UNIVERSIDAD CENTRAL DE CHILE
FACULTAD DE INGENIERÍA Y ARQUITECTURA
ESCUELA DE COMPUTACIÓN E INFORMÁTICA

Análisis del aprendizaje por refuerzo profundo para un asistente doméstico en un entorno simulado

Memoria para optar al título profesional de
Ingeniero Civil en Computación e Informática.

Profesor Guía:**Francisco Cruz**
Profesor Informante: **Alejandro Sanhueza**

Ithan Moreira
Javier Rivas

Santiago, Chile
2019

El presente trabajo está dedicado a nuestras familias por haber sido nuestro apoyo a lo largo de toda la carrera universitaria y a lo largo de nuestras vidas. A todas las personas especiales que nos acompañaron en esta etapa, aportando a nuestra formación tanto profesional y como ser humano.

Los autores.

Resumen

En la actualidad nos encontramos insertos en la cuarta revolución industrial, la revolución tecnológica. La tecnología está cambiando la visión del humano. Dentro de los múltiples cambios que está provocando la revolución digital, uno de ellos es pasar de máquinas programadas con una secuencia de pasos a máquinas o agentes inteligentes que aprendan de manera autónoma, explorando caminos, adquiriendo información y llegando a estrategias, sin la necesidad que el usuario final entienda el problema. Pese a los grandes avances, las máquinas aún presentan problemas cuando el entorno es dinámico (cambios continuos), como son los escenarios domésticos. Uno de los campos que busca resolver estos desafíos se llama aprendizaje profundo por refuerzo (DeepRL según sus siglas en inglés).

El presente proyecto consiste en desarrollar un agente inteligente que aprenda a realizar una tarea doméstica utilizando DeepRL interactivo como estrategia de control y aprendizaje. La situación doméstica simulada es que el robot identifique tres objetos diferentes que representaran las distintas prendas de vestir y con dos posibles colores rojo y azul que representaran el estado de la ropa (sucia o limpia). Dependiendo del estado, el robot debe desplazar los objetos de la mesa a un canasto ubicado a la derecha o bien a la izquierda. El proyecto planteado consiste en estudiar diferentes áreas que componen el problema centrándose en dos macro-temas DeepRL y aprendizaje interactivo, de estos macro-temas se hablará de aprendizaje por refuerzo, proceso de decisión de Markov, redes neuronales profundas y aprendizaje interactivo.

El método propuesto permite que el agente autónomo interactúe con el entorno aprendiendo políticas que permiten realizar la tarea de clasificar y separar la ropa sucia de la limpia. El método propuesto es la fusión de DeepRL con IRL(Interactive Reinforcement Learning) y con esta fusión el método obtiene mejores tiempos de aprendizaje de la tarea, evaluando la curva de aprendizaje, tiempo de convergencia y recompensas obtenidas, con estos datos podemos responder a nuestra pregunta

de investigación, la cual es: ¿IDeepRL será mejor que DeepRL en la clasificación de los objetos?.

Si bien IDeepRL mejora la curva de aprendizaje de los agentes DeepRL logra aprender la tarea y de igual manera logra clasificar los objetos, en un análisis más exhaustivo, IDeepRL logra ser una mejor metodología ya que el tiempo de aprender la tarea es más corto. Aún existen algunos temas a tener en cuenta, tales como ¿los maestros fueron los adecuados al momento de entrenar a los agentes?, ¿cómo se puede seleccionar un buen maestro para el agente?, o tal vez ¿la metodología fue la correcta para la tarea realizada?, estas preguntas surgen a partir de los resultados obtenidos, permitiendo la extensión de la aproximación IDeepRL en distintas direcciones, incluyendo aplicaciones en entornos domésticos.

Índice general

Índice de figuras

Capítulo 1

Introducción

1.1. Motivación

El campo del aprendizaje automático (machine learning del inglés) en los últimos años ha tomado relevancia debido al avance tecnológico. Las técnicas de machine learning están revolucionando la forma de resolver los problemas, dejando de lado a la programación como una secuencia de instrucciones para comenzar a pensar en cómo podría aprender una máquina para resolver el problema.

Al igual que la tecnología, este campo avanza a un ritmo muy rápido, en la actualidad se puede observar máquinas las cuales aprenden a diferenciar objetos, aprender a tomarlos y lanzarlos o dejarlos en un recipiente (Zeng et al., 2019), pero aún quedan bastantes desafíos a resolver, como por ejemplo acelerar la velocidad de aprendizaje de las máquinas o lograr que una máquina aprenda más de una tarea sin olvidar lo aprendido, resolver estos desafíos ayudaría a estar más cerca de la robótica doméstica y así mejorar la calidad de vida de las personas.

1.2. Definición del Problema

Parte de la tecnología se centra en lograr nuevos avances que mejoren la civilización(Tadele et al., 2014). Hace años atrás, la robótica se limitaba al área industrial (por ejemplo, en el área automotriz)(Schwab, 2016).El avance tecnológico permitió que los robot amplíen su dominio de aplicaciones en áreas tales como la medicina, militar, rescate, entretenimiento, entre otras. Otra de sus aplicaciones que está en

desarrollo es la robótica en entorno doméstico, esto requiere de un gran desafío debido a que los entornos domésticos son variables, a diferencia de los entornos industriales. Se espera que las personas interactuen con máquinas de manera cotidiana y, a su vez, que las máquinas comprendan y respondan al entorno doméstico (Schwab, 2016)(Tadele et al., 2014). Por otra parte, el avance tecnológico ha permitido el desarrollo de nuevas formas de resolver los problemas, soluciones que hacen que las máquinas aprendan a resolver tareas complejas sin necesidad de una programación secuencial (Mnih et al., 2015), una de las técnicas para resolver estos problemas es DeepRL. Por lo tanto, el problema abordado en esta tesis se centra en enseñar una tarea domestica a un robot simulado y que este sea capaz de resolverla de manera eficiente basándose en un modelo de DeepRL y una estrategia de aprendizaje interactivo.

1.3. Objetivos

1.3.1. Objetivo General

Desarrollar una estrategia de aprendizaje para agentes inteligentes en un entorno doméstico simulado utilizando aprendizaje por refuerzo profundo con retroalimentación interactiva.

1.3.2. Objetivos Específicos

En el presente trabajo, se han definido los siguientes objetivos específicos:

- Estudiar sobre DeepRL interactivo, estrategias de aprendizajes y entornos de simulación.
- Diseñar un entorno simulado del problema el cual consiste de un robot situado en una mesa donde se realizará la tarea.
- Desarrollar algoritmos de DeepRL interactivo para resolver la tarea de identificación y selección de prendas sucias.
- Desarrollar experimentos con DeepRL, utilizando entrenadores humanos y agentes inteligentes previamente entrenados.
- Analizar los resultados obtenidos según tiempo de aprendizaje, tiempo en desarrollar la tarea y desempeño en la tarea (porcentaje de error).

1.4. Hipótesis

La hipótesis del trabajo se centrará en sí DeepRL interactivo es capaz de mejorar el rendimiento general (tiempo en desarrollar la tarea, desempeño en la tarea, curva de aprendizaje) de un agente desarrollando una tarea doméstica en comparación con DeepRL. Para validar esta hipóstesis nos preguntamos si, ¿Interactive Deep Reinfocement Learning(IDeepRL) es mejor que Deep Reinforcement Learning(DeepRL) términos de desempeño?

1.5. Metodología de Trabajo

Dado que el proyecto es una investigación, se usa el método científico (Nola and Sankey, 2014) descrito a continuación :

- **Planteamiento del problema**: Un robot simulado, utilizando DeepRL interactivo como estrategia de control y aprendizaje, ¿es capaz de identificar si la ropa se encuentra sucia o limpia? y dependiendo de esto, ¿es capaz de desplazarla a un canasto?

- **Examen y análisis de los enfoques existentes**: Se realiza una revisión bibliográfica del estado del arte, para así obtener una visión detallada de los principales enfoques actualmente usados.

- **Construcción del escenario experimental**: Se desarrollaron algoritmos de DeepRL y se diseño un ambiente semi-controlado utilizando un software en el cual se realizan los experimentos.

- **Revisión de resultados y análisis**: Los resultados obtenidos se evalúan en esta etapa considerando la precisión obtenida en la tarea de control, rendimiento, curva de aprendizaje.

- **Resultados del informe**: Todos los resultados obtenidos son reportados a través de la tesis, así como también a través de una publicación científica en una conferencia internacional, actualmente en proceso de revisión.

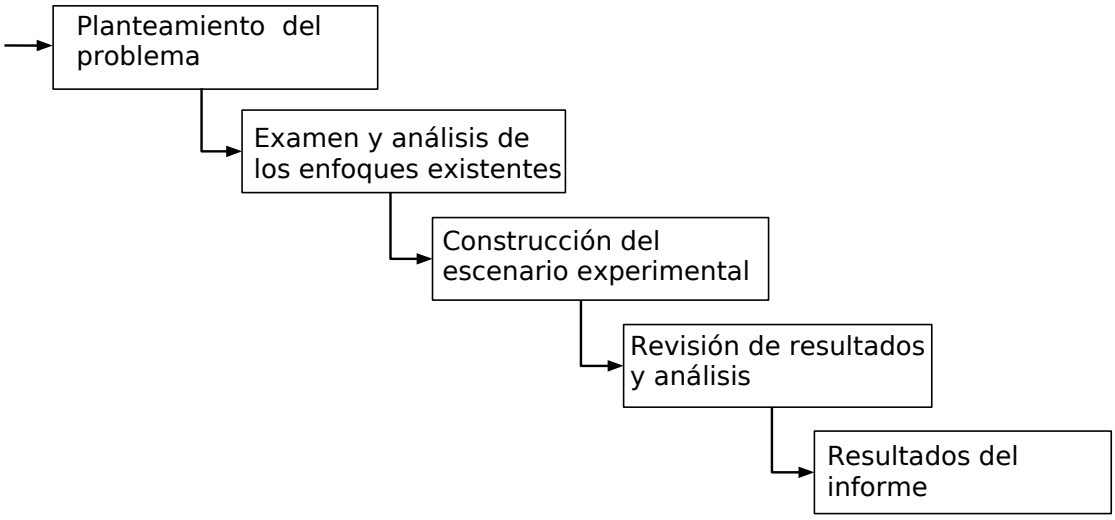

Figura 1.1: Pasos llevados a cabo en el método científico. Adaptada desde (Nola and Sankey, 2014).

