. El objetivo es encontrar una política, es decir, una función que asocie estados con acciones, que maximice la recompensa total esperada.

Uno de los retos más importantes dentro del aprendizaje por refuerzo es aprender directamente de datos de dimensiones altas, como imágenes o videos. Los avances recientes en aprendizaje profundo han mostrado que los datos crudos pueden ser usados como entrada para el aprendizaje. Uno de los avances más importantes fue el algoritmo DQN (Deep Q-Network) [14], el cual pudo combinar de manera satisfactoria el algoritmo Q-Learning con las redes neuronales convolucionales. Esta arquitectura fue usada principalmente para jugar juegos de Atari.

En el desarrollo de esta tesis, se ha elegido utilizar el algoritmo Q-Learning. Aunque existen otros métodos para abordar problemas similares, como los métodos de Monte Carlo, que emplean trayectorias completas o episodios para estimar valores. En el contexto de los videojuegos, estos métodos implicarían que el agente tendría que aprender de juegos completos en lugar de movimientos individuales. Esto conlleva a una convergencia más lenta hacia una solución óptima.

Por otro lado, el algoritmo SARSA es un enfoque *on-policy*, lo que significa que el aprendizaje del agente se ve influenciado por una política predefinida. Dado que la naturaleza estocástica de los videojuegos puede presentar dificultades al elegir el algoritmo SARSA, se considera menos adecuado en este contexto.

Por lo tanto, la elección del algoritmo Q-Learning se fundamenta en la capacidad del algoritmo para modelar los videojuegos como procesos de decisión de Markov, donde el agente aprende en cada acción que realiza. Además, destaca su habilidad para

adaptarse a la estocasticidad inherente de los videojuegos, permitiendo así la identificación de políticas óptimas.

De acuerdo con Sutton y Barto [17], de todas las formas de aprendizaje computacional, el aprendizaje por refuerzo es el tipo de aprendizaje más cercano al que los humanos y animales realizan, y la mayoría de los algoritmos importantes en aprendizaje por refuerzo están inspirados originalmente en sistemas biológicos de aprendizaje.

1.2 — Problema

Las interacciones de un agente y el ambiente en el que se encuentra requieren una cantidad considerable de datos y de tiempo para que el agente pueda aprender, de modo que el proceso de aprendizaje en este enfoque es lento, en particular, en el aprendizaje por refuerzo profundo. Además, el aprendizaje por refuerzo sufre de la poca capacidad que tiene de generalizar el conocimento aprendido a problemas nuevos pero relacionados.

Con base en lo anterior, el uso de la representación relacional permitir abstraer ciertas características del ambiente, de tal forma que se pueda acelerar el aprendizaje del agente, además de transferir este conocimiento a tareas similares.

1.3 — Preguntas de investigación

1. ¿Es posible que un agente aprenda en entornos de juegos dinámicos tipo Atari mediante la combinación de representación relacional y aprendizaje por refuerzo?

2. ¿Cómo se puede incorporar la representación relacional en el proceso de aprendizaje de un agente?

3. ¿La transferencia de conocimientos a tareas similares resulta efectiva al emplear modelos relacionales específicos de una tarea en particular?

1.4 — Hipótesis

La incorporación de la representación relacional en el aprendizaje por refuerzo permitirán al agente aprender una tarea de manera más rápida y transferir el conocimiento adquirido a una tarea nueva que es similar a la anterior.

1.5 — Objetivos

1.5.1. Objetivo general

Incorporar el uso de representación relacional dentro del aprendizaje por refuerzo profundo y su aplicación en juegos de Atari, para la transferencia de conocimiento.

1.5.2. Objetivos específicos

1. Implementar un sistema de detección de objetos que permita al agente identificar con precisión los elementos presentes en su entorno.
2. Establecer un conjunto de relaciones que posibilite la abstracción de características relevantes del ambiente, facilitando así el aprendizaje específico del agente para tareas particulares.
3. Implementar de manera eficiente el sistema de detección de objetos y el conjunto de relaciones en un algoritmo de aprendizaje, con el objetivo de evaluar la capacidad del agente para utilizar dichos elementos en la adquisición de habilidades para una tarea específica.
4. Transferir el conocimiento obtenido a partir de estas relaciones utilizadas en una tarea a otras similares, y demostrar que dicho conocmiento le es útil al agente.

1.6 — Alcances y Limitaciones

Los alcances de la presente tesis son los siguientes:

- Los ambientes elegidos para los experimentos son videojuegos de consola ATARI, en particular, se eligieron juegos de la categoría "*shoot'em up*", debido a la facilidad de generalizar el ambiente con relaciones en este tipo de juegos.
- El enfoque usado para la transferencia de conocimiento es la transferencia de parámetros estimados de una tarea a otra. En este caso se decidió utilizar los valores estimados de la función de valor Q.

Las limitaciones de la presente tesis son las siguientes:

- El enfoque propuesto es comparable solo con algoritmos y métodos de aprendizaje por refuerzo que involucre juegos de Atari.
- Se consideran únicamente las relaciones entre el agente con los objetos y las superficies del ambiente.
- La transferencia del conocimiento obtenido de un juego será aplicada en juegos de la misma consola (ATARI) y del mismo género (*shoot'em up*).

1.7 — Contribuciones

Los resultados obtenidos esperados a partir de este proyecto de tesis que permitirán contribuir al área de estudio son los siguientes:

- Proponer un enfoque basado en aprendizaje por refuerzo relacional en ambientes de juegos de Atari.
- La variante de un algoritmo de aprendizaje por refuerzo utilizando representación relacional.
- Presentar un método de aprendizaje por transferencia utilizando parámetros estimados previamente.

1.8 — Estructura del documento

El presente trabajo se estructura de la siguiente manera: El Capitulo 1 se presenta el propósito principal de la tesis, que es investigar el uso de representaciones relacionales en el aprendizaje por refuerzo, con el objetivo de mejorar la capacidad de transferencia de conocimiento del agente en tareas relacionadas en el contexto de los juegos de video. El Capitulo 2 proporciona una base teórica sólida para el desarrollo de la tesis y ayuda a contextualizar los conceptos y técnicas utilizados en el trabajo. En el Capitulo 3 se revisa la literatura existente sobre el aprendizaje por refuerzo y discute los avances notables que se han logrado en este campo en los últimos años. El Capitulo 4 describe la metodología propuesta para el trabajo y los pasos que se seguirán para alcanzar los objetivos de la tesis. Finalmente, en el Capitulo 5 presenta los hallazgos y conclusiones de los experimentos realizados para evaluar la efectividad de la representación relacional y la transferencia de conocimiento entre dos tareas similares, y el Capitulo 6 proporciona una visión general de los hallazgos de la investigación y sugiere posibles direcciones para futuras investigaciones en el campo de la representación relacional y el aprendizaje por refuerzo.

Capítulo 2

Marco Teórico

Esta sección describe los conceptos teóricos involucrados en el desarrollo de esta tesis.

2.1 — Procesos de decisión de Markov

El aprendizaje por refuerzo consiste en aprender cómo hacer un mapeo de situaciones a acciones, de tal manera que una señal de recompensa, representada con un número real, sea maximizada. Se formaliza el problema de aprendizaje por refuerzo utilizando conceptos de la teoría de sístemas dinámicos, en particular, los procesos de decisión de Markov.

Un proceso de decisión de Markov (o MDP) es un problema de optimización de control, esto es, busca la mejor acción en una situación determinada, en donde un agente aprende a través de prueba y error a tomar decisiones dentro de un ambiente. La solución de un proceso de decisión de Markov consiste en una secuencia de acciones que maximizan la recompensa esperada, llamada política.

De manera formal [17], tomando en cuenta una secuencia de pasos en el tiempo $t = 0, 1, 2, ...$, el agente se encuentra en una representación del ambiente definida por el estado $S_t \in \mathcal{S}$ donde $\mathcal{S}$ representa al conjunto de estados, y selecciona una acción A_t del conjunto de acciones $\mathcal{A}$. Como consecuencia, el agente recibe una recompensa $R_t \in \mathcal{R} \subset \mathbb{R}$ y cambia al estado a S_{t+1}. En un MDP finito, los conjuntos $\mathcal{S}, \mathcal{A}$ y $\mathcal{R}$ tienen un número finito de elementos, de modo que R_t y S_t son variables aleatorias cuyas distribuciones de probabilidad están bien definidas [7]. Entonces, la función p

definida como:

$$p(s', r|s, a) = P(S_t = s', R_t = r|S_{t-1} = s, A_{t-1} = a), \tag{2.1}$$

para cualesquiera $s, s' \in \mathcal{S}, r \in \mathcal{R}$ y $a \in \mathcal{A}$, representa la probabilidad de transición de pasar al estado s' con la recompensa r dado el estado s y la acción a. En general, la función $p : \mathcal{S} \times \mathcal{R} \times \mathcal{S} \times \mathcal{A} \longrightarrow [0, 1]$ describe la dinámica del ambiente. A partir de este concepto, se define la función de transición de estados como:

$$\phi(s'|s, a) = P(S_t = s'|S_{t-1} = s, A_{t-1} = a) = \sum_{r \in \mathcal{R}} p(s', r|s, a). \tag{2.2}$$

De la misma manera, las recompensas esperadas se pueden representar con una función $r : \mathcal{S} \times \mathcal{A} \longrightarrow \mathbb{R}$ como:

$$r(s, a) = \mathbb{E}[R_t|S_{t-1} = s, A_{t-1} = a] = \sum_{r \in \mathcal{R}} \sum_{s' \in \mathcal{S}} p(s', r|s, a). \tag{2.3}$$

Dado lo anterior, se puede caracterizar un MDP como una tupla $(\mathcal{S}, \mathcal{A}, \phi, r)$, donde $\mathcal{S}$ y $\mathcal{A}$ son los conjuntos de estados y acciones respectivamente, ϕ es la función de transición de estados y $r(s, a)$ es la función de recompensa. Se puede observar una representación gráfica de un proceso de decisión de Markov en la Figura 2.1.

Los procesos de decisión de Markov cumplen la propiedad de Markov: el estado futuro del sistema depende solo del estado actual, es decir, la distribución de probabilidad de los estados futuros es independiente de los estados anteriores. Esta propiedad se puede expresar con las siguientes probabilidades:

$$\begin{aligned} P(S_{t+1} = s'|S_t = s_t, A_t = a_t, S_{t-1} = s_{t-1}, A_{t-1} = a_{t-1}, ...) = \\ P(S_{t+1} = s'|S = s_t, A_t = a_t), \forall s \in S, a \in A, t = 0, 1, 2, 3, ... \end{aligned} \tag{2.4}$$

También cumple la propiedad estacionaria, la cual expresa que los parámetros del modelo permanecen iguales durante el paso del tiempo t:

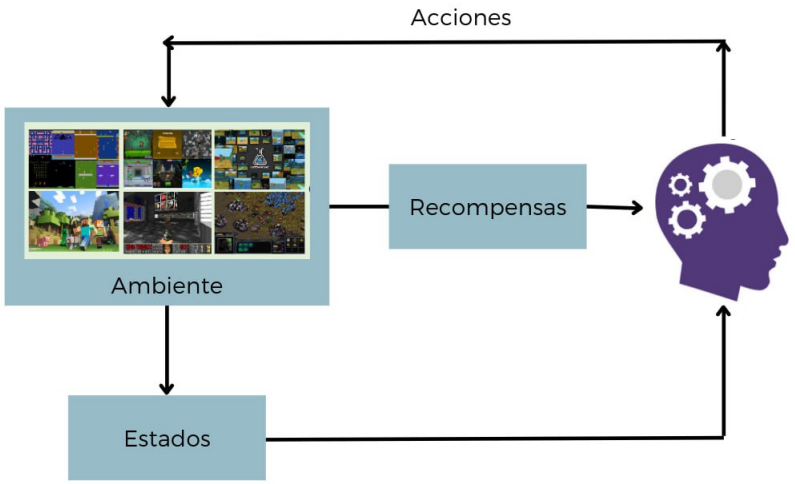

Figura 2.1. La interacción agente-ambiente en un proceso de decisión de Markov. El agente interactúa con el ambiente a través de acciones, de modo que el ambiente devuelve una señal de recompensa y el siguiente estado al agente en un proceso iterativo.

$$P(S_{t+1} = s'|S_t = s_t, A_t = a_t) = P(S_{t+2} = s''|S_{t+1} = s', A_{t+1} = a_{t+1}) \tag{2.5}$$

$$P(R_t = r|S_t = s_t, A_t = a_t) = P(R_{t+1} = r'|S_{t+1} = s', A_{t+1} = a'). \tag{2.6}$$

Dependiendo del número de pasos que se necesiten para llegar a un estado terminal, los procesos de decisión de Markov se pueden dividir en problemas de horizonte finito y de horizonte infinito. Los problemas de horizonte finito están definidos para un número determinado de pasos en el tiempo t, es decir, el tiempo esta acotado por algún número natural n, mientras que los problemas de horizonte finito no tienen un número fijo de pasos t.

