uc3m | Universidad Carlos III de Madrid

Grado Universitario en Ingeniería Telemática

2018 - 2019

Trabajo Fin de Grado

"Aplicación de Aprendizaje por Refuerzo Adversario en Juegos de Atari"

Jesús López Baeza-Rojano

Tutor

Francisco Javier García Polo

Leganés, 2019

RESUMEN

Actualmente, la inteligencia artificial y en particular el aprendizaje por refuerzo, se encuentra en todas partes, como la conducción autónoma, la robótica social y sistemas de vigilancia. El problema de tener tantos dominios de aplicación, es que también está expuesto a multitud de ataques que pueden causar grandes problemas. Por ejemplo, en un sistema de conducción autónoma controlado por aprendizaje por refuerzo, un ataque puede causar que el coche se estrelle causando catastróficas consecuencias. Por lo tanto, el estudio de la vulnerabilidad de estos sistemas frente a ataques está recibiendo cada vez más atención dentro de la comunidad científica.

Por eso, este trabajo se centra, precisamente, en el estudio de la vulnerabilidad de estos sistemas. En particular, el objetivo principal es disminuir el rendimiento de una red neuronal profunda aplicada a juegos de Atari mediante dos nuevos tipos de ataque. En el primero, se tratará de identificar zonas/regiones de ataque, es decir, aquellas regiones donde atacar induce más rápidamente el fallo del sistema. El segundo se basará en aprendizaje por refuerzo. En este ataque se tratará de aprender una política de ataque identificado, por cada estado en el que se puede encontrar el sistema, si es mejor atacar o no atacar. En ambos casos, se obtendrán conclusiones sobre el momento exacto en el que atacar afecta más a la red, es decir, baja más su rendimiento, y sobre el número de ataques de cada aproximación, puesto que el objetivo es maximizar el daño mientras se minimiza el número de ataques.

El primer paso para poder defenderte de un ataque, es conocer cómo se puede atacar y qué tipos de ataque puede haber. Por lo tanto, este documento sienta las bases sobre posibles sistemas de defensa futuros.

Palabras Clave: Aprendizaje automático, Refuerzo, Adversario, Atari, Deep Q Learning

AGRADECIMIENTOS

En primer lugar y más importante, me gustaría agradecer a mis padres el haberme brindado la oportunidad de estudiar una ingeniería y poder haberme formado en una universidad como esta. Su apoyo ha sido esencial durante todos estos años, sobre todo cuando más lo he necesitado. Agradecerles, por estar siempre cuando tienen que estar.

Agradecer también al resto de mi familia el apoyo e ilusión que han mostrado siempre por los objetivos que he ido logrando y a que terminara la carrera, especialmente a mis abuelos.

A mis amigos y compañeros de la carrera, que me han enseñado a distribuir el tiempo de trabajo y diversión, compaginando estos dos y centrándonos al máximo cuando el momento lo requería.

Sin olvidarme de mis amigos de toda la vida, que son los que de verdad me han apoyado y me han animado siempre a sacar adelante la carrera. También me han ayudado a desconectar después de días muy intensos de estudio, lo cual es imprescindible para coger fuerzas para el siguiente objetivo.

En estos dos últimos años, me gustaría agradecer el apoyo constante y la motivación que he recibido cada día a una persona muy especial para mi, Marta. Sin ella, la recta final de la carrera, que es lo más importante y cuando más cansado estás de los esfuerzos que has hecho hasta el momento, habría sido mucho más difícil.

Por supuesto, agradecer también a los profesores que me han acompañado durante toda la carrera, su paciencia en algunos momentos, y los conocimientos que he adquirido gracias a ellos.

Por último, agradecer a mi tutor Francisco Javier García Polo y a Rubén Majadas Sanz la preocupación e interés que han mostrado por mi durante el desarrollo del proyecto.

Gracias a todos.

ÍNDICE DE CONTENIDOS

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

1. INTRODUCCIÓN

El uso de la inteligencia artificial se ha venido incrementando en los últimos años debido a su amplio abanico de aplicaciones reales, automatización de procesos, y eficacia [1].

En particular, el aprendizaje automático [2] se utiliza cada vez más en entornos reales como la medicina [3], la conducción autónoma [4], la robótica [5], la industria [6], el sistema financiero [7], etc. Cada día se descubren nuevas aplicaciones en las que el aprendizaje automático juega un papel importante, agilizando y optimizando el trabajo.

Entre las técnicas de aprendizaje automático más destacadas se encuentra el *Deep Learning* o aprendizaje profundo [8], el cual se explicará más en profundidad posteriormente en la sección 2.

Dentro del *Deep Learning,* el *Deep Reinforcement Learning* o aprendizaje por refuerzo profundo [9] se ha utilizado con éxito en multitud de entornos incluyendo video-juegos como los que se van a analizar en este proyecto. En particular, en este proyecto se van a utilizar los conocidos juegos de Atari [10]. Estos juegos de arcade y aparentemente arcaicos, son de mucha ayuda a la hora de estudiar el comportamiento de los sistemas de aprendizaje gracias a la interfaz OpenAI Gym, que permite acceder fácilmente a los controles del juego, y así poder hacer uso de las técnicas de aprendizaje automático en estos entornos.

En este trabajo nos centraremos en dos de estos juegos: Breakout [11] y Pong [12].

El Breakout consiste en controlar el movimiento de la raqueta, de izquierda a derecha, para golpear a la pelota y así derribar los ladrillos de la parte de arriba de la pantalla, obteniendo puntos por la cantidad de ladrillos destruidos. Además, según donde se encuentren estos ladrillos, se obtendrán diferentes puntuaciones, obteniendo más puntos por destruir los ladrillos de más arriba. Se dispone de varias vidas, que se van consumiendo cuando la pelota se pierde por debajo de la pantalla sin haberla golpeado. El juego se gana cuando se derriban todos los bloques.

En el caso del juego Pong, la raqueta controlada por el jugador, con movimientos arriba y abajo, se encuentra en la parte derecha de la pantalla. El objetivo de este juego es batir

a la raqueta de la izquierda marcándole la mayor cantidad de puntos posibles. El vencedor de la partida será quien antes llegue a los 21 puntos.

En estos juegos es posible entrenar una red neuronal para que dado un estado (que se corresponde con una observación o *pantallazo* del juego en un determinado instante de tiempo) ejecute la mejor acción para ese estado (Figura 1).

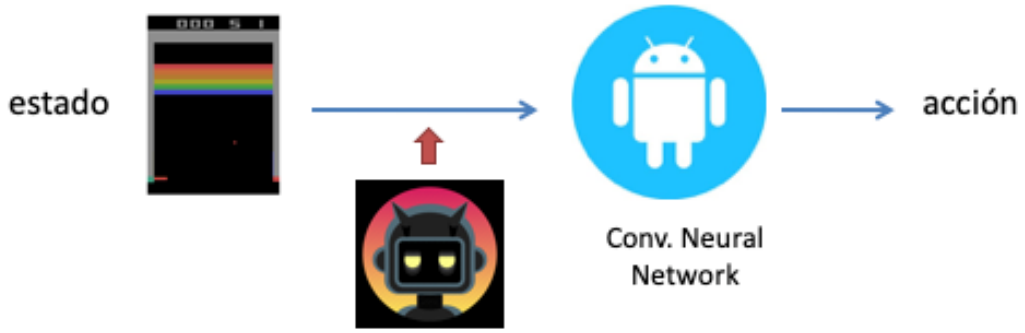


Figura 1. Una red neuronal ya entrenada percibe un estado y devuelve la acción a ejecutar. Un agente atacante puede modificar la imagen que recibe la red neuronal.

En ambos juegos el objetivo es modificar ligeramente las imágenes u observaciones que percibe la red neuronal en cada turno de juego para así inducir el fallo. Es decir, el objetivo de este trabajo es construir un agente mediante diferentes técnicas cuyo objetivo será *engañar* a la red neuronal haciéndola creer que está en una situación diferente a la que realmente se encuentra, para que de esta forma ejecute una acción *equivocada* o diferente a la que debería ejecutar.

1.1 – Motivación

Debido al auge del aprendizaje automático, y en particular al *Deep Learning* en entornos reales, existe una creciente preocupación dentro de la comunidad científica por la vulnerabilidad de este tipo de sistemas, ya que al estar altamente expuestos, corren el riesgo de poder ser atacados y presentar daños o alteraciones muy graves en sistemas automatizados no supervisados por un humano.

Por ello, son cada vez más necesarios estudios que analicen la vulnerabilidad de estos sistemas ante posibles ataques, y a su vez posibles mecanismos de defensa para garantizar la seguridad que involucre al entorno de estos sistemas automatizados.

Por otro lado, la inteligencia artificial y en particular el aprendizaje automático me ha parecido muy interesante de cara al futuro inmediato desde el segundo curso de la carrera, en el que empecé a ver las aplicaciones y la demanda que iba a tener en el futuro, en una asignatura llamada Teoría Moderna de la Detección y Estimación. En la cual, vimos cómo implementar algoritmos para sacar conclusiones sobre una gran cantidad de datos. Pudimos estimar valores futuros, y a su vez, tratar con datos reales aplicados a estos algoritmos, obteniendo conclusiones realmente veraces e interesantes.

Esta asignatura me inició en el gran mundo de la inteligencia artificial, interesándome por cada campo que hay dentro de este gigante con tanto futuro como cada día se muestra en las noticias, anunciando las grandes inversiones que multitud de empresas están haciendo en esta tecnología, para en el futuro, ahorrar costes y ganar productividad.

El hecho de hacer este proyecto y no otro, viene a raíz de una asignatura que he cursado este año en el primer cuatrimestre llamada Inteligencia Artificial en el Sector Financiero. Hasta el momento no había tenido ninguna asignatura en la que se profundizara en la inteligencia artificial. La asignatura me pareció muy interesante, y por ello me puse en contacto con el profesor para transmitirle mi interés en realizar el trabajo fin de grado con él. Me presentó la idea que mi tutor Javier estaba investigando, y al parecerme una propuesta muy interesante, no dudé en ponerme manos a la obra.

Al conocer en profundidad el *Deep Reinforcement Learning,* me he dado cuenta que es una de las ramas de la inteligencia artificial más aplicables a casos reales como la ya mencionada conducción autónoma. Parece que el futuro va encaminado hacia los coches autónomos sin la supervisión de una persona al volante, pero para que funcionen a la

perfección, no puede haber ni un mínimo error. La realidad ahora mismo es, que han surgido varios atropellos, incluso muertes por este tipo de coches [13], por lo que la seguridad ante posibles ataques externos o errores internos es imprescindible a la hora asegurar la integridad humana.

1.2 – Objetivos

En este trabajo nos centraremos en el análisis de la vulnerabilidad que tienen los sistemas de aprendizaje por refuerzo profundo.

En concreto, los objetivos serán:

- ***Estudio del estado del arte***. En este trabajo nos apoyaremos en investigaciones ya hechas y consolidadas, para poder ampliar los conocimientos de estos métodos aportando conclusiones y nuevas técnicas, como las que se van a implementar en este proyecto.

- ***Análisis e implementación de ataques sencillos existentes en la literatura***. Estos ataques modifican ciertos pixeles de la observación o imagen, o bien, aplican ruido a la misma mediante filtros (e.g., el de Gauss). En todos estos ataques se busca modificar lo mínimo posible la imagen en cada turno de juego para no ser detectados. Aún así, se comprobará que con mínimas modificaciones en la imagen, el rendimiento de la red cae sustancialmente.

- ***Definición e implementación de un nuevo ataque que identifique zonas/regiones de ataque***. La mayoría de ataques existentes en la literatura, como los que se describen en el punto anterior, atacan en cada turno de juego, independientemente de cuál sea la situación del mismo. En este ataque que proponemos, sin embargo, se trata de identificar zonas o regiones de ataque. Por ejemplo, si consideramos el caso del Breakout, no tiene sentido atacar cuando, por ejemplo, la pelota está subiendo, sino cuando está cayendo y está cerca de la raqueta. De esta forma, se induce el fallo de la red a la vez que se reducen el número de ataques. Algo similar ocurre en el juego del Pong. En este caso, el

objetivo es identificar cuando la pelota está en la zona centro-derecha y además se esté acercando a la raqueta de la derecha, la raqueta entrenada por la red neuronal a la que se pretende atacar. Es en estas zonas donde deberíamos atacar, y no en el resto donde atacar o no atacar no tiene ninguna importancia.

- ***Definición e implementación de un nuevo ataque basado en aprendizaje por refuerzo***. En este trabajo, se ha diseñado un nuevo ataque basado en aprendizaje por refuerzo. En este ataque, se definen los estados y las acciones posibles en cada uno de estos estados. Estas acciones serán dos: atacar o no atacar. De esta forma, mediante Q-Learning se trata de aprender una política de ataque que decida por cada estado del juego si lo mejor es atacar o no atacar. El objetivo nuevamente es identificar las zonas de ataque, y a la vez no ser detectado, causando el mayor daño posible y disminuyendo el número ataques al mínimo.

- ***Análisis de la vulnerabilidad de los sistemas de aprendizaje por refuerzo profundo a estos ataques en juegos de Atari***. Tras la experimentación realizada con todos los ataques propuestos se sacarán conclusiones sobre los resultados obtenidos tanto en Breakout como en Pong. El objetivo principal es comprobar la vulnerabilidad de la red de neuronas frente a este tipo de ataques. Para ello, se medirá el rendimiento de la red, el número de ataques realizado por cada técnica, etc.

1.3 – Estructura del documento

Este documento se divide en diferentes capítulos y apartados:

- ***2. Estado del arte***: En primer lugar, se hará un estudio sobre las tendencias e investigaciones que hay sobre las diferentes variantes del aprendizaje automático, aprendizaje por refuerzo, y aprendizaje por refuerzo adversario. En particular, este capítulo se dividirá en los siguientes apartados que describen el marco teórico en el que se enmarca el proyecto:
    - 2.1. Aprendizaje Automático
    - 2.2. Aprendizaje por Refuerzo
    - 2.3. Aprendizaje Profundo
    - 2.4. Aprendizaje por Refuerzo Profundo
    - 2.5. Aprendizaje por Refuerzo Adversario

- ***3. Descripción de la propuesta***: En este capítulo se describen en profundidad los tipos de ataques propuestos en este trabajo. Para ello, este capítulo se divide en los siguientes apartados:
    - 3.1. Descripción detallada de los filtros utilizados
    - 3.2. Identificación de regiones de ataque
    - 3.3. Ataque basado en aprendizaje por refuerzo

- ***4. Resultados***: En este capítulo se presentan los resultados obtenidos por los diferentes ataques propuestos en la sección anterior. Se divide en:
    - 4.1. Resultados en Breakout
    - 4.2. Resultados en Pong

- ***5. Planificación y presupuesto***: En este capítulo se describe la planificación y presupuesto del que se dispone para la realización de este proyecto. Por lo tanto, se dividirá en:
    - 5.1. Planificación
    - 5.2. Presupuesto

- ***6. Marco regulador***: En este capítulo se presentan las preocupaciones y estándares a seguir para desarrollar este proyecto. Se divide en los siguientes apartados:
    - 6.1. Riesgos

    - 6.2. Responsabilidades éticas
    - 6.3. Seguridad
    - 6.4. Estándares técnicos

- ***7. Entorno Socio-económico***: En este capítulo se describen los impactos a nivel global que genera este proyecto. Estos son:
    - 7.1. Impacto social
    - 7.2. Impacto económico
    - 7.3. Impacto medioambiental

- ***8. Conclusiones y trabajos futuros***: Por último, en este capítulo se detallan las conclusiones extraídas del estudio, teniendo en cuenta los objetivos del mismo.
    - 8.1. Conclusiones
    - 8.2. Trabajos futuros

2. ESTADO DEL ARTE

En este capítulo, se describirá el marco teórico en el cual se encuadra este trabajo. En primer lugar, se hará una breve introducción al Aprendizaje Automático que es el gran campo en el cual se encuentra este proyecto (Sección 2.1). Después, se describirá el Deep Learning o Aprendizaje Profundo (Sección 2.2) y se hará una breve introducción al Aprendizaje por Refuerzo (Sección 2.3). Más adelante, se describirán en qué consisten el Aprendizaje por Refuerzo Profundo (Sección 2.4) y, por último, el Aprendizaje por Refuerzo Adversario (Sección 2.5), siendo este último donde se encuentran el conjunto de técnicas que se emplearán en este trabajo. De esta forma, en este capítulo se parte de una visión global del marco teórico en el que se encuentran el conjunto de técnicas que se utilizarán, hasta llegar a la descripción detallada de las técnicas en particular que se emplearán.

2.1 – Aprendizaje Automático

El *Machine Learning o* Aprendizaje Automático es una de las principales ramas de la Inteligencia Artificial, cuyo objetivo principal es el de hacer que un sistema aprenda por sí mismo de forma automática, y no tenga que estar programado totalmente el comportamiento del mismo [2].

Aunque esta rama proviene de la ciencia computacional, difiere de la tradicional, ya que los algoritmos de esta ciencia computacional tradicional están específicamente programados para una solución concreta, o un determinado problema específico. Los algoritmos del Aprendizaje Automático en cambio, permiten a los computadores entrenarse con datos de entrada, además de utilizar modelos estadísticos adaptados a estos datos, con el fin de llegar a una solución dentro de un rango específico. Por lo que, el Aprendizaje Automático ayuda a los ordenadores a automatizar sus decisiones.

Un ejemplo de flujo de trabajo en Aprendizaje Automático lo podemos ver en la Figura 2. A partir de un conjunto de datos de ejemplo, se genera un modelo que puede ser de clasificación, predicción, clustering, optimización, etc. Una vez entrenado este modelo, dado un nuevo caso o ejemplo, se puede utilizar el modelo para generar una respuesta. Por ejemplo, supongamos que hemos entrenado un modelo de clasificación con ejemplos

que nos dicen dado un determinado tipo de día (nublado, soleado, temperatura, humedad) si ese día llovió o no. Una vez entrenado el modelo, dado un nuevo ejemplo que no ha sido utilizado previamente durante el entrenamiento, el modelo será capaz de predecir si para ese nuevo día lloverá o no. Los algoritmos de Aprendizaje Automático son capaces de identificar patrones de comportamiento en los datos.

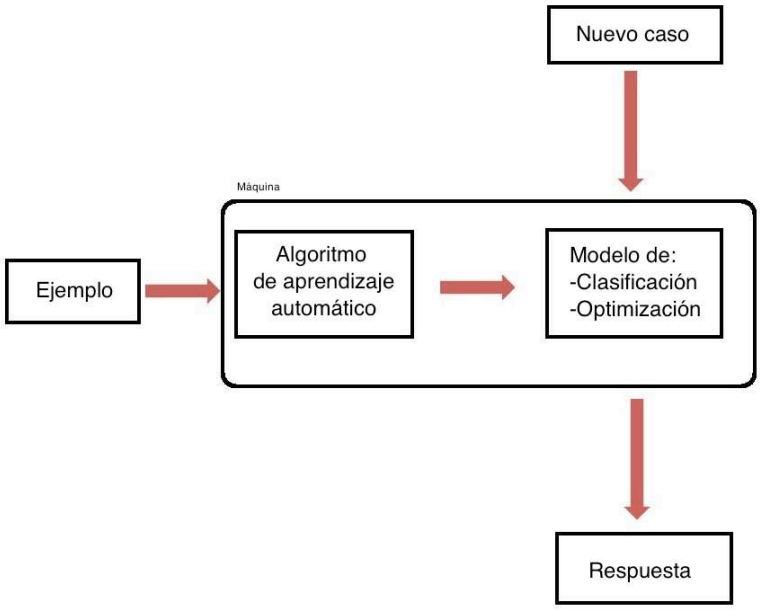


Figura 2. Esquema aprendizaje automático [14]

A día de hoy, el Aprendizaje Automático es aplicable a infinidad de tecnologías. Por ejemplo, en acciones cotidianas de las que no nos damos cuenta, como, por ejemplo, el desbloqueo del móvil por reconocimiento facial. Este sistema se basa en patrones aprendidos al escanear la cara en la cámara, de manera que compara estos patrones de la cara identificada como contraseña guardada para desbloquear el teléfono, con la cara que se muestra delante de la cámara para intentar acceder a los recursos internos.

El Aprendizaje Automático está en continuo desarrollo, lo que produce una gran satisfacción para la tecnología, optimizándola, y creciendo exponencialmente a lo largo de los años.

A su vez, el Aprendizaje Automático está clasificado en diversas categorías. Estas categorías están basadas en la manera de aprender que se le da al sistema. Se van a detallar

dos de los principales métodos de aprendizaje, el aprendizaje supervisado y no supervisado:

- En el **aprendizaje supervisado** se utilizan ejemplos etiquetados, es decir, ejemplos que contienen atributos de entrada y un atributo de salida que es el que se trata de clasificar o predecir. Volviendo al problema que se describía anteriormente, los datos de entrada serían las características del día (soleado, nublado, temperatura, humedad) y el atributo de salida sería si llueve o no ese día. En esta categoría podemos encontrar los árboles de decisión ID3, C4.5, o los árboles de regresión como M5P [2]. Un caso de uso común de aprendizaje supervisado es el uso de datos históricos para predecir futuros resultados inmediatos, como por ejemplo, filtrar los emails no deseados como spam [15].

- En el **aprendizaje no supervisado**, en cambio, no se dispone de un atributo de salida esperado, sino que se sacan conclusiones a partir de ejemplos sin etiquetar. El objetivo de este tipo de aprendizaje es descubrir patrones ocultos en el conjunto de datos, y así agruparlos en diferentes grupos. Por lo tanto, los algoritmos más comunes dentro de esta categoría son los algoritmos de *clustering* como por ejemplo, K-means [16]. Un ejemplo de aplicación puede ser tratar de agrupar los consumidores de un mercado. Por ejemplo, teniendo un conjunto de datos de los consumidores y sus compras, utilizando algoritmos de aprendizaje no supervisado, se puede saber la edad y sexo, que gracias a patrones, compran o pueden comprar artículos relacionados, para hacer campañas de marketing de esos productos destinados a esa edad o sexo específico. Este tipo de aprendizaje abarca un gran rango de aplicaciones reales.

En este proyecto nos vamos a centrar en otra técnica dentro del Aprendizaje Automático: el *Deep Learning* o Aprendizaje Profundo [17]. Lo que intenta este aprendizaje profundo es imitar el cerebro humano, por ello se basa en redes neuronales de multitud de capas. La disposición de estas capas es en cascada, es decir, la salida de una capa es la entrada de la siguiente capa, y así sucesivamente. En algunos casos, este diseño de multitud de redes neuronales artificiales, consigue superar a la capacidad cognitiva de un ser humano, y por ello se ha convertido en una de las técnicas de aprendizaje automático más potentes. Estos algoritmos se pueden utilizar tanto en aprendizaje supervisado, como en aprendizaje no supervisado.

Además, pueden aprender múltiples niveles de representación que corresponden con diferentes niveles de abstracción, formando así una jerarquía de conceptos. Actualmente, el Aprendizaje Profundo también se utiliza en Aprendizaje por Refuerzo, lo que ha permitido a éste último resolver tareas más complejas. En cualquier caso, antes de explicar en detalle en qué consiste el Aprendizaje por Refuerzo (Sección 2.3) y el Aprendizaje por Refuerzo Profundo (Sección 2.4), se dará una breve explicación al Aprendizaje Profundo.

2.2 – Aprendizaje Profundo

Como se anticipaba al final de la sección anterior, el *Deep Learning* o Aprendizaje Profundo es otro campo de la Inteligencia Artificial, que intenta imitar, y en algunos casos superar con creces, las habilidades cognitivas de los humanos.

El diseño de este aprendizaje está jerarquizado en diferentes niveles, formando una red neuronal artificial. En cada nivel jerárquico, se procesa información de entrada, generando salidas, las cuales serán las entradas del siguiente nivel de la jerarquía, consiguiendo así un proceso comprendido por una capa de entrada y otra de salida, y entre medias de estas dos, todos los niveles jerárquicos conectados entre sí.

Podemos diferenciar una red neuronal simple de una profunda en la capas que tiene cada una entre la entrada y la salida principal, como se muestra en la Figura 3.