1.6. Estructura del Documento

A continuación se expone un resumen de las partes que componen el documento como se aprecian en la Figura 1.1.

1. Introducción: Como primera parte se explican las motivaciones del tema, luego se define la problemática, sus objetivos y la metodología con la que se trabajará.

2. Marco teórico y estado del arte: Se estudia la problemática desde dos macro temas, por un lado el aprendizaje por refuerzo profundo que a su vez se divide en aprendizaje por refuerzo, proceso de decisión Markoviano y redes neuronales profundas. Y el segundo macro tema es el aprendizaje interactivo.

3. Escenario experimental: Se diseña el escenario experimental, indicando los posibles estados, acciones, recompensas y restricciones del problema.

4. Diseño e implementación del agente: Se define la arquitectura de la red neuronal, la implementación del enfoque interactivo y la representación continua.

5. Comparación y análisis de resultados: Se analizan los resultados obtenidos, que nos entregan los experimentos realizados.

6. Conclusion: Este capítulo resume las principales ideas del trabajo realizado, discute los resultados y propone futuras mejora

Capítulo 2

Marco teórico y estado del arte

A continuación, se expondrá documentación necesaria para abordar la problemática de la investigación. La investigación se centra en el aprendizaje por refuerzo profundo interactivo, pero para comprender el enfoque se estudiaron subtemas que lo componen. Como punto de partida se hablará del aprendizaje por refuerzo, este tema es la base de DeepRL, comparten gran parte de los fundamentos teóricos y diseño de algoritmos. Luego se hablará del aprendizaje por refuerzo interactivo, éste es un método utilizado para mejorar el rendimiento del agente. Para complementar la teoría de DeepRL se discute sobre las redes neuronales profundas y por último, se habla de DeepRL, haciendo la unión entre los temas expuestos (RL y DNN).

2.1. Aprendizaje por refuerzo

El aprendizaje por refuerzo (Reinforcement Learning del inglés RL) es un area de la inteligencia artificial que busca automatizar el aprendizaje de un agente. Esta basado en el conductismo el cual explica la interacción de un individuo el cual realiza una acción sobre el entorno para obtener una recompensa, como se puede apreciar en la Figura 2.1. Por ejemplo, el objetivo de un agente es entrar a una habitación que se encuentra cerrada, el agente interactúa con el entorno realizando una acción, si la acción le permite cumplir su objetivo, el agente, recibirá una recompensa positiva, o en caso contrariarío recibirá una recompensa negativa o neutra, con esta información el agente es capaz de diferenciar cuáles acciones cumplen de mejor manera el objetivo. El aprendizaje reforzado está compuesto

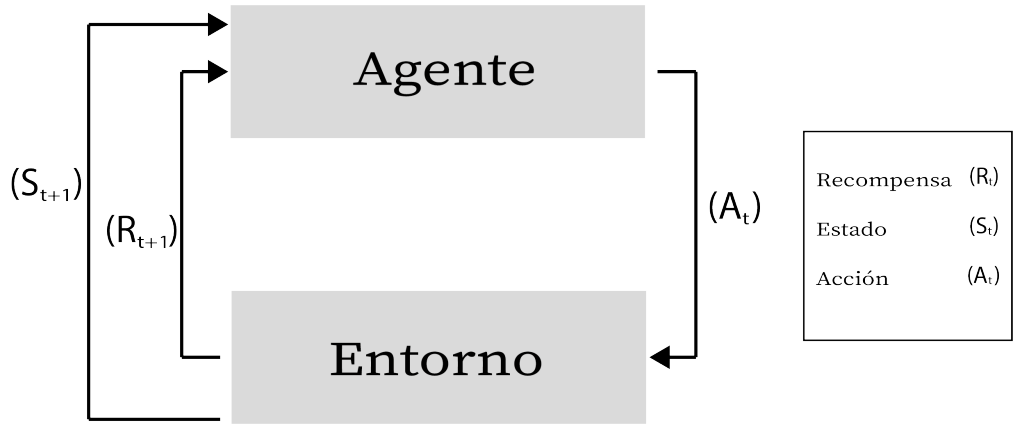

Figura 2.1: interacción agente-entorno en un proceso de decisión de Markov. Un agente interactúa en un entorno realizando una acción A_t, entonces el entorno le da un valor a la acción previa y la entrega como recompensa junto con un nuevo estado. Figura adaptada desde (Sutton and Barto, 1998).

de 4 elementos principales una política, una señal de recompensa, una función de valor y un modelo del medio ambiente.

- Una política define el comportamiento del agente ante diferentes estados entregados por el entorno, puede ser representada mediante una tabla de funciones. La política determina que acción tomar en cierta situación.
- Una señal de recompensa define la meta del agente, el único objetivo del agente es maximizar la recompensa total. En cada cambio de estado se le entrega al agente un valor de recompensa el cual define si es una buena o mala acción tomada para alcanzar la meta.
- Una función de valor le indica al agente que tan buena fue la acción que éste tomo, dependiendo del estado o la acción que se realizó.
- Opcionalmente, un modelo del entorno que es una vista a futuras acciones tomadas por el agente con la finalidad de maximizar la recompensa y tomar la mejor decisión de qué acción realizar.

2.1.1. Proceso de decisión Markoviano

Una problemática que trata la inteligencia artificial es que un agente artificial realice acciones pensando en futuras recompensas, incluso si esto significa sacrificar la recompensa actual para obtener una mayor recompensa futura (Sigaud and

Buffet, 2013). Por ejemplo, en el ajedrez un jugador podría decidir sacrificar piezas para lograr ganar la partida.

Este problema se aborda a través del proceso de decisión Markoviano o MDP (Markov Decision Processes del inglés), el cual plantea la retroalimentación evaluativa y asociativa, ya que toma diferentes acciones en diferentes situaciones, dado que la decisión no sólo influye en la recompensa actual sino que en las futuras recompensas, estados y situaciones (Sutton and Barto, 1998).

Un MDP finito es aquel en donde los estados, las acciones y recompensas (S, A y R) poseen un número finito de posibilidades y las cuales se pueden definir matemáticamente dentro de un tiempo (t), en este caso las variables aleatorias R_t y S_t dependen sólo del estado y la acción anterior como se aprecia en la ecuación (2.1).

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

La ecuación denotada en (2.1) es una función de cuatro argumentos, la cual nos indica una distribución de probabilidad para cada elección de S y A, es decir, las probabilidades de la ecuación (2.1) caracterizan completamente a un MDP finito y a partir de él, se puede calcular las recompensas esperadas para los pares de estado-acción en la ecuación (2.2), las probabilidades de transición de estado en la ecuación (2.3) y además se puede obtener las recompensas esperadas para los tres argumentos estado-acción-próximo estado denotada en la ecuación (2.4) (Sutton and Barto, 1998).

$$r(s, a) = \mathbb{E}[R_t|S_{t-1} = s, A_{t-1} = a] = \sum_{r\in\mathbb{R}} r \sum_{s'\in\mathbb{S}} p(s', r|s, a) \tag{2.2}$$

$$p(s'|s, a) = Pr\{S_t = s'|S_{t-1} = s, A_{t-1} = a\} = \sum_{r\in\mathbb{R}} p(s', r|s, a) \tag{2.3}$$

$$r(s, a, s') \doteq \mathbb{E}[R_t \mid S_{t-1} = s, A_{t-1} = a, S_t = s'] = \sum_{r\in R} r\frac{p(s', r|s, a)}{p(s'|s, a)} \tag{2.4}$$

2.1.2. *Diferencia-Temporal*

Diferencia Temporal (TD del inglés) es una metodología de aprendizaje creada al mezclar otras dos metodologías Monte Carlo (MC) y programación dinámica (PD).

A diferencia de las metodologías anteriores, en TD no requiere de un modelo del entorno y el agente aprende con cada nueva acción y no al finalizar un episodio (secuencia alterna de estados y pares de estado-acción)(Sutton and Barto, 1998). Por ejemplo, el tiempo que demora un Uber en llegar es una predicción que se hace en el momento, con la distancia como valor inicial, es una predicción sin tener el conocimiento del tráfico, a medida de que el vehículo recorre y conoce nuevos estados (lo que conlleva a aprender el comportamiento del entorno), el tiempo se actualiza haciendo una predicción más certera.