Como se ha mencionado, el objetivo del agente es maximizar la recompensa que obtiene al interactuar con el ambiente. Para esto se busca maximizar la recompensa acumulada, denotada por g_t, donde $R_t \in \mathbb{R}$ es una recompensa al tiempo t, y se define de la siguiente forma:

$$g_t = R_{t+1} + R_{t+2} + ... + R_T = \sum_{i=t}^{T} R_i, \tag{2.7}$$

donde T es el último paso del tiempo. De esta manera, este enfoque es útil puesto que el número de pasos en el tiempo es finito, de modo que se puede dividir la interacción del agente con el ambiente en episodios. Cada episodio tiene un estado terminal, el cual determina el momento en el que un episodio termina y otro puede comenzar.

En los casos donde la tarea es continua, es decir, el paso final es $T = \infty$, se añade el concepto de descuento, de tal manera que el agente intente seleccionar acciones para que la suma de recompensas descontadas que recibirá en el futuro sea maximizada. Se define la recompensa esperada con descuento de la siguiente forma:

$$G_t = R_{t+1} + \gamma R_{t+2} + \gamma^2 R_{t+3} ... + = \sum_{i=0}^{\infty} \gamma^i R_{t+i+1}, \tag{2.8}$$

donde $\gamma \in [0, 1]$ es llamada factor de descuento.

Si los valores de γ se acercan a 1, entonces el agente toma con mayor importancia las recompensas futuras. Si los valores de γ se acercan a 0, entonces el agente es "miope", es decir, solo toma en cuenta la información inmediata al tomar una decisión, sin considerar los efectos a largo plazo.

Una política π es una función $\pi : S \longrightarrow A$ que determina qué acción $a \in A$ debe realizar un agente dado un estado $s \in S$. Entonces, la función de valor de un estado s bajo una politica π , es la recompensa esperada acumulada de empezar en el estado s y seguir la política π. Esta función de valor se define de la siguiente manera:

$$v_\pi(s) = \mathbb{E}_\pi[G_t | S_t = s] = \mathbb{E}_\pi[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1} | S_t = s], \forall s \in S, \tag{2.9}$$

donde $\mathbb{E}_\pi[.]$ denota al valor esperado de una variable aleatoria dado que el agente sigue una política π. De manera similar, se define una función donde se involucra una acción a además del estado s. En este caso se define q_π como la función acción-valor de la siguiente forma:

$$q_\pi(s,a) = \mathbb{E}_\pi[G_t|S_t = s, A_t = a] = \mathbb{E}_\pi[\sum_{k=0}^{\infty} \gamma^k R_{t+k+1}|S_t = s, A_t = a]. \quad (2.10)$$

Para el caso de horizonte infinito, encontrar la política π^* que maximiza la recompensa esperada satisface la siguiente ecuación, llamada ecuación de Bellman:

$$v_\pi(s) = \mathbb{E}_\pi[G_t|S_t = s] = \sum_a \pi(a|s) \sum_{s',r} p(s', r|s, a)[r + \gamma v_\pi(s')], \forall s \in \mathcal{S}. \quad (2.11)$$

Esta ecuación representa la relación entre el valor de un estado y los valores de los estados sucesores. Entonces, la política que maximiza la ecuación de Bellman es la política óptima π^*:

$$v^*(s) = \max_a \sum_{s',r} p(s', r|s, a)[r6\gamma v^*(s'). \quad (2.12)$$

Ahora, dada la función óptima de acción-valor $q^* = \max_\pi q_\pi(s,a)$, podemos escribir la ecuación de Bellman óptima para q^* como:

$$\begin{aligned} q^*(s,a) &= \mathbb{E}[R_{t+1} + \gamma \max_{a'} q^*(S_{t+1}, a')|S_t = s, A_t = a] \\ &= \sum_{s',r} p(s', r|s, a)[r + \gamma \max_{a'} q^*(s', a'). \end{aligned} \quad (2.13)$$

La política π^* representa la solución de un proceso de decisión de Markov, por lo que existen diferentes métodos para encontrar esta política óptima, como son los métodos de Monte Carlo o los métodos de programación dinámica, como son el algoritmo de iteración de valor o el algoritmo de iteración de política. En este trabajo se utilizó el algoritmo Q-Learning, que es parte del aprendizaje por diferencias temporales.

2.1.1. Aprendizaje por diferencias temporales

En los métodos de aprendizaje por diferencias temporales, el agente aprende directamente de su experiencia del ambiente, sin tener un modelo que lo represente [17]. A este tipo de métodos se les conoce como *model free*.

Los métodos de diferencias temporales actualizan el valor estimado de v_π para

algun estado S_t, denotado por $V(S_t)$, observando la recompensa R_{t+1} y el valor estimado $V(S_{t+1})$. La forma más simple de los métodos de diferencias temporales hacen la siguiente actualización:

$$V(S_t) \longleftarrow V(S_t) + \alpha[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)], \tag{2.14}$$

donde γ es un factor de descuento y α es la razón de aprendizaje.

Q-Learning

Uno de los avances más importantes dentro del aprendizaje por refuerzo es el desarrollo del algoritmo Q-Learning [17], que es un algoritmo de diferencias temporales que no se basa en una política previa para encontrar la función acción-valor óptima. Este algoritmo define la siguiente regla de actualización:

$$Q(S_t, A_t) \longleftarrow Q(S_t, A_t) + \alpha[R_{t+1} + \gamma \max_a Q(S_{t+1}, a) - Q(S_t, A_t)], \tag{2.15}$$

donde Q es una función de valor que estima a q^*. El Algoritmo 1 describe el proceso de Q-Learning.

Algoritmo 1 : Q-Learning

Parámetros del algoritmo: $\alpha, \gamma \in [0, 1]$, $\epsilon > 0$ pequeño.
Inicializar $Q(s, a), \forall s \in S, a \in A$ de forma arbitraria.

1: Bucle para cada episodio:
2: Inicializar s
3: Bucle para cada paso en el episodio:
4: Escoger a' de s utilizando la politica derivada de Q (ϵ greedy)
5: Tomar acción a, observar R, s'
6: $Q(s, a') \longleftarrow Q(s, a') + \alpha[R + \gamma \max_a Q(s', a) - Q(s, a')]$
7: $s \longleftarrow s'$
8: Hasta que s sea un estado terminal

El algoritmo Q-Learning es aplicable en representaciones relacionales, ya que permite representar la función $Q(s, a)$ como una tabla que asigna cada par estado-acción a su recompensa acumulada esperada. Estos pares estado-acción pueden repre-

sentarse como entidades y sus relaciones. La actualización de la función Q se realiza mediante el algoritmo tradicional de Q-Learning, utilizando la recompensa observada y una estimación de $Q(s', a)$, donde s' es el estado sucesor de s. Una ventaja de utilizar representaciones relacionales es que permiten modelar relaciones complejas entre el agente y otras entidades en el entorno, lo que, a su vez, permite al agente razonar de manera más sofisticada sobre su entorno y tomar decisiones más informadas sobre sus acciones.

Este algoritmo desempeña un papel fundamental en el ámbito del aprendizaje por refuerzo profundo, ya que aprovecha redes neuronales profundas para aproximar y aprender directamente a partir de datos crudos, como imágenes o señales sensoriales. Esto habilita al agente para adquirir habilidades en tareas complejas y de alto nivel, como jugar videojuegos, controlar robots o tomar decisiones en entornos virtuales. Se menciona esta disciplina del aprendizaje computacional mas a fondo en el Capítulo 3.

2.2 — Representación Relacional

La representación relacional es una manera de representar un estado en el ambiente o el conocimento sobre éste en terminos de entidades y sus relaciones. Esta representación permite modelar de forma más expresiva los dominios de tal manera que se puedan generalizar algunos aspectos del ambiente. La estructura de la representación relacional se puede presentar como un conjunto de enunciados de lógica de predicados de primer orden, que describen las relaciones entre las entidades.

2.2.1. Lógica de predicados de primer orden

La lógica de primer orden [16] es un lenguaje formal que se utiliza para representar los aspectos del mundo en términos de objetos, sus atributos y las relaciones que existen entre ellos. Proporciona un marco de trabajo para representar dominios complejos que incluyen incertidumbre y cambio.

Las expresiones o fórmulas en la lógica de predicados de primer orden se construyen utilizando cuatro tipos de símbolos:

1. Constantes, son símbolos que representan objetos en el dominio de interés. Se denota con letras mayusculas (A, B, N, M, Z).

2. Predicados, son expresiones que pueden ser falsas o verdaderas, y que pueden tener un número determinado de argumentos. Se representan como notación de función con letras mayusculas ($P(x), Q(x), R(x, y, z)$).

3. Variables, son símbolos que representan objetos en el dominio de interés, pero que pueden ser argumentos de un predicado o función. Éstas se representan con letras minusculas (a, b, n, m, z).

Además, la lógica de predicados de primer orden utiliza los conectores tradicionales en la lógica proposicional: negacion ($\neg$), conjunción ($\wedge$), disyunción ($\vee$), implicación ($\Rightarrow$) y equivalencia ($\Leftrightarrow$), y se extiende la teoría utilizando tambíen los cuantificadores universales y de existencia:

- Cuantificador universal: $\forall x$ (para cualquier x).
- Cuantificador existencial: $\exists x$ (existe un x).

A través de estos símbolos y reglas, se puede construir y evaluar enunciados en lógica de predicados de primer orden que expresen argumentos sobre el ambiente y permite el razonamiento sobre ellos. En aprendizaje por refuerzo, es posible representar a los estados, las acciones, las recompensas y otros aspectos del ambiente en una forma relacional.

2.3 — Aprendizaje por Transferencia

El aprendizaje por transferencia o *transfer learning* (TL) es un paradigma del aprendizaje computacional que estudia cómo aplicar conocimiento existente, modelos y parámetros a nuevos problemas o tareas.

Un dominio consiste en un conjunto de datos y la distribución que genera estos datos [20]. Se denota con $\mathcal{D}$ y su distribución de probabilidad con $P(x, y)$, donde x es la entrada y y es la salida de la muestra de $\mathcal{D}$. Si X denota el conjunto de atributos y

Y el conjunto de etiquetas, de modo que $x_y \in X$ y $y_i \in Y$ para cualquier muestra, entonces un dominio se puede definir como $\mathcal{D} = \{X, Y, P(x, y)\}$.

De manera formal, podemos definir el aprendizaje por transferencia de la siguiente manera: dado un dominio fuente $\mathcal{D}_s$ con una muestra $\{x_i, y_i \overset{N_s}{\underset{i=1}{\}}}$ y un dominio de destino $\mathcal{D}_t$ con muestra $\{x_j, y_j \overset{N_t}{\underset{j=1}{\}}}$, donde $x \in X$ y $y \in Y$, el objetivo del aprendizaje por transferencia es utilizar la información del dominio fuente para aprender una función $f : X_t \longrightarrow Y_t$ tal que f puede alcanzar un riesgo mínimo de predicción en el dominio de destino (evaluado por la función de pérdida l):

$$f^* = \arg\min_f \mathop{\mathbb{E}}_{(x,y)\in\mathcal{D}_t} l(f(x), y), \tag{2.16}$$

cuando una de las siguientes condiciones se cumple:

1. Tienen diferentes conjuntos de atributos ($X_s \neq X_t$).
2. Tienen diferentes conjuntos de etiquetas ($Y_s \neq Y_t$).
3. Tienen diferentes distribuciones de probabilidad pero mismos conjuntos de atributos y etiquetas ($P_s(x, y) \neq P_t(x, y)$).

2.3.1. Aprendizaje por Transferencia en Aprendizaje por Refuerzo

Se define una tarea M como un proceso de decisión de Markov caracterizado por la tupla (S_M, A_M, T_M, R_M) donde S_M es el conjunto de estados, A_M es el conjunto de acciones, T_M es la función de transición y R_M es la función de recompensa.

El dominio de la tarea se define mediante el conjunto estado-acción $S_M \times A_M$, y el objetivo de la tarea se define a través de la función de transición T_M y la función de recompensa R_M. El espacio de tareas involucrado en el problema de aprendizaje por transferencia se denota por $\mathcal{M} = \{M\}$.

Sea Ω una distribución de probabilidad sobre el espacio de tareas $\mathcal{M}$, entonces denotamos por $\mathcal{E} = (\mathcal{M}, \Omega)$ al ambiente, que define el marco de trabajo del problema de transferencia de conocimiento.

