Universitat de les Illes Balears

escola politècnica superior

Trabajo de Fin de Grado

GRADO EN INGENIERÍA INFORMÁTICA

Machine Learning Playing Videogames

ÁLVARO CLAR LOPERA

Tutores

José María Buades Rubio Gabriel Moyà Alcover

Escuela Politécnica Superior
Universidad de las Islas Baleares
Palma, Julio de 2019

ÍNDICE GENERAL

ÍNDICE DE FIGURAS

ÍNDICE DE CUADROS

ACRÓNIMOS

AG Algoritmo Genético

EPS Escola Politècnica Superior

IA Inteligencia Artificial

ML Machine Learning

RN Red Neuronal

TFG Trabajo Final de Grado

RESUMEN

En el siguiente trabajo se expondrán diferentes soluciones y mejoras para llevar a cabo el aprendizaje de un conjunto de coches en un juego de carreras, con el objetivo de comprender y trabajar las técnicas y algoritmos de aprendizaje autómatico y su importancia y relevancia en los videojuegos.

Para situar el contexto del trabajo, se iniciará con una explicación de los orígenes de la Inteligencia Artificial (IA) y del Machine Learning (ML), así como las aplicaciones de los anteriores conceptos en el campo de los videojuegos.

Después de la introducción, se describirá el trabajo que fue realizado anteriormente, el cual sirve como base para el trabajo realizado.

A continuación, se mostrará el trabajo realizado, describiéndose cada uno de los cambios y mejoras que se han incorporado, junto a las diferentes metodologías desarrolladas para hacer posible el aprendizaje de los coches.

Una vez comentadas dichas metodologías, se mostrarán una serie de entrenamientos realizados junto a los resultados obtenidos.

Finalmente, se realizará una conclusión derivada de los resultados que se han obtenido, analizandolos y trantando de razonar el porqué de los mismos, junto a posibles mejoras y alternativas.

CAPÍTULO 1

INTRODUCCIÓN

1.1 Origen de la Inteligencia Artificial

La Inteligencia Artificial (IA) se define como una disciplina encargada de analizar y diseñar sistemas artificiales autónomos llamados agentes, de manera que sean capaces de mostrar un comportamiento inteligente. Estos agentes han de ser capaces de, mediante el uso de sensores, percibir el ambiente y actuar mediante actuadores con el objetivo de modificar dicho ambiente o bien conseguir un beneficio.(1)

Sin embargo, el concepto de IA no existe desde siempre. El pionero que puso la primera piedra angular sobre el mundo de la IA fue Alan Turing, considerado uno de los padres y precursores de las ciencias de la computación tal y como se conocen actualmente, quien en 1936 publicó un artículo llamado "Numeros Calculables", y es donde precisamente nace el concepto de "Máquina de Turing", que sentaría las bases de la informática actual y del concepto de máquinas que tendrán inteligencia.(2)

De todas formas, donde dicho concepto toma mayor fuerza es con el artículo de Alan Turing publicado en 1950, “Computing Machinery and Intelligence", en el volumen 59 de la revista “Mind”, en el que se habla de la posibilidad de que una máquina pueda imitar el comportamiento de un humano. A partir de dicho artículo, nace el famoso "Test de Turing", el cual pretende demostrar si una determinada máquina es inteligente o no. (3)

En cualquier caso, esta publicación sirvió como base para la creación de la nueva ciencia, que estaría formada por disciplinas como la Algoritmia y la Lógica. Aún así, el acontecimiento que sirvió para dar el nacimiento a la IA fue la Conferencia de Darthmouth, que tuvo lugar en 1956 en la universidad Darthmouth College. Fue un encuentro de dos meses de duración en el que se sentaron las directrices de la nueva ciencia, con el objetivo de estudiar cada aspecto del aprendizaje y de la inteligencia con el fin de poder dar una descripción precisa de estos y así poder implementarlos en las máquinas.(4)

1.2 Aplicaciones de la IA en videojuegos

Entre las múltiples aplicaciones de la IA, se encuentran los videojuegos, donde se puede ver dicho concepto en los comportamientos de las entidades que no son manejadas por un jugador (enemigos, personajes no jugables o NPCs y otros obstáculos).