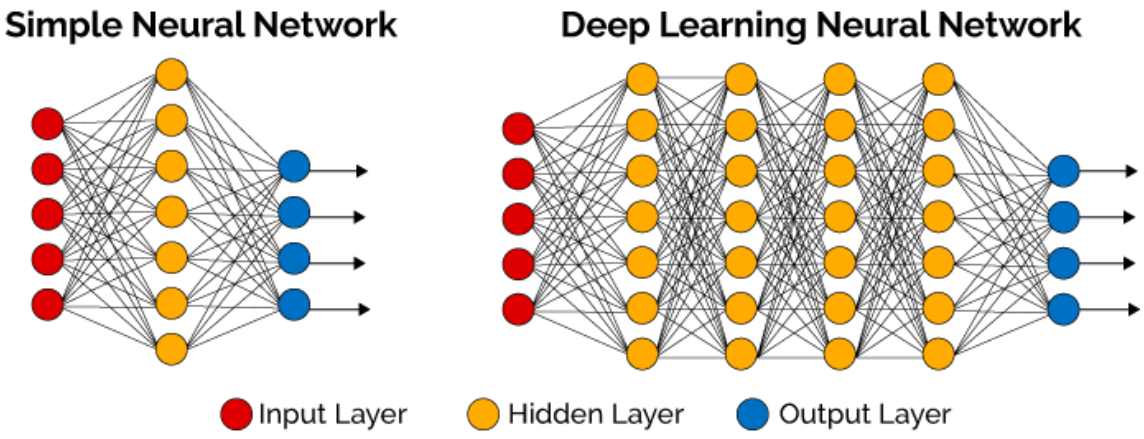


Figura 3. Deep Learning Neural Network [18]

En la Figura 3, se muestra a la izquierda una red de neuronas simple, y a la derecha una profunda. Se puede apreciar en la figura la clara diferencia que hay entre las dos redes, principalmente el número de capas que existen entre la capa de entrada y la capa de salida en cada una. El objetivo de esta jerarquización de niveles es englobar más dispersión en los datos de entrada, con el fin de abarcar muchas aplicaciones por la complejidad que presenta el conjunto de datos procesado.

Curiosamente, el aprendizaje profundo está inspirado en el cerebro humano, ese gran desconocido del que sabemos una pequeña parte de su potencial y diseño. Actualmente, es uno de los campos de la Inteligencia Artificial que mayores resultados y aplicaciones tiene, como por ejemplo, Siri, Alexa, Cortana dentro de los asistentes virtuales [19], coches autónomos y drones en el reconocimiento visual [20], chatbots como atención al cliente [21], reconocimiento facial para pagos en tiendas [22], diagnóstico de tumores dentro de la medicina [23], e infinidad de aplicaciones que nos hacen la vida más fácil.

2.3 – Aprendizaje por Refuerzo

En esta sección se describirá brevemente en qué consiste el aprendizaje por refuerzo antes de pasar a describir el Aprendizaje por Refuerzo Profundo (Sección 2.4) y el Aprendizaje por Refuerzo Adversario (Sección 2.5). El Aprendizaje por Refuerzo es una técnica de Aprendizaje Automático, que se basa en dar una recompensa o castigo al agente, en función de si la acción elegida por éste es correcta o errónea [24]. En Aprendizaje por Refuerzo existen tres conceptos básicos: acción, estado y recompensa.

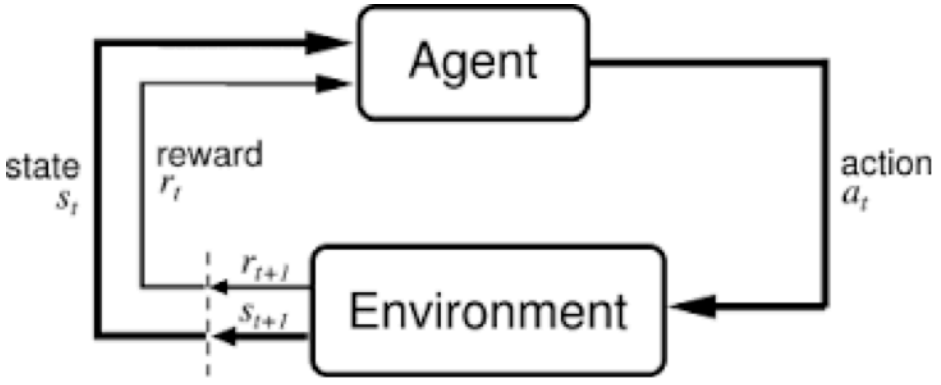


Figura 4. Esquema aprendizaje por refuerzo [25]

En aprendizaje por refuerzo, como se puede ver en la Figura 4, el agente se encuentra interactuando con un entorno. En cada instante de tiempo t, el agente recibe un estado st del entorno que define la situación en la que el agente se encuentra. Dependiendo de este estado st, el agente realiza una acción at que le lleva a otro estado st+1, y dependiendo de la acción realizada, el agente recibe una recompensa o un castigo rt.

Para entenderlo mejor, podemos imaginar a un bebé al lado de una chimenea. El bebé se encuentra en el estado actual en el salón al lado de la chimenea, el bebé realiza la acción de poner la mano en el fuego, por lo tanto, el bebé recibe el castigo de quemarse. El bebé aprenderá entonces que la ejecución de esa acción en ese estado no le lleva a conseguir un buen refuerzo, por lo que, tratará de evitar esa situación en el futuro.

El objetivo en Aprendizaje por Refuerzo es que el agente aprenda lo que está bien y lo que está mal, mediante recompensas o castigos, de forma que cada vez se comporte mejor, recibiendo más recompensas positivas y menos castigos, es decir, que se equivoque cada vez menos. En el caso real expuesto, el bebé no acercará la mano al fuego porque ha aprendido que si lo hace se quema, con lo que esto conlleva a más recompensas y menos castigos en el futuro.

A continuación, se detallarán en profundidad los principales componentes en Aprendizaje por Refuerzo:

- **Agente**: El agente es el que interactúa con el entorno. Es el que tiene la opción de realizar una acción u otra en un estado determinado. A su vez, es el que recibe la recompensa o el castigo. En nuestro ejemplo anterior, el agente sería el bebé. No obstante, por norma general, estos agentes serán agentes software que aprenden en entornos virtuales.

- **Acciones**: Se refiere al conjunto de acciones A que puede realizar el agente en cada uno de los estados.

- **Entorno**: Es el medio con el que interacciona el agente. En nuestro caso concreto, sería el salón junto con la chimenea.

- **Estados**: Se refiere al conjunto de situaciones o estados S en las que puede encontrarse el agente.

- **Recompensa**: Representa la recompensa o el castigo que recibe el agente, dependiendo de la acción realizada en un estado concreto. En nuestro caso, sería la quemadura del bebé en forma de castigo por haber realizado la acción incorrecta.

- **Política**: Define la estrategia π que el agente escoge a la hora de elegir una acción, es decir, representa el comportamiento que el agente tiene para escoger la acción con la que más recompensa obtendrá.

Existen varios tipos de algoritmos utilizados en Aprendizaje por Refuerzo para aprender la política π, pero en este trabajo nos centraremos en el algoritmo *Q-Learning* [24], el cual pondremos en práctica para una parte del análisis de este proyecto.

El algoritmo *Q-Learning* es el más utilizado entre todos los algoritmos del aprendizaje por refuerzo. Tiene como fin aconsejar o informar al agente mediante una tabla llamada Q, la cual representa los valores de la función de valor-acción para cada par estado-acción. De hecho, esta tabla tiene tantas filas como estados en los que pueda estar el agente, y tantas columnas como acciones pueda ejecutar. El objetivo de esta tabla es que el agente sepa qué acción es mejor elegir en cada estado.

La función de actualización de *Q-Learning* que permite actualizar cada una de las celdas de la tabla Q, se muestra en la Figura 5.

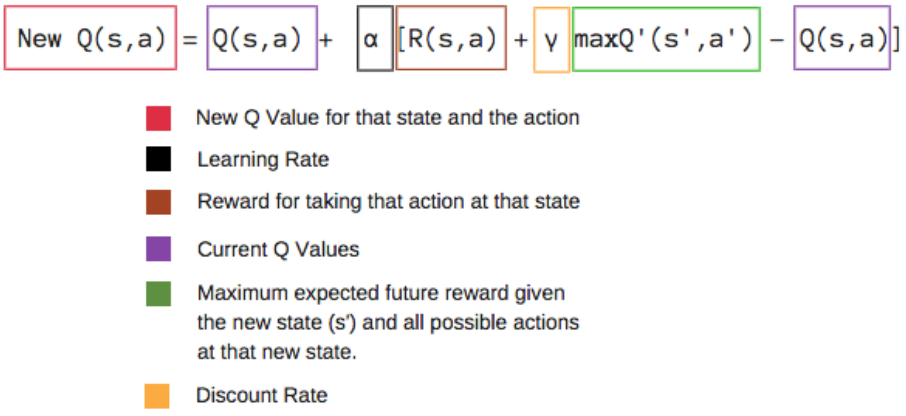


Figura 5. Algoritmo Q-Learning [26]

Como se puede ver en la Figura 5, se actualiza cada celda correspondiente a una acción y su estado. Cuando se empieza el entrenamiento, la tabla Q es inicializada a cero. Una vez inicializada la tabla, se irá rellenando con los valores máximos obtenidos de cada estado y su acción correspondiente, actualizándose iterativamente con los valores nuevos máximos para cada celda.

El proceso a iterar, con el objetivo de estabilizar los valores de la tabla Q, se estructuraría en los pasos que se describen en la Figura 6.

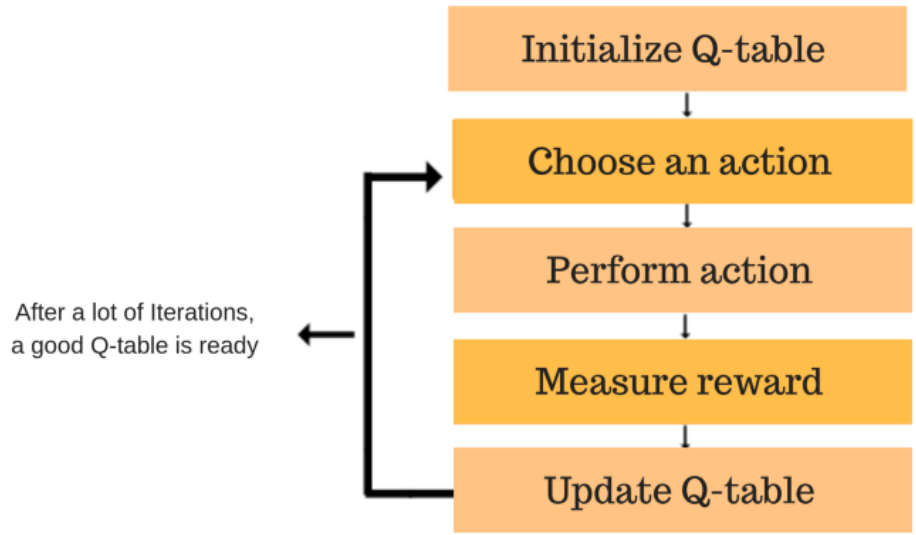


Figura 6. Estabilización de la tabla Q [26]

La descripción de cada uno de estos pasos sería la siguiente:

1. **Inicializar la tabla Q**: Lo primero es, crear una tabla Q de m filas por n columnas, en la que las filas corresponden con los m estados en los que se puede encontrar el agente durante su interacción con el entorno, y las n columnas con las acciones posibles. Inicialmente todas las celdas de esta tabla Q se inicializan con el valor cero.

2. **Elegir una acción**: Aquí entra en juego la estrategia de exploración. En este trabajo nos centraremos en la estrategia de exploración ε-*greedy* [24]. En esta estrategia tenemos el parámetro ε, llamado coeficiente de exploración, tomando

valores entre 0 y 1. En esta estrategia se explora con probabilidad ε (es decir, se elige una acción aleatoria entre las disponibles) y se explota la política aprendida con probabilidad 1-ε (es decir, se elige la mejor acción de la tabla Q). Cuando empieza el aprendizaje, ε es elevado, ya que permite al agente explorar e investigar el entorno escogiendo acciones aleatorias, ya que inicialmente no lo conoce. A medida que el agente empieza a conocer el entorno que le rodea, el factor de exploración comienza a disminuir, explotando así los valores ya conocidos.

3. **Realizar esa acción**: Una vez elegida la acción a realizar según el coeficiente de exploración, se lleva a cabo la misma.

4. **Recibir recompensa**: Según la acción escogida en el paso anterior, el agente recibe una recompensa, en función de si ha elegido un acción correcta, incorrecta o indiferente según el diseño de cada aplicación de este algoritmo.

5. **Actualizar la tabla Q**: Con los valores resultantes de la acción escogida, se actualiza la tabla Q utilizando la ecuación que se mostraba en la Figura 5, iterando este proceso tantas etapas de aprendizaje como el programador desee o hasta que el aprendizaje se detenga.

Como consecuencia de estos pasos, los valores de las celdas de la tabla Q empiezan a converger hacia los valores óptimos de cada acción para cada estado. Una vez obtenida una tabla Q con valores estabilizados, se puede obtener una política de comportamiento π que puede servir al agente para que dado un estado, elija la mejor acción posible, es decir, la acción que le conduce a obtener una mayor recompensa futura.

2.4 – Aprendizaje por Refuerzo Profundo

El *Deep Reinforcement Learning* o Aprendizaje por Refuerzo Profundo combina el Aprendizaje Profundo y el Aprendizaje por Refuerzo vistos en las secciones previas 2.2 y 2.3 respectivamente. En particular, se basa en el uso de redes neuronales profundas junto con algún algoritmo de Aprendizaje por Refuerzo, como por ejemplo *Q-Learning* [24].

Combinando estas dos técnicas, obtenemos como resultado un modelo increíblemente potente. En general, la red neuronal profunda se utiliza para aproximar la función Q. Por lo tanto, esta aproximación nos permite prescindir de la representación tabular de la función Q que se explicaba en la sección 2.3, lo que nos permite aprender en entornos mucho más complejos con un número infinito de estados. Por ejemplo, el algoritmo DQN [27] se ha utilizado con éxito en los juegos de Atari. Este algoritmo es capaz de identificar cada imagen u observación del juego. Es decir, en este caso, un estado sería una imagen o pantallazo en concreto del juego, y cada una de las características de estos estados sería cada uno de los píxeles de la imagen. La utilización de estas redes profundas en conjunción con Aprendizaje por Refuerzo permite aprender políticas que juegan muy bien y que superan incluso el rendimiento de jugadores humanos [27]. En el caso de nuestro proyecto, la red neuronal profunda también aprenderá de los píxeles de cada observación que se procesan en el juego como se describe en la Figura 7.

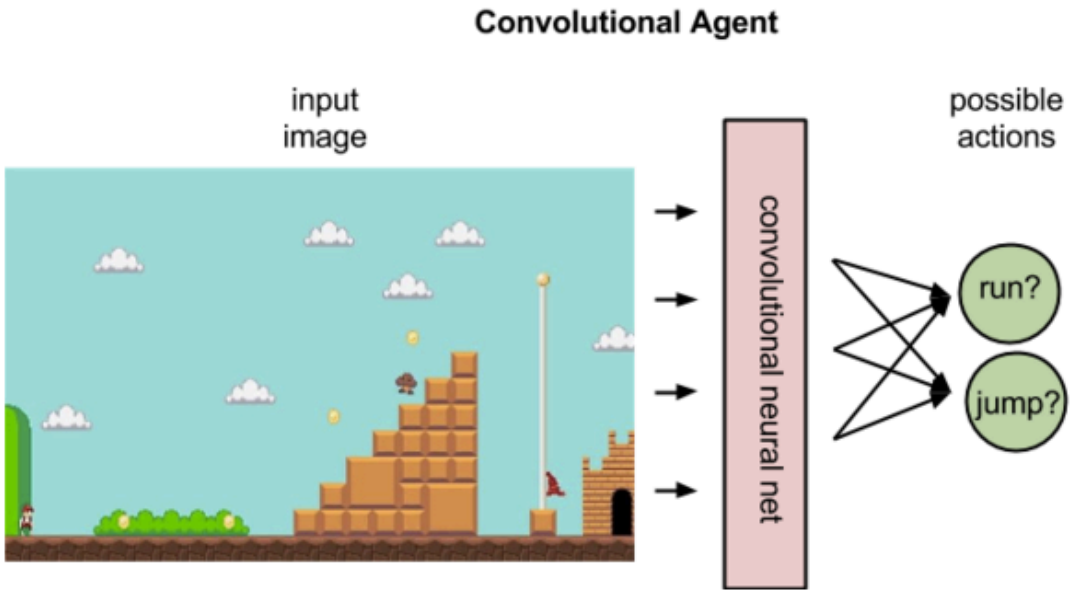


Figura 7. Deep Reinforcement Learning [25]

La Figura 7 ilustra la política que tiene el agente, mapeando el estado o imagen de entrada, a la mejor acción posible a realizar, gracias al procesamiento de imagen por parte de la red neuronal.

2.4.1 – Algoritmos DQN y DDPG

Existen en la literatura dos algoritmos principales dependiendo de si el espacio de acciones es discreto o no. En el caso de que el espacio de acciones sea discreto se suele emplear el algoritmo DQN [27], y en el caso de que el espacio de acciones sea continuo se suele utilizar el algoritmo DDPG [28]. A continuación se explican brevemente cada uno de estos algoritmos.

- **DQN (Deep Q-Network)**: Como se comentaba anteriormente, este método basado en aprendizaje por refuerzo profundo, combina las redes neuronales del aprendizaje profundo con el algoritmo Q-Learning del aprendizaje por refuerzo.

  La ventaja de esta técnica frente al aprendizaje por refuerzo simple es que el algoritmo de Q-Learning puede abordar problemas con un espacio de estados discreto, mientras que DQN puede afrontar problemas en el que el espacio de estados es infinito y de grandes dimensiones.

  Para solucionar las ineficiencias de los algoritmos como Q-Learning y así agilizar el procesado de los estados y acciones, DQN incluye dos componentes imprescindibles:

  - Memoria Replay: Buffer donde se almacenan y procesan datos de entrada en pequeñas trazas, entrenando así la red neuronal profunda de una forma más estable e independiente.

  - Red Target: Se crean dos redes profundas, una para guardar los valores de la Q, mientras que la otra se encarga de incluir todas las actualizaciones de esta en el entrenamiento. El objetivo principal de este diseño es el de coordinar cada ciertas actualizaciones la dos redes, ajustando así el valor objetivo de la tabla Q.

Aplicando la memoria replay y la red target, obtenemos una entrada y una salida mucho más estable para entrenar a la red neuronal y comportarse como si fuera aprendizaje supervisado.

- **DDPG (Deep Deterministic Policy Gradient)**: Otro método basado en el aprendizaje por refuerzo profundo, utilizando básicamente los mismos componentes adicionales u optimizaciones del aprendizaje profundo básico.

  Como se comentaba anteriormente, la diferencia principal entre el DQN y el DDPG es que el en Deep Q-Network se manejan sólo acciones de tipo discreto, mientras que en el DDPG trata con acciones continuas.

  Al tener una política determinista, el agente al principio no tiene muchas opciones para explorar el entorno. Para solventar este problema, se añade ruido a las acciones en el periodo de entrenamiento. Y si se desea unos datos de entrenamiento de calidad, se reduce el ruido aplicado a estas acciones.

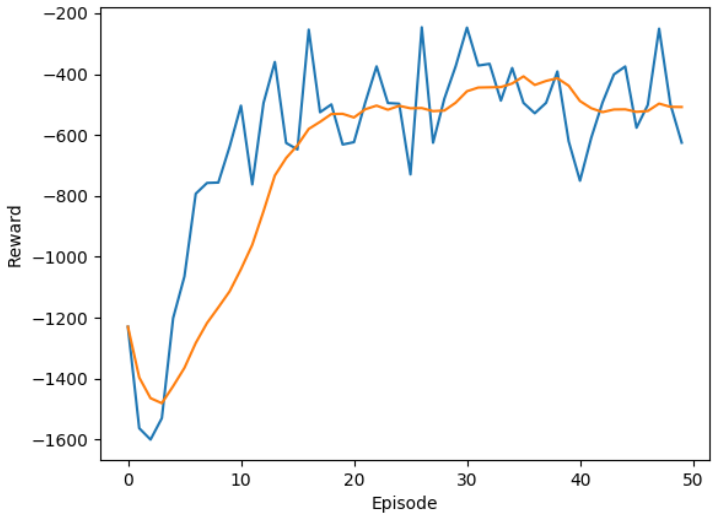


Figura 8. DDPG. Recompensa media en Pendulum [28]

En Figura 8, se refleja una gran mejora en poco tiempo, debido a la aplicación de este método en el juego Pendulum.

2.4.2 – Aplicaciones Reales

En cuanto a las aplicaciones reales del Aprendizaje por Refuerzo Profundo, encontramos una gran variedad entre ellas, y la gran mayoría serán cotidianas en nuestras vidas en un futuro muy cercano, ya que la tecnología está creciendo a pasos agigantados. Entre las más importantes están la conducción autónoma y la robótica.

- **Conducción autónoma**: Uno de los mayores e impactantes inventos, ya que el transporte es indispensable para la humanidad, que a pesar de su gran avance en este campo de la tecnología, queda mucho por investigar y perfeccionar. La conducción autónoma es una de las aplicaciones por las que se investiga y se desarrolla el aprendizaje por refuerzo profundo, ya que las pruebas corroboran el éxito de estas técnicas en los automóviles. El impacto social, medioambiental, económico, ético y la seguridad de estas técnicas aplicadas al sector del transporte, se expondrán más adelante en este trabajo [29].

Figura 9. Coche autónomo [30]

- **Robots**: Otra aplicación real muy útil es la inserción de estos algoritmos en el mundo de la robótica. La investigación en este sector es muy exhaustiva, ya que existe una gran demanda, tanto para la industria en cuanto a la automatización de procesos, como para el humano facilitándole las tareas del día a día.

  Dentro de la robótica, las aplicaciones reales más comunes son las habilidades cognitivas, tanto en reconocimiento de voz como en reconocimiento visual. Esto es gracias a la capacidad que tienen las redes neuronales profundas de procesar imágenes o sonidos, obteniendo como salida una serie de patrones, de los cuales se pueden obtener muchas conclusiones comparándolos con otros tipos de patrones resultantes de otros procesos.

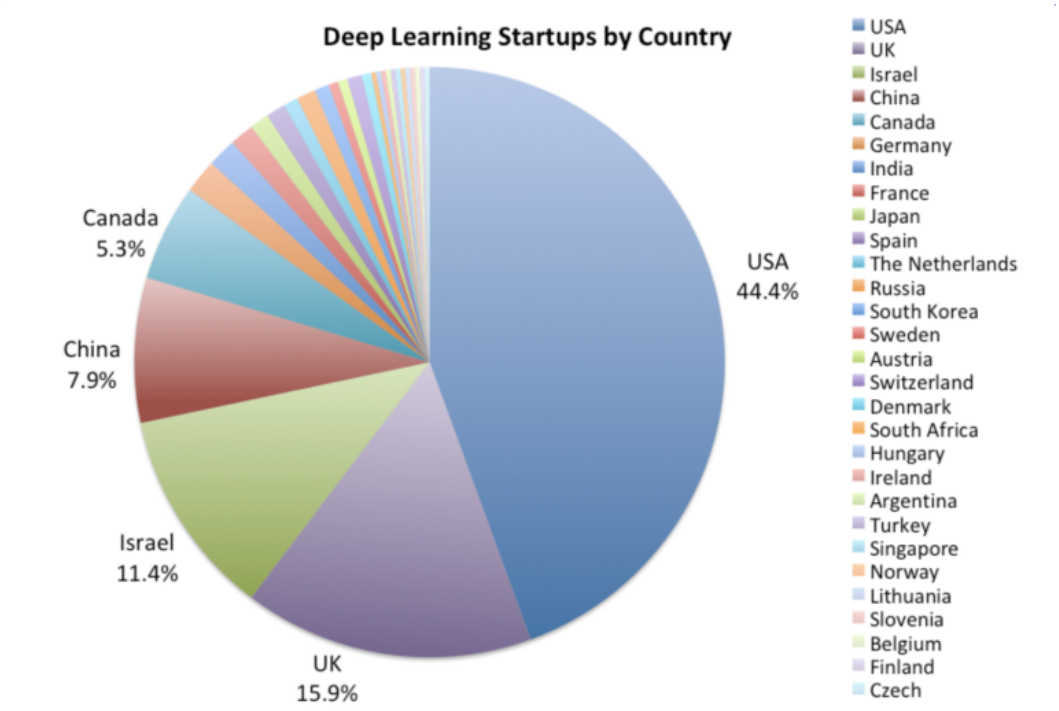


Figura 10. Gráfico Mundial Startups Deep Learning [31]

Este gráfico circular proporcionado por el blog Cognite Ventures [31] a día 9 de Septiembre del 2018, muestra el porcentaje de *startups* o proyectos impulsados o emprendedores que en cada país utilizan reconocimientos cognitivos implementando redes neuronales profundas para el procesado de imágenes o sonidos.

Como se puede observar, Estados Unidos es el país con mayor diferencia que utiliza estas técnicas de aprendizaje profundo, incluyendo en muchos casos el aprendizaje por refuerzo como complemento. Se debe a que el emprendimiento en este país es muy importante y está muy integrado en la mentalidad de sus trabajadores, intentando encontrar oportunidades y posibles aplicaciones de este método en cualquier sector.

Figura 11. China's E-Patrol Robot Sheriff [32]

El robot que se muestra en la Figura 11 aparentemente inofensivo, ha sido diseñado en china para detectar las caras de los delincuentes y terroristas que podría haber en el aeropuerto. Un robot que identifica las caras de todas las personas a las que alcanza el visor de su cámara, procesando a tiempo real y en un corto espacio de tiempo todos estos rostros, comparándolos gracias al aprendizaje por refuerzo profundo con los patrones guardados de estos terroristas, para alertar a sus compañeros de policía si en algún caso la cara identificada por este robot coincide con alguno de ellos.