En el contexto de un algoritmo de aprendizaje convencional, se parte de algún tipo de conocimiento relacionado con la tarea y, como resultado, se obtiene una solución de entre un conjunto de posibles soluciones. Si se representa el conjunto de conocimientos como $\mathcal{K}$ y el conjunto de soluciones o hipótesis como $\mathcal{H}$, se puede expresar el algoritmo de aprendizaje mediante la siguiente función:

$$\mathcal{A}_l : \mathcal{K} \longrightarrow \mathcal{H}. \tag{2.17}$$

En el ámbito del aprendizaje por transferencia, el objetivo es reducir el número de instancias requeridas para abordar una tarea específica y minimizar la necesidad de conocimiento previo de un experto en el dominio [8]. Esto se logra configurando y adaptando la estructura de un algoritmo de aprendizaje utilizando tareas previas observadas como base. Ahora, sea L el número de tareas extraídas de un conjunto $\mathcal{M}$, según una distribución Ω utilizada para definir tareas fuente. Un algoritmo de aprendizaje por transferencia se produce a través de la transferencia de conocimiento y una fase de aprendizaje. Si representamos el conocimiento adquirido al realizar las L tareas fuente como $\mathcal{K}_s^L$ y el conocimiento de la tarea de destino (si existe) como $\mathcal{K}_t$, la fase de transferencia se puede expresar mediante la siguiente función:

$$\mathcal{A}_{transfer} : \mathcal{K}_s^L \times \mathcal{K}_t \longrightarrow \mathcal{K}_{transfer}, \tag{2.18}$$

donde $\mathcal{K}_{transfer}$ es el conocimiento final transferido en la fase de aprendizaje. Por lo tanto, el algoritmo de aprendizaje puede definirse como:

$$\mathcal{A}_l' : \mathcal{K}_{transfer} \times \mathcal{K}_t \longrightarrow \mathcal{H}. \tag{2.19}$$

2.3.2. Tipos de transferencia de conocimiento

Para el estudio de la transferencia de aprendizaje en aprendizaje por refuerzo, se puede dividir la posible transferencia de conocimiento en tres categorías [20]:

1. **Transferencia de instancias:** En el aprendizaje por refuerzo, a diferencia de la programación dinámica, se depende de muestras en lugar de conocimiento

previo. Estas muestras se utilizan para estimar el modelo del proceso de decisión de Markov (MDP) o crear aproximaciones de funciones de valor y políticas. La transferencia de muestras de tareas anteriores simplifica el aprendizaje en nuevas tareas.

2. **Transferencia de representación:** En este caso, los algoritmos de RL emplean representaciones específicas para tareas y soluciones, como agregación de estados, redes neuronales o funciones base. Tras aprender en múltiples tareas, se aplican procesos de abstracción para cambiar estas representaciones. En esta categoría, se utilizan diversas estrategias, como la manipulación de recompensas, la expansión del MDP mediante opciones o la extracción de funciones base.

3. **Transferencia de parámetros:** La mayoría de los algoritmos de RL se caracterizan por tener una serie de parámetros que definen la inicialización y el comportamiento del propio algoritmo. La transferencia de soluciones iniciales (como políticas o o funciones de valor) se utilizan comunmente para inicializar el algoritmo de aprendizaje en el contexto de transferencia con una sola tarea fuente.

2.3.3. Métricas de desempeño

En el aprendizaje por refuerzo, se utilizan diversas métricas para evaluar el rendimiento de un agente en una tarea específica. Como resultado, también se han desarrollado varias métricas de rendimiento para evaluar los algoritmos de transferencia. A continuación, se presentan algunas de estas métricas:

1. **Mejora en rapidez de aprendizaje:** Consiste en reducir la cantidad de tiempo y experiencia que se necesita para aprender la tarea en cuestión. Se pueden utilizar medidas como tiempo para alcanzar un umbral o la proporción de área, que se define de la siguiente manera:

$$r = \frac{A_t - A_n}{A_n}, \tag{2.20}$$

donde A_t es el área bajo la curva de aprendizaje con conocimiento previo y A_n es el área bajo la curva de aprendizaje sin conocimiento previo.

2. **Mejora asintótica:** Una medida empírica de la calidad de $\mathcal{H}$ es comparar el rendimiento asintótico del aprendizaje por transferencia y del aprendizaje de una sola tarea utilizando una muestra grande disponible.

3. **Mejora inmediata:** Tambíen llamada *jumpstart improvement*, se refiere a mejorar el proceso inicial de aprendizaje en transferencia de conocimiento. En este contexto, después de observar tareas fuente, el algoritmo de transferencia puede proporcionar una hipótesis de inicio más efectiva para el agente en la tarea objetivo. Sin embargo, esta mejora inicial no siempre implica una aceleración en la velocidad de aprendizaje.

   Si se utiliza alguna medida M para conocer el rendimiento de la tarea fuente y la tarea objetivo, una manera de calcular la mejora inmediata o el *jumpstart* es encontrando la diferencia entre los valores iniciales de cada tarea, es decir, $M_0^s - M_0^t$, donde M_0^s es el valor inicial de la medida en la tarea fuente y M_0^t es el valor inicial de la medida en la tarea de destino.

   Es posible que la política óptima de la tarea fuente pueda ser muy diferente de la tarea objetivo, lo que podría ralentizar el proceso de aprendizaje. No obstante, la transferencia de políticas o soluciones más efectivas en lugar de una inicialización aleatoria se busca comúnmente en algoritmos de transferencia.

2.4 — Resumen

El aprendizaje por refuerzo está intrínsecamente relacionado con los procesos estocásticos y los problemas de optimización de control, y, por lo tanto, se sustenta en una teoría amplia. Sus aplicaciones se extienden desde la teoría de colas hasta la robótica. En este sentido, la representación relacional se integra de manera natural al realizar abstracciones del entorno, lo que permite al agente reconocer su estado actual y tomar decisiones óptimas. Además, esta representación resulta útil para abstraer ambientes similares.

En este contexto, el aprendizaje por transferencia se convierte en una elección lógica, ya que tanto las políticas aprendidas por el agente como las abstracciones del entorno pueden ser empleadas en tareas análogas.

Capítulo 3

Estado del Arte

El campo del aprendizaje por refuerzo ha experimentado un significativo avance en los últimos años, con la introducción de enfoques innovadores que han mejorado notablemente el rendimiento de los agentes. Los trabajos destacados incluyen el algoritmo DQN de Mnih et al., que alcanzó un rendimiento a nivel humano en juegos de Atari, así como los enfoques de Lillicrap et al. y Ammanabrolu et al., que ampliaron las capacidades de DQN para abordar espacios de acciones continuos y utilizar grafos de conocimiento para una mejor transferencia de conocimiento. Tessler et al. presentó un enfoque jerárquico en *Minecraft* con posibles aplicaciones más allá de los videojuegos, y Garnelo et al. abordó el razonamiento simbólico en el aprendizaje por refuerzo, demostrando cómo combinarlo con redes neuronales profundas puede superar enfoques puramente profundos o simbólicos.

Estos trabajos destacan la versatilidad del aprendizaje por refuerzo y su potencial para aplicaciones en una amplia variedad de campos.

3.1 — Aprendizaje por Refuerzo Profundo

El trabajo fundamental de Mnih et al. (2015) titulado *"Human level control through deep reinforcement learning"* [14] ha realizado contribuciones significativas al campo del aprendizaje por refuerzo profundo. Los autores presentan el algoritmo Deep Q-Network (DQN), que constituye una poderosa herramienta para el aprendizaje por refuerzo profundo y que ha demostrado la capacidad de aprender a jugar juegos de Atari al mismo nivel o incluso mejor que un ser humano. DQN es una red neuronal profunda que realiza predicciones de la recompensa esperada para una acción dada en un esta-

do específico. Este algoritmo se vale de una estrategia denominada *experience replay*, la cual implica almacenar las experiencias del agente en una memoria de repetición y tomar muestras aleatorias de esta memoria durante el proceso de entrenamiento.

Un trabajo relevante relacionado con el algoritmo DQN es el estudio de Lillicrap et al. (2019) [10], en el cual se aborda la adaptación del algoritmo DQN para lidiar con espacios de acciones continuos. Este enfoque emplea un proceso iterativo de optimización para determinar la acción que maximiza la función de valor-acción en entornos caracterizados por valores continuos. En contraposición al trabajo de Mnih, esta propuesta adopta un enfoque que se basa en un método *actor-critic* con aproximadores de funciones neuronales para aprender directamente una política a partir de información sensorial cruda. Esta capacidad de aprender políticas en situaciones que involucran espacios de acciones continuos y de alta dimensionalidad representa una diferencia clave en relación a la investigación previa.

En su estudio, Liang et al. (2015) [9] investigan el rendimiento del algoritmo DQN en el entorno de Arcade Learning Environment (ALE). Además, presentan una metodología de representación de atributos denominada Blob-PROST, que corresponde a *Binary Primitive Object Spatial Transform*. Blob-PROST es una representación binaria de atributos que codifica las relaciones espaciales entre dos colores diferentes en la pantalla del juego. El estudio se enfoca en evaluar el desempeño de esta representación de atributos en una serie de juegos de Atari dentro de la plataforma ALE.

3.2 — Transferencia de Conocimiento en Aprendizaje por Refuerzo

Ammanabrolu et al. (2020) presentó una nueva estrategia para mejorar el aprendizaje profundo por refuerzo mediante el uso de conocimiento previo definido como ”grafos de conocimiento”[2]. Estos grafos representan información acerca del dominio e incluyen entidades, relaciones entre entidades y atributos correspondientes a cada entidad. En este enfoque, se entrena un grafo de conocimiento utilizando información procedente de una tarea fuente. Posteriormente, el algoritmo Deep Q-Network se utiliza para incorporar el conocimiento del grafo en el proceso de toma de decisiones del agente. En su evaluación en diversas tareas de transferencia de conocimiento, se

observó que este enfoque superó en rendimiento a otros métodos existentes. Además, demostraron que el grafo de conocimiento es adaptable y puede ser actualizado durante el entrenamiento, permitiendo que el agente se ajuste a cambios en el entorno de manera efectiva.

El trabajo de Tessler et al. (2016), titulado *"A Deep Hierarchical Approach to Lifelong Learning in Minecraft"* [19], introduce una aproximación innovadora al aprendizaje por refuerzo en el contexto del juego *Minecraft*. Los autores abordan el desafío de permitir que un agente adquiera una amplia gama de habilidades y las adapte de manera continua en el entorno dinámico de *Minecraft*. Para lograrlo, proponen una arquitectura jerárquica que utiliza el módulo *"Deep Skill"*. Este módulo capacita al agente para aprender, reutilizar y combinar habilidades de alto nivel, las cuales se generan por separado mediante el entrenamiento del agente en tareas específicas. El enfoque se basa en una combinación de técnicas de aprendizaje por refuerzo y redes neuronales profundas, lo que, en conjunto, constituye un sólido marco de trabajo para abordar los desafíos inherentes a las tareas en entornos complejos de videojuegos, como es el caso de *Minecraft*.

3.3 — Representación Relacional en Aprendizaje por Refuerzo

Garnelo et al. (2016) propuso una innovadora manera de abordar tareas complejas en el contexto del aprendizaje por refuerzo [6]. En este trabajo, reconocen que el aprendizaje profundo es excepcional para aprender de la experiencia y sobresale en la captura de patrones y representaciones a partir de grandes cantidades de datos, especialmente cuando se trata de dimensiones elevadas, como imágenes o datos sensoriales en bruto. Sin embargo, identifican que ciertas tareas de aprendizaje por refuerzo requieren lo que ellos denominan razonamiento simbólico". Este razonamiento implica el uso de reglas lógicas y relaciones para comprender conceptos abstractos y símbolos.

Para abordar esta necesidad, presentan un enfoque híbrido que combina el aprendizaje profundo con un motor de razonamiento simbólico. Este enfoque se desglosa en dos componentes:

1. Componente de aprendizaje profundo: En esta parte, utilizan una red neuronal profunda para aprender una política basada en la experiencia, empleando una variante del algoritmo Deep Q-Network.

2. Motor de razonamiento simbólico: Este componente se encarga del razonamiento simbólico del agente en su entorno. Utilizan un motor de lógica de primer orden para representar de manera simbólica las acciones realizadas por el agente.

Para evaluar su enfoque, los autores lo aplicaron a una tarea de navegación en un laberinto tridimensional. Descubrieron que su enfoque tuvo un rendimiento superior en comparación con un enfoque puramente basado en aprendizaje profundo y uno basado en razonamiento simbólico.

3.4 — Análisis y Discusión

La Tabla 3.1 ofrece un resumen de los estudios relacionados con esta investigación. Vale la pena destacar que, aunque se han realizado investigaciones previas sobre la representación relacional en el aprendizaje por refuerzo, el enfoque propuesto aquí se distingue por incorporar elementos de visión computacional en el sistema de detección de objetos para la abstracción del entorno utilizando relaciones, además de tener como uno de los objetivos principales la transferencia del conocimiento adquirido.

En contraposición a investigaciones anteriores, el enfoque presentado utiliza imágenes para identificar las relaciones entre los objetos en el entorno. Esta estrategia tiene el potencial de mejorar la eficiencia del aprendizaje del agente y, al mismo tiempo, simplificar la transferencia de políticas aprendidas a juegos de naturaleza similar. Por otro lado, la representación relacional propuesta en este trabajo se distingue por solo utilizar relaciones espaciales para determinar el estado del agente, sin tomar en cuenta otros tipos de relaciones que puedan involucrar acciones o recompensas.