Los primeras aplicaciones de IA fueron vistas en juegos de mesa: en damas (Arthur Samuel) y en ajedrez (Claude Shannon), todo ello en los años 50.
(5)

En 1960 nació el Pong, un juego basado en la lógica. Este está considerado como el primer juego electrónico, en el que el jugador debe competir contra el rival manejado por la máquina, ganando el que llegue a una cantidad de puntos determinada. (6)

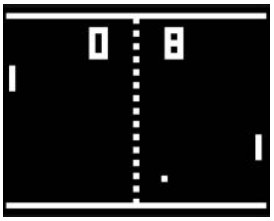

Figura 1.1: Pong, versión de Atari (1972)

Posteriormente, en 1978 y 1980 llegaron los videojuegos Space Invaders y Pac-Man, respectivamente, los cuales añadían múltiples enemigos que actuaban según patrones almacenados, y en el caso del segundo, se añadieron algoritmos de búsqueda en laberintos.

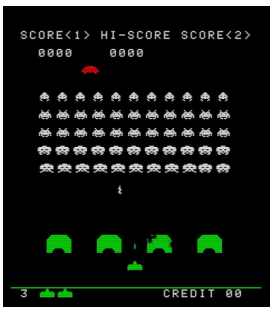

Figura 1.2: Space Invaders (1978)

A partir de los años 90 empezaron a surgir nuevas técnicas que sirvieron para ser aplicadas en los videojuegos, como son las máquinas de estados, y la que se detallará más en este trabajo, la Red Neuronal (RN). En 1996 nacería el que sería el primer videojuego con RNs, Battelcruiser 3000AD, donde las naves enemigas eran controladas con dichas redes. (7)

Figura 1.3: Pac-Man (1980)

Figura 1.4: Battlecruiser (1996)

1.3 Machine Learning

Dentro de las diferentes ramas que conforman la IA se encuentra el Machine Learning (Aprendizaje automático o de las máquinas).

El Machine Learning (ML) se define como el conjunto de técnicas que permiten dotar a las máquinas la habilidad de aprender por sí mismas, o dicho de otra forma, algoritmos que permiten dotar de inteligencia y una capacidad de aprendizaje. (8)

Como se puede apreciar por el título del trabajo, el ML será uno de los focos principales del mismo. Entre sus usos y aplicaciones, se encuentran también los videojuegos, donde se pueden encontrar casos en los que se introducen técnicas para permitir el aprendizaje de las diferentes entidades del juego.

Donde se puede ver ML y el uso de RNs es en una serie de trabajos realizados por el youtuber y programador "SethBling", los cuales trabajan ese tipo de técnicas en juegos como Super Mario Kart (Nintendo Entertainment System) y Super Mario World (Nintendo Entertainment System).

En el primero de ellos, la RN que controla al personaje aprende tras 15 horas de juego por parte del programador, de forma que está pudiera aprender de su juego. Realmente, esto suponía que la RN aprendiera a jugar como el youtuber, tratando de predecir la acción que haría el jugador. Este tipo de aprendizaje, se conoce como aprendizaje supervisado, en el que se le proporcionan datos de forma que la RN sea capaz de aprender a partir de esos datos, trantando de ajustar su comportamiento a lo que le dictan esos datos. (9)