En definitiva, existen infinidad de aplicaciones reales en las que pueden utilizarse el aprendizaje por refuerzo profundo, todas ellas muy útiles para el humano, sobre todo para hacer la vida de las personas más sencilla, de la manera más efectiva posible.

Muchas aplicaciones reales existentes del *Deep Reinforcement Learning,* se encuentran resumidas en la Figura 12.

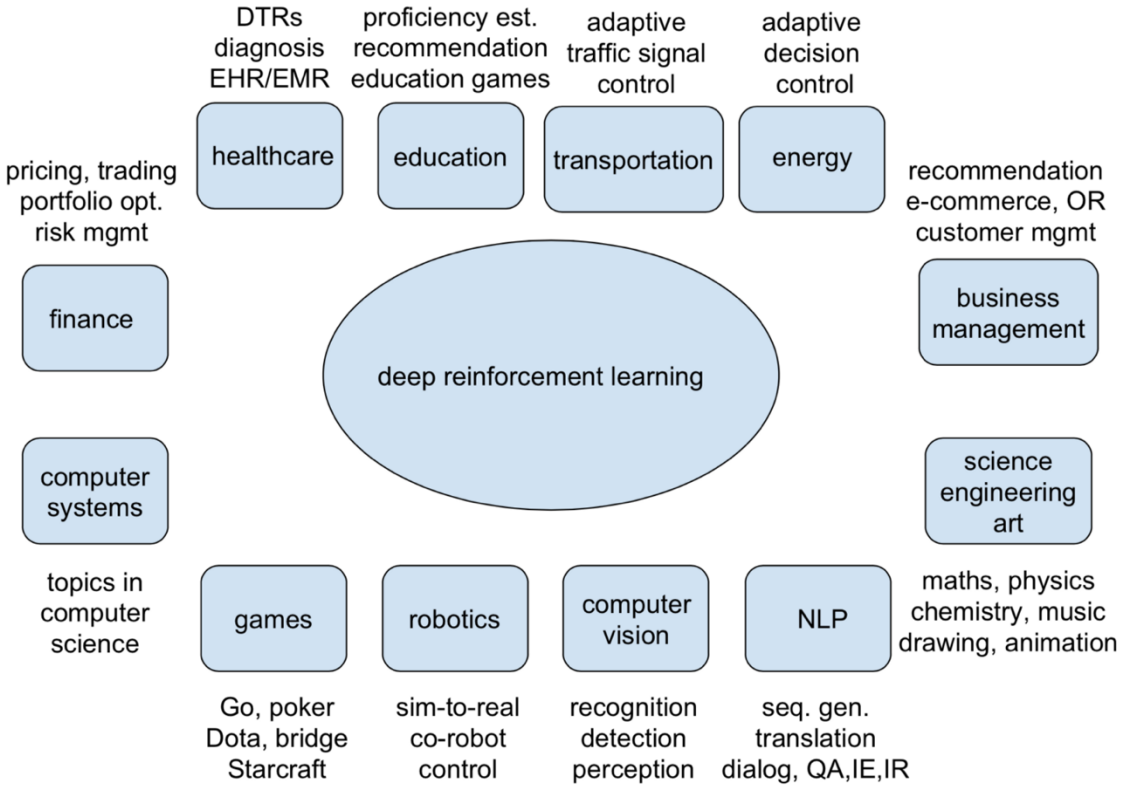


Figura 12. Aplicaciones Reales Deep Reinforcement Learning [33]

Figura 12 muestra varias de las aplicaciones reales que ahora mismo están utilizando las técnicas del aprendizaje por refuerzo profundo, así como la medicina estimando diagnósticos de enfermedades con antelación, la educación con juegos adecuados a la edad, la industria con sistemas automatizados en las fábricas, la energía tomando decisiones adaptadas al entorno, el sistema financiero estimando valores futuros de la bolsa o disposición de créditos, la gestión de negocios recomendando en *e-commerce* o tienda electrónica, etc.

Por lo tanto, se puede decir que el aprendizaje por refuerzo profundo es un método de futuro cercano y aplicable a casi cualquier ámbito. Teniendo como objetivo principal ser

de gran ayuda para que el humano tome más decisiones correctas que incorrectas e incremente su productividad.

2.4.3 – Aplicaciones en Video-Juegos: Atari

No obstante, aunque existen multitud de aplicaciones reales, en el caso de este proyecto nos centramos en los video-juegos, y, en particular, se utiliza la interfaz gráfica OpenAI Gym [34], la cual está bien documentada, y es de fácil implementación en cuanto a los algoritmos de Aprendizaje por Refuerzo Profundo. Nos centraremos en los juegos de Atari, algunos de los cuales se muestran en la Figura 13. En concreto, nos centraremos en los juegos de Breakout y Pong, que se corresponden con las dos primeras imágenes de la Figura 13.

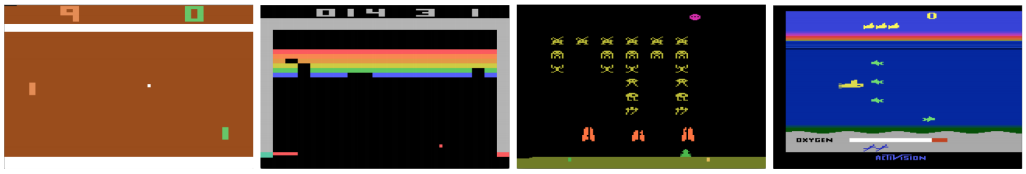


Figura 13. Juegos Atari [27]

Al ser juegos de simple configuración gráfica, al puro estilo Arcade, la implementación de algoritmos es bastante más simple que en los video-juegos que hay actualmente en el mercado, ya que estos juegos necesitan mucha más capacidad de procesamiento en las redes neuronales debido a que la cantidad de píxeles es mucho mayor que en los juegos de Atari. A medida que se perfeccione el rendimiento de los algoritmos y redes neuronales en estos juegos, se empezarán a introducir estas técnicas en juegos de gran resolución gráfica.

La Tabla 1 muestra los resultados totales obtenidos en diferentes juegos de Atari por varios algoritmos incluido DQN.

	B. Rider	Breakout	Enduro	Pong	Q*bert	Seaquest	S. Invaders
Random	354	1.2	0	−20.4	157	110	179
Sarsa [3]	996	5.2	129	−19	614	665	271
Contingency [4]	1743	6	159	−17	960	723	268
DQN	**4092**	**168**	**470**	**20**	**1952**	**1705**	**581**
Human	7456	31	368	−3	18900	28010	3690
HNeat Best [8]	3616	52	106	19	1800	920	**1720**
HNeat Pixel [8]	1332	4	91	−16	1325	800	1145
DQN Best	**5184**	**225**	**661**	**21**	**4500**	**1740**	1075

Tabla 1. Recompensa media total DQN [27]

En la parte de arriba de Tabla 1 se muestran las recompensas medias totales según el método utilizado. En la parte más baja de la tabla se encuentran los valores de la mejor puntuación obtenida durante todos los episodios de cada juego dependiendo del algoritmo usado.

Como se puede observar en la Tabla 1, los resultados obtenidos por el algoritmo DQN son de los más elevados por una gran diferencia en bastantes casos, superando incluso al humano en numerosas ocasiones.

Si nos centramos en los casos que se van a analizar y discutir en este proyecto, como son el Breakout y el Pong, podemos apreciar que el resultado obtenido de recompensa media está muy por encima del de una persona humana, por ello nos vamos a centrar en su estudio a fondo más adelante en este documento. Por lo tanto, se puede decir que el método DQN permite obtener políticas de comportamiento que cumplen las expectativas con creces, aplicándolo a estos juegos Atari en los que no es tarea fácil procesar los estados y las acciones con esa agilidad y eficacia.

Es importante destacar que aunque por simplicidad en este trabajo nos centraremos en los video-juegos de Atari para analizar la vulnerabilidad de los sistemas de Aprendizaje por Refuerzo Profundo, dicho análisis y las conclusiones que de él se obtengan pueden ser extrapolados a cualquier otro dominio.

2.5 – Aprendizaje por Refuerzo Adversario

Una vez introducido el contexto en el que se enmarca el Aprendizaje por Refuerzo Adversario, esta sección lo describe brevemente. El Aprendizaje por Refuerzo Adversario se dedica al estudio de los comportamientos que tiene el sistema cuando un atacante modifica los datos de entrada, con el fin de que este se equivoque y cometa errores o disminuya su rendimiento de aprendizaje.

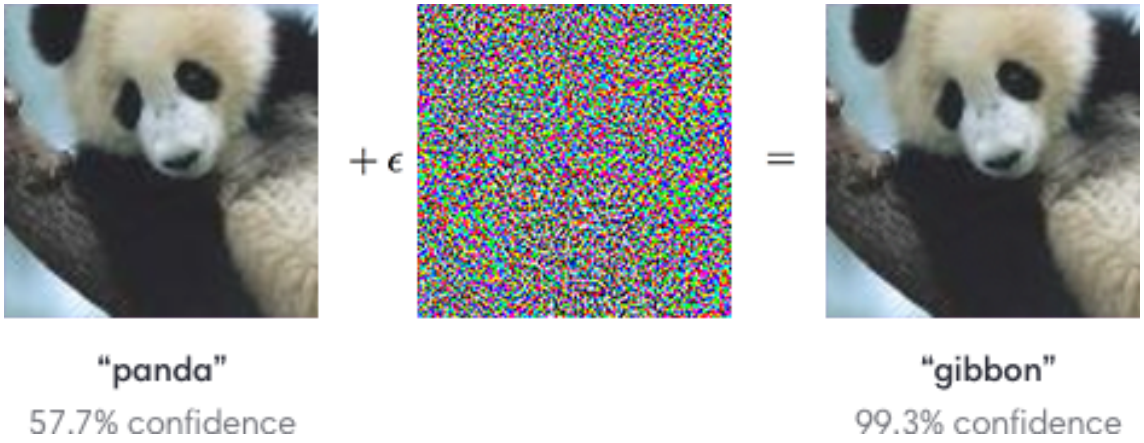


Figura 14. Vulnerabilidad Aprendizaje por Refuerzo Adversario [35]

Como se puede observar en Figura 14, teniendo una imagen de entrada de un panda con una seguridad de acierto de un 57,7%, el sistema puede ser engañado por el atacante, de tal manera que la imagen procesada resultante la identificará como un gibón en lugar de un panda con un 99,3% de certeza.

Este comportamiento se debe a que la imagen de entrada es modificada por un filtro o ruido añadido, lo que genera la confusión en el sistema, aunque a la vista del humano se aprecie que la imagen resultante es casi idéntica a la original.

El sistema se presenta bastante vulnerable a este tipo de modificaciones en los datos de entrada, ya que recoge pixel a pixel de esta imagen y los analiza de manera exhaustiva. Y es por esta razón por la que el humano no percibe la perturbación a simple vista, pero al sistema le causa una gran alteración en su comportamiento.

El mayor peligro de estos ataques es su implementación en las aplicaciones reales, como lo es en una aplicación muy delicada como la conducción autónoma, ya que en este sector está en juego la vida de los pasajeros.

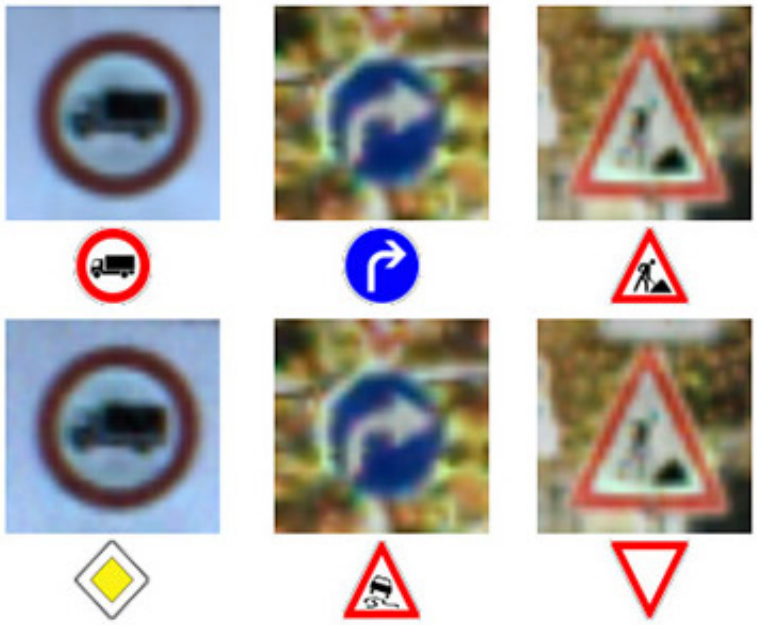


Figura 15. Aprendizaje Adversario Coches Autónomos [36]

La conclusión que se puede sacar de Figura 15, es la vulnerabilidad que presenta la conducción autónoma ante ataques que afectan gravemente al comportamiento de un vehículo. Por ejemplo en el caso de que el vehículo esté controlado por una política generada mediante Aprendizaje por Refuerzo Profundo, la modificación de una imagen puede causar que la acción que ejecute el sistema sea una diferente a la que realmente debería generar. Por ejemplo, un ataque podría hacer que una señal de girar a la derecha la interprete como una señal diferente, y el coche en lugar de girar a la derecha, se para, o incluso sigue recto. Por eso, ante cambios no apreciables para el ojo humano, las redes neuronales profundas que procesan estas imágenes se pueden comportar de manera disparatada y causar un accidente confundiendo una señal percibida por la cámara del vehículo como otra totalmente diferente.

Por lo tanto, es imprescindible analizar la vulnerabilidad de estos sistemas y tratar de implementar mecanismos de defensa pese a que son bastante difíciles de implementar. La razón principal por la que son tan difíciles de implementar es que, en primer lugar, sería

deseable conocer todos los tipos de ataque posibles para poder defenderse de todos ellos, ya que hay infinidad de métodos mediante los cuales se pueden modificar las imágenes de entrada al sistema. Y de esta manera no se puede saber exactamente con cuál de los métodos se está atacando en cada momento, teniendo que controlar gran cantidad de posibilidades de ataque.

Entre las posibles defensas, se destacan las siguientes [37]:

- Entrenamiento Adversario: Esta posible solución entrena al sistema con posibles casos de ataque, para el caso en el que el atacante ataque con cualquiera de estos métodos entrenados y el sistema detecte estos ataques, modificando así la imagen entrada del atacante por la original.

- Defensa de destilación: El objetivo principal de esta posible solución es, tener como salidas del sistema el porcentaje de cada clase de la salida en vez del tipo de clase que es. El propósito de este método de defensa es el de hacer que el sistema no se de cuenta de la entrada modificada y así atienda a los valores percibidos por las imágenes originales introducidas en las redes neuronales profundas.

- Autocodificadores: Son un tipo de redes neuronales que primero reducen las dimensiones de las imágenes de entrada, enviando estas a una capa oculta de menor nivel antes de devolver directamente la salida del sistema con las mismas dimensiones que la de la capa de entrada. Es decir, se encarga de comprimir la entrada, eliminando así posibles ruidos en los datos de entrada.

- Seguridad-profunda: El principal objetivo de este método de defensa es agrupar las entradas en clases, nombrando a cada grupo con un tipo de clase. Buscando así un rango seguro en el que las entradas agrupadas en clases se muestren intolerables.

Estas son unas posibles defensas, pero hay casos en los que estos métodos no son aplicables, ya que dependiendo de la técnica de ataque implementada, se necesita una defensa específica para ese tipo de ataque.

Por lo tanto, la investigación que necesita estudiar la defensa ante todos o al menos la mayoría de ataques está en proceso, y puede durar bastante tiempo hasta que se perfeccione.

Como se ha podido comprobar, mediante el Aprendizaje por Refuerzo Adversario es posible analizar la vulnerabilidad de este tipo de sistemas, y a partir de este análisis se podrían inventar posibles mecanismos de defensa. Pero también es cierto que la implementación de estos algoritmos de defensa es bastante compleja debido a la cantidad de posibles ataques existentes a la hora de modificar una entrada, y también al tipo de método que se utilice en el ataque.

3. DESCRIPCIÓN DE LA PROPUESTA

En este capítulo se van a describir los ataques llevados a cabo en una red neuronal profunda ya entrenada, para analizar la vulnerabilidad de este tipo de sistemas. Para ello, en primer lugar se van a presentar los filtros utilizados para modificar las imágenes u observaciones de cada uno de los dos juegos de Atari elegidos para esta investigación: Breakout y Pong. La aplicación de estos filtros se considera un ataque, puesto que puede inducir al error en la elección de la acción por parte del agente atacado.

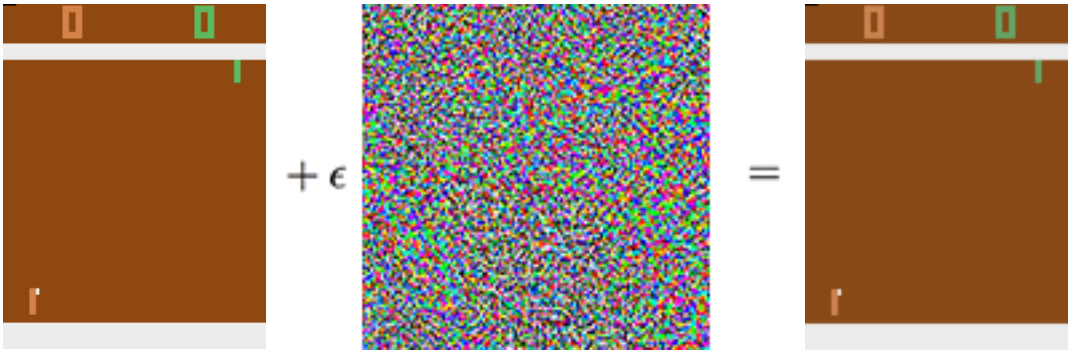


Figura 16. Filtro aplicado a imagen en Pong

A modo de introducción, la Figura 16 muestra un ejemplo de ataque que se va a llevar a cabo durante este trabajo en el juego Pong, aplicando un filtro a la imagen de entrada, y obteniendo una salida con una diferencia casi inapreciable para el humano con respecto a la imagen de entrada, pero con grandes efectos negativos en el sistema de aprendizaje, como se comprobará más adelante en la Sección 4.

Las observaciones procesadas una a una por la red neuronal profunda componen la secuencia de imágenes que, a su vez, construyen el video de la partida de cada juego. En la mayoría de las ocasiones se decide atacar siempre, es decir, por cada observación o, lo que es lo mismo, por cada acción que debe tomar este agente atacado. Con ello se consigue empeorar el rendimiento del sistema, pero también aumentan las probabilidades de ser detectado. Por ese motivo, en este trabajo se proponen dos tipos de ataque. En el primero, tratamos de identificar regiones del espacio de estados donde se debería atacar. El segundo, trata de aplicar técnicas de aprendizaje por refuerzo para identificar regiones del espacio para atacar y, además, también secuencias de ataque. En ambos casos se trata

de reducir el rendimiento del sistema lo máximo posible y, a la vez, reducir el número de ataques. Ambos ataques se explican en detalle en las secciones 3.2 y 3.3. Cabe destacar que todos los ataques se realizarán en fase de prueba o test, es decir, una vez que la red de neuronas atacada ha convergido a la política óptima y no continúa con el aprendizaje. En primer lugar, como se comentaba, se describirán los filtros utilizados para atacar cada una de las observaciones del juego.

3.1 – Descripción detallada de filtros

Se han empleado dos tipos de filtros durante la investigación de este trabajo, los cuales se explican en detalle a continuación:

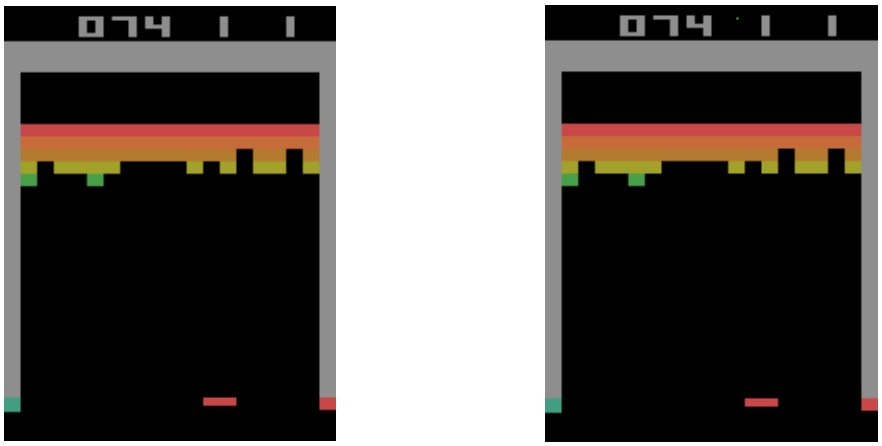
- **Modificación de píxeles**: La implementación de esta modificación, se divide, a su vez, en dos tipos:

    - **Modificación de un pixel**: La observación o imagen en la interfaz gráfica OpenAI Gym se representa como una matriz de 210x160x3 RGB. Lo que significa una matriz de 210 filas y 160 columnas, en la que cada celda de esta matriz representa el color de cada pixel de cada imagen del juego.

      Este método de ataque se resume en el cambio de color de un solo pixel de la imagen del juego con un valor aleatorio, es decir, un pixel aleatorio de la observación toma un valor de color aleatorio, modificando así la imagen original de entrada.

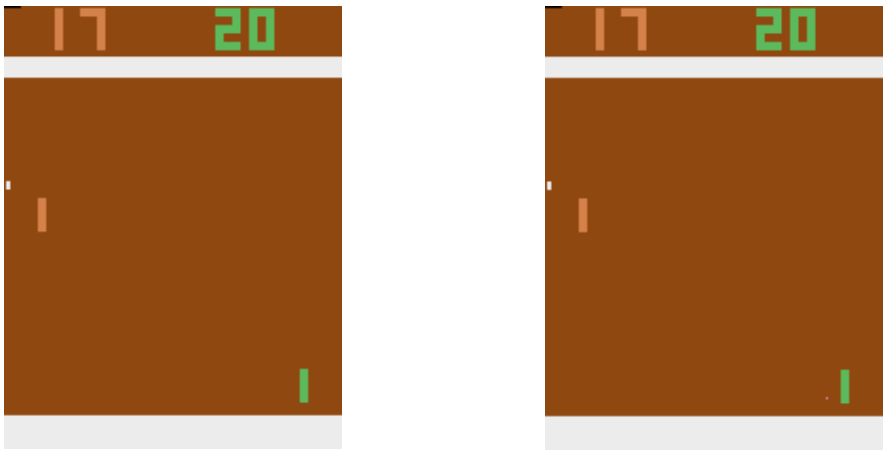
      Por lo tanto, cualquier pixel de la observación puede tomar el valor de 0 a 255 en RGB. Según si está en el campo de Red, Green o Blue, tomará el valor correspondiente a la combinación de colores que se represente mezclando los valores del RGB (e.g., [0, 255, 0] = Green).

Los ejemplos de este tipo de ataque en los juegos Breakout y Pong son los siguientes:



Original 1 Pixel

Figura 17. Original vs. 1 Pixel modificado en Breakout



Original 1 Pixel

Figura 18. Original vs. 1 Pixel modificado en Pong

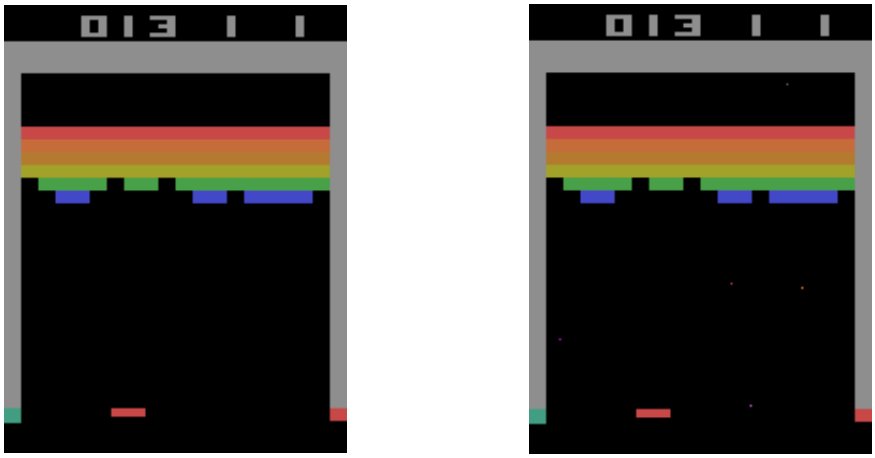
En la parte de arriba de la imagen de la derecha de la Figura 17 en el juego Breakout, entre el 074 y el 1, se puede apreciar el pixel aleatorio cambiado de color a verde, cuando vemos que en la imagen original justo ese pixel es negro.

De igual forma que se puede apreciar en la imagen de la derecha en Figura 18 en el juego Pong, el pixel modificado abajo a la derecha junto a la raqueta verde.

Los resultados obtenidos de la investigación sobre este ataque se ven reflejados en los epígrafes 4.1 y 4.2.