Articulo	Año	Enfoque	Contribuciones clave
Deep Q-Networks	2015	Deep Q-Learning	Introdujo la Deep Q-Network (DQN) para los juegos de Atari, marcando un hito en el aprendizaje profundo por refuerzo al lograr un rendimiento a nivel humano. Utilizó el proceso de *experience replay* y redes objetivo para aumentar la estabilidad.
Continuous control with deep RL	2019	Deep Q-Learning	Introduce un algoritmo de aprendizaje profundo por refuerzo para el control continuo, aprovechando un método de *actor-critic* con aproximadores de funciones neuronales.
State of the art Control of Atari Games using shallow RL.	2015	Deep Q-Learning Representación de atributos	Se investiga el algoritmo DQN en la plataforma ALE y se propone una metodología de representación de atributos llamada Blob-PROST.
Transfer Learning in RL	2016	Grafos de conocimiento, transferencia de conocimiento.	Propuso el aprendizaje por transferencia basado en grafos de conocimiento, superando a los métodos existentes.
Towards deep symbolic RL	2016	Aprendizaje profundo, razonamiento simbólico.	Presentó un enfoque híbrido que combina redes neuronales profundas con el razonamiento simbólico para tareas complejas, superando a los métodos puramente profundos y simbólicos en una tarea de navegación en un laberinto tridimensional.
Deep Hierarchical Approach to Lifelong Learning in Minecraft	2016	RL profundo jerárquico, Minecraft.	Introdujo un enfoque jerárquico de aprendizaje por refuerzo para el aprendizaje continuo en Minecraft con el Módulo de Habilidades Profundas, lo que permitió la adquisición y reutilización de habilidades a nivel alto. Demostró su capacidad de adaptación a diversas tareas en Minecraft, con implicaciones para entornos complejos y de naturaleza abierta más allá del ámbito de los videojuegos.

Capítulo 4

Desarrollo de la Propuesta

4.1 — Metodología

La metodología propuesta para este trabajo se basará en los siguientes pasos:

1. Implementación de un sistema de detección de objetos utilizando la librería Detecto en Python, basada en la arquitectura Faster R-CNN [11]. Este sistema permitirá al agente detectar los objetos presentes en el ambiente y generar relaciones espaciales.

2. Integración del sistema de detección de objetos en un ambiente de aprendizaje por refuerzo basado en juegos de video de Atari. A partir de esta integración, se definió un conjunto de relaciones que permita hacer una abstracción del ambiente y determinar el conjunto de estados en el que se encuentra el agente.

3. Adaptación del algoritmo Q-Learning para incorporar las representaciones relacionales y los modelos causales en el proceso de toma de decisiones del agente. Se investigó la eficiencia y la capacidad de generalización del algoritmo mediante la utilización de estas herramientas.

4. Realización de experimentos y pruebas utilizando juegos de disparos (shoot'em ups) de Atari para valorar la efectividad del enfoque propuesto, Se compararon los resultados obtenidos con los del algoritmo Q-Learning tradicional para demostrar las modificaciones en términos de velocidad de aprendizaje y transferencia de conocimiento.

4.2 — Arcade Learning Environment

Para este trabajo, se utilizó la librería Arcade Learning Environment (ALE) de Python para modelar juegos de Atari como entornos de aprendizaje por refuerzo [3]. En particular, se seleccionaron juegos de la categoría *shoot'em up*, donde el objetivo principal es sobrevivir oleadas de ataques enemigos mientras se intenta destruirlos o evadir obstáculos y enemigos. Estos juegos proporcionan desafíos de juego rápidos, con acción intensa y niveles que aumentan su dificultad a medida que el jugador avanza en el juego.

Como tarea de aprendizaje por refuerzo, los juegos de video son ideales debido a las siguientes razones:

1. Ambientes bien definidos: Los videojuegos presentan objetivos claros y acciones especificas que deben seguirse para lograr estos objetivos. La ubicación del agente se conoce en todo momento, lo que permite definir los estados de manera directa a través de la observación. Esto facilita el diseño y la evaluación de algoritmos de aprendizaje por refuerzo, ya que las acciones del agente y sus consecuencias pueden medirse y observarse con precisión.

2. Escenarios simulados: Los videojuegos pueden simular escenarios que pueden ser muy complejos o incluso imposibles de recrear en el mundo real. Esto permite investigar aspectos específicos del aprendizaje por refuerzo, como la toma de decisiones, la planificación estratégica y la adaptación a entornos dinámicos.

3. Escalabilidad: Los videojuegos ofrecen diferentes niveles de dificultad, lo que permite investigar la capacidad de un agente para aprender y adaptarse a tareas que se vuelven gradualmente más complejas.

4. Disponibilidad de datos: Los juegos generan una gran cantidad de datos, incluyendo información sobre estados, acciones y recompensas, que se pueden utilizar para el entrenamiento de un agente y la evaluación de algoritmos de aprendizaje por refuerzo.

5. Reproducibilidad: La naturaleza inherente de los juegos permite repetir tareas de forma consistente, lo que facilita la repetición de experimentos y la compa-

ración de resultados entre diferentes algoritmos, técnicas y enfoques de manera estandarizada.

En general, la combinación de ambientes bien definidos, la simulación de escenarios, la escalabilidad y la disponibilidad de datos, así como la capacidad de establecer estándares para la evaluación de elementos en el aprendizaje por refuerzo, hacen de los videojuegos una elección popular para la investigación en este ampo del aprendizaje computacional.

4.3 — Sistema de Detección de Objetos

El sistema de detección de objetos se incorporó al ambiente de aprendizaje por refuerzo utilizando Detecto para Python [1]. Detecto es una librería para la detección de objetos que se basa en redes neuronales convolucionales (CNN, por sus siglas en inglés), y se construye sobre el marco de aprendizaje profundo PyTorch. Utiliza arquitecturas de detección de objetos como Faster R-CNN, para realizar la tarea de detección de objetos en imágenes y videos. Los componentes clave de esta librería son los siguientes:

- Arquitecturas de detección de objetos: Faster R-CNN. Es una arquitectura que combina una red de regiones de interés (R-CNN) con una red neuronal convolucional. Primero, se utiliza una red neuronal convolucional para extraer las características de la imagen, para después generar propuestas de regiones de interés. Estas regiones se clasifican y se ajustan mediante una red neuronal para obtener la detección final de objetos.
- Modelos Pre-Entrenados. Detecto proporciona modelos pre-entrenados basados en la arquitectura mencionada con anterioridad. Estos modelos han sido entrenados en conjuntos de datos masivos, como COCO (*Common Objects in COntext*), que contiene imágenes con diferentes categorías de objetos. Los modelos pre-entrenados incluyen pesos ajustados que les permiten reconocer una amplia variedad de objetos.
- Proceso de detección de objetos: Al realizar la tarea de detección de objetos, Detecto realiza los siguientes pasos:

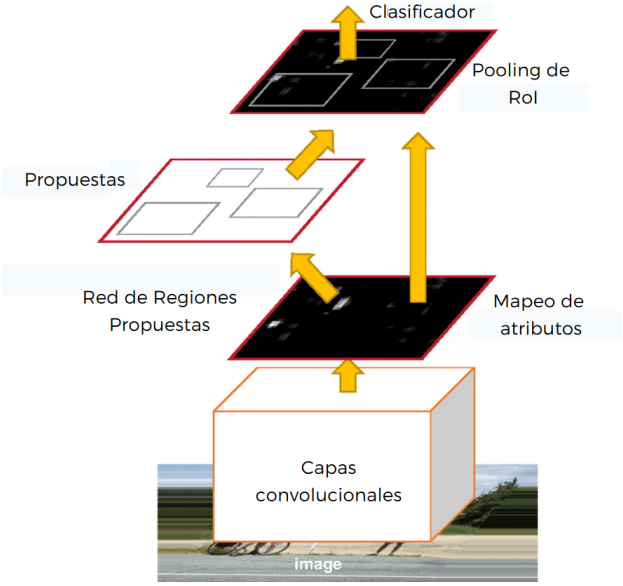

Figura 4.1. Proceso de detección de objetos a través de la librería Detecto [5].

El modelo toma como entrada una imagen, la cual es procesada para obtener regiones de interés que se clasifican para determinar la categoria del objeto detectado.

1. La imagen se pasa a través del *"backbone"* de la red neuronal convolucional para extraer características relevantes.
2. Se generan propuestas de regiones de interés que podrían contener objetos.
3. Cada propuesta se clasifica y se ajusta mediante una red neuronal para determinar la categoría del objeto y refinar su ubicación.
4. Si hay varias detecciones superpuestas de un mismo objeto, se realiza un post-procesamiento en los objetos detectados para filtrar predicciones de baja confianza.

Este proceso puede observarse de forma general en la Figura 4.1.

- *Bounding boxes* y categorías: Detecto proporciona un par de coordenadas de las *"bounding boxes"* que rodean cada objeto detectado en una imagen. Estas coordenadas cooresponden al extremo superior izquierdo y extremo inferior derecho del rectánculo que contiene al objeto. Estas *bounding boxes* indican la ubicación y el tamaño del objeto. Además, Detecto puede asignar una categoría a cada

objeto detectado.

- Entrenamiento personalizado: Es posible utilizar conjuntos de datos etiquetados con sus categorías correspondientes para entrenar un modelo propio de detección de objetos.

En este estudio, se emplearon imágenes obtenidas directamente de los juegos utilizados, en este caso *River Raid* y *Chopper Command*, con el objetivo de entrenar el modelo de detección de objetos. De esta manera, el modelo pudo clasificar de forma precisa los objetos presentes en una imagen del juego y utilizar esta información para comprender las relaciones espaciales entre el agente y dichos objetos.

4.4 — Representación Relacional

Para este trabajo, se utilizó la librería Detecto para desarrollar un sistema de detección de objetos. El sistema se entrenó utilizando un conjunto de datos etiquetados que contenía imágenes del juego River Raid y Chopper Command de Atari. Se eligió este juego para realizar las pruebas iniciales.

En particular, el sistema de detección de objetos proporciona información sobre la ubicación de cada objeto mediante el uso de *"bounding boxes"*. Estas cajas delimitan las coordenadas de la esquina superior izquierda y la esquina inferior derecha de cada objeto. Para lograr esto, se seleccionaron agentes, obstáculos, trofeos y paredes como elementos para la detección de objetos. Para obtener las relaciones entre los objetos y el agente, se definieron los centroides de cada objeto para calcular una distancia estimada entre ellos utilizando la distancia euclidiana. Esta metodología se puede visualizar en la Figura 4.3, donde se observa claramente la representación de los centroides y las distancias estimadas.

Es decir, si el sistema de detección define los extremos de la "bounding box"del agente como (x_l, y_l) y (x_r, y_r), se tiene que el centroide es igual al punto medio entre estos dos puntos extremos, por lo tanto, el centroide del agente se puede calcular mediante la siguiente expresión:

$$(x_a, y_a) = (\frac{x_l + x_r}{2}, \frac{y_l + y_r}{2}). \tag{4.1}$$

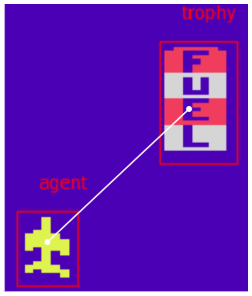

Figura 4.2. A partir del sistema de detección es posible encontrar las distancias entre los objetos a través de la distancia euclidiana.

Si se define de manera similar el centroide de un objeto (x_o, y_o), tenemos entonces que la relación espacial entre el agente y un objeto se basa en la siguiente expresión:

$$d((x_a, y_a), (x_o, y_o)) = d(Agent, Object) = \sqrt{(x_a - x_o)^2 + (y_a - y_o)^2}. \tag{4.2}$$

Se definieron dos relaciones de distancia: cerca y lejos, con base a dos umbrales preestablecidos $n, m \in \mathbb{R}$ distintos de 0 con $m < n$. Estas relaciones se representan de la siguiente manera:

$$CloseToObject(Agent, Object) \iff d(Agent, Object) < m,$$
$$FarFromObject(Agent, Object) \iff m < d(Agent, Object) < n.$$

Donde $CloseToObject$ y $FarFromObject$ tienen su valor de verdad determinado por la distancia entre el agente y los objetos circundantes [16]. En el enfoque propuesto, estos predicados forman parte de las relaciones que constituyen el conjunto de estados del ambiente.

Por otro lado, un problema dentro de los videojuegos que puede abordarse mediante el sistema de detección de objetos es la colisión del agente con las paredes. El sistema de detección de objetos puede identificar y registrar la presencia de las paredes en el escenario del videojuego. Cuando se detecta una posible colisión entre el agente y una pared, el agente puede tomar medidas apropiadas, como ajustar su trayectoria o aplicar consecuencias adecuadas al choque.