Figura 1.5: Super Mario Kart con RN

En el segundo, el youtuber crea una RN que pretende conseguir que Mario complete un nivel (Donut Plains 1, uno de los primeros y más sencillos niveles). Para ello, dicha red neuronal trata de percibir una serie de datos (suelo, paredes, enemigos y otros objectos) con el fin de obtener una salida (pulsar un botón u otro), todo ello de manera automática. En cada simulación, se comenzaba el nivel y esta terminaba si Mario moría o si terminaba quedándose quieto y sin realizar ninguna acción durante unos segundos. Entonces, entre cada simulación, se producía un cambio sobre la RN, de forma que esta fuera capaz de aprender de la simulación en que lo hacía mejor, hasta poder llegar a completar el nivel, objetivo el cual se acabó consiguiendo. En este caso, se produce lo que se conoce como aprendizaje no supervisado, en el cual, no existe ningún tipo de ayuda o información que permita a la RN aprender. Este tipo de aprendizaje será explicado con mayor profundidad más adelante, puesto que supone el núcleo principal del trabajo. (10)

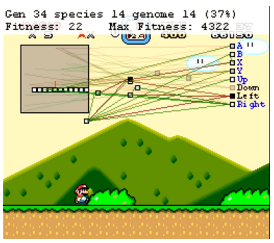

Figura 1.6: Super Mario World con RN

Sin embargo, los casos anteriores no son los únicos que traen las RN al mundo de los videojuegos. En el año 2014, la compañía de IA DeepMind, la cual fue absorbida por Google en ese mismo año, desarrolló un algoritmo capaz de superar juegos de la videoconsola Atari2600. Dicho algoritmo utilizaba una RN, y fue capaz de superar 43 de los 49 diferentes juegos en los que la red fue entrenada. Esta RN fue capaz de superar en un 75 % la puntuación humana en más de la mitad de juegos. Además, el hecho de que fuera capaz de aprender en juegos de mucha variedad, desde los juegos de boxeo hasta los de carreras, demostró la gran capacidad de adaptación que tenía la RN. (11), (12)

1.4 Estructura de la memoria

Una vez vistos los origenes y conceptos básicos de la IA y del ML, se comentará el formato que tendrá esta memoria.

En primer lugar, se explicará con detalle el planteamiento del problema a cubrir, y junto a ello se describirá el programa realizado previamente para poder resolver el problema: los circuitos, los coches y el funcionamiento del juego en sí (vueltas, colisiones y demás aspectos). Debido a que la base del programa está implementada por el alumno Mateu Morro Ribot, muchas de las cuestiones que conforman el programa no serán detalladas con tanta precisión, debido a que mencionado alumno estudio el Grado en Matemáticas y hay mucha formulación y teoría matemática en el planteamiento de dichas cuestiones. En este trabajo, la explicación está limitada a explicar el funcionamiento de cada aspecto para dar a comprender el programa ya realizado.

En segundo lugar, se detallarán los cambios y mejoras que se han incorporado a nivel del programa, de forma que todos estos se comentarán en cada apartado. Entre los anteriores aspectos, se incluyen los nuevos métodos que se han desarrollado para permitir el aprendizaje de los coches.

Finalmente, se dedicará un importante espacio a comentar los resultados obtenidos utilizando diferentes estrategias y parámetros, con el objetivo de sacar conclusiones respecto al trabajo realizado.

CAPÍTULO 2

TRABAJO PREVIO

En un trabajo previo se desarrolló un videojuego de carreras en el que se pretendía poner en práctica algoritmos y técnicas de ML, de forma que se pudiera ver como los coches podían aprender a conducir. Dicho trabajo, en el que está basado este TFG, fue realizado por el alumno Mateu Morro Ribot, del grado de Matemáticas, el cual fue entregado día 19 de febrero de 2019. A continuación se detallarán las cuestiones y problemas planteados en dicho trabajo previo, con el fin de dar a conocer las bases que dan forma al nuevo trabajo desarrollado.