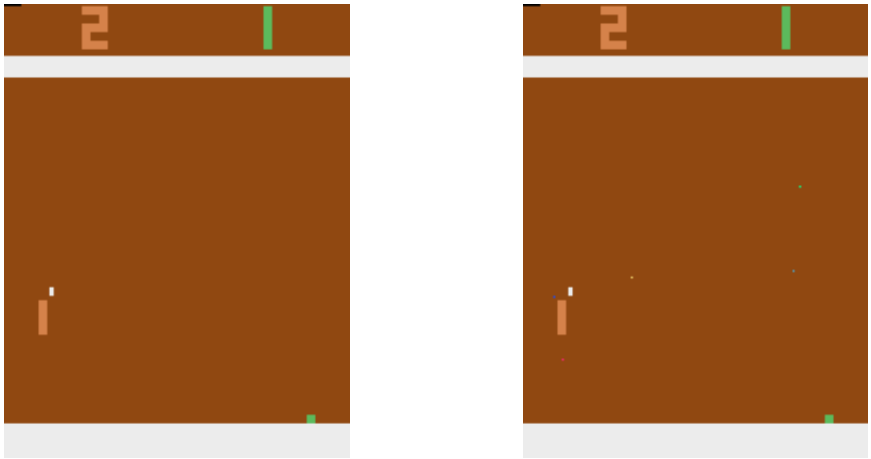
- **Modificación de 5 píxeles**: Esta técnica se basa en el mismo procedimiento que en el del anterior. La diferencia que hay entre ellas es que en esta técnica se modifican 5 píxeles aleatorios de la observación.

  Una ejemplificación de este tipo de ataque en los juegos Breakout y Pong es el siguiente:



Original 5 Píxeles

Figura 19. Original vs. 5 Píxeles modificados en Breakout

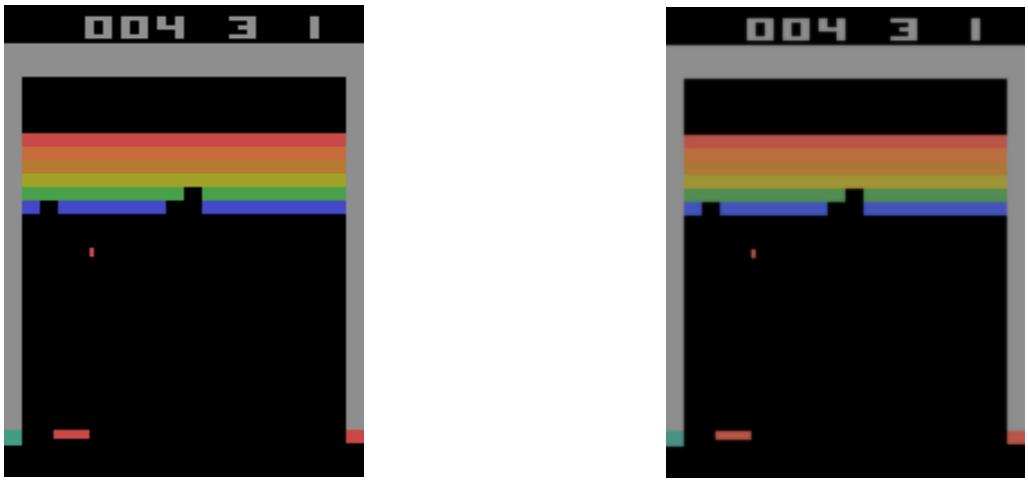


Original 5 Píxeles

Figura 20. Original vs. 5 Píxeles modificados en Pong

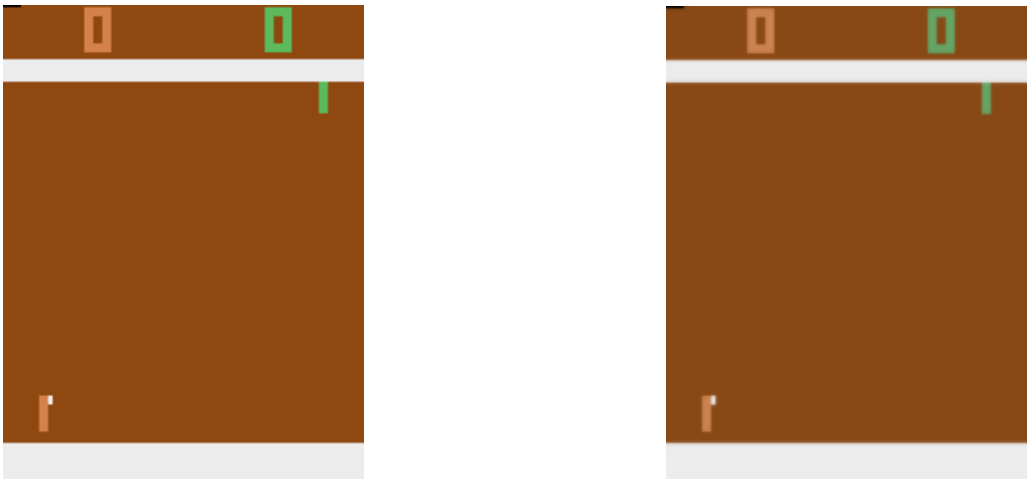
Como se aprecia en Figura 19 y Figura 20, la imagen modificada de la derecha contiene cinco píxeles con diferente color del que deberían tener, como en la original de la izquierda.

- **Gauss**: En este caso, se utiliza como ataque un filtro gaussiano. Este filtro modifica a simple vista humana tan poco la imagen, que no se percibe ningún cambio de la original a esta modificada. En cambio, el sistema se ve afectado severamente, como se explicará más adelante en los resultados del experimento.



Original Gauss

Figura 21. Original vs. Gauss en Breakout



Original Gauss

Figura 22. Original vs. Gauss en Pong

A simple vista, no se percibe apenas modificación, de la imagen original de la izquierda, a la imagen con el filtro gaussiano aplicado a esta en la observación de la derecha de Figura 21 para el Breakout y Figura 22 para el Pong.

En este documento, se detallarán los resultados obtenidos al aplicar estos ataques vistos. Estos métodos implementados son ligeras modificaciones como muestran las figuras anteriores, como la modificación de la imagen de entrada en un solo pixel, cinco píxeles o el filtro de gauss. Todos estos, muy poco apreciables a simple vista, pero con un gran factor de alteración en el sistema de las redes neuronales profundas.

3.2 – Identificación de regiones de ataque

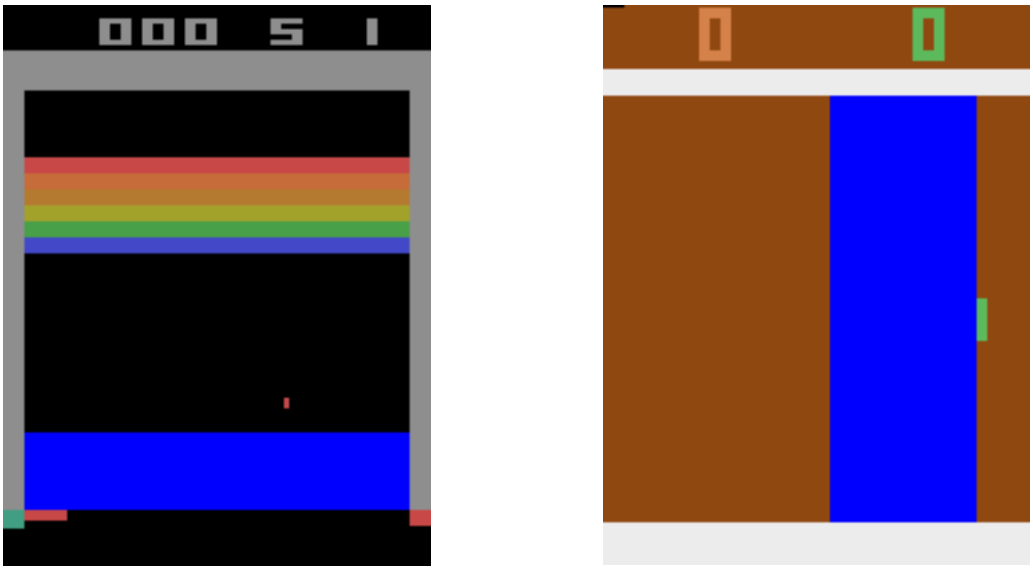
Si asumimos que cada ataque tiene un coste, y que además, el número de ataques debería ser el mínimo posible con el fin de no ser detectado por el agente atacado, lo evidente es no atacar siempre, es decir, no en cada observación del juego, sino en momentos clave, o lo que es lo mismo, en ciertas regiones del espacio de la imagen del juego.

Por ejemplo, en el Breakout lo lógico es atacar cuando la pelota esté en la zona media/baja de la pantalla y además esté cayendo, cuanto más cerca de la raqueta mejor.

Cabe destacar que los ataques son los mismos que cuando se decide atacar siempre, pero ahora la misión es identificar **cuándo** atacar.

La identificación de **cuándo** atacar induce más rápidamente al error, teniendo como efecto colateral la reducción del número de ataques, para que el atacado no pueda descubrir el momento en el que ha sido atacado, y ni siquiera que está siendo atacado.

Para el desarrollo de este tipo de ataques, se aplicarán los filtros descritos en la sección anterior pero, en este caso, sólo se aplicarán cuando la pelota está en un área determinada de la pantalla, es decir, en la zona cercana a la raqueta, y además teniendo en cuenta que la pelota se está acercando y no alejando a ésta. Estas áreas se describen en la Figura 23 para cada uno de los juegos considerados.



Breakout Pong

Figura 23. Región de ataque

La Figura 24 muestra la representación gráfica de este tipo de ataque. En esta figura, dado un nuevo estado st que se corresponde con una imagen del juego en ese instante de tiempo, el atacante puede decidir atacar o no atacar. Si la pelota no se encuentra en la región de ataque, el atacante pasará el estado st sin modificar al agente atacado, que devolverá una acción at. En cambio, si la pelota se encuentra en la región de ataque, el atacado pasará el estado st + ϵ, es decir, la imagen original con un pequeño ruido.

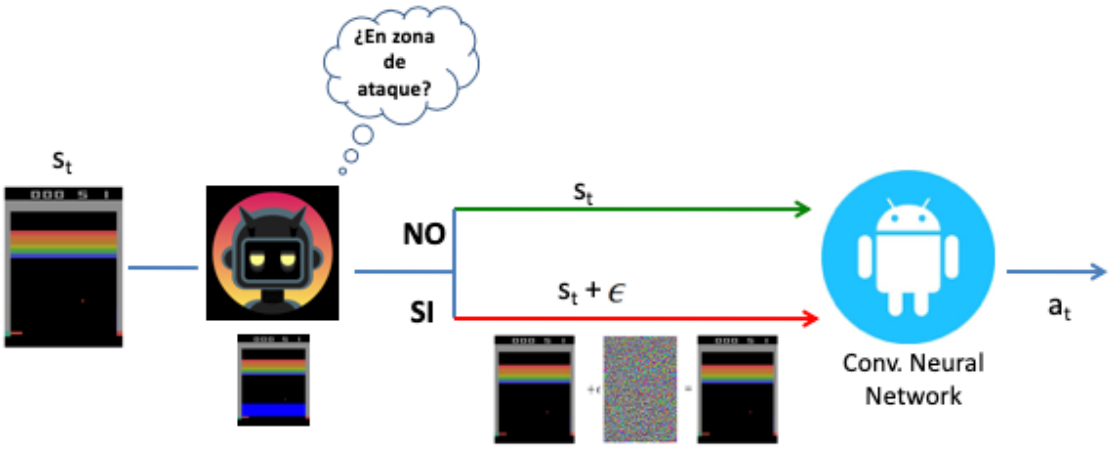


Figura 24. Representación gráfica del ataque en regiones de ataque

Siguiendo la misma estructura que en el epígrafe anterior, los tipos de ataques, es decir, los diferentes filtros de ϵ que se van a probar en este ataque, se definen de la siguiente manera:

- **Modificación de un pixel**: Esta vez se aplica el mismo ataque de modificación de un solo pixel de la imagen original de entrada, adaptándolo a una región mucho más restringida, como la que se aprecia en la Figura 23.

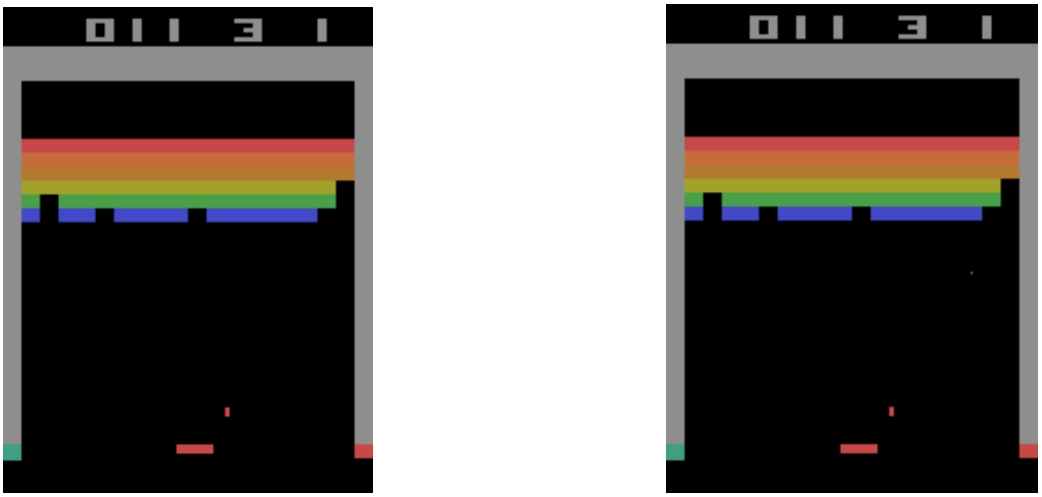
  En Figura 23 la región diseñada para atacar se muestra en color azul. Además de tener en cuenta que la pelota esté dentro de esta zona, hay contemplar también que se esté acercando a la raqueta.

  El efecto de este ataque tiene sentido que sea el mismo o similar al que se realiza en cada observación y sin ninguna restricción, ya que cuando más confunde a la raqueta es justo cuando va a golpear la pelota, es decir, cuando la pelota se encuentra bajando y en la región inferior de la pantalla, muy cerca de la raqueta, en cuanto al juego Breakout se refiere.

  En el caso del juego Pong, se aplica la misma estrategia. En cambio, en este juego, las restricciones cambian, ya que la raqueta entrenada por la red neuronal profunda se encuentra en la parte derecha de la pantalla. Por lo que, la región seleccionada para este ataque se acota a la parte derecha de la imagen, en una zona cercana a la raqueta, y además, se efectúa cuando la pelota se está acercando a esta, es decir, cuando la pelota se dirige hacia la derecha.

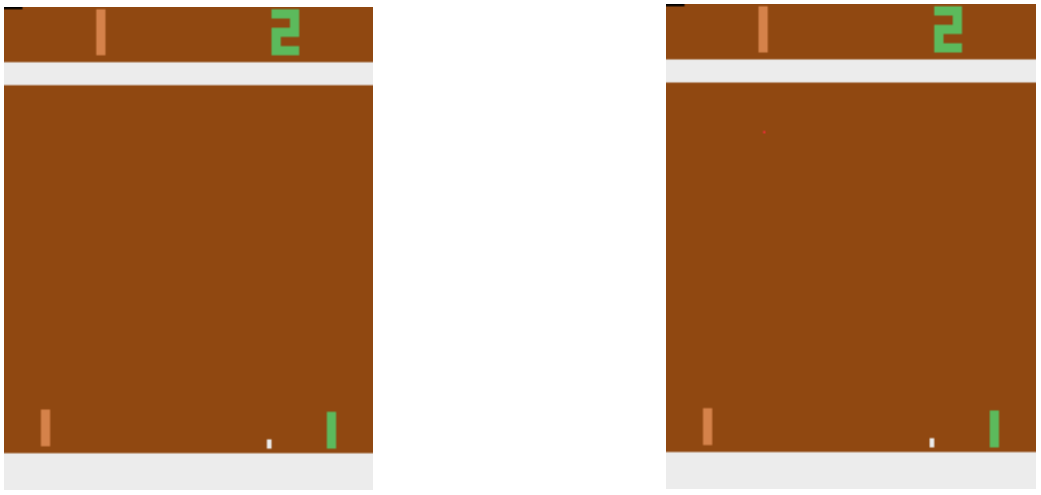
  Lo que se pretende conseguir con esta estrategia es, causar el mayor daño posible a la red durante su entrenamiento, minimizando al mismo tiempo el número de ataques, consiguiendo así no ser detectado por el sistema.

Un ejemplo del momento en el que se aplica este método en cada juego se refleja a continuación:



Original 1 Pixel Zona Ataque

Figura 25. Original vs. 1 Pixel Zona Ataque en Breakout



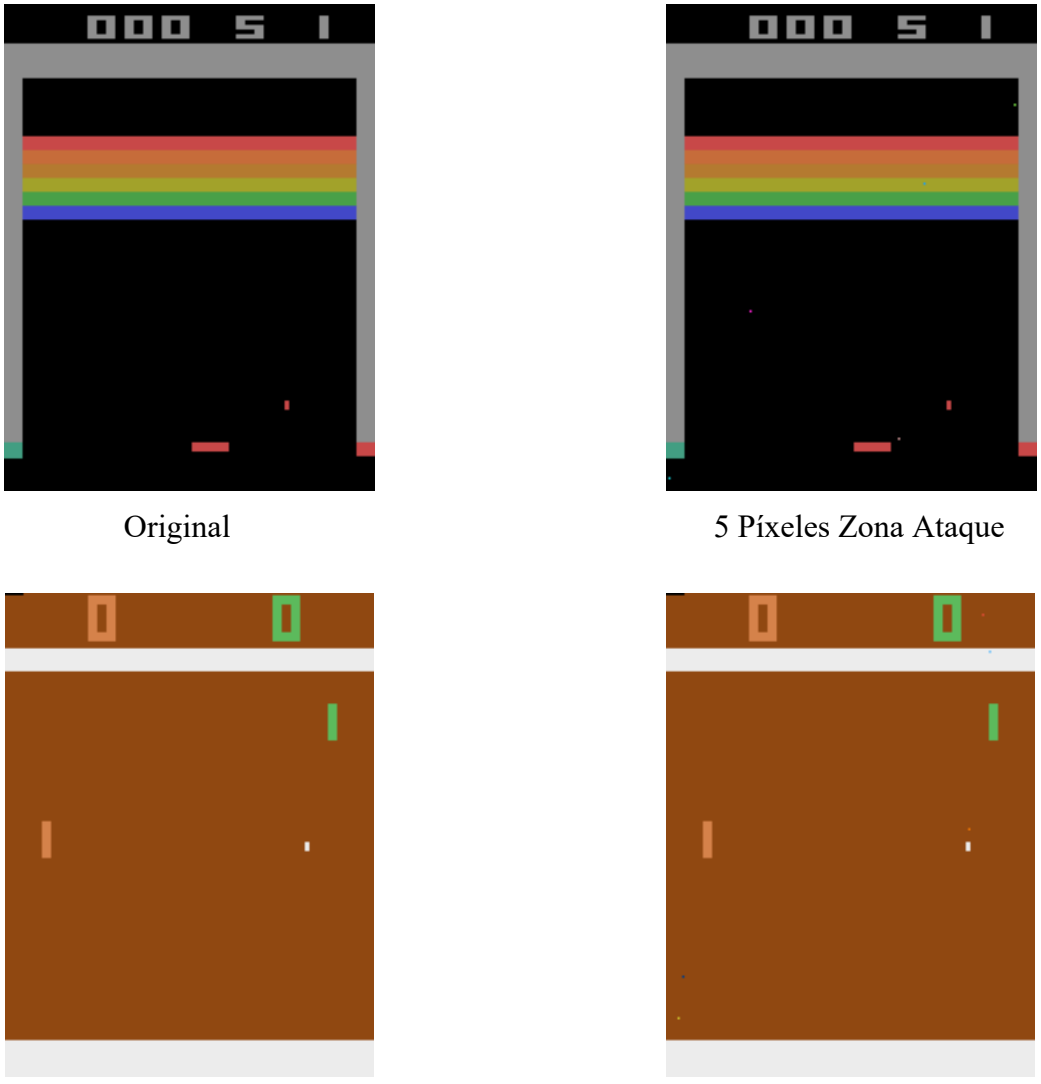
Original 1 Pixel Zona Ataque

Figura 26. Original vs. 1 Pixel Zona Ataque en Pong

En Figura 25 y Figura 26, se aprecia que la pelota se encuentra en la región determinada por Figura 23, cada uno con su respectivo juego.

Además, se muestra un pixel cambiado de color en la imagen atacada con esta técnica con respeto a la imagen original de la izquierda de cada figura. Un pixel verde en la parte superior de la pantalla en el Breakout en Figura 25, y un pixel rojo en la parte superior del juego Pong en Figura 26.

- **Modificación de 5 píxeles**: La idea principal de este ataque, es modificar la imagen de entrada, cambiando de color a un valor aleatorio de RGB cinco píxeles de ésta, pero en la región acotada, y además cuando la pelota se está acercado a la raqueta dentro de esta región, como se muestra a continuación:



Original 5 Píxeles Zona Ataque

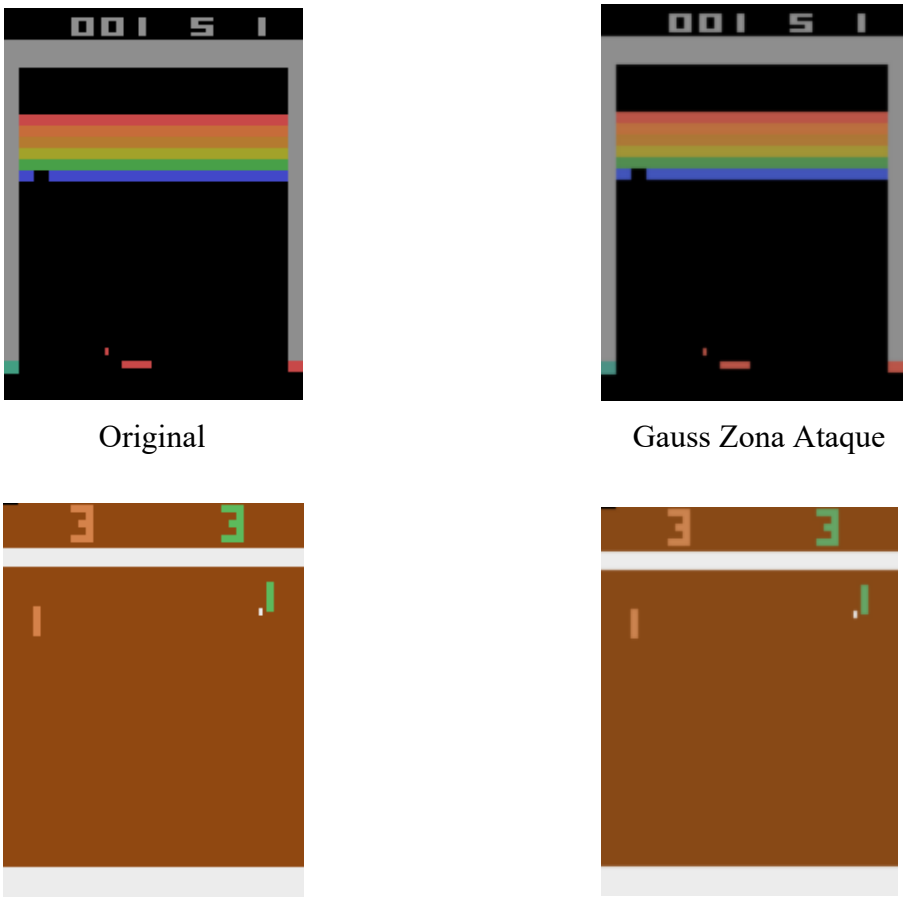
Original 5 Píxeles Zona Ataque

Figura 27. Original vs. 5 Píxeles Zona Ataque

Como se puede observar en Figura 27, las imágenes de la derecha modificadas difieren de las observaciones originales de la izquierda en que, estas muestran cinco píxeles cambiados de color a valores aleatorios. Además, se aprecia que la pelota se encuentra dentro de la región de ataque en cada juego.

- **Gauss**: El objetivo principal de este ataque es aplicar el filtro de gauss a la imagen u observación de entrada, de manera que el cambio entre la original y la modificada por este filtro no se aprecie apenas.

  La diferencia entre este método y el detallado en el epígrafe 3.1 es, que esta técnica se aplica sólo cuando la pelota se encuentra en el área detallada anteriormente y además acercándose a la raqueta. A continuación se muestra un ejemplo de esta técnica:



Original Gauss Zona Ataque

Original Gauss Zona Ataque

Figura 28. Original vs. Gauss Zona Ataque

Figura 28 presenta la leve modificación cuando la pelota se encuentra dentro de la región de ataque definida para cada juego, que produce el filtro de gauss con sigma 0.5 a simple vista en la imagen de la derecha con respecto a una imagen original de entrada como la de la izquierda. A simple vista, la imagen de la derecha modificada se aprecia un poco más oscura que la original de la izquierda, pero esto es si nos fijamos con detenimiento, sino pasa desapercibida. Precisamente por eso, la idea de este filtro es, no percibir apenas cambio visual, pero si percibir alteración en el rendimiento del sistema.