Para estos casos, la distancia euclidiana no es suficiente, pues la orientación del

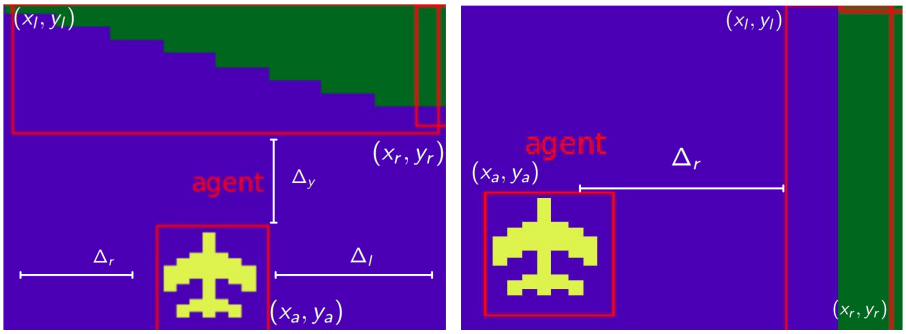

Figura 4.3. Ejemplos de detección de objetos aplicado a las paredes cercanas al agente. La distancia relativa entre las coordenadas del centroide del agente y las coordenadas de la pared permiten saber si el agente se acerca a una pared.

centroide del agente con el de la detección de una pared conduce a la toma de decisiones erróneas por parte del agente, asi que es necesario agregar condiciones extras para determinar la ubicación espacial de las paredes respecto al agente. Supongamos ahora que las coordenadas de una pared son (x_l, y_l) y (x_r, y_r). Entonces, si el centroide del agente es el punto (x_a, y_a), se definen las siguientes cantidades como las distancias relativas entre el agente y las líneas rectas que representan los bordes de la pared:

$$\begin{aligned} \Delta_l &= |x_a - x_r|, \\ \Delta_r &= |x_a - x_l|, \\ \Delta_y &= |y_a - y_l|. \end{aligned} \tag{4.3}$$

En la Figura 4.3 se puede observar dos ejemplos de cómo se utilizó la distancia relativa de las coordenadas del agente con las coordenadas de los vértices extremos de la pared para determinar si el agente se acerca a una pared en algún momento durante el entrenamiento. Observe que cuando el agente se acerca de frente a una pared, es necesario calcular las distancias a ambos bordes de la pared para que el agente pueda determinar si es más conveniente moverse a la izquierda o la derecha.

Por lo tanto, los siguientes casos fueron considerados para la detección de paredes alrededor del agente:

1. Pared lateral izquierda: Sea $k \in \mathbb{R}$ diferente de 0, si la coordenada y_a se encuentra dentro del intervalo formado por y_l y y_r, es decir, $y_a \in (y_l, y_r)$, además, $\Delta_l > \Delta_r$

y $\Delta_l < k$, entonces la pared se encuentra a la izquierda del agente.

2. Pared lateral derecha: Sea $k \in \mathbb{R}$ diferente de 0. Si $y_a \in (y_l, y_r)$, $\Delta_r < k$ y $\Delta_r > \Delta_l$, entonces la pared se encuentra a la derecha del agente.

3. Pared superior: Si $x_a \in (x_l, x_r)$ y $y_a < y_r$, entonces la pared se encuentra arriba del agente.

4. Pared inferior: Si $x_a \in (x_l, x_r)$ y $y_a > y_l$, entonces la pared se encuentra debajo del agente.

5. Pared superior derecha: Sean $k, n \in \mathbb{R}$ diferentes de 0. Si $\Delta_r < k$, $\Delta_y < n$ y $\Delta_l > \Delta_r$, entonces la pared se encuentra arriba y a la derecha del agente.

6. Pared superior izquierda: Sean $k, n \in \mathbb{R}$ diferentes de 0. Si $\Delta_l < k$, $\Delta_y < n$ y $\Delta_r > \Delta_l$, entonces la pared se encuentra arriba y a la izquierda del agente.

Cada una de estas condiciones permite asignar un valor de verdad a la relación $CloseToWall$, lo que significa que el sistema de detección proporciona información sobre la proximidad del agente a las paredes en el entorno.

Como se puede observar, a través del sistema de detección, es posible generar las relaciones $CloseToObject(Agent, Object)$ y $FarFromObject(Agent, Trophy)$. La relación $CloseToObject$ es verdadera si la distancia entre el agente y el objeto es menor que m, es decir, cuando $d(Agent, Object) < m$. Por otro lado, se tiene que la relación $FarFromObject$ es verdadera cuando la distancia entre el agente y el objeto se encuentra en el rango $m < d(Agent, Object) < n$. Como consecuencia, se tiene un conjunto de relaciones que permiten conocer el estado del agente dentro del ambiente haciendo una abstracción de lo que sucede a su alrededor. Por lo tanto, para este trabajo se definieron las siguientes relaciones para ser utilizadas como estados dentro del ambiente:a

$$CloseToObject(Agent, Object),$$
$$FarFromObject(Agent, Object),$$
$$CloseToWall(Agent, Wall),$$

Donde $Object$ puede tomar los valores $obstacle$ o $trophy$, y $Wall$ puede tomar los valores R, L, UR, UL, F, B para denotar a la pared lateral derecha, la pared lateral

izquierda, la pared superior derecha, la pared superior izquierda, pared superior y pared inferior respectivamente. Además, se utilizaron variables auxiliares X, Y y D para determinar la ubicación de los objetos respecto al agente, de modo que X toma los valores $left$ y $right$ y Y toma los valores up y $down$ y finalmente, D toma los valores $close$ y far. Puesto que la lógica de primer orden permite darle valor de verdad a estas relaciones, se tiene que los estados del ambiente tambien incluye disyunciones entre las relaciones, es decir, los estados dentro del ambiente tienen la siguiente posible representación:

$$CloseToObject(Agent, Object) \vee CloseToWall(Agent, Object),$$
$$FarFromObject(Agent, Object) \vee CloseToWall(Agent, Object),$$

4.5 — Proceso de Decisión de Markov

El ambiente utilizado en este trabajo se modeló como un Proceso de Decisión de Markov, o MDP, donde sus componentes se definieron utilizando el sistema de detección de objetos y la representación relacional.

4.5.1. Estados

Tomando en cuenta la definición propuesta de las relaciones espaciales del agente con su entorno, el conjunto de estados S se ha establecido con un total de 119 relaciones que permiten llevar a cabo una abstracción de los elementos circundantes al agente.[1] Estas relaciones se definieron teniendo en cuenta las diversas combinaciones posibles de variables, como obstáculos, trofeos y paredes, priorizando aquellas combinaciones que son detectables. Para ser más precisos, se incluyeron únicamente las relaciones que pueden surgir durante el proceso de detección durante el entrenamiento del agente.

[1] Puesto que las variables $Object, X, Y$ y D tienen dos posibles valores y $Wall$ puede tomar 7, se tienen $2^4 \cdot 7 = 112$ relaciones. Luego, considerando el caso donde el agente solo esta cerca de alguna pared, se tienen $112 + 7 = 119$ relaciones.

4.5.2. Acciones

La libreria ALE permite definir el conjunto de acciones como números enteros, es decir, el conjunto de acciones está predefinido de acuerdo a los movimientos posibles del control de Atari, por lo que la lista de acciones es la siguiente:

1. NOOP: Se refiere a no realizar acción alguna, es decir, corresponde a no presionar ningun botón ni a mover el mando direccional.
2. FIRE: Corresponde al botón de disparo.
3. UP: Movimiento hacia arriba.
4. RIGHT: Movimiento hacia la derecha.
5. LEFT: Movimiento hacia la izquierda.
6. DOWN: Movimiento hacia abajo.
7. UPRIGHT: Movimiento en diagonal hacia arriba y la derecha.
8. UPLEFT: Movimiento en diagonal hacia arriba y la izquierda.
9. DOWNRIGHT: Movimiento en diagonal hacia abajo y la derecha.
10. DOWNLEFT: Movimiento en diagonal hacia abajo y la izquierda.
11. UPFIRE: Combinación de movimiento hacia arriba con disparo.
12. RIGHFIRE: Combinación de movimiento hacia la derecha con disparo.
13. LEFTFIRE: Combinación de movimiento hacia la izquierda con disparo.
14. DOWNFIRE: Combinación de movimiento hacia abajo con disparo.
15. UPRIGHTFIRE: Combinación de movimiento en diagonal hacia arriba y la derecha con disparo.
16. UPLEFTFIRE: Combinación de movimiento en diagonal hacia arriba y la izquierda con disparo.

17. DOWNRIGHTFIRE: Combinación de movimiento en diagonal hacia abajo y la derecha con disparo.

18. DOWNLEFTFIRE: Combinación de movimiento en diagonal hacia abajo y la izquieda con disparo.

Dado que estas 18 acciones pueden representarse por números enteros en el orden en el que fueron enumeradas, el conjunto de acciones entonces esta definido de la siguiente manera:

$$\mathcal{A} = \{0, 1, 2, ..., 15, 16, 17\}$$

(4.4)

4.5.3. Recompensa

En este trabajo, los estados definidos por relaciones se denominan $r-$estados. Teniendo el conjunto de estados representado como un conjunto de relaciones y el conjunto de acciones definido como un conjunto de números enteros que representan cada posible acción que se puede ejecutar con el mando de Atari, la función de recompensa se definió de la siguiente forma, dados $k, m, n \in \mathbb{R}$:

$$r(r - estado, accion) = \begin{cases} k & \text{si el agente se acerca a un trofeo.} \\ m & \text{si el agente se aleja de un obstáculo.} \\ n & \text{si el agente se aleja de una pared.} \end{cases} \tag{4.5}$$

En este trabajo, se ha optado por asignar una recompensa más significativa al agente por la adquisición de trofeos en contraste con la recompensa otorgada por evitar obstáculos. Asimismo, se ha definido que la máxima recompensa que el agente puede obtener se relaciona con la acción de mantener distancia de las paredes, ya que esta priorización permite al agente evitar colisiones de manera efectiva. Para lograr esto, se estableció la siguiente relación de valores: $m < k < n$.

4.6 — Algoritmo RQ-Learning

Luego, la incorporación de estos elementos al algoritmo Q-Learning está basada en la detección de objetos, pues a través de ella se puede definir el estado actual del agente. Además, puesto que la detección de objetos resulta en un subconjunto $S' = \{R|R \in S \text{ es una relación}\}$ del conjunto S, pues es posible que el agente detecte varias relaciones al mismo en un paso en el tiempo t, es necesario definir un criterio de selección de estado. Para ello se decidió por elegir la relación ó $r-$estado con el mayor Q-valor, y en caso de que hubiera empates, se elige una relación al azar. Formalmente, la elección del $r-$estado se puede representar con la siguiente función:

$$r - estado = \begin{cases} R_i & \text{si } \max Q(S', a) = R_i, \forall a \in \mathcal{A}, i = 1, 2, .., |S'| \\ \mathcal{R} & \text{de otra forma.} \end{cases} \tag{4.6}$$

Donde $\mathcal{R}$ es una variable aleatoria con función de probabilidad $P(\mathcal{R} = R_i) = \frac{1}{|S|}, i = 1, 2, ..., |S'|$. Por lo tanto, el algoritmo Q-Learning propuesto, que incluye la incorporación del sistema de detección de objetos es el siguiente, se puede observar en el Algoritmo 2. A esta variante se le ha denominado como algoritmo RQ-Learning.

Algoritmo 2 : RQ-Learning

Parámetros del algoritmo: $\alpha \in [0, 1]$, $\gamma \in [0, 1]$, $\epsilon > 0$ pequeño (entre 0.1 y 0.01).
Inicializar $Q(s, a), \forall s \in S, a \in A$ de forma arbitraria.

Bucle para cada episodio:
  Inicializar S (conjunto de relaciones)
  Bucle para cada paso en el episodio:
    $S' = Deteccion\_Objetos(imagen) \subset S$
    $r - estado = mejor\_relacion(S')$
    Escoger a' de s utilizando la politica derivada de Q (ϵ greedy)
    Observar $R(r - estado, a') = R$
    $S'' = Deteccion\_Objetos(siguienteimagen) \subset S$
    $sig.r - estado = mejor\_relacion(S'')$
    $Q(r-estado, a') \longleftarrow Q(r-estado, a') + \alpha[R + \gamma \max_a Q(sig.r-estado, a) - Q(r - estado, a')]$
    $r - estado \longleftarrow sig.r - estado$
  Hasta que $r - estado$ sea un estado terminal

El algoritmo propuesto permite utilizar las relaciones encontradas a través del sistema de detección de objetos para definir el estado en el que se encuentra el agente. La estrategia $\epsilon-$greedy está basada a partir de un modelo lineal definido de la siguiente manera:

$$\epsilon(t) = 1 - mt, \tag{4.7}$$

donde t representa el número de episodios y m es el gradiente de la línea que representa a $\epsilon(t)$. La elección de este modelo fue basada en los experimentos realizados en el trabajo de Machado et al.[12].