2.1 Consideraciones previas

A modo de clarificar todas las explicaciones que se realizarán a continuación, hay que considerar una serie de aspectos:

- La mayor parte de los módulos que componen el programa (desarrollo de los coches, circuitos, detección de colisiones y demás aspectos) serán explicados de forma más breve que en el propio trabajo de Mateu Morro. Esto es debido al hecho de que el objetivo de este trabajo es ampliar y mejorar la versión realizada por este, con el fin de tratar de resolver cuestiones que quedaron pendientes en el anterior trabajo. Otro motivo por el que las descripciones de dichos elementos serán más breves, es debido a que el planteamiento de los mismos conlleva una gran base de matemáticas, la cual no es el objetivo de este trabajo desarrollado.
- Se comentarán, además, las pruebas realizadas en el anterior trabajo con el fin de visualizar las carencias o cuestiones que podían quedar abiertas para ser resueltas en una ampliación.
- A la hora de explicar las RNs de los coches, se hará una explicación más profunda del concepto de RN a modo de introducción, y seguidamente se explicará el tipo de red utilizada en el trabajo de Mateu.

El problema planteado en este trabajo es el siguiente: para poner en práctica el uso de ML en videojuegos, se pretende crear un pequeño videojuego que utilice algoritmos que permitan conseguir que unos coches puedan aprender a conducir.

Para lograr dicho objetivo, será necesario que los coches aprendan a correr en varios circuitos. Esto es importante puesto que, si los coches aprendieran a correr un circuito concreto, en el que, por ejemplo, hay demasiadas curvas hacia la derecha, estos únicamente sabrían girar hacia la derecha, y en el caso de que se encontraran curvas hacia la izquierda, estos no tendrían los mismos conocimientos que para girar a la derecha, y probablemente no realizarían esta acción correctamente. Por tanto, es por esta razón que es necesario que los coches afronten la prueba de recorrer diversos circuitos con diferentes situaciones: rectas, curvas más abiertas, más cerradas, en forma de "S", etc.

Por tanto, el objetivo primordial será conseguir que los coches aprendan a conducir, siendo solventes en la mayor cantidad de situaciones.

La manera en que se va a dotar a los coches de la posibilidad de aprender a conducir y mejorar sus prestaciones, es mediante una RN. Esta se encargará de dictaminar las acciones de cada coche en cada instante de tiempo, según la situación que el coche perciba, de forma que en cada simulación o ejecución realizada el coche verá modificada su RN, y así conseguir mejorar su conducción.

Los aspectos fundamentales para entender el funcionamiento del juego desarrollado son los siguientes:

- Los coches.
- Los circuitos por los que circulan los coches.
- La manera en que el coche se desplaza: velocidad y giro.
- Las colisiones y la forma de detectarlas.
- La red neuronal de un coche.

2.2 Circuitos

Los circuitos creados para el juego, tanto los que servirán para el entrenamiento de los coches como los que servirán para probar el aprendizaje de los mismos, son creados a partir de un fichero para cada circuito con un formato específico.

Cada circuito está formado por diferentes cuadriláteros o segmentos conectados entre sí. Precisamente, cada fichero tiene 3 líneas, lo que significa que cada columna representa una tripleta de valores. Cada tripleta representa lo siguiente:

- El primer valor representa la coordenada x del punto que marca el comienzo de un segmento, y determina el inicio de un segmento.

- El segundo valor representa la coordenada y del punto que marca el comienzo de un segmento, y determina el inicio de un segmento.

- El tercer valor indica la anchura que tiene el segmento en ese punto.

De esta manera, cada tripleta se conecta con la siguiente formando un segmento.

2.3 Coches

Cada coche desarrollado consiste en un rectángulo definido por 4 puntos que representan las esquinas del mismo.

Evidentemente, es importante conocer la posición, puesto que esto permite saber en que punto del circuito está el coche, además de la distancia que ha recorrido.

Además, de cada coche es de donde saldrán los sensores que detectarán los puntos y distancias de colisión, como se puede observar en la figura 2.1 .

Para simular cada coche, se introduce una variable temporal que permite separar un instante de tiempo i con el instante de tiempo i + 1, de forma que esa diferencia siempre será fija.