En consecuencia a estos tipos de ataque, el sistema se debe comportar de manera similar, tanto en el juego Pong, como en el juego Breakout, ya que los dos están implementados en base al mismo código, es decir, la red neuronal profunda es la misma y la estructura principal es muy parecida.

La principal diferencia es que cada juego aprende de diferente forma, ya que según el juego, necesita más o menos episodios de entrenamiento para tener un buen comportamiento.

3.3 – Ataque basado en aprendizaje por refuerzo

En este ataque, se ha modelizado el proceso de ataque como un proceso de decisión de Markov (MDP) donde hemos identificado los estados, las acciones, y la función de refuerzo [24].

En un MDP la transición entre estados se define en tiempo discreto. Es decir, en un instante inicial, el agente parte desde un determinado estado del entorno. En el instante siguiente, el agente ejecuta una acción, trasladándose al siguiente estado.

La Figura 29 muestra la representación gráfica de este ataque. Al contrario que el ataque descrito en la sección 3.2, el agente atacante no decide atacar en función de si la pelota está o no en una determinada región de la pantalla, sino en función de una política de comportamiento π, representada en una tabla Q, y aprendida mediante Q-Learning y una estrategia de exploración *ε-greedy*. Mediante esta política π, el agente puede decidir si atacar o no en un determinado estado st. En el caso de que no se ataque, el estado st se

entrega al agente atacado sin modificaciones. En el caso de que se ataque, se modificará el estado st, y se entregará al agente atacado un estado st + ϵ. El agente atacado ejecutará una acción at de acuerdo a este estado que ha recibido. Dependiendo de si el ataque fue bien o no, el agente atacante recibirá un refuerzo rt que le servirá para actualizar su política de comportamiento.

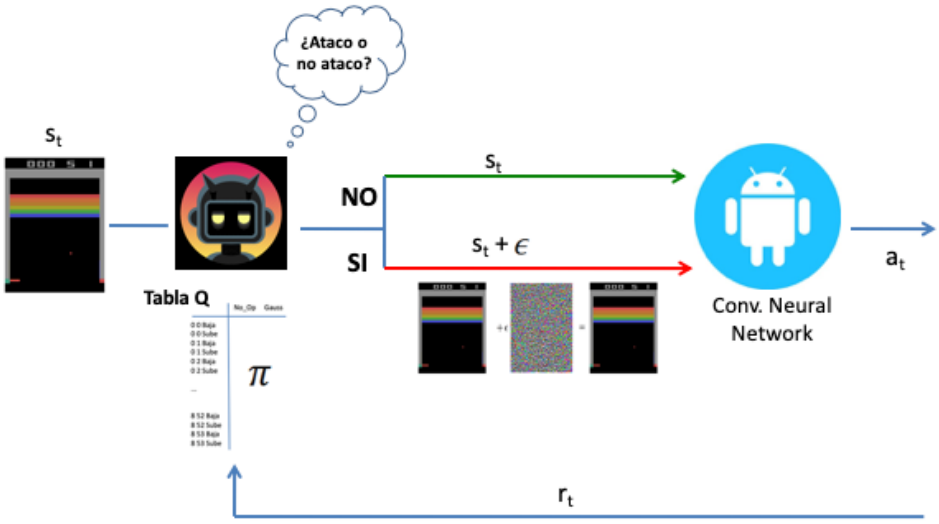


Figura 29. Representación gráfica del ataque basado en aprendizaje por refuerzo

A continuación se describen cada uno de los elementos de este MDP para aprender la política de comportamiento: los estados, las acciones, y la función de refuerzo.

3.3.1 – Descripción de los estados

Todo MDP está compuesto por un conjunto de estados finito, que en el caso de nuestro proyecto equivalen a las celdas estructuradas en la siguiente figura del juego Breakout:

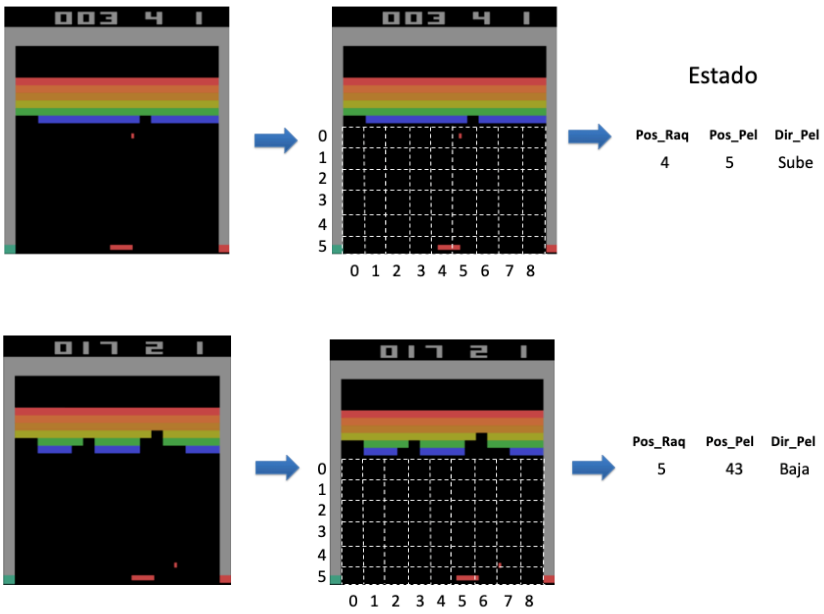


Figura 30. Definición de estados en Breakout

En Figura 30 se muestra un ejemplo de traducción de una observación o *pantallazo* del juego a un estado. Para la definición del estado, se selecciona una región de esta observación que se corresponde con la parte inferior de la pantalla y que tiene un tamaño de 96x144, es decir, 96 filas y 144 columnas de píxeles. Por lo tanto, se descarta la parte superior de la imagen que se corresponde con los ladrillos, puntuación, etc. Esta región de la pantalla se divide utilizando un grid de tamaño 6x9 y donde cada celda de este grid tiene un tamaño de 16x16 píxeles. Estas celdas nos sirven para identificar a *grosso modo* la localización de la pelota y la raqueta dentro de la pantalla. De esta forma, la pelota puede encontrarse en una de estas 54 celdas, y la raqueta en alguna de las 9 celdas inferiores. De esta forma, para la imagen inferior de la figura, podemos decir que la raqueta está en la celda 5, que la pelota está en la celda 43, y que la pelota baja. La combinación de estas tres características conforman la tupla del estado de la observación actual.

Lo mismo se aplica para el juego Pong, como se representa a continuación:

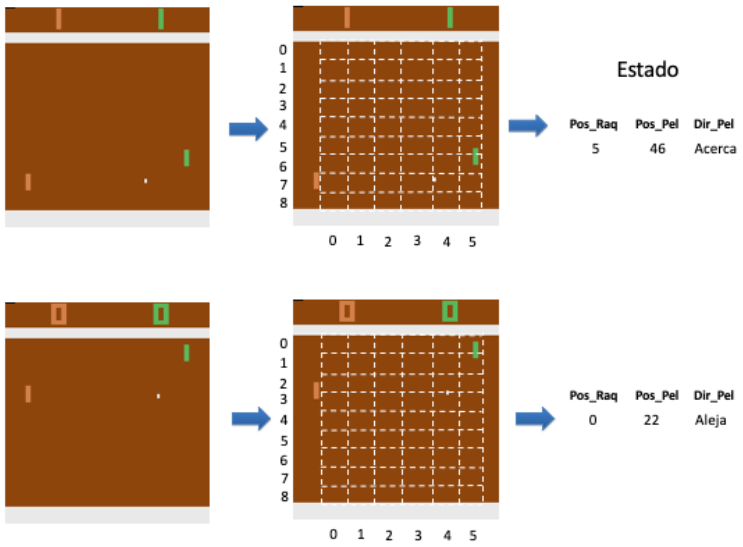


Figura 31. Definición de estados en Pong

En el caso del Pong, como se aprecia en Figura 31, se aplica el mismo grid que en el juego Breakout. La diferencia que hay en este juego es, que la raqueta se desplaza de arriba abajo, no como en el Breakout, la cual se desplaza de izquierda a derecha.

Por lo tanto, en este grid se cambiarán las filas por las columnas del juego Breakout, es decir, en vez de ser de 6x9 será de 9x6, lo que equivale nuevamente a 54 celdas pero en este caso de 18x21 píxeles al abarcar más pantalla que en el juego Breakout. Este nuevo grid ocupa una región de la pantalla de 144 filas y 96 columnas de píxeles. Como en el caso anterior, las celdas del grid pueden utilizarse para identificar la posición de la pelota y raqueta, y la dirección de la pelota, si se acerca o se aleja de la raqueta de la derecha. Combinando la posición de cada una y la dirección de la pelota en estas celdas, se obtiene el estado para cada observación concreta, como refleja en la parte de la derecha Figura 31.

En ambos juegos, el tamaño del espacio de estados de la tabla Q será de 9x54x2, es decir, 972 estados. Por lo tanto, esta representación de los estados nos permite reducir en gran medida su número, para así poder utilizar una representación tabular de la política de comportamiento.

3.3.2 – Descripción de acciones

Como se muestra en Figura 32, tanto para el Breakout (izquierda), como para el Pong (derecha), las posibles acciones elegidas en la tabla Q a la hora de entrenar al agente se dividen en dos:

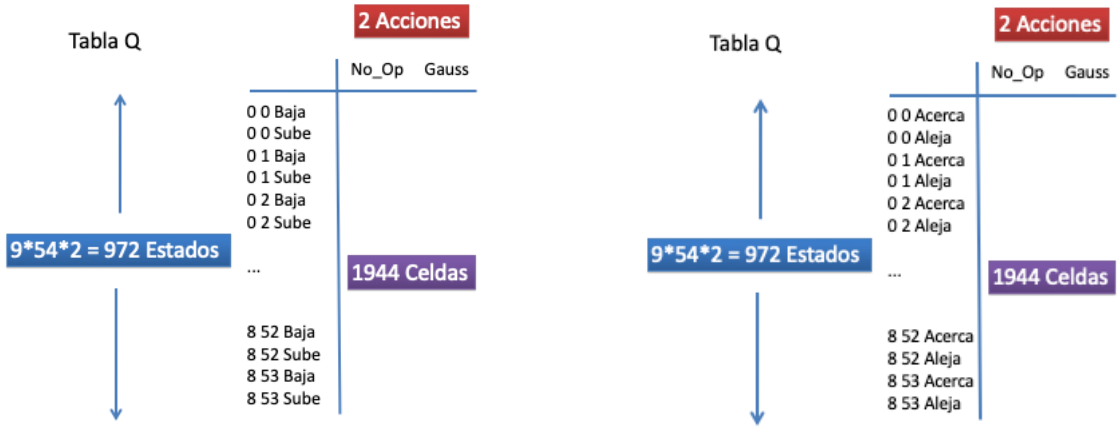


Figura 32. Tabla Q para cada juego

- No aplicar ningún ataque (No Op en la Figura 32): Como su propio nombre indica, no se aplicaría ningún tipo de filtro a la imagen original de entrada, es decir, la imagen u observación permanecería tal y como es, y la red entrenada no percibiría cambios o ataques. Por lo que, el juego se comportaría con un funcionamiento correcto y sin alteraciones de ningún tipo.

- Filtro de Gauss (Gauss en la Figura 32): En este caso, se añade el filtro de Gauss a la entrada, modificando esta con un sigma bajo, de manera que el cambio de la original a la modificada no se aprecie apenas visualmente, para poder demostrar que el sistema ve afectado su rendimiento, mientras que a simple vista no se percibe ninguna modificación.

  Este tipo de técnica podría ser un posible ataque, en el cual el sistema no sepa si se está atacando y cuando se está atacando. Por el resultado de las imágenes modificadas que se han representado anteriormente en este documento aplicando los tres tipos de ataque (modificación de un pixel, modificación de cinco píxeles

y Gauss), el ataque que menos se percibe a simple vista es este de Gauss. Por lo tanto, es un buen candidato a ser un gran ataque contra la red neuronal a la hora de no percatarse de este ataque. Se comprobarán sus resultados más adelante.

El objetivo de aplicar cualquiera de los dos tipos de ataque en cada estado es, saber qué acción en cada estado es la mejor: si atacar o no atacar.

3.3.3 – Función de refuerzo

El objetivo principal de este epígrafe es definir la función de refuerzo aplicando el algoritmo Q-Learning que, tanto en Breakout como en Pong, se lleva a cabo para que el agente aprenda, de manera que esta función enseñe a éste a saber en qué estado tiene que realizar una acción u otra.

En nuestro caso hay dos posibles acciones a ejecutar: no aplicar ningún filtro, y aplicar el filtro de Gauss a la imagen original de entrada, como se ha explicado en la Sección 3.3.2.

La idea principal de esta función es rellenar la tabla Q de valores, optimizando estos en cada iteración, con el fin de que el agente, una vez optimizada esta tabla, al cabo de bastantes episodios de actualización, elija la mejor acción para cada estado posible. Es decir, que el agente elija la acción que más afecte al sistema en cada estado, de manera que el rendimiento de la red neuronal disminuya considerablemente a medida que pase el tiempo. Para ello, la función de refuerzo se define como se muestra en la Ecuación 1.

$$r = \begin{cases} 10 & si\ se\ ataca\ y\ modifica\ la\ política\ del\ atacado \\ -1 & si\ se\ ataca\ y\ no\ modifica\ la\ política\ del\ atacado \\ 0 & en\ caso\ contrario \end{cases} \qquad (1)$$

El refuerzo será de 10 si se ataca y se modifica la política del atacado. En este caso, hemos conseguido engañar al sistema para que ejecute una acción diferente de la que debería ejecutar, es decir, hemos conseguido que ejecute una acción at en el estado st + ϵ diferente

a la acción a*t que debería ejecutar en el estado st sin modificar. En cambio, el refuerzo será de -1 si hemos atacado pero aún así el sistema devuelve la acción óptima. En este caso, penalizamos puesto que hemos atacado pero sin ninguna repercusión en el atacado, es decir, at=a*t. Mediante estas penalizaciones, tratamos de evitar atacar en aquellos estados donde no sea necesario. Por último, el refuerzo será de 0 si no atacamos.

4. RESULTADOS

Los resultados de este estudio se van a dividir en dos grandes bloques. En cada uno se evaluará el rendimiento del sistema cuando es atacado con diferentes técnicas y en diferentes situaciones, tanto para el juego Breakout (Sección 4.1), como para el juego Pong (Sección 4.2). A continuación se detallan los resultados en cada uno de estos juegos.

4.1 – Resultados en Breakout

En esta sección se presentan los resultados de tres ataques diferentes: ataque en cada observación, ataque en una zona específica (descrito en la Sección 3.2), y el ataque basado en aprendizaje por refuerzo (descrito en la Sección 3.3).

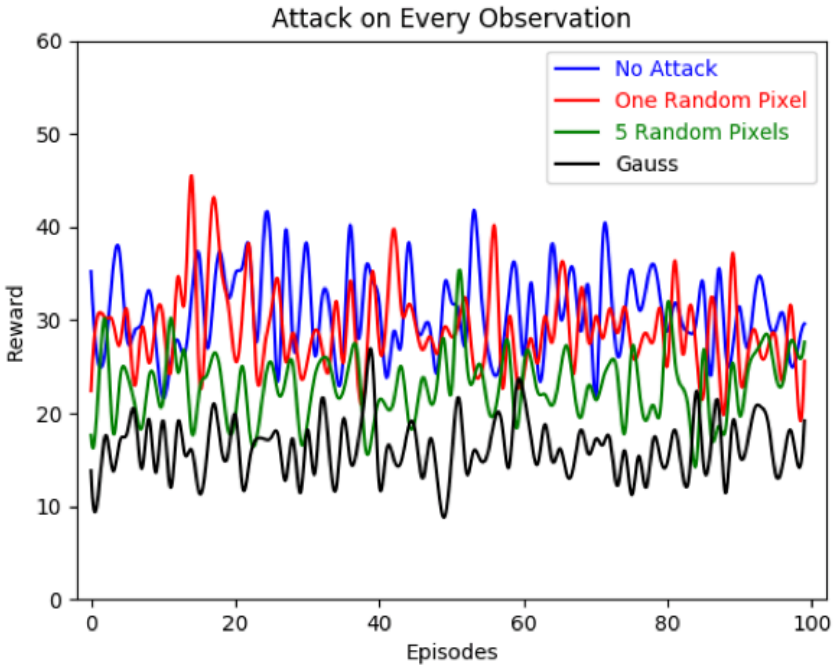
4.1.1 – Ataque en cada observación

Aplicando los diferentes filtros y modificaciones de la imagen de entrada original a cada observación de la secuencia del juego, se obtienen diferentes resultados.

De estos resultados se sacan conclusiones sobre el rendimiento de la red neuronal profunda cuando se encuentra expuesta a diferentes técnicas de ataque, como son la modificación de un píxel, de cinco píxeles y la aplicación del filtro de Gauss, en este caso, aplicadas a cada observación del juego sin ninguna restricción.

Los resultados obtenidos de este proceso, teniendo en cuenta que se aplican a cada observación, se reflejan en la siguiente Figura 33:

A



B

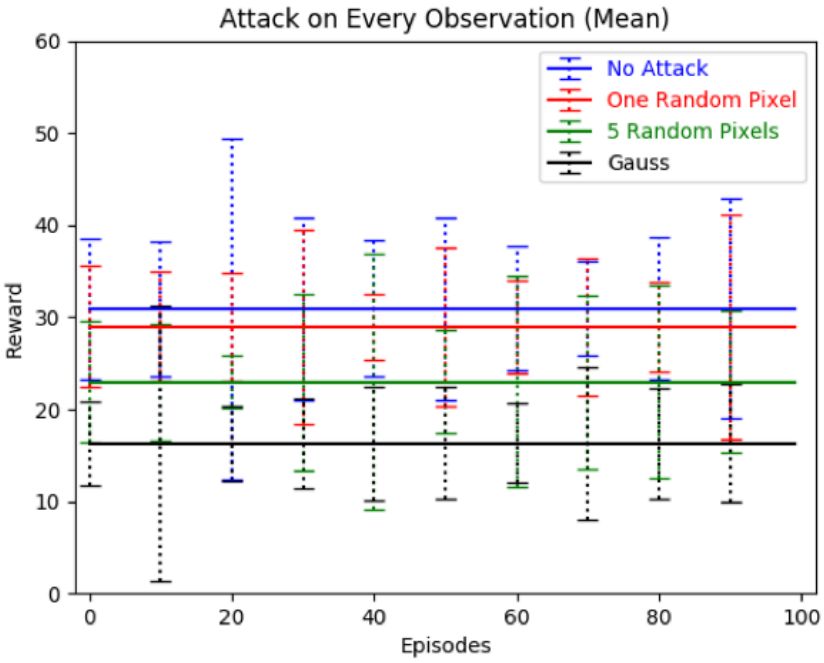


Figura 33. Ataque en Cada Observación y Media en Breakout

La Figura 33 A muestra la recompensa obtenida por el agente durante cien episodios cuando éste es atacado mediante cuatro técnicas diferentes: no aplicar ningún ataque, modificar un pixel aleatorio, modificar cinco píxeles aleatorios o aplicar el filtro de Gauss con parámetro sigma de valor 0.5.

Como en la Figura 33 A no se puede observar claramente el rendimiento de cada ataque ya que están las líneas superpuestas entre sí, en la Figura 33 B se puede apreciar la recompensa media obtenida por el agente al haberse aplicado cada tipo de ataque mencionado anteriormente, además de sus desviaciones estándar como explicación a la diferencia entre el valor máximo y el mínimo obtenido en las ejecuciones realizadas para estas figuras.

En la Figura 33 B se aprecia que ya con sólo cambiar un pixel de cada observación aleatorio el sistema disminuye su rendimiento, de igual manera que los demás ataques disminuyen el rendimiento notoriamente. El ataque que más afecta a la red neuronal profunda como se puede observar en esta figura, es la aplicación del filtro de Gauss, ya que, como se explicaba anteriormente en este documento, el objetivo del filtro de Gauss es no percibir apenas la modificación a simple vista de la imagen u observación original de entrada, pero sí dañar lo máximo posible el sistema disminuyendo notablemente su rendimiento, como se aprecia en la Figura 33 B.

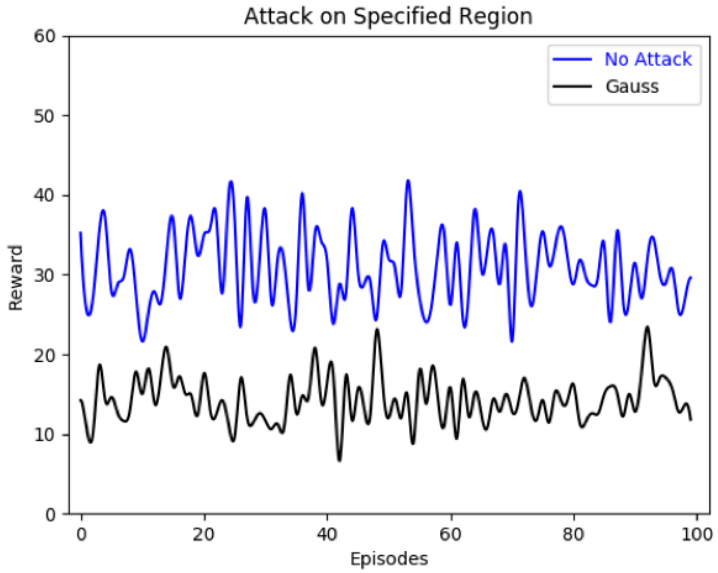
Como se describía anteriormente, los resultados obtenidos tanto en la Figura 33 A como en la Figura 33 B se refieren a cuando se ataca a la red neuronal profunda en cada observación sin ningún tipo de restricción. En la Sección 4.1.2 se mostrarán los resultados del ataque definido en la Sección 3.2.

4.1.2 – Ataque en la región específica

Esta vez se aplicará el ataque de Gauss sólo en la región detallada por la Figura 23 y además cuando la pelota se esté acercando a la raqueta, ya que, como se pudo comprobar en la sección anterior, este filtro de Gauss es el ataque que más afecta al sistema y que menos se percibe a simple vista.

Los resultados obtenidos de este proceso, teniendo en cuenta en este caso que se aplican sólo cuando se cumplen las condiciones detalladas anteriormente, se reflejan en la siguiente Figura 34:

A



B

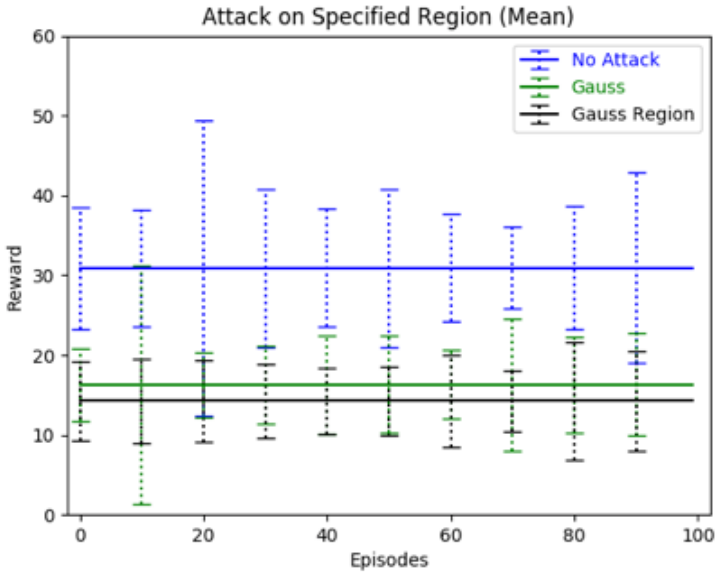


Figura 34. Ataque en la Región Específica y Media en Breakout

La Figura 34 A representa la recompensa obtenida por el agente cuando no se aplica ningún ataque y cuando se aplica el filtro de Gauss. Estos resultados se obtienen al aplicarse este filtro cuando la pelota se encuentra en la región detallada en la Figura 23, y además se está acercando a la raqueta.

Por otro lado, en la Figura 34 B se muestra la media de las ejecuciones de cien episodios junto con sus desviaciones típicas representadas cada 10 de estos.

Este resultado es muy importante, ya que, se aprecia en la Figura 34 B (con la línea negra) que el sistema es alterado casi de igual manera que si se ataca en cada observación (como representa la línea verde). De esta forma, en este ataque que hemos diseñado hemos conseguido reducir el número de ataques y dañar el sistema incluso más que atacando en cada observación, lo que implica que se reduce la probabilidad de ser detectado.

Cabe destacar que el número de ataques realizados de media por episodio en la región específica del Breakout cuando la pelota se está acercando a la raqueta es de 120 frente a los 1520 realizados en el caso de atacar en cada observación. Si nos fijamos, el porcentaje de ataques se reduce en más de un 90% en cada episodio, consiguiendo así tener muchas menos probabilidades de ser descubierto.