Para calcular el valor-estado TD requiere hacer una predicción como se muestra en la la fórmula (2.5). Para realizar estas predicciones existen dos categoría principales on-police y off-police, la primera, busca estimar el valor de la acción sobre un estado para una política actual, un ejemplo de esto es el algoritmo SARSA, la segunda categoría, aproxima la acción-valor aprendida a la acción-valor óptima independiente de la política que se siga (Sutton and Barto, 1998), un ejemplo es el algoritmo *Q-learning*.

$$V(S_t) \leftarrow V(S_t) + \alpha[R_{t+1} + \gamma V(S_{t+1}) - V(S_t)] \tag{2.5}$$

2.1.3. Exploración y explotación

Que el agente aprenda no es el único desafío, otro dilema es cómo balancear la exploración y la explotación, un agente que no cuente con un método de exploración repetirá la acción aprendida que le traiga una recompensa mayor (no necesariamente positiva), es necesario que de vez en cuando el agente explore nuevos caminos (exploración) en busca de nuevas acciones que le traigan mayor recompensa, un buen balance entre exploración y explotación trae beneficios en tiempo de aprendizaje y la calidad de las políticas aprendidas. Demasiada exploración impide maximizar la recompensa a corto plazo, dado que el agente siempre intenta nuevos caminos. Por otro lado, demasiada explotación minimiza las recompensas a largo plazo porque el agente siempre repite las acciones y éstas podrían no ser las mejores (Sutton and Barto, 1998).

Un ejemplo de esto es un robot que recoja basura, si explota sólo se limitará a limpiar un área determinada donde anteriormente ha sido exitoso y no tratará de limpiar áreas nuevas. En contraparte, si el robot explora, conocerá diferentes áreas por limpiar, pero decaerá su eficacia al limpiar. Un buen equilibrio seria que el robot limpiara el área que conoce y de vez en cuando explorara nuevas áreas para así mejorar su rendimiento poco a poco.

Alguno de los métodos más conocidos que tratan de resolver el dilema son ϵ-*greedy* y *softmax* (Tokic, 2010).

- **ϵ-*greedy*:** El agente tiene una probabilidad aleatoria (ϵ) entre 0 y 1 de realizar exploración en vez de la acción que maximiza su recompensa, este algoritmo puede tener variantes, como por ejemplo, que en el comienzo del aprendizaje el agente tenga una mayor probabilidad de explorar.
- ***softmax*:** Selecciona una acción basándose en probabilidades de la función valor-acción. Si las probabilidades son altas se realiza explotación en cambio si son bajas se realiza exploración.

2.2. Aprendizaje por refuerzo interactivo

El aprendizaje por refuerzo interactivo o IRL (Interactive Reinforcement Learning del inglés) es un método utilizado para mejorar el tiempo de aprendizaje del agente, esto se logra incluyendo agentes externos (entrenadores), los cuales pueden ser humanos u otros agentes artificiales.

Dentro del IRL existen 2 formas de relación entre el entrenador y el agente, la primera el entrenador interfiere cambiando la recompensa asignada por el entorno o confirmándola como se puede apreciar en la Figura 2.2, y la segunda, el entrenador ayuda en la toma de decisión al agente como se puede ver en la Figura 2.3, ya que éste sabe cómo actuar frente al estado actual (Cruz et al., 2016).

También existen distintas formas de ayudar al agente a aprender, esto nos lleva a otros problemas ya que si el entrenador ayuda demasiado al agente, este último no aprende debido a que descansa la mayor parte del tiempo en las decisiones del agente externo (Taylor et al., 2014). Se debe considerar la calidad de las decisiones del agente externo para determinar si el aprendizaje sigue mejorando, dado que el agente externo también podría cometer errores(Cruz et al., 2016) (Sutton and Barto, 1998).

2.2.1. Enseñar con un presupuesto

El entrenador posee un número limitado de oportunidades de ayudar al agente, debido a que las personas no tienen paciencia infinita y en algún momento dejarán de enseñar. Por lo que se debe crear un modo de asignación de oportunidades

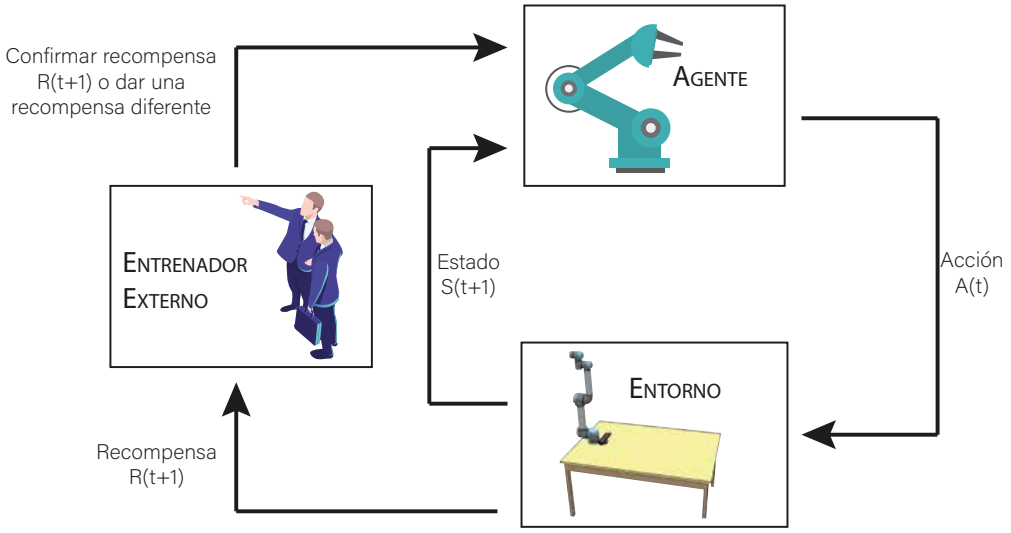

Figura 2.2: Interacción del entrenador en la política de recompensa entre el agente y el entorno, el entrenador puede confirmar o cambiar la recompensa entregada por el entorno.

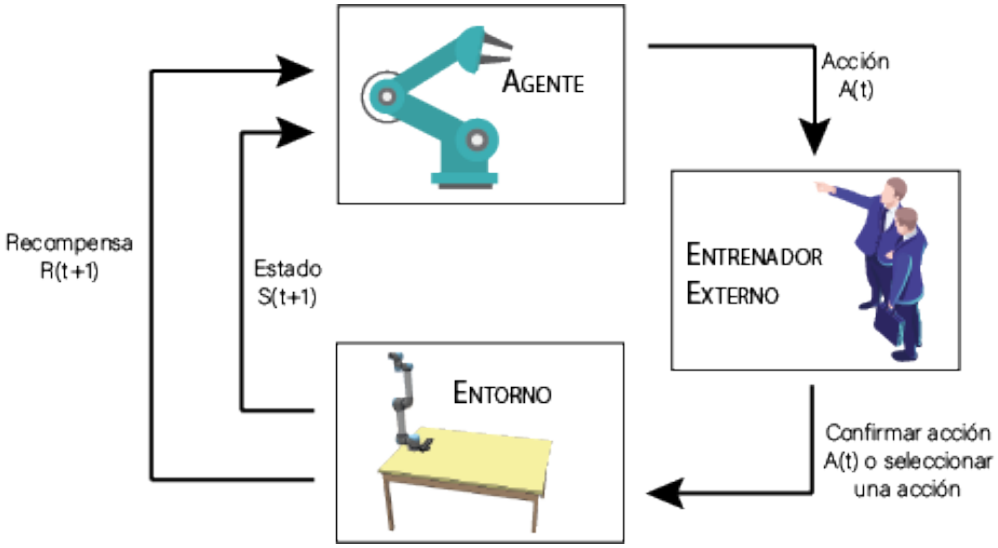

Figura 2.3: Interacción del entrenador en la política de acción entre el agente y el entorno, el entrenador puede cambiar la acción tomada por el agente o asiste al agente en la selección de la acción. El entrenador posee conocimiento del entorno y la tarea a realizar.

de ayuda en la politica de acción, los cuales se nombran a continuación y fueron propuestos por (Taylor et al., 2014).

- **Asesoramiento temprano:** El entrenador ayuda un limitado número de veces al agente sólo en la etapa temprana del aprendizaje, cuando el agente no entienda aún como funciona el entorno.
- **Consejos alternos:** En esta forma de interacción el entrenador ayuda al agente cada cierta cantidad de estados, para que el agente pueda explorar el entorno antes de que el entrenador gaste su limitado número de interferencias.
- **Asesoramiento de importancia:** Esta forma de interacción está basada en la importancia de la decisión suponiendo que no todas la decisión poseen la misma importancia, por ejemplo cruzar o no la calle es una decisión importante, ya que puede venir un auto y en estos casos actuaría el entrenador para ayudar al agente, y no en decisiones de poca importancia como detenerse a mirar una vitrina (tienda). El entrenador debe saber diferenciar la importancia de la decisión observando el estado actual.
- **Corrección de errores:** Esta forma de interacción parte con un intercambio de información entre el agente y el entrenador antes de llevar a cabo la acción, para que el entrenador pueda reconocer si la decisión es errónea o no y en caso de ser errónea el entrenador interfiera y cambiaria la acción a realizar.
- **Asesoramiento predictivo:** Esta forma de interacción genera un modelo predictivo de las acciones que tomará el agente, si las predicciones funcionan correctamente no es necesario una comunicación entre el entrenador y el agente como en la interacción de corrección de errores. Si el modelo predictivo no funciona correctamente el entrenador pierde oportunidades de ayudar al agente dado que posee oportunidades limitadas.