En cada instante de tiempo, el coche tiene asociadas unas distancias de colisión para cada uno de los sensores. Estos sensores funcionan de forma que cada uno parte del punto central de la parte trasera de cada coche, los cuales tienen estos ángulos respecto a la horizontal:

$\beta = [10°, 26°, 42°, 58°, 74°, 90°, 106°, 122°, 138°, 154°, 170°]$

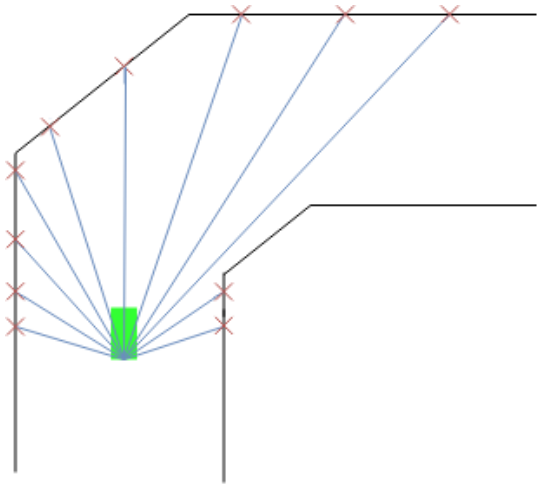

Figura 2.1: Distancias de colisión detectadas con los sensores del coche.

Mediante una serie de cálculos matemáticos que utilizan expresiones vectoriales y matriciales, se consigue calcular para cada sensor el primer punto en el que intersecta con una barrera del circuito, o bien con un obstáculo, y por tanto la distancia y punto de colisión.

2.4 Vueltas

Cada coche realizará una serie de vueltas al circuito. En este trabajo previo, se considera que cada vehículo completará un circuito si este ha realizado 2 vueltas.

Cada circuito, como ya se ha comentado, está formado por un conjunto de cuadrilátero o segmentos conectados entre sí. Por tanto, la manera de conocer si un coche ha completado una vuelta, es ver si ha recorrido todos los segmentos del circuito.

2.5 Velocidad y aceleración del coche

Como se ha podido ver anteriormente, el vehículo es capaz no solo de girar para tomar curvas, sino que además es capaz de acelerar y frenar. Dichos cambios de velocidad son los que pueden permitir que un vehículo no solo complete un circuito, sino que además pueda completarlo en el menor tiempo posible, ya que, como se verá más adelante, la distancia no es el único criterio a tener en cuenta para poder mejorar la conducción de los coches.

2.6 Red neuronal

Una Red Neuronal (RN) es un modelo matemático que pretende simular el comportamiento del cerebro. Este está formado por unas 86 billones de neuronas, las cuales están conectadas entre sí mediante sinapsis . Del mismo modo que ocurre en el cerebro humano, las RNs están formadas por unidades llamadas nodos o neuronas, conectadas con otras neuronas. Estas conexiones tendrán un cierto peso asociado ω (13)

El modelo más sencillo de RN es el Perceptrón. Se trata de una red formada por n neuronas de entrada, y una neurona de salida k. Cada entrada se conecta con la neurona de salida, transmitiendole un valor, junto al peso asociado a la conexión. Esta neurona obtendrá el valor total que recibe utilizando una regla de propagación, y mediante una función de transferencia dará una salida.(14)

Como se puede ver en la figura 2.2, además de todos los pesos asociados a cada entrada, se considera una entrada adicional, ω_0, cuyo valor es $-\theta_k$. Este valor θ_k se llama umbral (threshold en inglés), y este indica un valor mínimo para que la neurona se active, dando como valor de salida 1 en ese caso y 0 en el caso contrario. (15). El valor ω_0 pues, sirve para que el criterio de evaluación para saber si una neurona se activa sea comprobar si se ha superado el valor 0.