4.1.3 – Ataque Basado en Aprendizaje por Refuerzo

Este ataque es el descrito en la Sección 3.3 de este documento. Como se describía en esa sección, en este ataque, el entrenamiento del agente atacante se lleva a cabo mediante la actualización de la tabla Q, la cual modifica los valores de sus celdas en cada episodio con el objetivo de converger a valores en los que el agente seleccione la mejor acción para cada estado. En nuestro caso, el agente aprende a atacar en los estados que más afecte al rendimiento del sistema, reduciendo así el número de ataques y maximizando el daño causado.

La recompensa obtenida por el juego al entrenar al agente a atacar en los estados donde más alteración provoca al sistema, se refleja en la siguiente Figura 35:

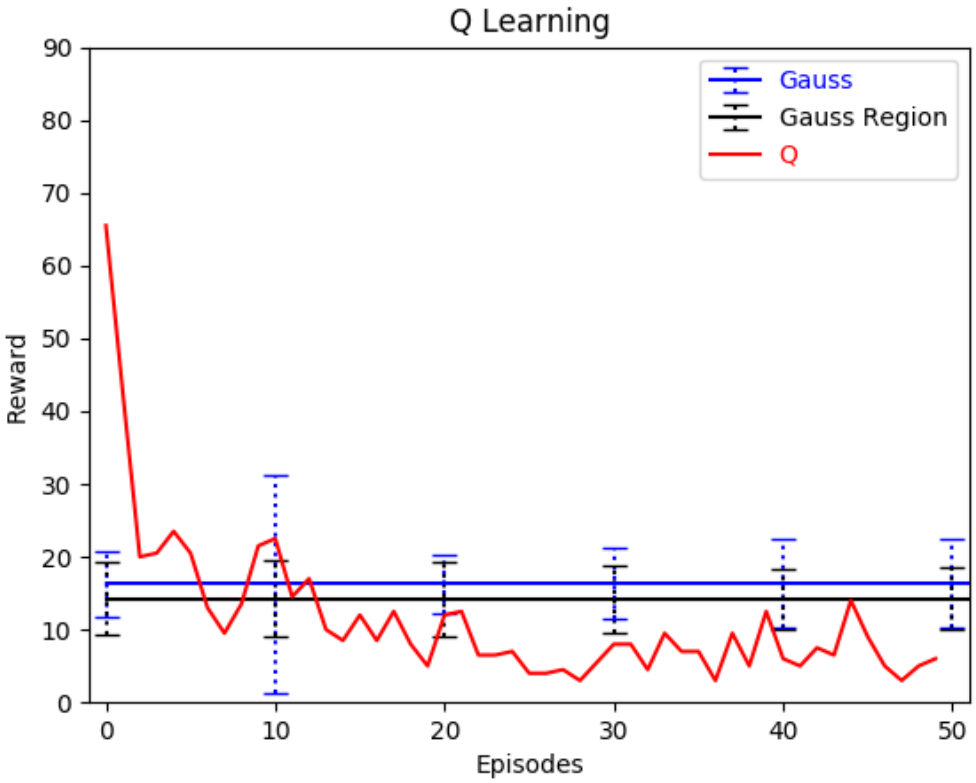


Figura 35. Q Learning en Breakout

Como se puede apreciar en la Figura 35, la línea de rojo representa la recompensa obtenida por el juego al entrenar con Q-Learning al agente para atacar al sistema en el mejor estado durante el proceso de test. Por otro lado, se muestran las medias de las recompensas obtenidas por el agente durante el proceso de entrenamiento de la red neuronal profunda en color azul y negro junto con sus desviaciones típicas. La diferencia que hay entre estas dos líneas rectas es que la azul representa la recompensa media cuando se aplica el filtro de Gauss a cada observación sin ningún tipo de restricción, y la negra representa la recompensa media obtenida por el agente cuando se aplica el filtro de Gauss sólo cuando la pelota está en la región definida en la Figura 23 y además se está acercado a la raqueta.

También se puede apreciar una convergencia por parte de la curva roja hacia el episodio siete con una recompensa de más o menos diez de media. Por lo que, se puede decir que esta curva que representa la recompensa obtenida por el juego entrenando al agente a atacar o no según el estado en el que esté, afecta al sistema de igual manera que lo hacen los ataques de Gauss bien en cada observación, o bien en la región diseñada cuando la pelota está cayendo.

Se puede comprobar que los resultados del proceso de Aprendizaje por Refuerzo Profundo con redes neuronales profundas descritos en las Secciones 4.1.1 y 4.1.2 y del proceso de Q-Learning mediante la actualización de la tabla Q, son similares aunque se puede observar como el ataque basado en refuerzo obtiene un resultado ligeramente mejor que los ataques anteriores.

Durante este proceso de prueba se reduce el rendimiento del sistema como se muestra en la Figura 35 aplicando el filtro de Gauss, apoyándose en los valores actualizados de la tabla Q, para saber en qué estado realizar la mejor acción posible. De igual manera que se incrementa el número de ataques con éxito durante el transcurso de los episodios, como se muestra en la siguiente Figura 36:

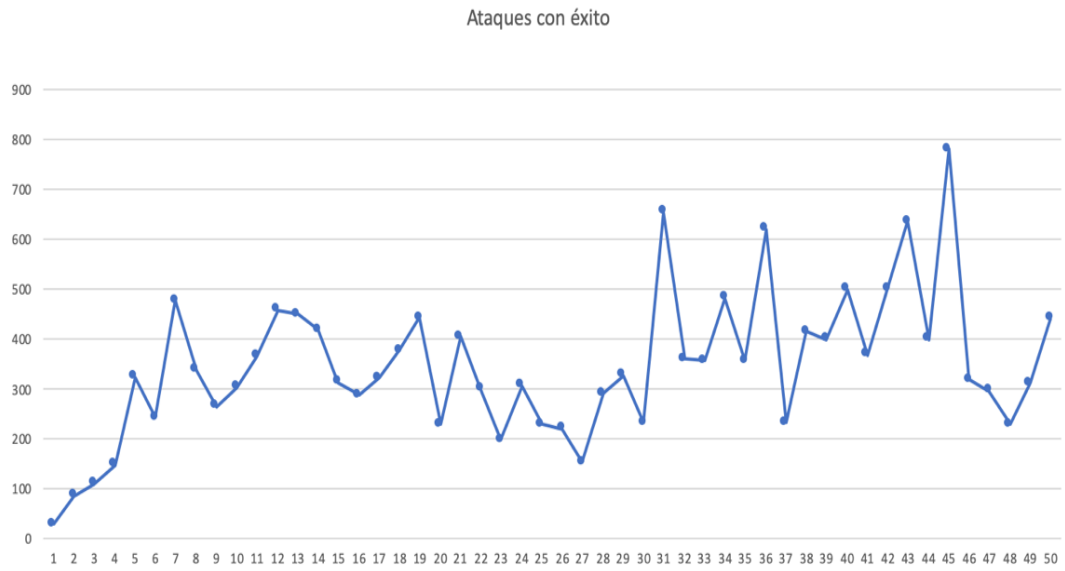


Figura 36. Ataques con éxito Q Learning en Breakout

Se puede observar en la Figura 36 una tendencia creciente en el número de ataques efectivos, es decir, ataques que han hecho equivocarse de acción a la raqueta del juego. Por lo tanto, se puede decir que el aprendizaje que ha tenido el agente para atacar de mejor forma para cada estado, ha sido efectivo y ha hecho que el sistema disminuya su rendimiento notablemente.

Por último, cabe destacar que el número de ataques realizados de media por episodio aplicando el filtro de Gauss en cada observación es de 1520, frente a los 120 que se realizan cuando se aplica Gauss sólo cuando la pelota se encuentra en la región determinada y además se está acercando a la raqueta. En cambio, el ataque basado en refuerzo ataca 541 veces de media por episodio.

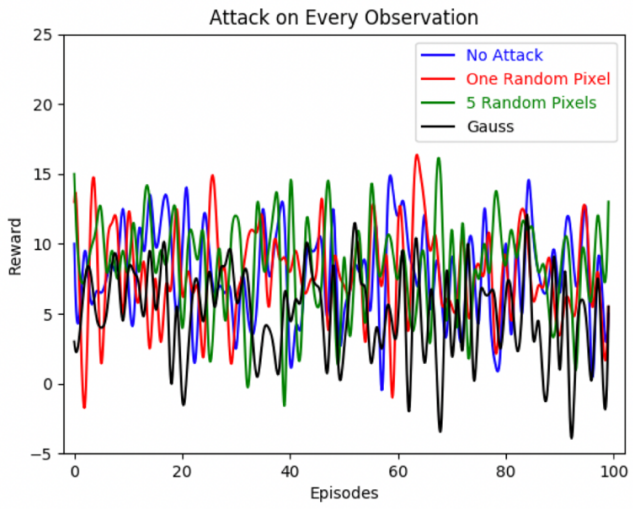
4.2 – Resultados en Pong

Como en el caso de la Sección 4.1, los resultados en el juego Pong se van a dividir en tres ataques diferentes: ataque en cada observación, ataque en una zona específica (descrito en la Sección 3.2), y el ataque basado en aprendizaje por refuerzo (descrito en la Sección 3.3).

4.2.1 – Ataque en cada observación

Los resultados obtenidos al atacar a la red neuronal profunda modificando un pixel, cinco píxeles, y aplicando el filtro de Gauss, teniendo en cuenta que se aplican a cada observación, se reflejan en la siguiente Figura 37:

A



B

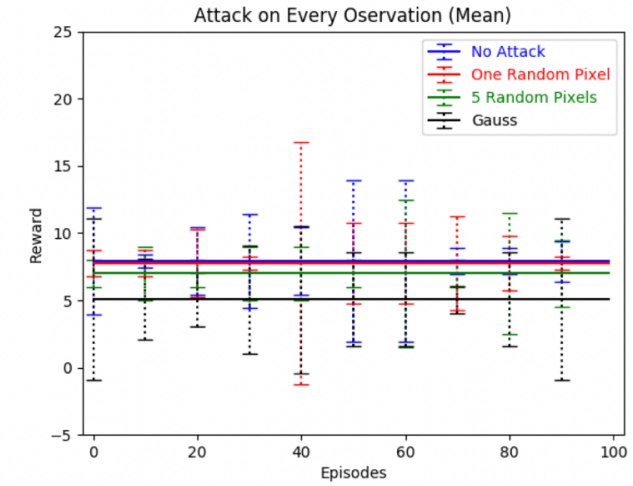


Figura 37. Ataque en Cada Observación y Media en Pong

La Figura 37 A muestra nuevamente la recompensa obtenida por el agente durante cien episodios cuando éste es atacado mediante cuatro técnicas diferentes: no aplicar ningún ataque, modificar un pixel aleatorio, modificar cinco píxeles aleatorios o aplicar el filtro de Gauss.

Como en la Figura 37 A no se puede observar claramente el rendimiento de cada ataque ya que están las líneas superpuestas entre sí. En la Figura 37 B se puede apreciar la recompensa media obtenida por el agente al haberse aplicado cada tipo de ataque mencionado anteriormente, además de sus desviaciones típicas.

En la Figura 37 B se aprecia que el ataque que más afecta a la red neuronal profunda es la aplicación del filtro de Gauss, ya que, como se explicaba anteriormente en este documento, el objetivo del filtro de Gauss es no percibir apenas la modificación a simple vista de la imagen u observación original de entrada, pero sí dañar lo máximo posible el sistema disminuyendo notablemente su rendimiento, como se aprecia en la Figura 37 B. Cabe destacar también, que los ataques en 1 y 5 píxeles escogidos de forma aleatoria, apenas afectan a la red neuronal. Por lo tanto, en este juego, la red neuronal es menos sensible a estos ataques.

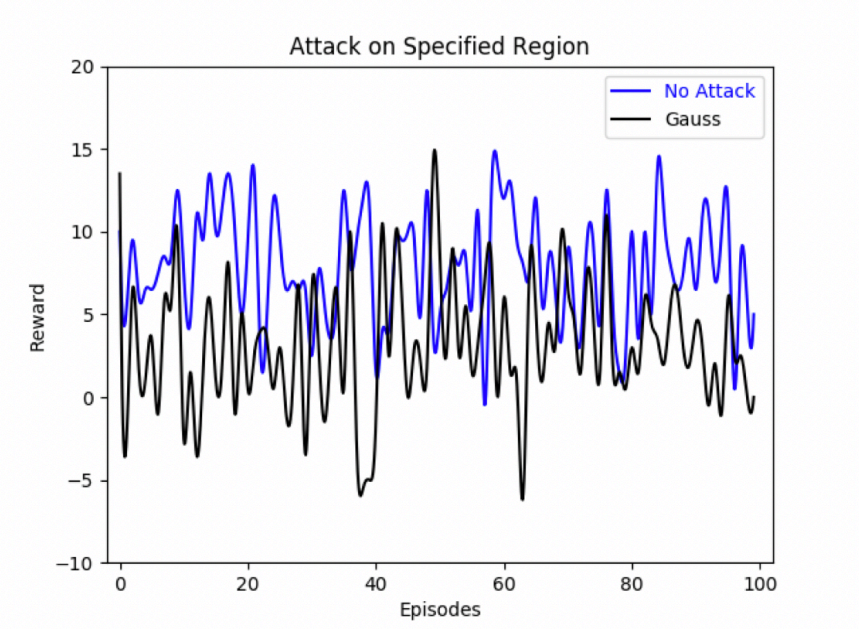
Como se describía anteriormente, los resultados obtenidos tanto en la Figura 37 A como en la Figura 37 B se refieren a cuando se ataca a la red neuronal profunda en cada observación sin ningún tipo de restricción. En la Sección 4.2.2 se mostrarán los resultados del ataque definido en la Sección 3.2.

4.2.2 – Ataque en la región específica

Esta vez se aplicará el ataque de Gauss sólo en la región detallada por la Figura 23 y además cuando la pelota se esté acercando a la raqueta, ya que, como se pudo comprobar en la sección anterior, este filtro de Gauss es el ataque que más afecta al sistema y que menos se percibe a simple vista.

Los resultados obtenidos de este proceso, teniendo en cuenta en este caso que se aplican sólo cuando se cumplen las condiciones detalladas anteriormente, se reflejan en la siguiente Figura 38:

A



B

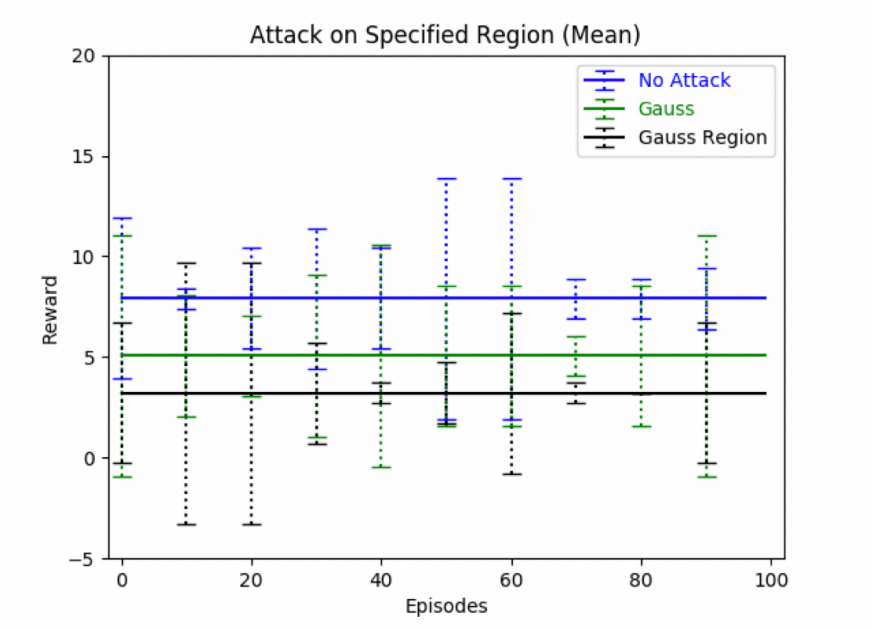


Figura 38. Ataque en la Región Específica y Media en Pong

La Figura 38 A representa la recompensa obtenida por el agente cuando no se aplica ningún ataque y cuando se aplica el filtro de Gauss. Estos resultados se obtienen al aplicarse este filtro cuando la pelota se encuentra en la región detallada en la Figura 23, y además acercándose a la raqueta.

Por otro lado, en la Figura 38 B se muestra la media de las ejecuciones de cien episodios junto con sus desviaciones estándar.

Nuevamente, este resultado es de gran significado, ya que, se aprecia en la Figura 38 B (con la línea negra) que el sistema es alterado incluso más que si se ataca en cada observación (como representa la línea verde). De esta forma, en este ataque que hemos diseñado hemos conseguido reducir el número de ataques y dañar el sistema más que atacando en cada observación, lo que implica que se reduce la probabilidad de ser detectado.

Cabe destacar que el número de ataques realizados de media por episodio en la región específica del juego Pong cuando la pelota se está acercando a la raqueta es de 861 frente a los 3697 realizados en el caso de atacar en cada observación. Si nos fijamos, el porcentaje de ataques se reduce en más de un 75% en cada episodio, consiguiendo así tener muchas menos probabilidades de ser descubierto.

4.2.3 – Ataque Basado en Aprendizaje por Refuerzo

Este ataque es el descrito en la Sección 3.3 de este documento. Como se describía en esa sección, en este ataque, el entrenamiento del agente atacante se lleva a cabo mediante la actualización de la tabla Q, la cual modifica los valores de sus celdas en cada episodio con el objetivo de converger a valores en los que el agente seleccione la mejor acción para cada estado. En nuestro caso, el agente aprende a atacar en los estados que más afecte al rendimiento del sistema, reduciendo así el número de ataques y maximizando el daño causado.

La recompensa obtenida por el juego al entrenar al agente a atacar en los estados donde más alteración provoca al sistema, se refleja en la siguiente Figura 39:

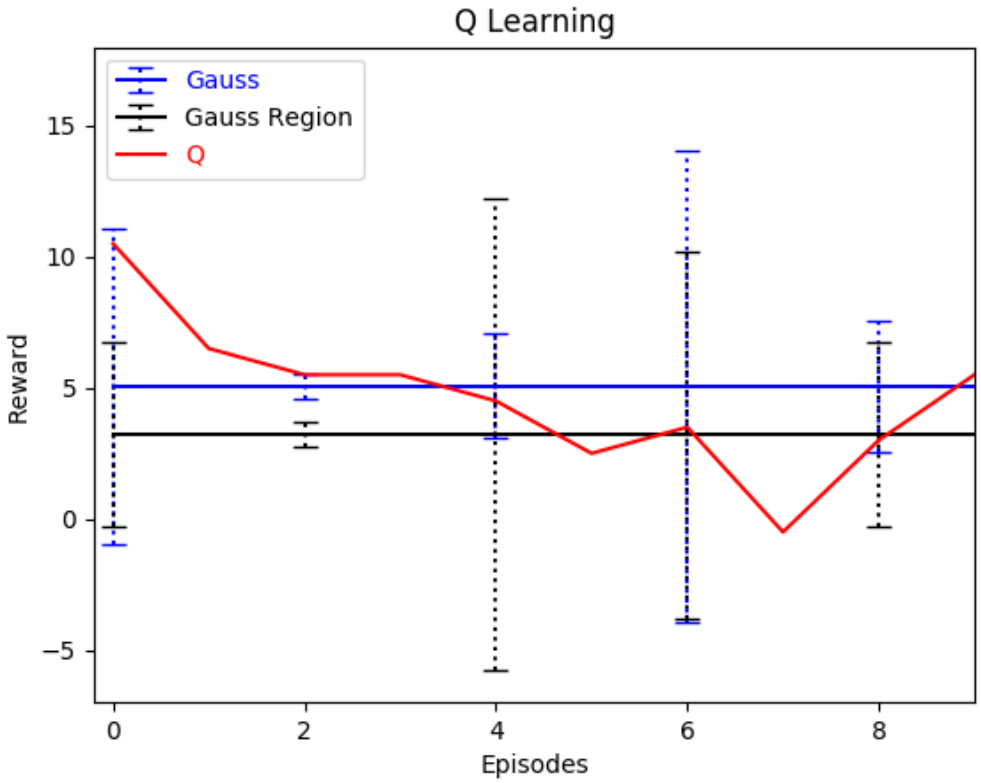


Figura 39. Q Learning en Pong

Como se puede apreciar en la Figura 39, la línea de rojo representa la recompensa obtenida por el juego al entrenar con Q-Learning al agente para atacar al sistema en el mejor estado posible. Por otro lado, se muestran las medias de las recompensas obtenidas por el agente durante el proceso de entrenamiento de la red neuronal profunda en color azul y negro junto con sus desviaciones. Como en el caso del juego Breakout, la diferencia que hay entre estas dos líneas rectas es que la azul representa la recompensa media cuando se aplica el filtro de Gauss a cada observación sin ningún tipo de restricción, y la negra representa la recompensa media obtenida por el agente cuando se aplica el filtro de Gauss sólo cuando la pelota está en la región definida en la Figura 23 y además se está acercado a la raqueta.

También se puede apreciar una tendencia negativa en la curva roja, convergiendo a un valor de media de cuatro. Por lo que, se puede decir que esta curva que representa la recompensa obtenida por el juego entrenando al agente a atacar o no según el estado en el que esté, afecta al sistema de igual manera que lo hacen los ataques de Gauss bien en cada observación, o bien en la región diseñada cuando la pelota está cayendo.

Se puede comprobar que los resultados del proceso de Aprendizaje por Refuerzo Profundo con redes neuronales profundas descritos en las Secciones 4.2.1 y 4.2.2 y del proceso de Q-Learning mediante la actualización de la tabla Q, son similares aunque se puede observar como el ataque basado en aplicar el filtro de Gauss cuando la pelota está acercándose a la raqueta y además se encuentra dentro de la región específica obtiene un resultado ligeramente mejor que los otros dos ataques.

Durante este proceso de prueba se reduce el rendimiento del sistema como se muestra en la Figura 39 aplicando el filtro de Gauss, apoyándose en los valores actualizados de la tabla Q, para saber en qué estado realizar la mejor acción posible. De igual manera que se incrementa el número de ataques con éxito durante el transcurso de los episodios, como se muestra en la siguiente Figura 40:

Figura 40. Ataques con éxito Q Learning en Pong

Se puede observar en la Figura 40 una clara tendencia creciente en el número de ataques que han hecho equivocarse de acción a la raqueta del juego. Por lo tanto, se puede decir que el aprendizaje que ha tenido el agente para atacar de mejor forma para cada estado, ha sido efectivo y ha hecho que el sistema disminuya su rendimiento notablemente.

Por último, cabe destacar, como en el caso del Breakout, que el número de ataques realizados de media por episodio aplicando el filtro de Gauss en cada observación es de 3697, frente a los 861 que se realizan cuando se aplica Gauss sólo cuando la pelota se encuentra en la región determinada y además acercándose a la raqueta. En cambio, el ataque basado en refuerzo ataca 2535 veces de media por episodio.

5. PLANIFICACIÓN Y PRESUPUESTO

5.1 – Planificación

La planificación de este proyecto se va a dividir en dos grandes bloques, ya que, inicialmente se parte de una planificación ideal a seguir, pero la realidad es que no siempre es posible adecuarse perfectamente a los tiempos de esta planificación inicial debido a que surgen imprevistos durante el transcurso del trabajo.

Esta planificación se ha llevado a cabo haciendo uso de una metodología ágil llamada SCRUM [38]. El objetivo de esta metodología es ser flexible en cuanto a la modificación continua del proyecto en el transcurso de éste. Por eso, se plantean tres entregas en el periodo de tiempo total del trabajo.