2.3. Redes neuronales profundas

Las redes neuronales profundas o DNN (Deep Neural Networks según sus siglas en inglés) son redes neuronales que tienen múltiples capas entre la capa de salida y la de la entrada (Hinton et al., 2012). Considerando la problemática de la investigación donde nuestra entrada sensorial es la visión del robot (imágenes) se estudian las redes neuronales convolucionales o CNN (Convolutional Neural Network según sus siglas en inglés) las cuales son un tipo de red neuronal que se diseñaron específi-

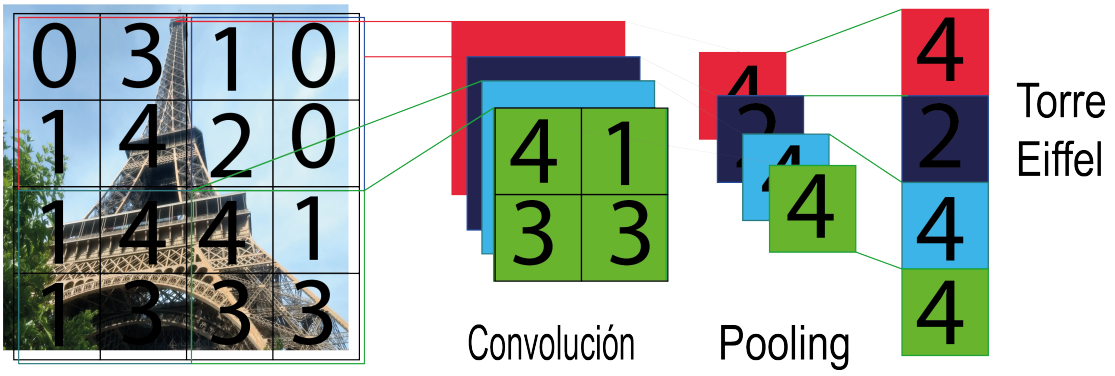

Figura 2.4: Ejemplo de una CNN, tiene una capa de entrada la cual pasa por una convolución y la imagen de entrada pasa a ser cuadrantes de un tamaño inferior y luego el pooling (en este caso Max-pooling).

camente para encontrar patrones en imágenes, este diseño se basó en el perceptrón multicapa y en la corteza visual de algunos animales (Haykin et al., 2009). La arquitectura de una CNN permite que una imagen de alta resolución sea tratada de tal manera que ésta cambia su tamaño (amplitud y profundidad) a través de filtros y muestreos (Krizhevsky et al., 2012). Este tratamiento hace que la red neuronal no genere tantas conexiones y el tiempo que tarda en encontrar patrones importantes sea menor en comparación con una red neuronal común como el perceptrón multicapa.

La estructura básica de la CNN se compone de la convolución o extracción de características la cual consiste en evaluar cuadrantes (pixel por pixel) en busca de características relevantes. Al encontrar un patrón éste se extrae y se baja la relevancia del cuadrante y se siguen buscando patrones, todos los cuadrantes son de igual tamaño, como se muestra en la figura 2.4. El otro componente básico de la CNN es la agrupación (pooling en inglés) la cual resume la información de los cuadrantes disminuyendo aún más el tamaño de la imagen (Krizhevsky et al., 2012).

2.4. Aprendizaje por refuerzo profundo

La teoría de DeepRL viene dada por la teoría del aprendizaje por refuerzo (RL) y las redes neuronales profundas o DNN. Esta complementación permite potenciar RL el cual busca una función valor-acción (2.6) óptima dado un estado (s) y una acción (a), limitada por la probabilidad de una acción dado un estado (política π),

para maximizar las recompensa (r) actual y futura en el tiempo (t) con una taza de descuento (γ) entre 0 y 1, esta función se aproxima con las DNN y permite trabajar con espacios de observación de alta dimensión (Gu et al., 2016). Como por ejemplo píxeles de una imagen, más específicamente, se utilizan arquitecturas como la CNN la cual detecta patrones relevantes en una imagen, estos patrones determinan un estado en específico y con esos patrones la CNN aproxima la función de valor de manera que el agente maximice su recompensa futura (Mnih et al., 2015).

$$Q^*(s,a) = \max_{\pi} \mathbb{E}[r_i + \gamma r_{t+1} + \gamma^2 r_{t+2} + ...|S_t = s, a_t = a, \pi] \tag{2.6}$$

Algoritmos como Deep Q-lerning (Mnih et al., 2013) en su estructura incorporan variantes de Q-lerning para estabilizar la función valor-acción. Una de estas variantes es incorporar una memoria que almacena información de estados pasados, con estos datos la DNN entrena, logrando que el agente aprenda de manera equitativa (Adam et al., 2012). Esta memoria tiene una dimensión finita y cuando es superada sobrescribe la información más antigua, cuando la red entrena, selecciona de manera aleatoria información de estados pasados, es importante que la información almacenada sea uniforme para que la DNN no caiga en mínimos locales. Un punto importante en el proceso de almacenar estados pasado es el tener un correcto balance entre exploración y explotación.

Capítulo 3

Escenario experimental

El escenario experimental será desarrollado en un software de simulación de robots llamado V-REP (Rohmer et al., 2013), en su versión edu V3.6.3 el cual nos entrega herramientas y objetos para generar un escenario y poder entrenar al agente. El agente deberá completar su objetivo el cual será clasificar algunas figuras geométricas por su color y derivarlas (moverlas) a una ubicación designada, y así optimizar la recompensa obtenida.

Se entrenan 3 agentes de manera distinta en el mismo escenario para verificar el impacto de la metodología propuesta agent-IDeepRL y human-IDeepRL vs DeepRL autónoma y generar una comparativa entre las metodologías de aprendizaje.

3.1. Escenario

V-REP cuenta con diferentes elementos para crear el escenario (robots, mesas, sillas y más). Para la creación del escenario se utilizaron tres mesas, dos para dejar los objetos (figuras de colores) una vez organizados y una para decidir cual objeto tomar. Además se usó un brazo robótico manipulador con 7 grados de libertad que cuenta con 6 ejes y un grip de ventosa. Junto con las mesas y el brazo robótico, nuestro escenario contiene una mesa más alta que las demas la cual es utilizada para colocar el brazo robótico y un sensor de profundidad que cuenta con una cámara RGB-D. Para la simulación de la ropa se utilizaron tres objetos (cubos, cilindros y discos) de diferentes colores (ver Figura 3.1), los s objetos representan diferentes tipos de prendas, y los colores si está sucia o limpia, el escenario completo se muestra en las figuras 3.2 y 3.3.

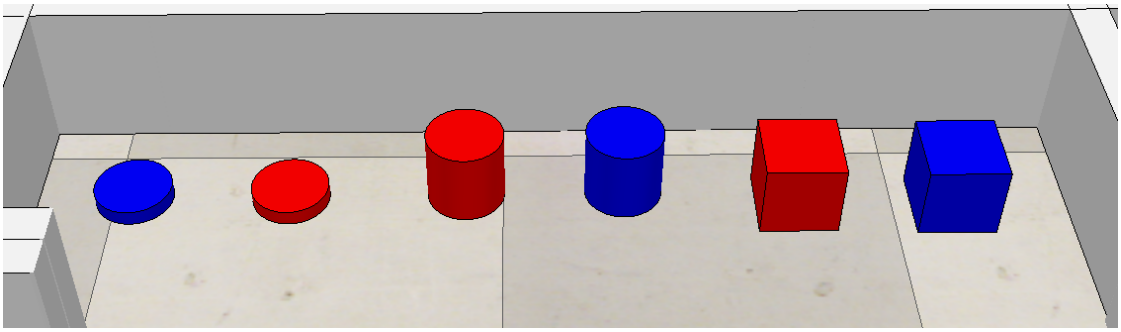

Figura 3.1: Objetos simulados en V-rep, son 3 tipos de objetos (cubos, cilindros y discos) que representan diferentes prendas de vestir, el color representa el estado de la ropa (sucia o limpia), el agente tiene que detectar estas diferencias y clasificarlas de manera correcta.

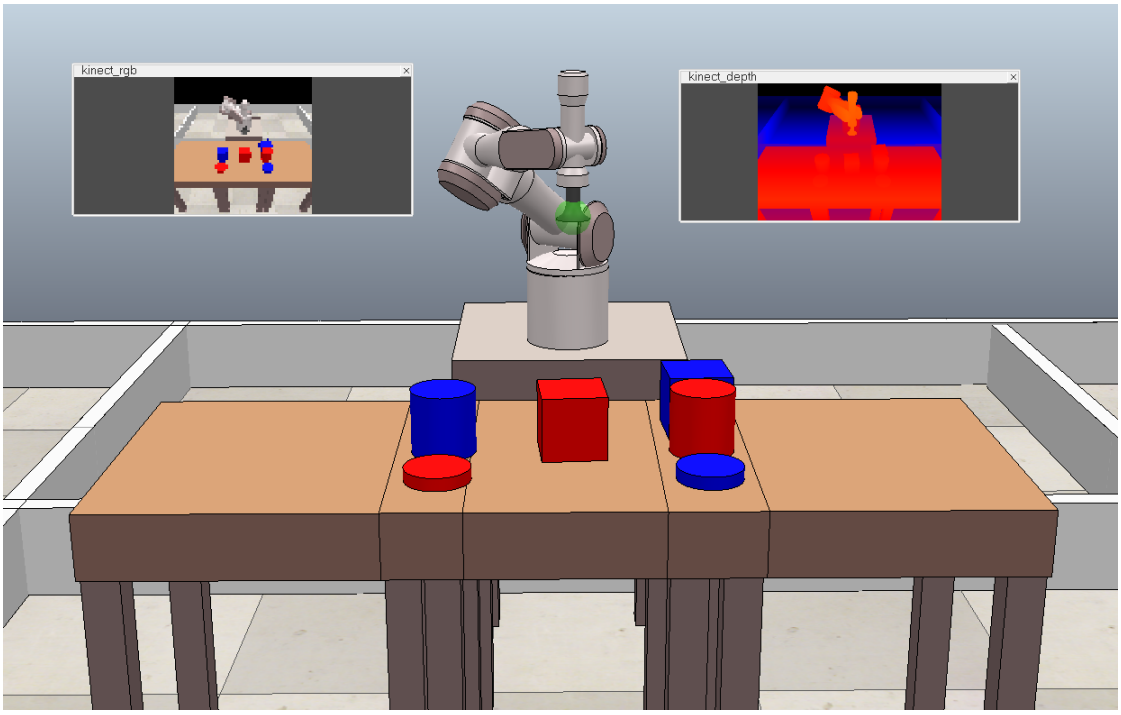

Figura 3.2: Escenario experimental desde una perspectiva frontal, en la imagen se puede observar que en la mesa central se encuentran los diferentes objetos a clasificar, al momento de iniciar o reiniciar el escenario los objetos solo se posicionan en la mesa central. Las mesas laterales se utilizan para dejar los objetos, cuando un objeto es colocado en las mesas laterales desaparece y queda clasificado.