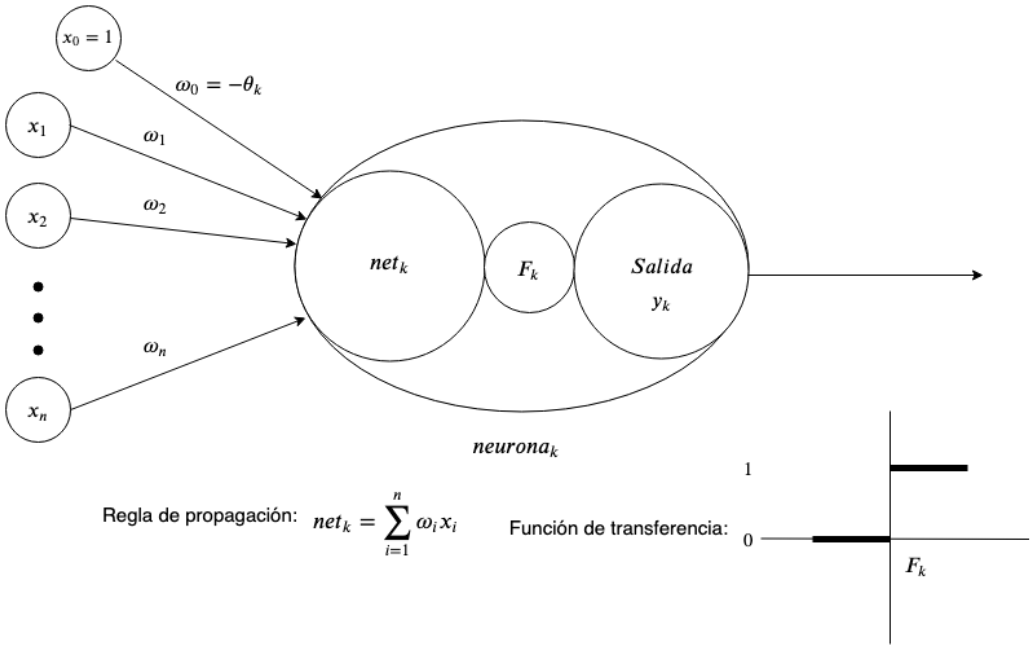

Figura 2.2: Esquema del perceptrón

Sin embargo, en el caso de la RN de cada coche, existen importantes diferencias con el perceptrón anteriormente descrito. Para empezar, esta se compone de diversas capas, concretamente 3: la capa de entrada, la capa intermedia y la capa de salida.

Como se puede apreciar en la figura 2.3, hay 21 neuronas que forman la capa de entrada. Esta capa consiste en un conjunto de neuronas que simplemente contienen un valor de entrada para la siguiente capa de neuronas, es decir, en ellas no se computa ninguna operación como la regla de propagación ni la función de activación, solamente representan un valor de entrada.

El motivo por el que hay 21 neuronas de entrada es debido a que el coche tiene 11 sensores que le otorgan al vehículo de un ángulo de visión frontal de 160º, y cada uno de estos sensores detectará una distancia y punto de colisión, donde esas 11 distancias serán los valores de entrada para la red neuronal, mientras que el resto de entradas se trata de 10 de lecturas por parte del sensor central de los 10 instantes de tiempo anteriores.

Estas últimas diez entradas permiten detectar la distancia en línea recta a la que está el vehículo de una colisión, de forma que:

- Si a cada instante de tiempo la distancia que detecta el sensor central es más grande, quiere decir que el coche ha salido de una curva y que hay una recta, con lo cual el vehículo puede acelerar.
- Si a cada instante de tiempo la distancia que detecta el sensor central es más pequeña, esto significa que el coche está terminando una recta y que empieza