El diagrama de bloques que se presenta en la Figura 4.4 ofrece una representación visual del enfoque que se ha propuesto. Se puede apreciar que la función llamada $mejor\_relacion$ equivale a la selección precisa de $r-$estados basada en el subconjunto de relaciones identificadas por el sistema de detección de objetos.

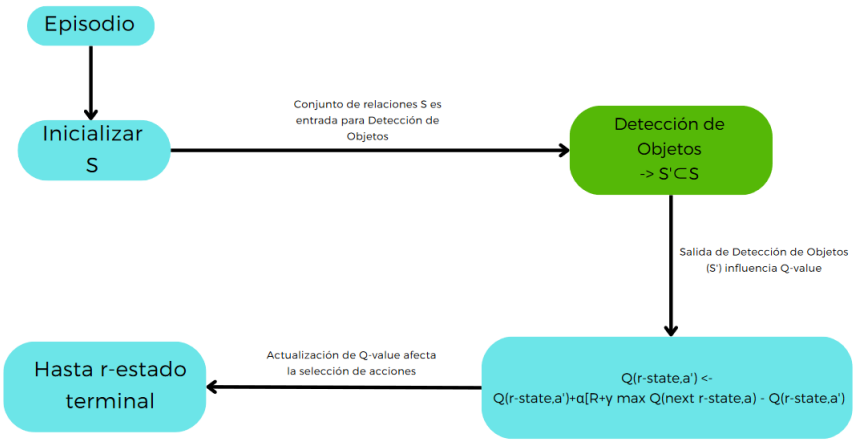

Figura 4.4. A partir del sistema de detección es posible encontrar las distancias entre los objetos a través de la distancia euclidiana.

Es relevante destacar que la imagen utilizada para el proceso de detección de objetos es capturada en tiempo real del juego en cada episodio. Además, con el fin de permitir una comparación adecuada de los resultados obtenidos en este estudio con otros trabajos similares [3], se ha decidido procesar la imagen del juego cada 4 fotogramas para analizar las interacciones entre el agente y el entorno.

4.6.1. Transferencia de conocimiento

En este trabajo, se utilizó aprendizaje por transferencia para comprobar que un agente aprendiera una tarea en particular y, utilizando esta experiencia, tratar de aprender otra tarea similar. En el contexto de este trabajo, el objetivo fue trasladar la experiencia previa del agente del juego *River Raid* al juego *Chopper Command* a través de la forma tabular de la función de valor. Es decir, se utilizó la $Q-$tabla generada en el entrenamiento del agente en *River Raid* como punto de partida para que el agente aprendiera a jugar *Chopper Command*. Esta transferencia de parámetros es posible ya que el algoritmo Q-learning inicializa los valores de esta tabla de manera arbitraria, de modo que se puede aprovechar para utilizar una Q-tabla con valores estimados en otra tarea similar.

La manera en la que se hizo la transferencia fue directa, pues ambos juegos comparten el mismo conjunto de acciones $\mathcal{A}$ y las relaciones definidas son aplicables en el contexto de los dos juegos, por lo que el conjunto de estados $\mathcal{S}$ también es el mismo para las dos tareas.

4.7 — Resumen

En este capítulo, se describe la metodología empleada en este trabajo, la cual consta de cuatro fases fundamentales: la definición del sistema de detección de objetos, la obtención de la representación relacional a partir de esta detección, la integración de dicha representación en el proceso de aprendizaje del agente y, por último, la transferencia de la experiencia previa a tareas similares. Estos componentes conforman un nuevo enfoque dentro del aprendizaje por refuerzo, con el objetivo de abstraer las características del entorno a través de relaciones y evaluar la efectividad del aprendizaje del agente.

El sistema de detección de objetos permitió una identificación óptima de las relaciones espaciales del agente con su entorno, lo que planteó el desafío de incorporar estas relaciones al proceso de aprendizaje por refuerzo. Sin embargo, al definir las relaciones como los estados de un Proceso de Decisión de Markov (MDP), fue posible utilizar este resultado para proponer el algoritmo RQ-Learning. Además, se demostró

que es factible transferir el conocimiento al utilizar los parámetros aprendidos en una tarea del agente en otra tarea similar, permitiendo así llevar a cabo la transferencia de conocimiento de manera efectiva.

Capítulo 5

Experimentos y Resultados

La sección experimental de este trabajo de tesis se dividió principalmente en tres etapas: la implementación del sistema de detección de objetos, la aplicación de la representación relacional en un contexto de aprendizaje por refuerzo y la transferencia de conocimientos obtenidos a partir de los elementos mencionados a una tarea similar. La figura 5.1 ilustra esta división del trabajo propuesto.

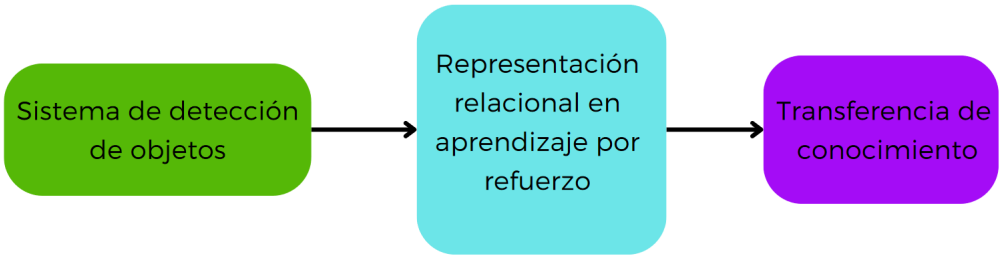

Figura 5.1. División fundamental de este trabajo en etapas.

La introducción del sistema de detección de objetos facilitó la representación del entorno del juego *River Raid* como un Proceso de Decisión de Markov. Esta representación se basó en la detección de objetos, definiendo un conjunto de estados que incorporan las relaciones espaciales entre el agente y los objetos circundantes. Estos estados se consideraron en el contexto de las dos tareas experimentales planteadas.

Posteriormente, empleando estas relaciones espaciales, el agente aplicó el algoritmo Q-Learning para aprender dinámicamente sobre el juego. Se exploró incluso la posibilidad de proponer una variante del algoritmo, aprovechando el conjunto de estados derivados para el proceso de aprendizaje del agente.

Con el agente ahora competente en la toma de decisiones en el juego *River Raid*, se llevó a cabo una transferencia de conocimiento. La Q-tabla resultante se utilizó como parámetro de inicialización para el aprendizaje del agente en el juego *Chopper Command*.

5.1 — Sistema de Detección de Objetos

Durante el entrenamiento del sistema de detección, se obtuvo una gráfica que representa la función de pérdida, la cual se muestra en la Figura 5.2. En dicha gráfica se observa cómo la pérdida converge a un valor mínimo de 0.40 después de aproximadamente 10 episodios. Este resultado permitió que el modelo detectara y clasificara correctamente las imágenes extraídas del juego durante el entrenamiento del agente.

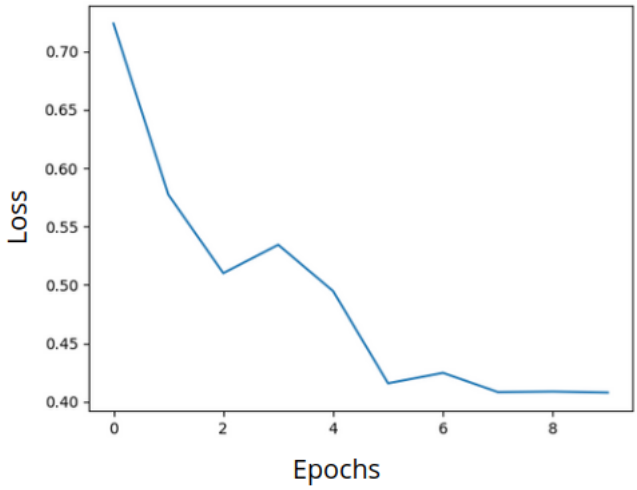

Figura 5.2. Función de pérdida del sistema de detección de objetos.

Para las siguientes pruebas, se entrenó el sistema de detección de objetos utilizando un conjunto de 150 imágenes etiquetadas. Estas imágenes se dividieron en dos conjuntos: el conjunto de entrenamiento, que contiene 120 de estas imágenes, y el conjunto de prueba, que contiene las 30 imágenes restantes. Es importante mencionar que el conjunto de 150 imágenes incluye las imágenes de los 2 juegos con los que se planificó inicialmente comenzar las pruebas (River Raid y Chopper Command).

5.2 — Representación Relacional en Aprendizaje por Refuerzo

Las pruebas iniciales fueron realizadas dentro del juego River Raid, en el cual el jugador controla un avión de combate y debe navegar a través de un río lleno de obstáculos y enemigos. El objetivo principal es destruir puentes, tanques, barcos y aviones enemigos, al mismo tiempo que se evitan colisiones con los objetos y se mantiene un suministro adecuado de combustible. A medida que el jugador avanza, la dificultad aumenta con la aparición de enemigos más rápidos y obstáculos más desafiantes. River Raid es conocido por su jugabilidad intensa y desafiante, así como por su diseño de niveles generados de manera procedimental, lo que garantiza una experiencia de juego diferente en cada partida. La Figura 5.3 muestra una imagen durante un episodio del juego River Raid.

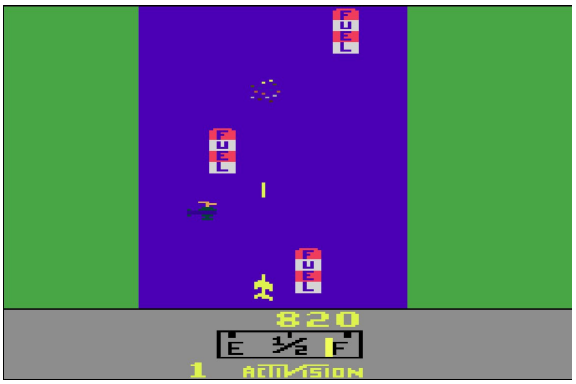

Figura 5.3. River Raid es un juego de disparos desarrollado para la consola Atari y lanzado en 1982.

La elección de los parámetros para la implementación del enfoque propuesto, que consiste en el algoritmo RQ-Learning, se basó en investigaciones previas, en particular, se tomó en consideración el trabajo de Machado et al. [12], donde se calcula la recompensa total acumulada por episodio.

Dentro de los parámetros seleccionados, se definió la estrategia ϵ-greedy, la cual utiliza una función lineal que tiene un valor mínimo de 0.1 una vez que se alcanzan los 1500 episodios de entrenamiento, es decir, $\epsilon(1500) = 0.1$. Además, se fijó una razón de aprendizaje de (α) en 0.0025 y un factor de descuento (γ) en 0.99. Es fundamental

destacar que la elección de α en 0.0025 proporciona un proceso de aprendizaje más estable. De la misma manera, el valor de γ, establecido en 0.99, refleja la importancia otorgada a las recompensas futuras en el proceso de toma de decisiones. Estos parámetros fueron cuidadosamente seleccionados para optimizar el rendimiento del algoritmo RQ-Learning en nuestro estudio. El resultado de este experimento se puede apreciar en la Figura 5.4.

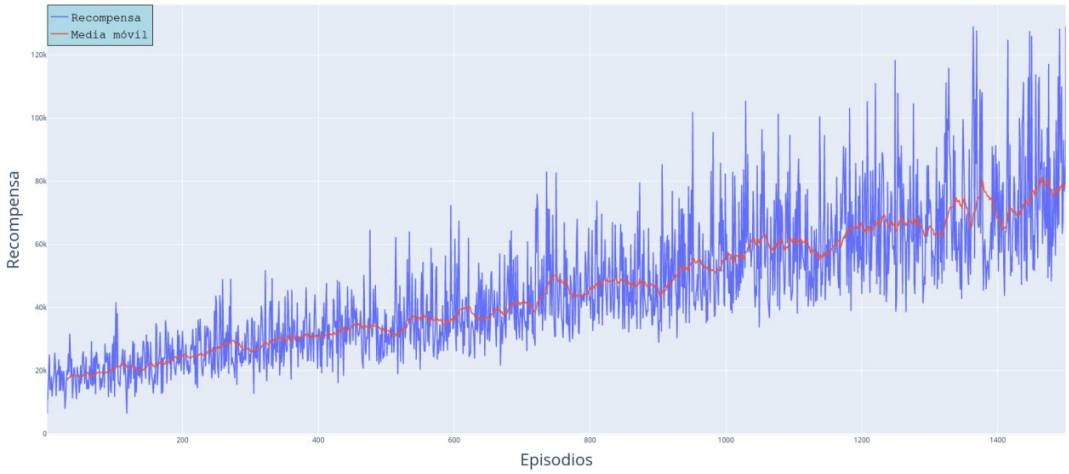

Figura 5.4. Resultados de entrenamiento del agente a través de 1500 episodios en *River Raid*.