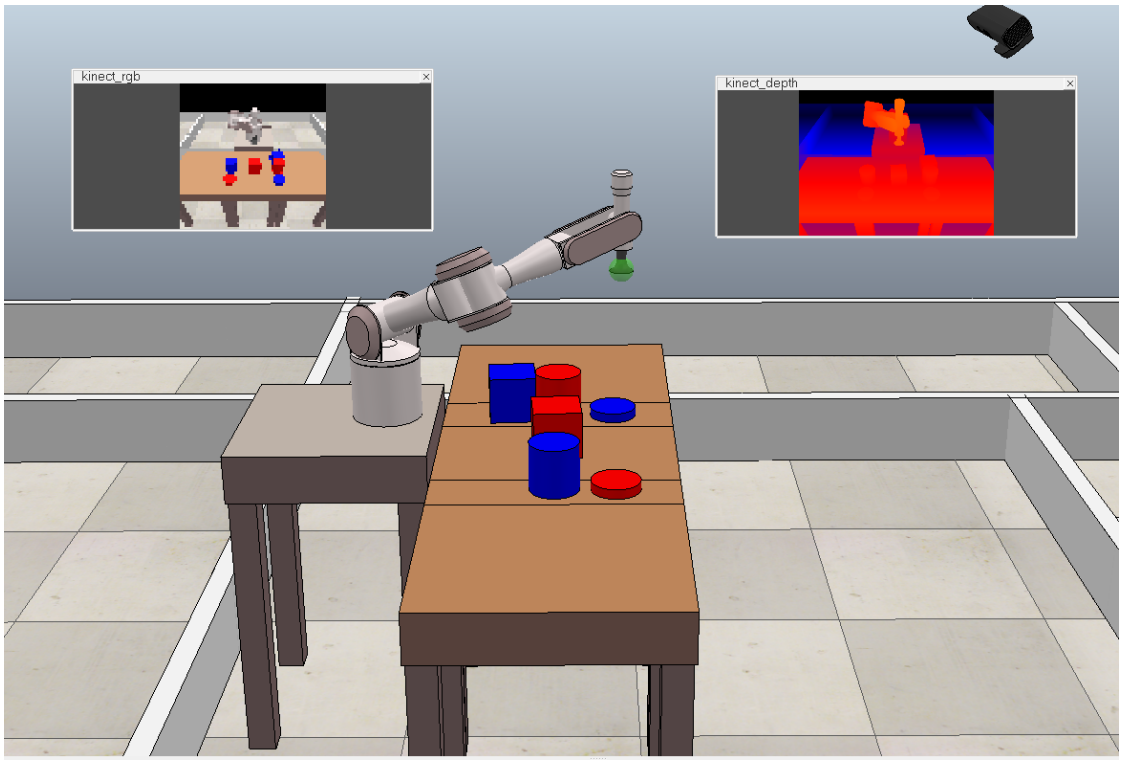

Figura 3.3: Escenario experimental desde una perspectiva lateral, se observa la distribución de las mesas en el escenario, la distancia del brazo robótico y el sensor Kinect. Los recuadros en las esquinas superiores son las imágenes captadas por el sensor Kinect, la imagen que procesa la red neuronal es la que se encuentra a la izquierda.

Para manipular el entorno simulado, crear los agentes y los experimentos se utiliza el lenguaje de programación Python versión 3.7. La unión de V-REP y Python se realiza mediante una librería de V-REP (Remote API) y con ésta se pueden obtener parámetros de los objetos, sensores, robot, entre otros. Por ejemplo, V-REP es capaz de entregar la posición y orientación de un objeto. En este caso el control del brazo robótico es a través de un target y la trayectoria hasta el target es calculada a través de cinemática inversa (inverse kinematics del inglés) la cual nos ayuda en el movimiento del brazo y sus ejes, el robot tomará las figuras con un agarre de ventosa para luego transportarlos a su lugar correspondiente.

Los estados son revisados mediante el sensor kinect el cual entrega la imagen actual del brazo y su entorno.

3.2. Restricciones (acotaciones o límites)

La tarea se tomará como completa si el robot es capaz de clasificar y mover las figuras geométricas a sus respectivos lugares (rojo izquierda del robot y azul derecha del robot), la ropa será simulada por figuras geométricas (cilindro, disco y cuadrado) de distintos colores (rojo y azul). No se utilizaran otros colores distintos de azul y rojo, para la tarea de tomar el objeto no se utilizar una mano robótica sino sólo un grip de ventosa. El robot no busca el objeto que clasificará sino que clasifica el que levanta, se tienen solo 2 figuras geométricas de cada tipo y cada uno de un color.

Al utilizar inverse kinematics en el simulador V-REP, el brazo puede accidentalmente botar algunos objetos que estén sobre la mesa, o al momento de mover los objetos éstos pueden botar a otros. Se utilizo un entorno donde variaba la posición de los objetos pero no existe una interferencia externa distinta a que se caiga un objeto por causa del movimiento del brazo, como puede ser la interferencia de un humano en el trayecto del robot.

3.3. Definición del proceso de decisión de Markov

Para modelar el problema se utilizó como referencia una representación a través de un MDP. En un MDP las accciones del agente influyen directamente al entorno y el agente aprende mediante la recompensa y los nuevos estados. En esta sección se definirán las acciones, estados y recompensas.

3.3.1. Acciones

Las acciones del agente independientemente del tipo de figura (cubo, disco o cilindro) son 4. Estas acciones son realizables de manera autónoma o con el asesoramiento de otro agente. Las acciones son las siguientes:

- **Tomar objeto:** El agente agarra con el grip de ventosa uno de los objetos aleatoriamente. Si el brazo esta en uno de los dos lados con un objeto tomado este vuelve a la posición del objeto cuando estaba en la mesa central.

- **Mover derecha:** El brazo se mueve desde cualquier posición a la mesa derecha, si se encuentra en esta posición el brazo se mantiene en la misma posición.
- **Mover izquierda:** El brazo se mueve desde cualquier posición a la mesa izquierda, si se encuentra en esta posición el brazo se mantiene en la misma posición.
- **Soltar:** Suelta el objeto. Si el brazo tiene un objeto en el grip y se encuentra en una de las mesas laterales, el brazo baja y deja el objeto. En caso de que el brazo este en la mesa central y se elije la opción soltar, el brazo sólo mantendrá la posición.

Por ejemplo para completar un episodio, el cual consta en clasificar seis objetos ubicados en la mesa central, de los cuales 3 son de color azul y 3 de color rojo. El episodio termina en 18 pasos, los cuales serian los siguientes, tomar objeto, si es rojo mover a la izquierda, soltar objeto, tomar objeto, si es azul mover a la derecha, soltar objeto, tomar objeto, si es azul mover a la derecha, soltar objeto, tomar objeto, si es rojo mover a la izquierda, soltar objeto, tomar objeto, si es rojo mover a la izquierda, soltar objeto, tomar objeto, si es azul mover a la derecha, soltar objeto, si completa la clasificación de todos los objetos de forma correcta termina el episodio, en caso de que clasifique mal un objeto el episodio termina automáticamente.

3.3.2. Estados

El agente representará los estados a través de imágenes entregadas por una cámara, la imagen es procesada por una CNN la cual detecta patrones relevantes en la imagen. Estos patrones son los utilizados para diferenciar un estado de otro, la red neuronal debería identificar parámetros como los que se muestran a continuación:

- **PosicionBrazo:** Localización del brazo (target o ventosa) en el espacio.
- **posicionObjeto:** El objeto se puede encontrar en la mesa central, mesa para objetos sucios o mesa para objetos limpios.
- **EstadoVentosa:** La ventosa se encuentra con un objeto tomado o sin un objeto.
- **EstadoObjeto:** Estado en el que se encuentra el objeto (correctamente clasificado o incorrectamente).