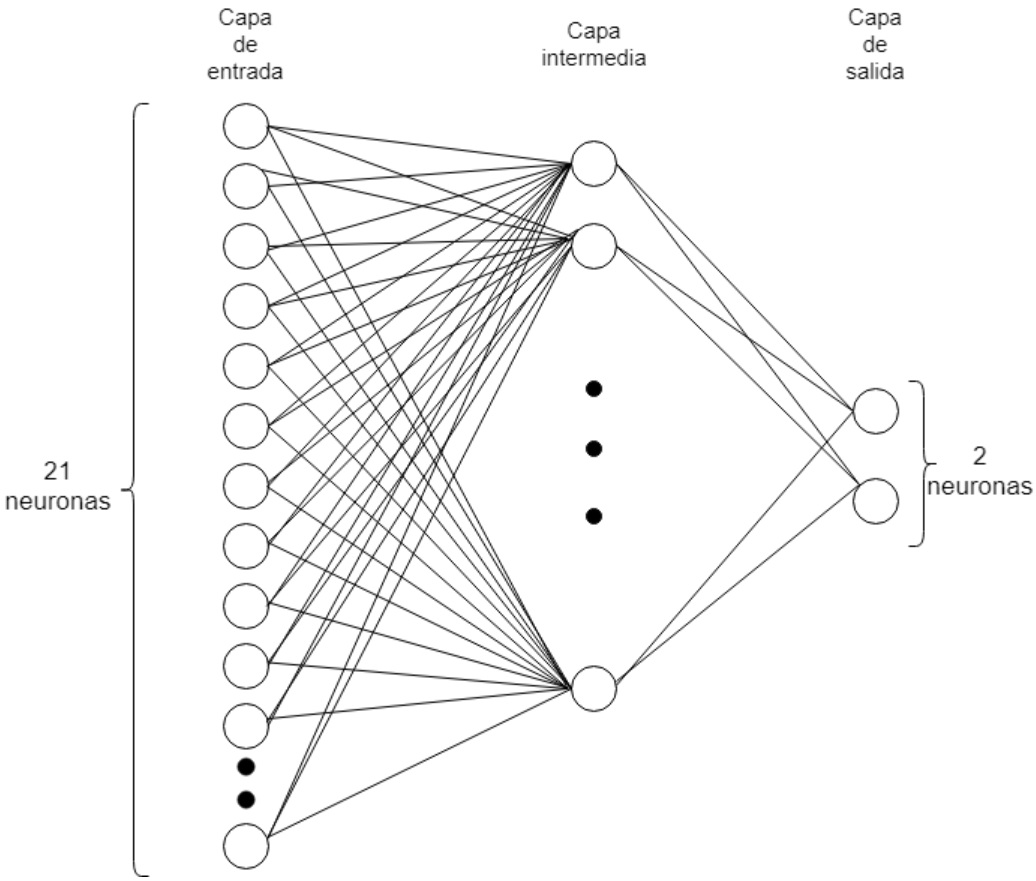

Figura 2.3: Red neuronal de un coche

una curva, con lo cual el vehículo debe frenar.

Por tanto, por un lado:

- Las 11 primeras entradas, correspondientes a los 11 sensores propios del vehículo, son los que permiten saber a que lado debe girar el coche: o bien derecha o bien izquierda.
- Las 10 entradas restantes, correspondientes a las 10 lecturas del sensor central, servirán para controlar la velocidad del vehículo.

A continuación está la capa intermedia, la cual recibe esos valores de entrada y los procesa con la regla de propagación (suma ponderada de pesos y valores) y calcula el valor de salida mediante la función de transferencia, valor el cual será suministrado a las neuronas de la siguiente capa. Existen diferentes funciones de transferencia, y para este trabajo se han utilizado varias, las cuales se verán más adelante en esta memoria. En la figura 2.3 se puede ver que hay una sola capa intermedia, lo cual no siempre es así, ya que pueden tenerse más de una capa de neuronas intermedias, pero debido a la naturaleza del problema, es suficiente con añadir 1 capa, puesto que un mayor número podrían generar una mayor cantidad de estados y posibilidades, que harían

la convergencia más complicada a la hora de encontrar una solución. En este caso, se utilizaron un total de 25 neuronas en una sola capa intermedia.