Es destacable el aumento progresivo de la recompensa promedio a medida que avanzan los episodios. La línea roja en el gráfico representa la media móvil con una ventana de tamaño $n = 30$. Se observa que el agente logró obtener puntajes superiores a los alcanzados por otros algoritmos en varias ocasiones, aunque también se observa una mayor variabilidad en estas recompensas. Es relevante resaltar que algunos valores atípicos por debajo de la media son comúnmente causados por la aleatoriedad inherente del juego. Esta característica adquiere una gran importancia en este trabajo, ya que este tipo de perturbaciones permite evaluar la robustez de la política generada y cómo el ruido en los datos afecta el entrenamiento del agente.

Para comprobar la tendencia que sigue la recompensa promedio a través del paso de los episodios, se realizaron pruebas cada 50 episodios para comprobar la efectividad de la política generada hasta ese momento. En la Figura 5.5 se puede observar en los resultados de estos experimentos realizados cada 50 episodios que la recompensa va

Figura 5.5. Resultados de la prueba de rendimiento del agente cada 50 episodios.

aumentando conforme avanzan los episodios, pues tienen una tendencia lineal positiva.

El mismo experimento se llevó a cabo en el juego *Chopper Command*, donde el jugador asume el control de un helicóptero militar y se enfrenta a una serie de desafíos. En este juego, la tarea principal es proteger una serie de bases terrestres y convoyes militares de los ataques de aviones enemigos y misiles. Mientras se realizan estas misiones de defensa, el jugador debe ser hábil para derribar a los enemigos antes de que causen daño a las bases o convoyes. A medida que se avanza en el juego, la dificultad aumenta con la aparición de aviones enemigos más rápidos y la intensificación de los ataques. *Chopper Command*, al igual que *River Raid*, se destaca por su jugabilidad emocionante y desafiante. Ambos juegos ofrecen intensas experiencias de juego en las que los jugadores deben sortear obstáculos y enfrentarse a enemigos mientras avanzan a lo largo de la partida. Además, al igual que en *River Raid*,*Chopper Command* emplea un diseño de niveles generados de manera procedural, lo que garantiza una experiencia única en cada sesión de juego. Una imagen del juego *Chopper Command* se puede observar en la Figura 5.6.

Los juegos *River Raid* y *Chopper Command* comparten ciertas similitudes como juegos de disparos de desplazamiento vertical con una acción intensa. Sin embargo, se diferencian en términos del vehículo controlado y el enfoque de las misiones. En *River Raid*, el jugador pilotea un avión de combate y se enfrenta al desafío de destruir puentes,

Figura 5.6. Captura de pantalla de *Chopper Command*: Pilotando un helicóptero militar en una intensa batalla aérea defensiva.

tanques, barcos y aviones enemigos mientras navega por un río lleno de obstáculos. Por otro lado, *Chopper Command* permite al jugador tomar el control de un helicóptero militar y se centra en la defensa de bases y convoyes militares contra ataques aéreos enemigos.

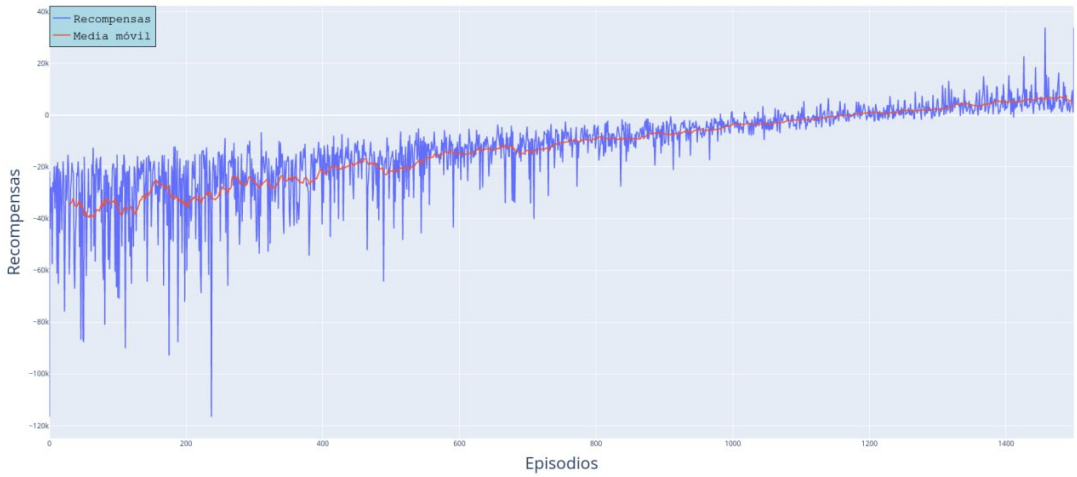

Figura 5.7. Resultados de entrenamiento del agente a través de 1500 episodios en *Chopper Command*.

Estas diferencias en mecánica, vehículos y objetivos de misión ofrecen experiencias de juego únicas en cada título, a pesar de las similitudes en su género de disparos y enfoque en la acción. A pesar de esto, ambas tareas pueden modelarse como un MDP, donde las acciones, estados y recompensas son las mismas. Los resultados de este experimento se pueden apreciar en la Figura 5.7. Debido a las distintas características

mencionadas en ambos juegos, se observa que la gráfica en este caso comienza con recompensas negativas, pero a medida que avanzan los episodios, estas se estabilizan. Esto demuestra empíricamente que el algoritmo RQ-Learning está aprendiendo a resolver el problema presentado, a pesar de las diferencias entre ambos juegos.

El siguiente paso consistió en evaluar la política generada después de los 1500 episodios de entrenamiento, donde se calculó el promedio de las recompensas obtenidas y el puntaje interno del juego (*score*). Los resultados se presentan en detalle en la Tabla 5.1, destacando los mejores resultados. La desviación estándar se representa con el número entre paréntesis. Es notable que en ambos juegos, la desviación estándar es significativamente alta debido a la naturaleza inherente de la aleatoriedad en el juego, lo que genera perturbaciones en la aplicación del algoritmo y, como resultado, variaciones comunes durante el periodo de entrenamiento del agente.

Tabla 5.1. Comparación del promedio de puntaje en 100 episodios de prueba entre diferentes enfoques y RQ-Learning.

	DQN	**Sarsa(λ) + Blob-PROST**	**RQ-Learning**
River Raid	3166.2 (**125.2**)	**4141.9** (574.4)	2103.4 (524.5)
Chopper Command	841.4 **(144.3)**	**1647.5** (389.2)	474 (147.4)

Dado lo anterior, se realizó un análisis de varianza bidireccional [4] para comparar la media de los tres algoritmos tomando en cuenta la influencia de los dos juegos propuestos. Se definió el siguiente conjunto de hipótesis:

H_0 : La media de los tres algoritmos son iguales vs. H_1 : Al menos la media de un algoritmo es diferente.

Para calcular los estadísticos F y los p-valores, se utilizó el software estadístico R, que arrojó los resultados descritos en la Figura 5.8 utilizando un nivel de significancia de $\alpha = 0.05$. La variable Dataset corresponde a la elección de los juegos utilizados (*River Raid* y *Chopper Command*).

De acuerdo con los resultados del análisis de varianza bidireccional, se pueden extraer las siguientes conclusiones:

	Df	Sum Sq	Mean Sq	F value	Pr(>F)
Algorithm	2	1621464	810732	0.365	0.709
Dataset	1	4071093	4071093	1.833	0.225
Algorithm:Dataset	2	56850	28425	0.013	0.987
Residuals	6	13323713	2220619		

Figura 5.8. Resultados obtenidos del ANOVA bidireccional para determinar si hay diferencia significativa entre los 3 algoritmos. Se tomaron en cuenta la media y la desviación estándar de 3 algoritmos con 2 tareas distintas (*Dataset*).

1. El estadístico F para los Algoritmos es 0.365, con un p-valor de 0.709. Dado que este valor es mayor que el nivel de significancia, se puede afirmar con un $95\,\%$ de confianza que la elección del algoritmo no tiene un impacto significativo en el resultado final

2. El estadístico F para la elección del juego es 1.833, con un p-valor de 0.225. De manera similar, este valor supera el nivel de significancia, lo que indica que la elección del juego no afecta el resultado final de manera significativa.

3. El estadístico F para la interacción entre los Algoritmos y los juegos no es significativo, pues su valor es 0.013 y el p-valor es 0.987. Esto sugiere que no hay evidencia significativa de interacción entre la elección del algoritmo y el juego.

En consecuencia, según el análisis anterior, no se observan diferencias significativas entre los tres algoritmos, los dos juegos y su interacción. La hipótesis nula no se rechaza, y con un nivel de significancia del $95\,\%$, se confirma que no existen evidencias sólidas que sugieran diferencias significativas entre las medias de los algoritmos.

5.3 — Transferencia de conocimiento

Hasta el momento, se ha explorado el proceso de aprendizaje del agente en los juegos *River Raid* y *Chopper Command*, logrando desarrollar una política efectiva mediante entrenamiento y optimización de recompensas. Entonces, se puede plantear la pregunta: ¿es posible transferir el conocimiento adquirido por el agente en River Raid a otros juegos y tareas relacionadas?

En esta sección, se aborda el concepto de transferencia de conocimiento, aprovechando los avances obtenidos en *River Raid* para aplicarlos a *Chopper Command*. El objetivo es evaluar la capacidad del agente para adaptarse a nuevos desafíos y utilizar la experiencia previa para acelerar el aprendizaje en un entorno diferente. A través de la transferencia de conocimiento, el objetivo es maximizar la eficiencia y el rendimiento del agente, aprovechando la información y la experiencia previa para acelerar el aprendizaje en nuevas tareas. Esto no solo resulta en un agente más versátil y adaptable, sino que también sienta las bases para futuras aplicaciones de aprendizaje computacional en diversos entornos y dominios.

La prueba inicial se llevó a cabo en el juego *Chopper Command*, que comparte varias similitudes con *River Raid*. Se plantea que el conocimiento generado por el agente en River Raid puede ser aprovechado para aprender las dinámicas de *Chopper Command*. Es importante mencionar que ambos juegos cuentan con obstáculos dentro de cada ambiente, aunque las acciones son las mismas. Además, puesto que se está utilizando el mismo conjunto de estados en ambas tareas, la dimensión de la Q-tabla será igual.

Para lograr el objetivo, se realizó una transferencia de parámetros, utilizando los valores de la Q-tabla generada durante el entrenamiento de 1500 episodios en *River Raid*. En términos formales, dado que el MDP propuesto tiene como dominio el conjunto $S \times A$, el algoritmo de transferencia toma como entrada el conjunto de Q-valores calculados en la tarea fuente y el conocimiento disponible de la tarea de destino, devolviendo un conjunto de Q-valores que contienen el conocimiento de ambas tareas. Por lo tanto, si Q_s^* es la función de valor estimada de la tarea fuente y Q_t^* es la función de valor estimada para la tarea de destino, se tiene que $\mathcal{K}_s = \{Q_s^*(s,a) | s \in S, a \in A\}$, $\mathcal{K}_t = S \times A$ y $\mathcal{K}_{transfer} = \{Q_t^*(s,a) | s \in S, a \in A\}$.

El experimento que se realizó con relación al aprendizaje por transferencia consistió en utilizar la política generada por el agente durante un entrenamiento de 1500 episodios en el juego *River Raid* para que aprendiera a jugar el juego *Chopper Command*.

El resultado de la Figura 5.9 muestra a través de 1500 episodios, donde un agente entrenó en el juego *Chopper Command* sin una política previa y con una política generada del juego *River Raid*, cómo el agente comenzó el entrenamiento con cierto conocimiento sobre algunas acciones de acuerdo a las relaciones encontradas por el

detector de objetos.

Figura 5.9. Resultados durante 1500 episodios del entrenamiento de un agente en el juego Chopper Command con política previa y sin ella, donde se contabilizó la recompensa total acumulada por episodio.

En este caso, se puede observar que el aprendizaje es inestable durante el transcurso de los episodios, aunque el entrenamiento con política previa muestra un inicio más favorable y menor variabilidad que el entrenamiento sin política previa. Hay que destacar que la función de recompensa no fue optimizada de tal manera que el agente se adaptara mejor a este ambiente, pues el movimiento en *River Raid* es de abajo hacia arriba y en *Chopper Command* puede ser de izquierda a derecha o viceversa, por lo que la modificación de esta función, asi como el incremento o decremento de los umbrales para las distancias entre el agente y el entorno podrían cambiar estos resultados de manera significativa. Sin embargo, se llevó a cabo una prueba de 100 episodios utilizando las políticas resultantes de ambos conjuntos de entrenamientos para obtener los resultados que se muestran en la Tabla 5.2.

Se puede observar que el entrenamiento con política produce una solución a esta tarea que permite alcanzar un puntaje máximo más alto en comparación con un agente entrenado sin una política previa como guía. No obstante, es importante destacar que la variabilidad en el rendimiento es mayor en el caso del entrenamiento con política. Puesto que el entrenamiento con política previa fue mas estable aproximadamente del episodio 1000, se puede conjeturar que el agente tuvo un comportamiento menos

Tabla 5.2. Resultados de puntuación en el juego Chopper Command en una prueba de 100 episodios.