La CNN al identificar el estado y retorna una acción. Dado los parámetros mencionados anteriormente una forma de diferenciar un estado es el siguiente:

$$S_t =< PosicionBrazo, posicionObjeto, estadoVentosa, estadoObjeto > \quad (3.1)$$

Dentro de los múltiples posibles estado existen 2 estados finales, el exitoso, debido a que el objeto es movido a la posición correcta, como muestra la ecuación (3.2) y el erróneo en el cual debido a que el agente falló al clasificar y movió un objeto a un área errónea o que no puede finalizar la tarea, sea por que no logra tomar el objeto o supero el numero de movimientos.

$$S_f =< Derecha, mesaObjetosLimpios, sinObjeto, Correcto > \quad (3.2)$$

3.3.3. Recompensas

Existen diferentes tareas para finalizar un episodio y si bien el completar la tarea de manera correcta significa clasificar todos los objetos, el hecho de clasificar un objeto correctamente es considerado parte del objetivo. Por esta razón al completar el objetivo de clasificar todos los objetos se entrega una recompensa de 1 y clasificar correctamente un objeto una recompensa de 0.4. Por ejemplo, si el agente clasifica 6 figuras de manera correcta, las primeras 5 le entregaran una recompensa de 0.4 cada una y la última una recompensa de 1, esto hace una recompensa total de 3. Por otra parte es necesario que el agente aprenda a clasificar en la menor cantidad de pasos posibles por esta razón se penalizará con una recompensa negativa de -0.01 por cada paso cuando la cantidad de pasos sea mayor a 18 (que es mínimo número de acciones para completar la tarea satisfactoriamente), por ejemplo, si el agente realiza la tarea completa de manera exitosa en 20 pasos, su recompensa total será 2.98, considerando 3 de recompensa menos 0.01 x 2 pasos sobre las 18 acciones. Por último, al clasificar de manera incorrecta se le da una recompensa de -1 y se finaliza el episodio.

La función de recompensa se puede observar de manera mas organizada en la siguiente ecuación (3.3)

$$r(s) = \begin{cases} 1 & \text{si todos los objetos estan organizados} \\ 0.4 & \text{si el objeto esta bien clasificado} \\ -1 & \text{si el objeto fue mal clasificado} \\ -0.01 & \text{si pasos } > 18 \end{cases} \tag{3.3}$$

Capítulo 4

Diseño e implementación del agente

En este capítulo se describe el diseño de los agentes, éstos se diferencian en el enfoque del aprendizaje. Los agentes implementados son los siguientes:

- **Agente autónomo:** El agente al iniciar el algoritmo no cuenta con información útil del entorno ni de su objetivo.
- **Agente interactivo:** Un maestro entrena al agente cuando éste aún no conoce el entorno. Se utiliza asesoramiento temprano (Taylor et al., 2014), con este enfoque se implementan dos maestros, el primer maestro es un agente previamente entrenado (agent-IDeepRL) y el segundo maestro es un humano con conocimiento del entorno (human-IDeepRL).

La base entre los agentes es la misma, estos perciben toda la información del entorno mediante una representación visual (Desai and Banerjee, 2017) que es entregada luego que el agente realice una acción.

La representación visual es procesada por una red neuronal convolucional que estima los valores Q, este algoritmo (Deep Q-learning) permite que los agentes interactúen mediante la experiencia adquirida por acciones pasadas y al mezclarse con redes neuronales (aproximadores de funciones) permite generalizar estados y aplicar Q-learning en espacios de observación continuos.

Para guardar la experiencias pasadas se implementa una técnica llamada experience replay (Adam et al., 2012). Este algoritmo guarda en una memoria información útil (experiencia) con la que se entrena el agente.

Si las decisiones son tomadas solamente por la red neuronal, la cual procesa la representación visual y la expresa en valores Q, el hecho de que las decisiones sólo sean tomadas por la red hace que no exista un balance óptimo entre exploración y explotación. Para evitar esto se utilizó la metodología ϵ-greedy que posee un factor de exploración ϵ, esto permite que en algunas ocasiones el agente tome decisiones que antes nunca intento debido a la convergencia de la red neuronal hacia el óptimo.

4.1. Enfoque interactivo

Como se mencionó anteriormente se entrenaron agentes con un entrenador externo el cual puede ser humano u otro agente artificial, para la tarea que se realizó se aplicó policy shaping, que consta de la interacción del maestro en la toma de decisión o acción, es decir la acción que realizará el robot como se aprecian en la figura 2.3. Dentro del entrenamiento interactivo existen distintas técnicas de enseñanza, la que se aplica en nuestro caso es enseñar con un presupuesto (P) de manera temprana (Taylor et al., 2014) como se aprecia en el algoritmo 4.1, el algoritmo llena la memoria con 1000 transiciones, en las primeras 100 transiciones el maestro es quien toma las decisiones y el resto son acciones aleatorias. Cuando se decide por una acción el agente la realiza sobre el entorno, luego se recibe una recompensa, se observa el nuevo estado (Taylor et al., 2014) y se guarda la información en la memoria. En caso de que el estado sea un estado final o que se superen los 250 pasos, se reinicia el episodio.

Esta técnica intenta reducir el tiempo que tarda el agente en entender el entorno considerando que un maestro tiene un conocimiento mayor de la problemática a resolver y del funcionamiento del entorno en comparación al agente. Además, no es factible que el maestro esté disponible todo el tiempo enseñando al agente.

4.2. Representación visual

Para la representación visual se utilizó como guía el algoritmo Deep Q-learning (Mnih et al., 2013), este algoritmo es similar al de Q-learning pero utiliza CNN como aproximación de la función-Q, incorpora una memoria, con datos pasados, de la cual se sacan lotes de menor tamaño para entrenar a la red (Roderick et al., 2017).

Algorithm 4.1 Algoritmo de pre entrenamiento y llenado de la memoria de entranamiento incluyendo interactive feedback.

```
Inicializar memoria M
Agente observa el estado inicial s_0
while len(M) ≤ 1000 do
    if interactivo es usado AND len(M) > 900 then
        Obtener acción a_t del entrenador
    else
        tomar una acción random a_t
    end if
    Realizar acción a_t
    Observar r_t y Siguiente estado s_{t+1}
    M.add(< s_t, a_t, r_t, s_{t+1} >)
    if s_t es terminal OR time-pasos > 250 then
        Reiniciar episodio
    end if
end while
```

Nuestra arquitectura se compone de una imagen de tamaño 64x64 pixeles, dada la problemática de clasificación se mantiene en escala RGB para que la red aprenda a diferenciar colores, la primera capa convolucional tiene 4 filtros de tamaño 8x8, luego viene un maxPooling de tamaño 2x2, la segunda capa convolucional tiene 8 filtros de tamaño 4x4 y le sigue un maxPooling de igual tamaño que el anterior, luego tenemos la última capa convolucional de 16 filtros de tamaño 2x2. Finalmente tenemos la última capa que está totalmente conectada y posee 256 neuronas, la cual está conectada a la capa de salida que tiene 4 posibles acciones, la arquitectura de la red se puede apreciar en la figura 4.1.

Dada la característica de la hipótesis, que a grandes rasgos es comparar metodologías de aprendizaje, es necesario destacar que todos los agentes tienen la misma arquitectura para que estén en igualdad de condiciones. Además, se debe aclarar que pueden existir mejores arquitecturas que optimicen el aprendizaje en términos de tiempo pero el foco de la tesis es la comparacion de metodologias y no necesariamente el encontrar la arquitectura óptima.

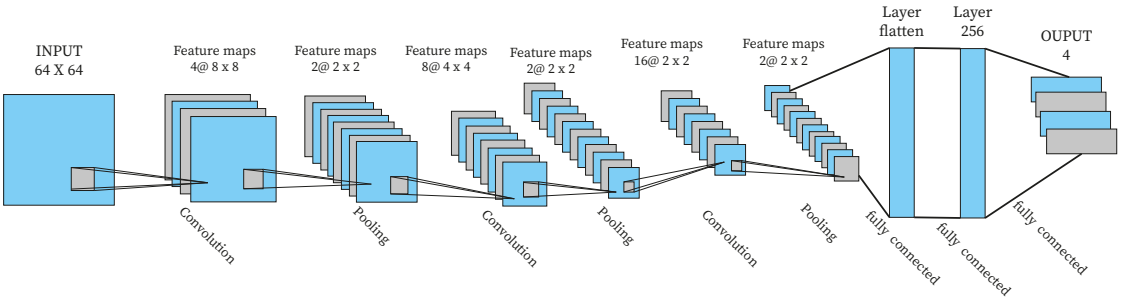

Figura 4.1: Arquitectura de la red neuronal utilizada para procesar la imagen obtenida del entorno, posee una parte convolución la cual se encarga de encontrar patrones relevantes en la imagen y una parte clasificatoria que se encarga procesar los patrones y los transforma en acciones realizadas sobre el entorno.

4.3. Representación continua

Debido a la complejidad de trabajar con imágenes y con objetos con distintas características en un entorno dinámico, no es factible generar una tabla con los estados y sus acciones posibles, es por esto que en nuestra tarea se aplicó la representación continua mediante 2 formas unidas. La primera es mediante la red neuronal, la cual es un aproximador de la función Q(St,At) la cual nos permite generalizar los estados, utilizar Q-learning en espacios continuos y seleccionar una acción a realizar, La segunda es mediante la técnica de experience replay (Adam et al., 2012) la cual consiste en guardar en memoria una tupla de datos importantes los cuales son estado, acción, recompensa, estado siguiente <S(t),A(t),R(t),S(t+1)>, estos datos guardados en memoria son utilizados para entrenar al agente (Sutton and Barto, 1998).