Por lo tanto, la planificación se dividirá en planificación inicial y planificación real, como muestran los siguientes diagramas de Gantt:

PLANIFICACIÓN INICIAL

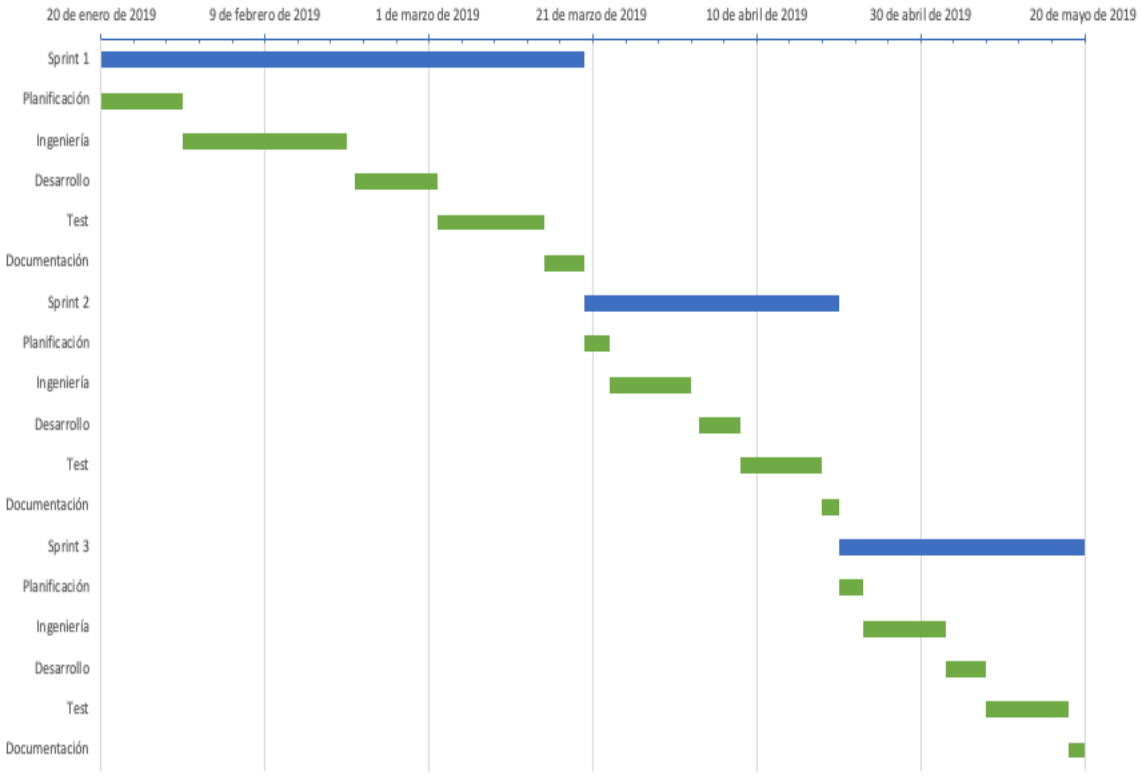


Figura 41. Diagrama de Gantt Planificación Inicial

Como se puede apreciar en la Figura 41, el diagrama de Gantt está planificado para realizar las tareas en cuatro meses justos (del 20 de Enero al 20 de Mayo). Además, la planificación se divide en tres *sprints* o entregas principales que están representadas mediante las barras azules, las cuales engloban las tareas de cada entrega, como muestran las barras verdes.

Cada entrega se divide, a su vez, en planificación (organización de tareas), ingeniería (diseño de algoritmos y código), desarrollo (implementación de los algoritmos diseñados), test (pruebas del código implementado), y documentación (formalización de los resultados obtenidos durante las pruebas).

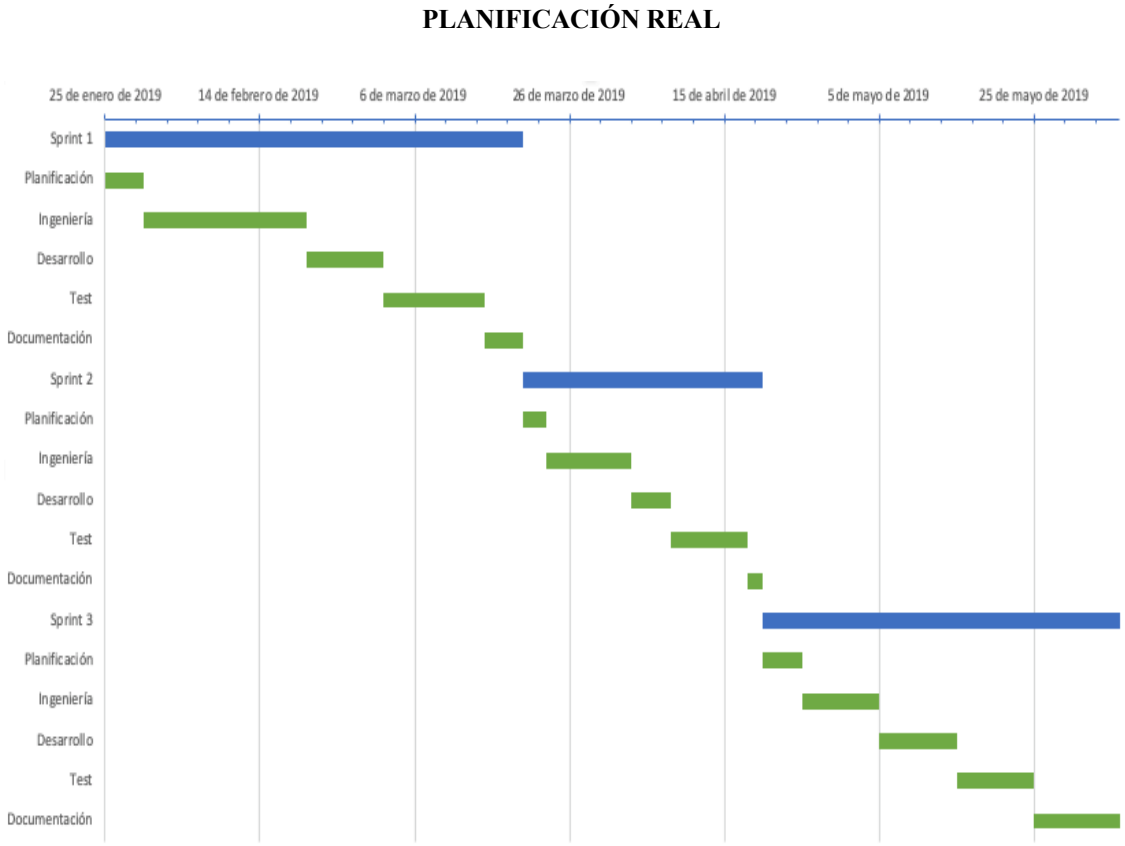


Figura 42. Diagrama de Gantt Planificación Real

Como se puede observar en la Figura 42, el comienzo del trabajo (25 de Enero) y el final de este (5 de Junio) se han retrasado debido a imprevistos, al igual que las tareas de este se han modificado, de manera, que unas tareas han llevado más tiempo y otras menos según la cantidad y el tipo de imprevisto ocasionado.

5.2 – Presupuesto

En esta sección se detalla el presupuesto del trabajo, el cual define los costes generados por la investigación del proyecto.

La siguiente Tabla 2 muestra el presupuesto total del proyecto.

Duración Total	5 meses
Presupuesto Total	55.000 €

Tabla 2. Presupuesto Total Proyecto

En la Tabla 3 se representan los costes asociados al equipo o personal del proyecto. El jefe de proyecto sería el tutor mientras que el resto de personal sería el estudiante autor del trabajo.

Rol	Coste
Jefe de Proyecto	10.560 (€/mes)
Analista	7.040 (€/mes)
Programador	3.520 (€/mes)

Tabla 3. Costes de Equipo

En la Tabla 3 los costes generados por el personal del equipo se han calculado de la siguiente manera: El jefe de proyecto cobra 60 € la hora, por 8 horas al día de la jornada completa, por 22 días laborales de trabajo, se obtiene un coste total de 10.560 € para este perfil; El analista cobra 40 € la hora, por 8 horas al día, por 22 días laborales, se obtiene un coste total de 7.040 €; Y el programador del código cobrará 20 € la hora, por 8 horas al día, por 22 días laborales al mes, resultando 3.520 € el coste total de este perfil de personal.

La Tabla 4 muestra los costes relacionados con las herramientas de trabajo necesarias para el equipo del proyecto.

Herramienta	Coste
IMac 21’ 16 GB RAM 512 SSD	2.200 €
MacBook Pro 13’ 8 GB RAM 256 SSD	2.000 €

Tabla 4. Costes de Material

El presupuesto no es elevado, ya que, es una investigación y no se utilizaría de cara a la venta al cliente, sino con fines de avanzar en la tecnología y en muchas aplicaciones reales a las que este proyecto puede ser destinado.

6. MARCO REGULADOR

6.1 – Riesgos

Posibles riesgos de la Inteligencia Artificial pueden ser los ataques que pueden ocasionar grandes daños en los sistemas y que estos se comporten de forma anómala.

Como se ha estudiado en este proyecto, los sistemas que cuentan con esta tecnología se encuentran expuestos a infinidad de tipos de ataque, por lo que, son sistemas muy vulnerables, y por ello, se necesita estudiar e investigar a fondo los tipos de ataque posibles para contar con una solución a cada uno de ellos.

6.2 – Responsabilidades éticas

La Inteligencia Artificial está jugando un papel muy importante en la tecnología, ya que, como se ha visto anteriormente en este documento, es aplicable a infinidad de sectores, proporcionando automatización y eficiencia a diversos sistemas, mejorando el rendimiento de estos exponencialmente.

Pero también se plantea si estos sistemas son sustitutos del humano, dejando al humano sin trabajo en muchos casos. Incluso no sólo podrían quitar el puesto laboral a los humanos, sino también dañar a estos teniendo un comportamiento erróneo o diseñado incluso con un propósito maligno.

Por eso, es imprescindible la regulación de esta tecnología en casos en los que el humano puede ser de ayuda y también protegiéndolo de posibles comportamientos anómalos.

La regulación de estos sistemas se refiere al control que debe haber por los organismos reguladores de esta tecnología aplicada a numerosos sectores. El objetivo de estos organismos es hacer el seguimiento del proceso y diseño de estos sistemas destinados a las aplicaciones reales existentes, para así proporcionar seguridad al ser humano en el ámbito laboral garantizando un trabajo, y también garantizando seguridad personal.

6.3 – Seguridad

La seguridad que se tiene que tener en cuenta ante comportamientos anómalos o ataques de cualquier tipo en estos sistemas tiene que ser exhaustiva, ya que se encuentra en juego la vida de los seres humanos debido a la potencia de estos sistemas en muchas aplicaciones reales.

La Figura 43 muestra los niveles de autonomía que se encuentran legalizados a día de hoy en el ámbito de los coches autónomos.

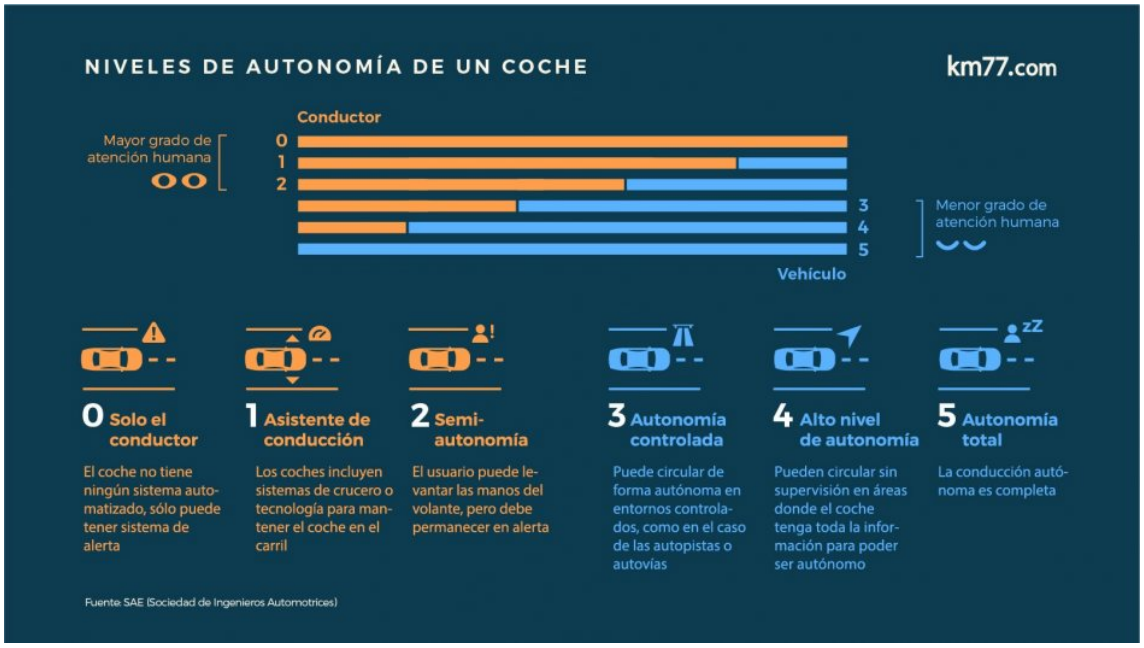


Figura 43. Niveles de Autonomía de un Coche [39]

En la Figura 43 se representan los niveles de conducción autónoma posibles estandarizados de 0 a 5 de menor a mayor en cuanto a autonomía se refiere, teniendo el nivel cero una autonomía nula en la que el conductor no dispone de ningún sistema automatizado, y el nivel cinco una autonomía completa en la que el vehículo está completamente automatizado sin necesitar la conducción o atención del conductor en ningún momento.

Hoy en día, el nivel legal máximo de autonomía se encuentra en el cuatro debido a alguna incidencia producida por estos sistemas ya mencionada anteriormente en el documento presente [13].

Por lo tanto, se necesita un estudio más exhaustivo de esta tecnología para proporcionar la seguridad íntegra del ser humano en diversas aplicaciones reales.

6.4 – Estándares técnicos

La Inteligencia Artificial avanza de forma continua y afecta a gran parte de la industria. La sociedad se ve afectada ante este cambio de la tecnología, ya que, cambian los negocios, la producción y los perfiles buscados en el mundo laboral. Así, afecta a multitud de sectores en los que la estandarización juega un papel muy importante: fabricación inteligente, robótica, vehículos autónomos ya mencionados, sanidad, reconocimiento visual o ciberseguridad. En todos estos sectores existen unos estándares que deberán ser actualizados para incorporar esta nueva tecnología.

El sistema europeo de estandarización es un elemento esencial para garantizar un enfoque centrado en las personas para la Inteligencia Artificial, asegurando que se beneficia a la sociedad. Los organismos europeos de normalización, CEN [40] y CENELEC [40] trabajan conjuntamente con las organizaciones internacionales ISO [41] e IEC [42], para llevar a cabo esta regulación y normalización sobre el uso de esta tecnología.

7. ENTORNO SOCIO-ECONÓMICO

7.1 – Impacto social

El impacto que este proyecto genera en la sociedad es de gran envergadura, ya que, esta investigación se centra en la vulnerabilidad de los sistemas frente a ataques de todo tipo. Existen una gran cantidad de aplicaciones reales en las que es necesario estudiar el comportamiento que tienen los sistemas autónomos frente a ataques, para poder defenderse lo mejor posible y que no aparezcan daños colaterales.

Como se ha detallado anteriormente en este documento, la rama de la Inteligencia Artificial investigada en este proyecto tiene infinidad de aplicaciones reales como la conducción autónoma, la medicina, los robots, la industria, etc. Además, estas aplicaciones son todas de mucha ayuda para el ser humano, haciéndole más fácil su vida.

Por lo tanto, el impacto social que tiene la Inteligencia Artificial, y más concretamente la rama investigada en este trabajo, es positivo, ya que está pensada para la eficiencia y la automatización de procesos y los posibles mecanismos de defensa frente a estos ataques, lo cual es de gran ayuda para el ser humano que busca cada vez sistemas más autónomos y seguros.

Figura 44. Impacto Social Inteligencia Artificial [43]

7.2 – Impacto económico

Estas técnicas de la Inteligencia Artificial están presentes en numerosas empresas como las que viven hoy en día de la tecnología, y cada día más empresas se suman a la utilización de estos algoritmos para la automatización de procesos.

Uno de los sectores más importantes que están innovando con esta tecnología es el de los coches autónomos, ya que, se busca la conducción autónoma sin la supervisión de un humano, simplemente un transporte en el cual te transporte de un lugar a otro sin que se necesite estar pendiente de la carretera y del volante.

Otro sector muy importante en el que se está invirtiendo en investigación, es la medicina, debido a que con este tipo de tecnología se pueden salvar muchas vidas encontrando patrones en diversas características del humano, de manera que, se diagnostique una enfermedad con mucha antelación con respecto a los diagnósticos de enfermedades que se practican hoy en día.

La automatización en la industria presenta grandes eficiencias y ahorros en costes de maquinaria y personal. Por lo que, el impacto económico en este sector es notable.

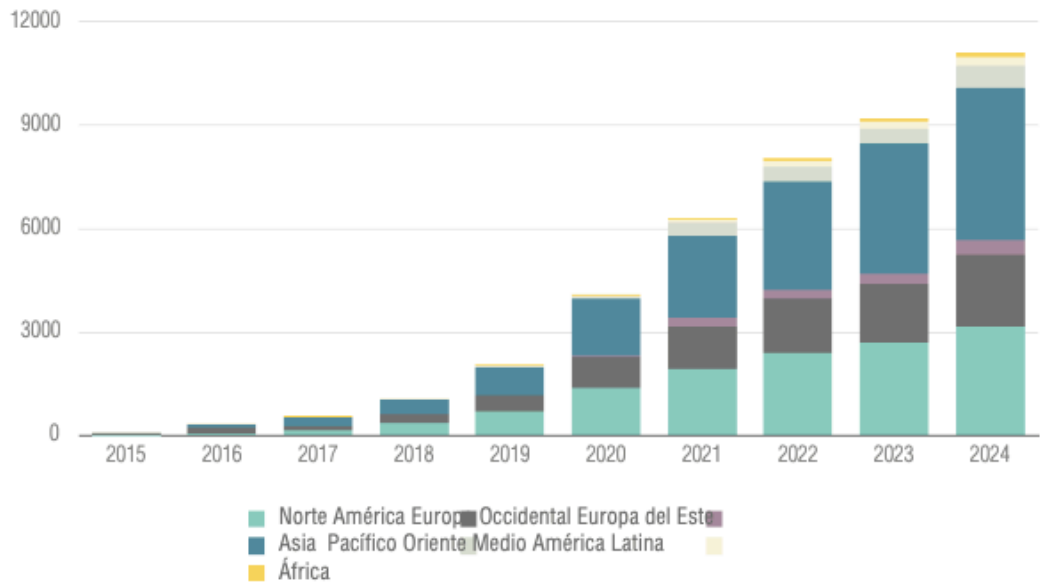


Figura 45. Rendimientos empresariales derivados de la IA (En millones de dólares) [44]

7.3 – Impacto medioambiental

En la agricultura hoy en día ya existen maquinarias que recorren los campos y realizan sus labores sin nadie que los conduzca. También se utilizan sensores de monitorización de riego, enviando la información a tiempo real a los dispositivos móviles para que todo esté controlado y el propietario de este sistema pueda monitorizar la calidad de su producción. El objetivo de este control es el ahorro de agua y comercializar mejor los productos obtenidos.

El *blockchain* [45] tendrá un alto impacto en disminuir el desperdicio de alimentos. Esto se debe a que la información en esta cadena se presenta de forma clara pudiendo controlar el estado de cada alimento en cada eslabón por el que pasa este.

No sólo se ahorra en agua, sino que también se ahorra en energía, ya que, con estos sensores enviando información a tiempo real, el consumidor puede determinar los momentos en los que debe hacer uso de la electricidad y cuando no, a diferencia de hacer uso de ella cuando se cree que se debe utilizar.

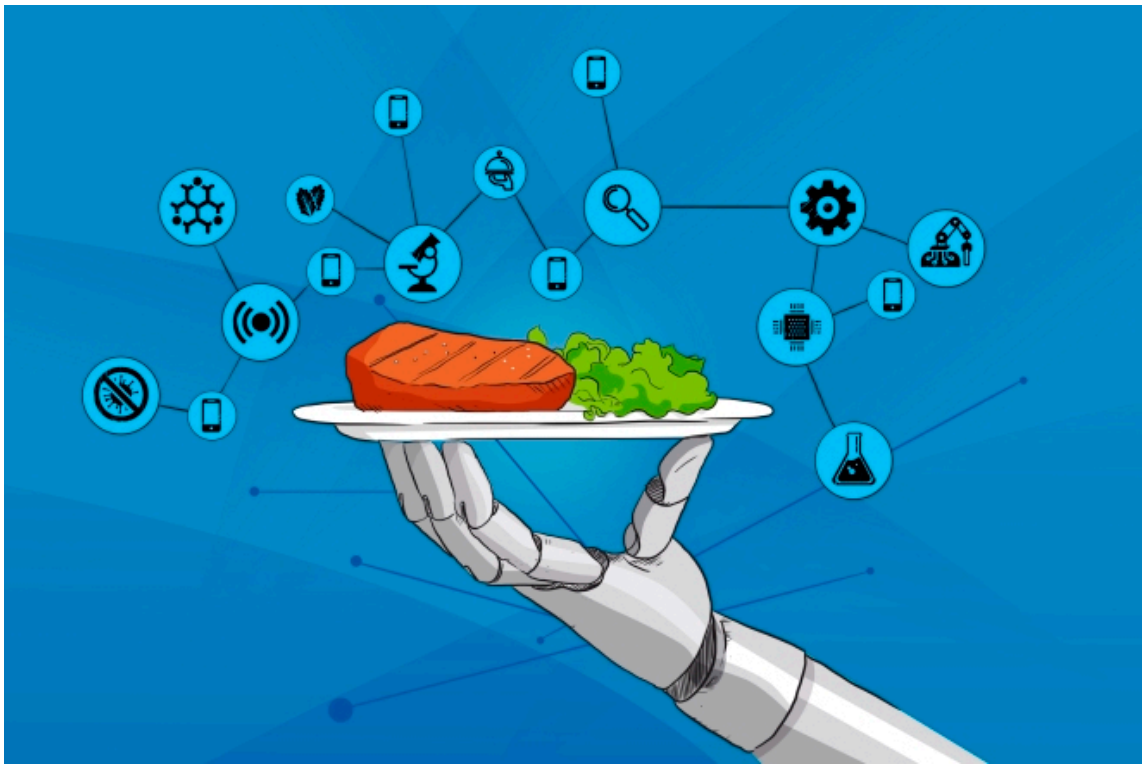


Figura 46. Impacto medioambiental [46]

8. CONCLUSIONES Y TRABAJOS FUTUROS

8.1 – Conclusiones

La investigación exhaustiva que se ha llevado a cabo en este proyecto genera diversos resultados de los que se pueden sacar conclusiones muy positivas para el avance de esta tecnología, pudiendo experimentar el efecto que causa en el sistema de cada juego los ataques detallados anteriormente en este documento.

En primer lugar, hay ligeras diferencias entre los resultados del juego Breakout y del juego Pong. Pero lo más importante es que en los dos casos el sistema disminuye su rendimiento notablemente al aplicar diferentes tipos de ataque.

En cuanto al juego Breakout, como se aprecia en los resultados (Sección 4.1), destaca la disminución de más de un 90% en los ataques realizados cuando la pelota está cayendo y dentro de la región representada en la Figura 23 frente a los ataques realizados en cada observación. Por lo tanto, se puede decir que es un resultado importante, ya que se reduce sustancialmente la probabilidad de ser detectado, y además, se consigue disminuir incluso un poco más el rendimiento del sistema que si se ataca en cada observación. En especial, destacan los resultados de la aplicación del filtro de Gauss a la observación, debido a que es el ataque que menos se aprecia a simple vista y es el que más afecta al rendimiento de la red neuronal profunda. En cuanto a los resultados obtenidos mediante aprendizaje por refuerzo, destacan la disminución que hay en número de ataques frente al número de ataques que se realizan al aplicar estos en cada observación, y también, el efecto que causa este método en el sistema, apreciando la bajada de rendimiento de éste de media, incluso, por debajo de todos los demás ataques.

Por otro lado, en el caso del juego Pong, como se puede apreciar en la Sección 4.2, destaca nuevamente la reducción de más de un 75% en los ataques realizados cuando la pelota se encuentra dentro de la región específica y además acercándose a la raqueta frente a los ataques realizados en cada observación. Por lo que, este resultado es de gran significado, ya que, se reduce notablemente la probabilidad de ser descubierto, y a su vez, el rendimiento del sistema más que si se ataca en cada observación. De igual manera que en el juego Breakout, el ataque estrella, es decir, el que afecta sustancialmente a la red neuronal profunda sin apreciarse apenas a simple vista en la imagen modificada, es el filtro de Gauss. En los resultados obtenidos aplicando Q Learning haciendo que el agente

aprenda cuándo atacar y cuándo no, se destacan la reducción del número de ataques realizados frente al número de ataques que se realizan aplicando estos en cada observación, y también, el efecto que causa este método en el sistema, apreciando la bajada de rendimiento de éste de media, aunque en este juego el sistema se presenta menos sensible ante algunos ataques que en el juego Breakout.

Cabe destacar, tanto en el juego Breakout como en el juego Pong, el aumento de ataques con éxito que realiza el agente enseñado mediante Q Learning a medida que transcurren los episodios, apreciando claramente un aumento de éstos, los cuales hacen que el juego se equivoque de acción y no ejecute la que debería realizar en un comportamiento normal del juego.

Por último, se puede concluir este proyecto aclarando que, después de detallar las conclusiones descritas anteriormente, la investigación de este proyecto ha sido un éxito para el avance de la tecnología basada en Inteligencia Artificial. Y por lo tanto, se ha cumplido con el objetivo principal de este estudio, cuya finalidad es extrapolar estos resultados realizados en video-juegos a todas las aplicaciones reales posibles detalladas en la Sección 2.4.2 de este documento.

Queda en evidencia en este trabajo, por tanto, que las redes neuronales en Deep Learning son propensas a grandes caídas de rendimiento incluso con ataques que no son apreciables a simple vista, y que aparentemente son muy sencillos de llevar a cabo. Es necesario, por tanto, investigar mecanismos de defensa sobre todo cuando se aplican a situaciones reales. En este trabajo, se han presentado resultados en varios juegos, en los que la caída de este rendimiento por ataques implica perder una partida. No obstante, en la vida real, estos ataques pueden implicar la pérdida de una vida.