Chopper Command	μ	σ
Con política	**589**	274.55
Sin política	474	**147.39**

errático dentro del juego debido a la poca variabilidad a partir de este punto en la recompensa total acumulada, lo que le permitió obtener mejores puntajes que el agente con entrenamiento sin política.

Al considerar las dos curvas de aprendizaje obtenidas, podemos evaluar la mejora en la transferencia de conocimiento mediante la proporción de áreas. Si representamos el área bajo la curva de aprendizaje con conocimiento previo como A_t y el área bajo la curva de aprendizaje sin conocimiento previo como A_n, entonces la métrica r se define como:

$$r = \frac{A_t - A_n}{A_n}.$$

Esta métrica cuantifica la mejora en la transferencia de conocimiento [18]. En nuestro experimento, calculamos esta proporción utilizando las curvas de aprendizaje de la Figura 5.8, y obtuvimos un valor de r igual a 0.02194. Esto indica una mejora del aprendizaje de aproximadamente el $2\,\%$ despues de 1500 episodios al utilizar una política basada en el conocimiento adquirido de una tarea previa. Además, al considerar la mejora inmediata (*jumpstart*), observamos que la curva de aprendizaje cuando el agente utiliza una política previamente aprendida comienza con una diferencia positiva significativa en comparación con el caso en el que comienza sin política alguna. La diferencia entre las recompensas iniciales es de 9948.37, lo que sugiere que la transferencia de la política de la tarea fuente a la tarea de destino es efectiva cuando las dos tareas son similares y se pueden representar como MDPs con elementos equivalentes, como lo son el conjunto de estados, el conjunto de acciones o la función de recompensa.

Por último, al analizar el comportamiento asintótico de ambas curvas de aprendizaje, se observa que en un punto, el aprendizaje con transferencia tiene un rendimiento inferior al aprendizaje sin el uso de una política previa. Esto indica que la transferencia

de la política generada en *River Raid* puede ser beneficiosa durante la fase de exploración cuando el agente está aprendiendo a jugar *Chopper Command.*

En resumen, la aplicación de la transferencia de conocimiento en el contexto de juegos como *River Raid* y *Chopper Command* demuestra la prometedora capacidad de la inteligencia artificial para adaptarse a nuevas tareas y entornos mediante la reutilización de la experiencia previa. Este enfoque no solo aumenta la eficiencia del aprendizaje, sino que también sienta las bases para la creación de agentes más versátiles y adaptables en una amplia variedad de dominios. La posibilidad de aprovechar el conocimiento adquirido en un juego para mejorar el rendimiento en otro o aprovecharlo para encontrar una solución más rápido ofrece un interesante campo de investigación y aplicaciones futuras en el área del aprendizaje computacional y la inteligencia artificial.

Además, la transferencia de conocimiento presenta un desafío para comprender cómo las máquinas pueden aprender de manera similar a cómo los humanos generalizan y aplican lo que han aprendido en un contexto a otro. Esta investigación no solo contribuye al desarrollo de sistemas de aprendizaje más eficientes, sino que también tiene el potencial de aportar ideas a numerosas áreas, desde la robótica hasta la toma de decisiones autónomas en una amplia gama de aplicaciones. La transferencia de conocimiento es un paso significativo hacia la creación de agentes de inteligencia artificial capaces de enfrentar una variedad de desafíos de manera adaptativa y, finalmente, mejorar la autonomía y versatilidad de las máquinas en nuestro mundo en constante cambio.

5.4 — Resumen

En esta sección se resumen los resultados de los experimentos realizaods de acuerdo a la metodología previamente expuesta. En términos generales, se han alcanzado los siguientes resultados:

- El sistema de detección de objetos se entrenó con éxito utilizando imágenes de los juegos empleados en los experimentos. Su integración en las tareas de aprendizaje por refuerzo se realizó de manera adecuada, aprovechando la información obtenida de la detección de objetos para establecer relaciones espaciales entre el agente y el ambiente.

- Estas relaciones definieron un conjunto de estados que permitieron modelar la tarea como un Proceso de Decisión de Markov (MDP). Esto posibilitó la aplicación del algoritmo *Q-Learning* para guiar al agente en el proceso de aprendizaje. Tras 1500 episodios de entrenamiento, se verificó que el agente adquirio un entendimiento sólido del juego al emplear este conjunto de relaciones para tomar decisiones. Además, se comprobó que el enfoque propuesto es estadísticamente comprobable con otros métodos, como DQN o Sarsa(λ)+Blob-PROST.

- La transferencia de conocimiento se logró mediante la transferencia de parámetros entre tareas, en particular, los valores estimados $Q(s, a)$. Es importante destacar que esta transferencia, aunque a largo plazo no supera el rendimiento del entrenamiento sin una política previa, muestra una mejora significativa en las etapas iniciales del entrenamiento, lo que se conoce como *jumpstart improvement*.

En conjunto, estos hallazgos destacan el potencial de la representación relacional en el contexto del aprendizaje por refuerzo y sus implicaciones en una variedad de entornos y tareas, como el aprendizaje por transferencia.

Capítulo 6

Conclusiones y Trabajo futuro

6.1 — Conclusiones

En este trabajo, se propuso la utilización de representación relacional en el aprendizaje por refuerzo con el objetivo de verificar la capacidad de un agente para aprender. Además, se exploró la posibilidad de utilizar esta representación para realizar una abstracción del entorno que permitiera transferir el conocimiento adquirido en una tarea a otra similar. Las tareas experimentales se llevaron a cabo en juegos de Atari de la categoría *shoot'em up*.

Para lograr esto, se incorporó con éxito un sistema de detección de objetos que actuó como radar para el agente, permitiéndole "observar" los objetos en el entorno durante el entrenamiento. El uso de este sistema proporcionó información geométrica detallada de los objetos, lo que permitió, a través de la representación relacional, abstraer ciertas características del entorno que se utilizaron para definir los estados del agente.

La implementación exitosa del sistema de detección de objetos y el conjunto de relaciones condujo a una variante del algoritmo *Q-learning*, denominada *RQ-Learning*. En esta variante, se definieron conceptos relevantes para la tarea, como la función de recompensa y la función selectora de la mejor relación dado un subconjunto del conjunto de estados.

En cuanto a los resultados de esta parte de la investigación, se observó un aumento constante en la recompensa máxima del agente a medida que avanzaba a lo largo de los episodios, lo que sugiere la posibilidad de alcanzar una política óptima con un mayor

tiempo de entrenamiento.

La transferencia de parámetros entre los juegos *River Raid* y *Chopper Command* se realizó de manera exitosa utilizando la Q-tabla resultante del entrenamiento como parámetro de inicialización para la nueva tarea. Se evidenció una mejora inmediata (*jumpstart*) entre la recompensa inicial con transferencia y la recompensa inicial sin transferencia.

A pesar de que la orientación entre los dos juegos es diferente, el agente logró encontrar una política que le permitió alcanzar puntajes razonables en el juego, además de desarrollar una estrategia coherente con la función de recompensa establecida. Los resultados preliminares en la transferencia de conocimiento indican que es factible transferir conocimiento entre juegos que comparten características similares representadas en forma de relaciones.

6.2 — Trabajo Futuro

Tomando como base las relaciones identificadas por el sistema de detección, el enfoque propuesto sugiere la utilización de un modelo causal derivado de estas relaciones para representar el efecto inmediato de una acción en un conjunto específico de relaciones, como se ilustra en la Figura 6.1.

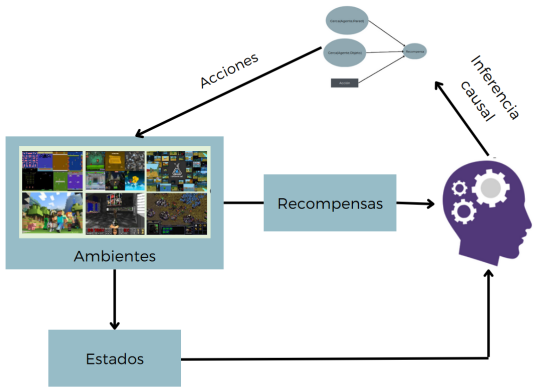

Figura 6.1. Incorporación de modelos causales basados en las relaciones obtenidas de la detección de objetos para diferentes tareas.

La premisa subyacente es que el agente pueda aprender un modelo causal simultáneamente con la adquisición de una política, todo basado en esta representación relacional abstracta. Una vez que el agente haya desarrollado un modelo parcial [13], podrá emplearlo para seleccionar acciones adecuadas, como aquellas que maximicen las recompensas, con el fin de acelerar aún más el proceso de aprendizaje. Este aspecto se considera como una dirección para investigaciones futuras.

Además, el enfoque propuesto puede aplicarse en diversas áreas, como sistemas multiagente, donde la interacción y las relaciones entre agentes puede variar según la tarea a realizar. Asimismo, en el ámbito de la robótica, el agente podría beneficiarse al comprender las relaciones entre él y su entorno. También, en la gestión de tráfico y vehículos autónomos, es crucial conocer las relaciones y dependencias entre vehículos, peatones y el entorno.

Referencias

[1] Alankbi. *Github - alankbi/detecto: Build fully functioning computer vision models with Pytorch*. URL: https://github.com/alankbi/detecto.

[2] Prithviraj Ammanabrolu y Mark O. Riedl. "Transfer in Deep Reinforcement Learning Using Knowledge Graphs". En: *Conference on Empirical Methods in Natural Language Processing*. 2019.

[3] Marc G. Bellemare et al. "The Arcade Learning Environment: An Evaluation Platform for General Agents (Extended Abstract)". En: *International Joint Conference on Artificial Intelligence*. 2012.

[4] John M. Chambers y Trevor J. Hastie, eds. *Statistical Models in S*. Routledge, nov. de 2017. DOI: 10.1201/9780203738535.

[5] Umer Farooq. "From R-CNN to mask R-CNN - Umer Farooq - medium". En: (jun. de 2018). URL: https://medium.com/@umerfarooq_26378/from-r-cnn-to-mask-r-cnn-d6367b196cfd.

[6] Marta Garnelo, Kai Arulkumaran y Murray Shanahan. "Towards Deep Symbolic Reinforcement Learning". En: *ArXiv* abs/1609.05518 (2016).

[7] Abhijit Gosavi. *Simulation-Based Optimization*. Springer US, 2015. DOI: 10.1007/978-1-4899-7491-4.

[8] Alessandro Lazaric. "Transfer in Reinforcement Learning: A Framework and a Survey". En: *Adaptation, Learning, and Optimization*. Springer Berlin Heidelberg, 2012, págs. 143-173. DOI: 10.1007/978-3-642-27645-3_5.

[9] Yitao Liang et al. "State of the Art Control of Atari Games Using Shallow Reinforcement Learning". En: *ArXiv* abs/1512.01563 (2015). URL: https://api.semanticscholar.org/CorpusID:1970845.

[10] Timothy Lillicrap et al. "Continuous control with deep reinforcement learning". En: *CoRR* (sep. de 2015).

[11] Tsung-Yi Lin et al. “Feature Pyramid Networks for Object Detection”. En: *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2016), págs. 936-944.

[12] Marlos C. Machado et al. “Revisiting the Arcade Learning Environment: Evaluation Protocols and Open Problems for General Agents”. En: *ArXiv* abs/1709.06009 (2017).

[13] Arquímides Méndez-Molina et al. “Causal Based Q-Learning”. En: *Res. Comput. Sci.* 149 (2020), págs. 95-104.

[14] Volodymyr Mnih et al. “Human-level control through deep reinforcement learning”. En: *Nature* 518 (2015), págs. 529-533.

[15] E. Morales. “Scaling Up Reinforcement Learning with a Relational Representation”. En: *Proc. of the Workshop on Adaptability in Multi-Agent Systems (AORC-2003)* (2002), págs. 15-26.

[16] Luc De Raedt et al. *Statistical Relational Artificial Intelligence*. Springer International Publishing, 2016. DOI: 10.1007/978-3-031-01574-8.

[17] Barto A. Sutton R. *Reinforcement Learning: An Introduction, 2nd edition*. The MIT Press, 2020.

[18] Matthew E. Taylor y Peter Stone. “Transfer Learning for Reinforcement Learning Domains: A Survey”. En: *J. Mach. Learn. Res.* 10 (dic. de 2009), págs. 1633-1685. ISSN: 1532-4435.

[19] Chen Tessler et al. “A Deep Hierarchical Approach to Lifelong Learning in Minecraft”. En: *AAAI Conference on Artificial Intelligence*. 2016.

[20] Jindong Wang y Yiqiang Chen. *Introduction to Transfer Learning*. Springer Nature Singapore, 2023. DOI: 10.1007/978-981-19-7584-4.