El algoritmo de aprendizaje para el agente autónomo está compuesto por 2 partes la primera consta de 1000 acciones que realiza el agente de manera aleatoria para conocer el entorno, la segunda parte es cuando el agente comienza a entrenar utilizando la política ϵ-greedy y el agente después de cada acción seleccionada por ϵ-greedy (acción aleatorio o acción tomada por la CNN) entrena a la CNN con la tupla estado, acción, recompensa, estado siguiente <S(t), A(t), R(t), S(t+1)> guardados en memoria, luego disminuye el ϵ como se aprecia en la linea 12 del algoritmo 4.2.

Algorithm 4.2 Algoritmo de entrenamiento, poblar y extracción información desde batch memory.

Realizar algoritmo de pre entrenamiento
for cada episodio **do**
    Observar estado s_t
    repeat
        Seleccionar acción a_t usando ϵ-greedy
        Realizar acción a_t
        Observar r_t y siguiente estado s_{t+1}
        M.add($< s_t, a_t, r_t, s_{t+1} >$)
        Poblar de manera random B desde M
        Entrenar CNN usando datos de B
        $s_t \leftarrow s_{t+1}$
        $\epsilon \leftarrow \epsilon * \epsilon$_decay
    until s_t es terminal OR time-pasos > 250
end for

Capítulo 5

Comparación y análisis de resultados

En esta sección mostramos los resultado de los experimentos, el objetivo es comparar a los diferentes agentes y determinar que agente es mejor en la tarea domestica descrita con anterioridad. Se realizaron experimentos con tres tipos de agentes: DeepRL, agent-IDeepRL, y human-IDeepRL, lo cuales consisten en aprendizaje autónomo del entorno y la tarea, un maestro agente entrenado el cual ayuda al agente alumno a entender entorno y la tarea, y un maestro humano el cual ayuda al agente alumno a entender la tarea y el entorno, respectivamente. Todos los agentes se desempeñaron en el mismo entorno, resolvieron la misma tarea y estuvieron bajo los mismos hiperparámetros (Tabla 5.1).

Tabla 5.1: Parámetros globales

	Epsilon inicial(ϵ)	***Epsilon decay***	***Aprendizaje***	***Episodios***
Valor	1	0.9995	0.001	350

Para el primer experimento utilizamos al agente con aprendizaje autónomo (DeepRL), que no posee ninguna información del entorno ni del objetivo. En sus primeros episodios repite muchas acciones y supera considerablemente el contador de 18 pasos que corresponde a la mínima cantidad de acciones para finalizar la tarea exitósamente. Alrededor del episodio 25 llega a la recompensa mínima de -3 y cercano al episodio 100 comienza a obtener recompensa acumulada positiva. El total desconocimiento del entorno y no contar con información adicional del objetivo aumenta la cantidad de errores en la etapa inicial, se puede decir que el agente comienza a

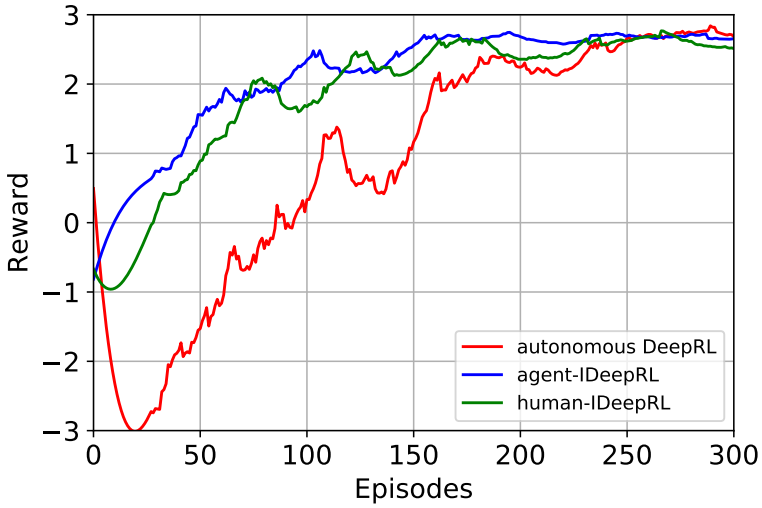

Figura 5.1: Gráfico comparativo con los tres métodos propuestos. La línea roja representa el agente autónomo (DeepRL) que tiene que descubrir el entorno sin ninguna ayuda. Las líneas azul y verde son los agentes con un entrenador externo, un asesor artificial (agent-IDeepRL) y un asesor humano (human-IDeepRL), respectivamente.

completar la tarea de manera reiterada en el episodio 250 como se aprecia en la gráfica Figura 5.1 (color rojo).

Para el resto de experimentos se utilizó la metodología IDeepRL con dos tipos de maestros los cuales son maestro humano (human-IDeepRL) y un maestro agente entrenado (agent-IDeepRL). Ambos maestros cuentan con conocimiento del entorno y de su objetivo, los maestros sólo ayudan al agente aprendiz en una etapa inicial (100 episodios) brindando información del entorno al agente. Los resultados obtenido son muy similares entre ellos, como se puede ver en la gráfica de la Figura 5.1 (color verde para human-IDeepRL y azul para agent-IDeepRL).

Al momento de comparar los agentes nos damos cuenta que existe una gran diferencia entre las metodologías de aprendizaje (Figura 5.1), los agentes que obtienen ayuda de un maestro, sin importar si es agent-IDeepRL o human-IDeepRL, cometen menos errores desde un comienzo y aprende la tarea en menos tiempo (episodios), el agente autónomo posee un número mayor de errores Porqué en un comienzo intenta comprender cómo funciona el entorno y el objetivo que debe cumplir, esto no se observa en la metodología IDeepRL puesto que el maestro le entrega información útil del entorno en su etapa inicial. Debido a que existen distintas formas de aplicar IDeepRL es que sólo nos referimos a un entrenador con

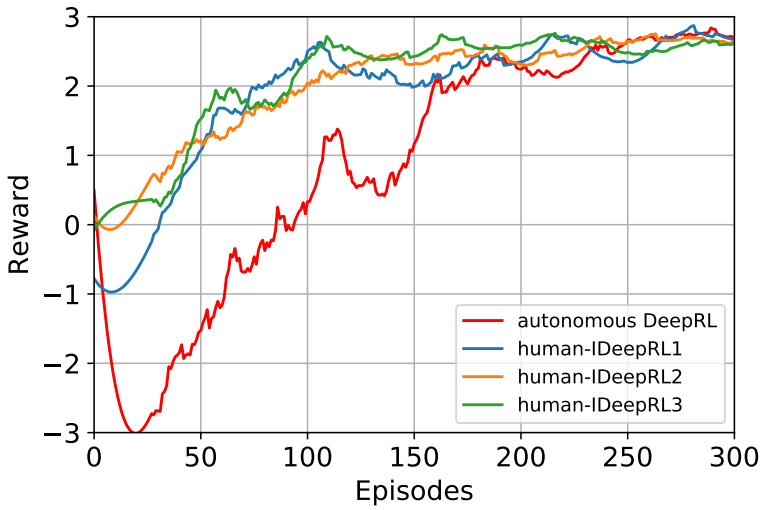

Figura 5.2: Gráfico comparativo con agentes entrenados por diferentes personas usando IDeepRL humana. El entrenador tiene un presupuesto de 100 acciones para asesorar al agente alumno. Aunque cada persona tiene inicialmente una comprensión diferente del entorno teniendo en cuenta los objetivos y los posibles movimientos, todos los agentes convergen en un comportamiento similar en términos de recompensa recolectada.

oportunidades finitas de ayuda, consecutivas y al comienzo del entrenamiento del agente, a esta estrategia se le llama asesoramiento temprano (Early advising).

Ya que trabajamos con maestros de distintos tipos, humanos y agentes entrenados es que nos preguntamos cómo seleccionar a un buen maestro, debido a los resultados obtenidos y mostrados en la Figura 5.2, se logra apreciar las diferentes curvas de aprendizaje, cada una es diferente ya que sus maestros humanos enseñan de manera diferente y aun así se obtienen una curva de aprendizaje mejor que un agente autónomo como se aprecia en la Figura 5.2 línea de color rojo y agentes con entrenador humano grafica de color verde, amarilla y celeste. En la estrategia Human-IDeepRL participaron 11 personas en los experimentos, de los cuales 4 hombres y 7 mujeres con edades entre los 16 y 24 años de edad.