8.2 – Trabajos Futuros

Después de haber estudiado a fondo el efecto que producen en el comportamiento del sistema los ataques diseñados en este documento, se plantean diferentes soluciones a estas vulnerabilidades.

Como se comentaba anteriormente en este trabajo, es necesario conocer detalladamente los ataques propuestos para poder defenderse de ellos, ya que es el objetivo principal de los desarrolladores de estos tipos de sistemas de Inteligencia Artificial implementados en las aplicaciones reales.

Por lo tanto, como trabajo futuro posible a esta investigación podría ser el estudio exhaustivo de posibles defensas ante estos ataques detallados a lo largo de este proyecto.

Existen varios tipos de defensas, como los mencionados en la Sección 2.5 en este documento. Éstos sólo son unos cuantos ejemplos de la infinidad de defensas que existen para controlar posibles ataques al sistema, ya que, como se comentaba anteriormente, para paliar cada tipo de ataque y poder controlar todos éstos se necesitan muchas defensas, lo cual no es trivial.

BIBLIOGRAFÍA

[1] Stuart J. Russell y Peter Norvig. *Artificial intelligence: a modern approach*. Prentice Hall, 1995. Disponible en: https://www.cin.ufpe.br/~tfl2/artificial-intelligence-modern-approach.9780131038059.25368.pdf

[2] T. Mitchell. *Machine Learning*. McGraw Hill, 1997. Disponible en: http://profsite.um.ac.ir/~monsefi/machine-learning/pdf/Machine-Learning-Tom-Mitchell.pdf

[3] Z. Jin, Y. Sun, y AC. Cheng. *Predicting cardiovascular disease from real time electrocardiographic monitoring: An adaptive machine learning approach on a cell phone.* Minneapolis, USA, 2009. Disponible en: https://www.ncbi.nlm.nih.gov/pubmed/19964449

[4] V. Talpaert, I. Sobh, B. Kiran, P. Mannion, S. Yogamani, A. El-Sallab, y P. Perez. *Exploring applications of deep reinforcement learning for real-world autonomous driving systems.* Bangalore, 2019.

[5] J. Kober, A. Bagnell, y J. Peters. *Reinforcement Learning in Robotics: A Survey.* Bielefeld University, Germany, 2013. Disponible en: https://www.ias.informatik.tu-darmstadt.de/uploads/Publications/Kober_IJRR_2013.pdf

[6] M. Hossain, M. Al-Hammadi, y G. Muhammad. *Automatic Fruit Classification Using Deep Learning for Industrial Applications.* IEEE Transactions on Industrial Informatics, 2018. Disponible en: https://ieeexplore.ieee.org/document/8488544

[7] R. Trippi y E. Turban. *Neural Networks in Finance and Investing: Using Artificial Intelligence to Improve Real World Performance.* San Diego, USA, 1993. Disponible en: https://www.researchgate.net/publication/234797465_Neural_Networks_in_Finance_and_Investing_Using_Artificial_Intelligence_to_Improve_Real_World_Performance

[8] M. Wani. *Introduction to Deep Learning*. Springer Singapore, 2019. Disponible en: https://www.researchgate.net/publication/331790067_Introduction_to_Deep_Learning

[9] K. Arulkumaran, P. Deisenroth, M. Brundage, y A. Bharath. *Deep Reinforcement Learning: A brief survey.* IEEE Signal Processing Magazine, 2017. Disponible en: https://www.gwern.net/docs/rl/2017-arulkumaran.pdf

[10] Atari: *Reach high scores in Atari 2600*. Recuperado de: https://gym.openai.com/envs/#atari, 14 de Junio 2019.

[11] Atari: *Breakout-v0*. Recuperado de: https://gym.openai.com/envs/Breakout-v0/, 14 de Junio 2019.

[12] Atari: *Pong-v0*. Recuperado de: https://gym.openai.com/envs/Pong-v0/, 14 de Junio 2019.

[13] El País: *Tesla reconoce otro accidente mortal en un vehículo que circulaba con piloto automático*. Recuperado de: https://elpais.com/tecnologia/2019/05/17/actualidad/1558075375_210626.html, 14 de Junio 2019.

[14] Wikipedia: *Esquema flujo Aprendizaje Automático*. Recuperado de: https://es.wikipedia.org/wiki/Aprendizaje_autom%C3%A1tico#/media/File:Esquema_aprendizaje.jpg, 14 de Junio 2019.

[15] Blog Logicalis: *Learning machine, los usos del aprendizaje supervisado*. Recuperado de: https://blog.es.logicalis.com/analytics/learning-machine-los-usos-del-aprendizaje-supervisado, 14 de Junio 2019.

[16] Ligdi González: *Aprendizaje No Supervisado: K-Means Clustering*. Recuperado de: http://ligdigonzalez.com/aprendizaje-no-supervisado-k-means-clustering/, 14 de Junio 2019.

[17] SmartPanel: *¿Qué es el Deep Learning?*. Recuperado de: https://www.smartpanel.com/que-es-deep-learning/, 14 de Junio 2019

[18] Planetachatbot: *Deep Learning fácil con DeepCognition*. Recuperado de: https://planetachatbot.com/deep-learning-f%C3%A1cil-con-deepcognition-9af43b2319ba, 14 de Junio 2019

[19] Social Geek: *Con 'Deep Learning' Apple logró que Siri suene menos robótica*. Recuperado de: https://socialgeek.co/tech/deep-learning-apple-logro-que-siri-suene-menos-robotica/, 14 de Junio 2019.

[20] Medium: *How to easily do Object Detection on Drone Imagery using Deep Learning*. Recuperado de: https://medium.com/nanonets/how-we-flew-a-drone-to-monitor-construction-projects-in-africa-using-deep-learning-b792f5c9c471, 14 de Junio 2019.

[21] Sigmoidal: *Deep Learning chatbot – analysis and implementation*. Recuperado de: https://sigmoidal.io/chatbots-for-b2c-and-deep-learning/, 14 de Junio 2019.

[22] Packtpub: *Admiring the many faces of Facial Recognition with Deep Learning*. Recuperado de: https://hub.packtpub.com/admiring-many-faces-facial-recognition-deep-learning/, 14 de Junio 2019.

[23] Health IT Analytics: *Deep Learning Malware Can Fake Cancer on Medical Images*. Recuperado de: https://healthitanalytics.com/news/deep-learning-malware-can-fake-cancer-on-medical-images, 14 de Junio 2019.

[24] R. Sutton y A. Barto. *Reinforcement Learning: An Introduction.* Bradford, 2017. Disponible en: http://incompleteideas.net/book/bookdraft2017nov5.pdf

[25] Skymind. *A Beginner's Guide to Deep Reinforcement Learning*. Recuperado de: https://skymind.ai/wiki/deep-reinforcement-learning, 14 de Junio 2019.

[26] FreeCodeCamp. *An introduction to Q-Learning: reinforcement learning*. Recuperado de: https://medium.freecodecamp.org/an-introduction-to-q-learning-reinforcement-learning-14ac0b4493cc, 14 de Junio 2019.

[27] V. Mnih, A. Graves, D. Silver. *Playing Atari with Deep Reinforcement Learning.* Toronto, 2013. Disponible en: https://www.cs.toronto.edu/~vmnih/docs/dqn.pdf

[28] Towards Data Science: *Deep Deterministic Policy Gradients Explained*. Recuperado de: https://towardsdatascience.com/deep-deterministic-policy-gradients-explained-2d94655a9b7b, 14 de Junio 2019.

[29] M. Vitelli y A. Nayebi. *CARMA: A Deep Reinforcement Learning Approach to Autonomous Driving.* Stanford, USA, 2016. Disponible en: https://web.stanford.edu/~anayebi/projects/CS_239_Final_Project_Writeup.pdf

[30] Blog Mapfre: *¿Cómo serán los seguros de los coches autónomos?*. Recuperado de: https://blogmapfre.com/motor/como-seran-los-seguros-de-los-coches-autonomos/, 14 de Junio 2019.

[31] Cognite Ventures: *The Cognitive Computing Startup List*. Recuperado de: http://www.cogniteventures.com/the-cognitive-computing-startup-list/, 14 de Junio 2019.

[32] Geek: *China's Robot Police Use Facial Recognition to Catch Criminals*. Recuperado de: https://www.geek.com/tech/chinas-robot-police-use-facial-recognition-to-catch-criminals-1689694/, 14 de Junio 2019.

[33] Medium: *Reinforcement Learning Applications*. Recuperado de: https://medium.com/@yuxili/rl-applications-73ef685c07eb, 14 de Junio 2019.

[34] Open AI Gym: *Gym*. Recuperado de: https://gym.openai.com/, 14 de Junio 2019.

[35] Open AI: *Attacking Machine Learning with Adversarial Examples.* Recuperado de: https://openai.com/blog/adversarial-example-research/, 14 de Junio 2019.

[36] Spectrum IEEE: *Slight Street Sign Modifications Can Completely Fool Machine Learning Algorithms*. Recuperado de: https://spectrum.ieee.org/cars-that-think/transportation/sensors/slight-street-sign-modifications-can-fool-machine-learning-algorithms, 14 de Junio 2019.

[37] Towards Data Science: *About Adversarial Examples*. Recuperado de: https://towardsdatascience.com/about-adversarial-examples-2a7a7b4d2670, 14 de Junio 2019.

[38] Proyectos Ágiles: *Qué es SCRUM*. Recuperado de: https://proyectosagiles.org/que-es-scrum/, 14 de Junio 2019.

[39] SAE. Recuperado de: https://www.20minutos.es/noticia/2825372/0/clasificiacion-coches-autonomos/

[40] CEN-CENELEC: *Making Standards for Europe*. Recuperado de: https://www.cencenelec.eu/Pages/default.aspx, 14 de Junio 2019.

[41] International Organization for Standarization: *When the world agrees*. Recuperado de: https://www.iso.org/home.html, 14 de Junio 2019.

[42] International Electrotechnical Commission: *International Standards and Conformity Assessment for all electrical, electronic and related technologies.* Recuperado de: https://www.iec.ch/renewables/standardization.htm, 14 de Junio 2019.

[43] Alainet: *Inteligencia artificial y trabajo en América Latina*. Recuperado de: https://www.alainet.org/es/articulo/198957, 14 de Junio 2019.

[44] Andrés P. y Luis M-I. *El impacto económico de la inteligencia artificial.* Universidad de Valencia, España, 2018. Disponible en: https://www.researchgate.net/publication/329840275_El_impacto_economico_de_la_inteligencia_artificial

[45] Blockchain: *Conectamos las criptomonedas con el mundo*. Recuperado de: https://www.blockchain.com/es/, 14 de Junio 2019.

[46] El Mercurio Campo: *Inteligencia Artificial: más alimentos y menos impacto ambiental*. Recuperado de: http://www.elmercurio.com/campo/noticias/noticias/2019/04/16/inteligencia-artificial-mas-alimentos-y-menos-impacto-ambiental.aspx?disp=1, 14 de Junio 2019.

ANEXO. RESUMEN EN INGLÉS

ABSTRACT

Nowadays, artificial intelligence in particular reinforcement learning, is everywhere, like autonomous driving, social robotics and surveillance systems. The problem of having so many application domains, is that it is also exposed to a multitude attacks that can cause major problems. Therefore, the study of the vulnerability of these systems to attacks is receiving more and more attention within the scientific community.

Therefore, this work focuses, precisely, on the study of the vulnerability of these systems. In particular, the main objective is to decrease the performance of a deep neural network applied to Atari games by means of two new attack types. In the first, we will try to identify areas/regions of attack, that is, those regions where attacking induces system failure more quickly. The second will be based on reinforcement learning. In this attack we will try to learn an identified attack policy, for each state in which the system can be found, if it is better to attack or not attack. In both cases, conclusions will be obtained about the exact moment in which attack affects the network the most, that is, its performance decreases more, and the number of attacks of each approach, since the objective is to maximize the damage while minimizing the number of attacks.

The first step to be able to defend yourself against an attack is to know how to attack and what types of attack there may be. Therefore, this document lays the foundations for possible future defense systems.

Key words: Machine Learning, Reinforcement, Adversary, Atari, Deep Q Learning

1. INTRODUCTION

The use of artificial intelligence has been increasing in recent years due to its wide range of real applications, process automation, and efficiency [1].

In particular, machine learning [2] is increasingly used in real environments such as medicine [3], autonomous driving [4], robotics [5], industry [6], the financial system [7], etc. Every day new applications are discovered in which machine learning plays an important role, speeding up and optimizing work.

Within the technique of Deep Learning [8], Deep Reinforcement Learning [9] has been used successfully in many environments including video games such as those that will be analyzed in this project. In particular, the well-known Atari games will be used in this project [10]. These arcade games and apparently archaic, are very helpful when it comes to studying the behavior of learning systems thanks to the OpenAI Gym interface, which allows easy access to game controls, and thus be able to make use of the techniques of machine learning in these environments.

In this paper we will focus on two of these games: Breakout [11] and Pong [12].

In both games, the objective is to slightly modify the images or observations that the neural network perceives in each game turn in order to induce the failure. That is to say, the objective of this work is to build an agent by means of different techniques whose objective is to deceive the neural network making it believe that it is in a different situation from the one it is really in, so that in this way it executes a wrong action or different from which should execute.

1.1 – Motivation

Due to the rise of machine learning, and in particular to Deep Learning in real environments, there is a growing concern within the scientific community for the vulnerability of this type of systems, since being highly exposed, they run the risk of being attacked and present damages or very serious alterations in automated systems not supervised by a human.

Therefore, studies are increasingly needed to analyze the vulnerability of these systems to possible attacks, and in turn possible defense mechanisms to ensure the security that involves the environment of these automated systems.

Thanks to a subject in the second course of the career, I discovered machine learning. Then I started to get interested in Deep Reinforcement Learning, and knowing it in depth, I have realized that it is one of the branches of artificial intelligence more applicable to real cases such as the autonomous driving. It seems that the future is heading towards the autonomous cars without the supervision of a person at the wheel, but for them to work perfectly, there can not be a minimum error. The reality right now is that there have been several abuses, including deaths by this type of car, so that security against possible external attacks or internal errors is essential when it comes to ensuring human integrity.

1.2 – Objectives

In this work we will focus on the analysis of the vulnerability of deep reinforcement learning systems.

In particular, the objectives will be:

- ***Study of the state of art.*** In this work we will rely on already done and consolidated research, in order to expand the knowledge of these methods, providing conclusions and new techniques.

- ***Analysis and implementation of simple attacks existing in the literature.*** These attacks modify certain pixels of the observation or image, or they apply noise to it by means of filters (e.g., the Gaussian one).

- ***Definition and implementation of a new attack that identifies attack zones/regions.*** In this attack we propose, however, it is about identifying areas or regions of attack.

- ***Definition and implementation of a new attack based on learning by reinforcement.*** It is about learning an attack policy that decides for each state of the game if it is better to attack or not attack through Q-Learning.

- ***Analysis of the vulnerability of learning systems by deep reinforcement to these attacks in Atari games.*** The main objective is to verify the vulnerability of the neural network in front of this type of attacks. For this, the performance of the network, the number of attacks made by each technique, etc. will be measured.

2. STATE OF ART

First of all, there will be a brief introduction to Machine Learning, which is the great field in which this project is located (Section 2.1). Next, Deep Learning (Section 2.2) will be described and a brief introduction to Reinforcement Learning will be made (Section 2.3). Later on, what will be described is Deep Reinforcement Learning (Section 2.4) and, finally, the Adversary Reinforcement Learning (Section 2.5), the latter being where the set of techniques that will be used in this work are found.

2.1 – Machine Learning

Machine Learning is one of the main branches of Artificial Intelligence, whose main objective is to make a system learn by itself automatically, and do not have to be fully programmed the behavior of it [2].

Although this branch comes from computational science, it differs from the traditional computational science are specifically programmed for a specific solution, or a specific problem. Machine Learning algorithms, on the other hand, allow computers to train with input data, in addition to using statistical models adapted to these data, in order to arrive at a solution within a specific range. So, Machine Learning helps computers automate their decisions.

2.2 – Deep Learning

Deep Learning is another field of Artificial Intelligence, which tries to imitate, and in some cases far surpass, the cognitive abilities of humans.

The design of this learning hierarchized at different levels, forming an artificial neural network. At each hierarchical level, input information is processed, generating outputs, which will be the inputs of the next level of the hierarchy, thus obtaining a process comprised of an input layer and an output layer, and in between these two, all the hierarchical levels connected to each other.

2.3 – Reinforcement Learning

Reinforcement Learning is a technique of Machine Learning, which is based on giving a reward or punishment to the agent, depending on whether the action chosen by the latter is correct or erroneous [24]. In Reinforcement Learning there are three basic concepts: action, state and reward. The process of this type of learning is shown in Figure 4.

2.4 – Deep Reinforcement Learning

Deep Reinforcement Learning combines Deep Learning and Reinforcement Learning seen in the previous sections 2.2 and 2.3 respectively. In particular, it is based on the use of deep neural networks together with some algorithm of Reinforcement Learning, such as Q-Learning [24].

There are two main algorithms in the literature depending on whether the action space is discrete or not. In the case that the action space is discrete, the DQN algorithm [27] is usually used, and in the case that the action space is continuous, the DDPG algorithm is usually used [28].

As for the real applications of Deep Reinforcement Learning, we find a great variety among them, and the vast majority will be daily in our lives in the very near future, as technology is growing by leaps and bounds. Among the most important are autonomous driving and robotics.

However, although there are many real applications, in the case of this project we will focus on video games, and, in particular, the OpenAI Gym graphical interface [34], which is easy to use. We will focus on the Atari games, some of which are shown in Figure 13. Specifically, we will focus on the Breakout and Pong games, which correspond to the first two images in Figure 13.

The Adversarial Reinforcement Learning is dedicated to the study of the behaviors that the system has when an attacker modifies the input data, in order that this is wrong and makes mistakes or decrease their learning performance.

3. DESCRIPTION OF THE PROPOSAL

In this chapter we will describe the attacks carried out on a deeply trained neural network to analyze the vulnerability of this type of systems. To do this, first of all, the filters used to modify the images or observations of each of the two Atari games chosen for this research will be presented: Breakout and Pong. The application of these filters is considered an attack, since it can lead to error in the choice of action by the attacked agent.

3.1 – Detailed description of filters

- ***Modification of one pixel***: This attack method is summarized in the color change of a single pixel of the game image with a random value, that is, a random pixel of the observation takes a random color value, thus modifying the original input image.

- ***Modification of five pixels:*** This technique is based on the same procedure as the previous one. The difference between them is that in this technique 5 random pixels of the observation are modified.

- ***Gauss:*** In this case, a Gaussian filter is used as an attack. This filter modifies the image so little in human eyes, that no change is perceived form the original to this modified one. Instead, the system is severely affected, as will be explained later in the results of the experiment.

3.2 – Identification of attack regions

For the development of this type of attack, the filters described in the previous section will be applied but, in this case, they will only be applied when the ball is in a certain area of the screen, that is, in the area close to the racket, and also taking into account that the ball is approaching and not moving away from it. These areas are described in Figure 23 for each of the games considered.

3.3 – Attack based on reinforcement learning

In this attack, the attack process has been modeled as a Markov decision process (MDP) where we have identified the states, actions, and reinforcement function [24].

In an MDP the transition between states is defined in discrete time. That is, at an initial moment, the agent starts from a certain state of the environment. In the next instant, the agent executes an action, moving to the next state.

In both games, the size of the state space of the Q table will be 9x54x2, that is, 972 states. Therefore, this representation of the states allows us to greatly reduce their number, in order to use a tabular representation of the behavior policy.

The possible actions chosen in table Q when training the agent are divided into two: no filter would be applied to the original input image, or, the Gauss filter is added to the entrance, modifying it with a low sigma, so that the change from the original to the modified one is not visually appreciated, to be able to show that the system has affected its performance, while at a glance no change is perceived.

The objective of applying any of the two types of attack in each state is to know what action in each state is the best: whether to attack or not to attack.

The main idea of the reinforcement function designed is to fill the Q table of values, optimizing these in each iteration, so that the agent, once this table is optimized, after several update episodes, choose the best action for each possible status. That is, the agent chooses the action that most affects the system in each state, so that the performance of the neural network decreases considerably as time passes. For this, the reinforcement function is defined as shown in Equation 1.

4. RESULTS

The results of this study will be divided into two large blocks. In each one, the performance of the system will be evaluated when it is attacked with different techniques and in different situations, both for the Breakout game (Section 4.1) and for the Pong game (Section 4.2). Below are the results in each of these games.

4.1 – Results in Breakout

In this section the results of three different attacks are presented: attack in each observation, attack in a specific zone (described in Section 3.2), and attack based on reinforcement learning (described in Section 3.3).

4.1.1 – Attack in each observation

By applying the different filters and modifications of the original input image to each observation of the game sequence, different results are obtained.

From these results conclusions are drawn about the performance of the deep neural network when it is exposed to different attack techniques, such as the modification of a pixel, five pixels and the application of the Gauss filter, in this case, applied to each observation of the game without any restriction.

The results obtained from this process, taking into account that they apply to each observation, are shown in Figure 33.

4.1.2 – Attack in the specified region

This time Gauss attack will be applied only in the region detailed by Figure 23 and also when the ball is approaching the racket, since, as it was possible to verify in the previous section, this Gaussian filter is the attack that most affects the system and less is perceived at a glance.

The results obtained from this process, taking into account in this case that they apply only when the conditions detailed above are met, are shown in Figure 34.

4.1.3 – Attack based on Reinforcement Learning

This attack is the one described in Section 3.3 of this document.

The reward obtained by the game when training the agent to attack in the states where more alteration causes the system, is shown in Figure 35.

In the same way that the number of successful attacks increases during the course of the episodes, as shown in Figure 36.

4.2 – Results in Pong

As in the case of Section 4.1, the results in the Pong game will be divided into three different attacks.

4.2.1 – Attack in each observation

The results obtained by attacking the deep neural network modifying one pixel, five pixels, and applying the Gaussian filter, taking into account that they apply to each observation, are shown in Figure 37.

4.2.2 – Attack in the specified region

The results obtained from this process, taking into account in this case that they apply only when the conditions detailed above are met, are shown in Figure 38.

4.2.3 – Attack based on Reinforcement Learning

The reward obtained by the game when training the agent to attack in the states where more alteration causes the system, is shown in Figure 39.

In the same way that the number of successful attacks increases during the course of the episodes, as shown in Figure 40.

5. CONCLUSIONS AND FUTURE WORKS

5.1 – Conclusions

First of all, there are slight differences between the results of the game Breakout and the game Pong. But the most important thing is that in both cases the system significantly decreases its performance when applying different types of attacks.

Regarding the game Breakout, as can be seen in the results (Section 4.1), the decrease of more than 90% in the attacks made when the ball is falling and within the region represented in Figure 23 stands out against the attacks made in each observation. Therefore, it can be said that it is an important result, since the probability of being detected is substantially reduced, and in addition, the performance of the system is even reduced a little more than if it is attacked in each observation. In particular, the results of the application of Gaussian filter to the observation stand out, because it is the attack that is least visible to the naked eye and is the one that most affects the performance of the deep neural network. Regarding the results obtained through reinforcement learning, the decrease in the number of attacks against the number of attacks made when applying these in each observation, and also the effect that this method causes on the system, highlighting the performance drop of this average, even, below the rest of attacks.

On the other hand, in the case of the Pong game, as can be seen in Section 4.2, again highlights the reduction of more than 75% in the attacks made when the ball is in the specified region and approaching the racket in front of the attacks made in each observation. Therefore, this result is of great significance, since the probability of being discovered, and in turn, the performance of the system, is significantly reduced more than if it is attacked in each observation. In the same way as in the game Breakout, the star attack, that is, the one that substantially affects the deep neural network without appreciating only the naked eye in the modified image, is the Gaussian filter. In the results obtained by applying Q Learning making the agent learn when to attack and when not, the reduction in the number of attacks made against the number of attacks carried out by applying these in each observation are highlighted, as well as the effect caused by this method in the system, appreciating the decrease in performance of this average, although in this game the system is less sensitive to some attacks than in the game breakout.

It is evident in this work, therefore, that neural networks in Deep Learning are prone to large drops in performance even with attacks that are not visible to the naked eye, and that apparently are very simple to carry out. It is necessary, therefore, to investigate defense mechanisms especially when applied to real situations. In this work, results have been presented in several games, in which the fall of this performance by attacks involves losing a game. However, in real life, these attacks can involve the loss of a life.

5.2 – Future works

After having studied the effect that the attacks designed in this document have on the behavior of the system, different solutions to these vulnerabilities are proposed.

As mentioned earlier in this paper, it is necessary to know in detail the proposed attacks to be able to defend against them, since it is the main objective of the developers of these types of Artificial Intelligence system implemented in real applications.

Therefore, as possible future work to this investigation could be the exhaustive study of possible defenses against these detailed attacks throughout this project.

There are several types of defenses, such as those mentioned in Section 2.5 in this document. These are just a few examples of the infinity of defenses that exist to control possible attacks on the system, since, as mentioned above, to alleviate each type of attack and to control all these many defenses are needed, which is not trivial.