Durante los experimentos, se explicó a los participantes cómo ayudar al robot a completar la tarea dando consejos utilizando el mismo guion para todos ellos. Debido a que el maestro tiene un presupuesto de 100 acciones para asesorar, la retroalimentación interactiva se consume dentro de los primeros 6 episodios, teniendo en cuenta que la cantidad mínima de acciones para completar un episodio son 18 acciones. Incluso con una cantidad tan pequeña de retroalimentación, el

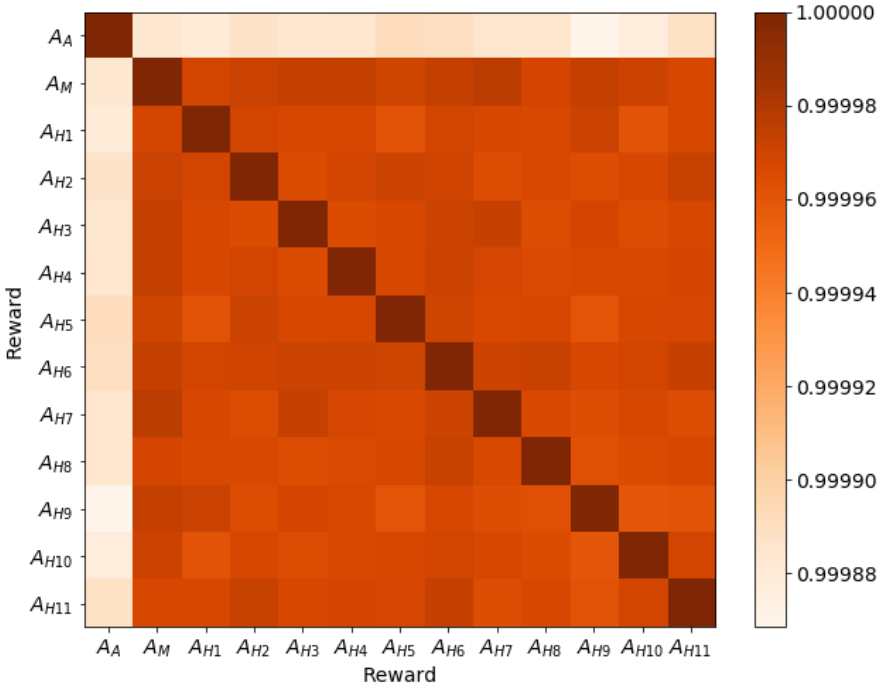

Figura 5.3: Gráfico comparativo de la correlación de Pearson de todos los resultados obtenidos en las 3 metodologías, la única metodología que se aleja del comportamiento es el DeepRL en comparacion con las otras dos metodologias agent-IDeepRL y human-IDeepRL.

alumno recibe un conocimiento importante del asesor que se complementa con el método de repetición de la experiencia (Experience replay).

Como se aprecia en los gráficos mostrados anteriormente se logra observar un comportamiento similar entre los agentes human-IDeepRL y agent-IDeepRL, en cambio se ve un comportamiento diferente en el agente DeepRL. Utilizamos el coeficiente de correlación de Pearson (Fig. 5.3) para comparar la relación entre los resultados de todos los experimentos incluyendo los 11 entrenadores humanos, el entrenador artificial y el agente autónomo. Nos percatamos de que efectivamente DeepRL tiene un comportamiento diferente a los otros agentes pero su coeficiente de correlación sigue siendo un valor muy cercano a 1 por lo que se podría decir que al aplicar IDeepRL cambia la magnitud de la curva de aprendizaje pero el comportamiento de los agentes es similar. Además al comparar los resultados obtenidos en agentes human-IDeepRL y agent-IDeepRL se desempeñan de una manera similar tanto en magnitud y comportamiento. En resumen IDeepRL acelera la magnitud del aprendizaje pero no modifica el comportamiento del agente.

Como análisis final se comprueba que con la ayuda de un maestro en DeepRL el tiempo de aprendizaje es mucho menor, el error cometido al principio, se ve reducido con la ayuda de un maestro que tenga conocimientos del entorno y objetivo.

Cabe destacar que desde el episodio 150 aproximadamente el agente realiza acciones basadas en su aprendizaje o entrenamiento, ya que el valor de ϵ de la política ϵ-greedy en este punto solo realiza un 1 % de acciones aleatorias.

Otro punto a tener en cuenta es que debido a que el robot con sus movimientos puede botar algún otro objeto que esté sobre la mesa, es que sus recompensas fluctúan entre 2.5 y 3 lo cual es muy difícil de controlar debido a que el brazo del robot se mueve gracias a inverse kinematics por ende no controlamos sus movimientos completamente.

Capítulo 6

Conclusion

En este trabajo presentamos una comparación entre dos modelos, por un lado el clásico DeepRL el cual tiene que entender su entorno por sí solo y el IDeepRL el cual cuenta con un entrenador que en este caso le entrega asesoramiento del entorno mediante una retroalimentación la cual es entregada en los primeros episodios.

En términos de resultados la metodología IDeepRL es notoriamente superior que DeepRL, en tareas más complejas y que requieran una mayor cantidad de tiempo entrenarlas, contar con un entrenador, el cual le entrega información útil para comprender el entorno, trae grandes beneficios en términos de tiempo y recursos. Pero también puede limitar al aprendiz, enseñando una estrategia que no necesariamente es la óptima. Para seleccionar un buen maestro es necesario tener en cuenta que un agente que obtiene los mejores resultados, en términos de recompensa acumulada, no es necesariamente el mejor candidato (Cruz et al., 2018). Más bien un buen maestro podría ser uno con una pequeña desviación estándar de los resultados obtenidos (recompensa acumulada).

Con esta investigación aprendimos sobre DeepRL interactivo, estrategias de aprendizajes y entornos de simulación. Por medio de un software diseñamos un ambiente con los elementos necesarios para realizar los experimentos (robot, mesas, objetos), utilizando programación diseñamos diferentes algoritmos con los cuales el agente aprendió a resolver la tarea. Se realizaron diferentes experimentos para medir el rendimiento de la metodología planteada. Para finalizar, analizamos los resultados obtenidos considerando el tiempo de aprendizaje, tiempo que tarda en desarrollar la tarea y porcentaje de error.

Un punto importante a considerar es que el diseño de los agentes se podría optimi-

zar para obtener mejores resultados (rendimiento de los agentes individualmente), tanto en la parte de los hiperparámetros como la arquitectura de la CNN. Pero estos cambios no alteran los resultados de la comparación de metodologías debido a que todos los agentes se rigieron sobre la misma arquitectura.

Como trabajo futuro, consideramos el uso de diferentes tipos de entrenadores artificiales, con el fin de seleccionar un mejor maestro para nuestro problema, podría traer muchos más beneficios en términos de tiempo y recompensas que acentúan aún más el potencial de IDeepRL, además de flexibilizar la enseñanza permitiendo una mayor exploración a nuevas estrategias.

Capítulo 7

Lista de acrónimos

AI – Artificial Intelligence.

ANN – Artificial Neural Network.

IDeepRL – Interactive Deep Reinforcement Learning.

Human-IDeepRL – Human Interactive Deep Reinforcement Learning.

DeepRL – Deep Reinforcement Learning.

CNN – Convolutional Neural Network.

DNN – Deep Neural Network.

IRL – Interactive Reinforcement Learning.

MDP – Markov Decision Process.

NN – Neural Network.

RL – Reinforcement Learning.

Bibliografía

Adam, S., Busoniu, L., and Babuska, R. (2012). Experience replay for real-time reinforcement learning control. *Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on*, 42:201 – 212.

Cruz, F., Magg, S., Nagai, Y., and Wermter, S. (2018). Improving interactive reinforcement learning: What makes a good teacher? *Connection Science*, 30(3):306–325.

Cruz, F., Magg, S., Weber, C., and Wermter, S. (2016). Training agents with interactive reinforcement learning and contextual affordances. *IEEE Transactions on Cognitive and Developmental Systems*, 8(4):271–284.

Desai, N. and Banerjee, A. (2017). Deep reinforcement learning to play space invaders.

Gu, S., Lillicrap, T., Sutskever, I., and Levine, S. (2016). Continuous deep q-learning with model-based acceleration. In *International Conference on Machine Learning*, pages 2829–2838.

Haykin, S. S. et al. (2009). *Neural networks and learning machines/Simon Haykin.* New York: Prentice Hall,.

Hinton, G., Deng, L., Yu, D., Dahl, G., Mohamed, A.-r., Jaitly, N., Senior, A., Vanhoucke, V., Nguyen, P., Kingsbury, B., et al. (2012). Deep neural networks for acoustic modeling in speech recognition. *IEEE Signal processing magazine*, 29.

Krizhevsky, A., Sutskever, I., and Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105.

Mnih, V., Kavukcuoglu, K., Silver, D., Graves, A., Antonoglou, I., Wierstra, D., and Riedmiller, M. (2013). Playing atari with deep reinforcement learning. *arXiv preprint arXiv:1312.5602*.

Mnih, V., Kavukcuoglu, K., Silver, D., Rusu, A. A., Veness, J., Bellemare, M. G., Graves, A., Riedmiller, M., Fidjeland, A. K., Ostrovski, G., et al. (2015). Human-level control through deep reinforcement learning. *Nature*, 518(7540):529.

Nola, R. and Sankey, H. (2014). *Theories of scientific method: an introduction*. Routledge.

Roderick, M., MacGlashan, J., and Tellex, S. (2017). Implementing the deep q-network. *arXiv preprint arXiv:1711.07478*.

Rohmer, E., Singh, S. P., and Freese, M. (2013). V-rep: A versatile and scalable robot simulation framework. In *2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, pages 1321–1326. IEEE.

Schwab, K. (2016). *La cuarta revolución industrial*. Debate.

Sigaud, O. and Buffet, O. (2013). *Markov decision processes in artificial intelligence*. John Wiley & Sons.

Sutton, R. S. and Barto, A. G. (1998). *Reinforcement Learning: An Introduction*. Cambridge, MA, USA: Bradford Book.

Tadele, T. S., de Vries, T., and Stramigioli, S. (2014). The safety of domestic robotics: A survey of various safety-related publications. *IEEE robotics & automation magazine*, 21(3):134–142.

Taylor, M. E., Carboni, N., Fachantidis, A., Vlahavas, I., and Torrey, L. (2014). Reinforcement learning agents providing advice in complex video games. *Connection Science*, 26(1):45–63.

Tokic, M. (2010). Adaptive ε-greedy exploration in reinforcement learning based on value differences. In *Annual Conference on Artificial Intelligence*, pages 203–210. Springer.

Zeng, A., Song, S., Lee, J., Rodriguez, A., and Funkhouser, T. (2019). Tossingbot: Learning to throw arbitrary objects with residual physics. *arXiv preprint arXiv:1903.11239*.