Finalmente se encuentra la capa de salida, la cual procesa las entradas que recibe de la capa anterior, realizando el mismo tratamiento ya descrito para producir la salida, finalmente, de la red neuronal. Esta salida está formada por 1 neurona, que representa la velocidad angular, y por ello es lo que controla el giro del coche.

2.7 Entrenamiento realizado. Tipos de aprendizaje.

En esta sección se comentarán los diferentes entrenamientos y tipos de aprendizaje realizados.

Para poder realizar el entrenamiento de los coches, se optarón por dos soluciones diferentes: aprendizaje supervisado y aprendizaje no supervisado.

2.7.1 Aprendizaje supervisado

Este tipo de aprendizaje se caracteriza por la presencia de un agente denominado como "supervisor", en el cual dicho agente se encarga de mostrar a las redes que intentan aprender el como deben actuar en cada situación.

En cada situación a cada RN se le presentará un conjunto o par de valores: la entrada actual que percibe el vehiculo, y la salida deseada por el supervisor. Si la salida obtenida por la red que está tratando de aprender no se corresponde con la salida deseada, se modificarán los pesos de la red neuronal con el objetivo de conseguir que la salida de la red sea la deseada o lo más parecida a esta.(16)

Para poder obtener esos valores (entrada y salida deseadas), conocidos como datos de entrenamiento, el usuario realizaba los circuitos varias veces para poder extraer esa información, de forma que luego se puede empezar con el entrenamiento en cuestión.

A partir de este momento, la idea es aplicar el algoritmo del Descenso del Gradiente, el cual tiene como objetivo conseguir ajustar los pesos y umbrales de las RNs para que estas se parezca lo máximo posible a los datos de entrenamiento.

Descenso del Gradiente

El algoritmo del descenso del gradiente se basa en buscar el mínimo de una función determinada.

Dada una situación, se tienen un conjunto de entradas para cada neurona, y se tienen un conjunto de salidas deseadas para esa neurona. Se calcula la salida mediante, en este caso, la función sigmoidal, y se compara con la salida deseada, la cual no tiene porque ser única, sino que puede tener *i* salidas. Dicha diferencia entre la salida deseada

y la salida real se obtiene calculando la función de error cuadrático medio:
$E = \frac{1}{2}\sum_{i=1}^{n}(d_i - y_i)^2$, donde:

- E representa el error,
- d_i representa el valor de salida deseado para un conjunto de entradas determinado,
- y representa el valor de salida real para dicha entrada.

La idea del algoritmo es conseguir el conjunto de pesos y umbrales que minimizan ese error.

Debido al objetivo de buscar el mínimo de dicha función, este requiere el uso del Gradiente, un vector que indica las derivadas parciales para una función en un punto determinado. El resultado dado por el gradiente indica la dirección en la que la pendiente es mayor, que a su vez indica hacia donde disminuye con mayor intensidad el valor de la función.

Dicho paso se realiza tantas veces hasta llegar a ese mínimo, donde el resultado del gradiente sería 0, puesto que no habría ninguna inclinación en ese punto. Dicho punto en el que la función tiene un mínimo global, se traduce en que los valores de los pesos y umbrales para dicha neurona permiten dar una salida que se parece lo máximo posible a la deseada. (17)

Sin embargo, para que el algoritmo pueda encontrar el mínimo global de la función, es necesario aplicar sucesivos descensos, lo cual introduce una nueva variable: el ratio de aprendizaje η. Este valor η indica lo grande o pequeño que va a ser el descenso en cada iteración, de forma que:

- Un valor η grande puede hacer que los descensos sean demasiado grandes y por tanto el algoritmo puede entrar en bucle infinito puesto que no podrá converger.
- Un valor η pequeño hará que los pasos dados por el algoritmo sean más pequeños, lo cual conducirá a una convergencia. El problema es que esto implica una mayor cantidad de iteraciones y por tanto una mayor ineficiencia a nivel temporal. (18)

$$w_k \to w'_k = w_k - \eta\frac{\partial C}{\partial w_k}$$
$$b_l \to b'_l = b_l - \eta\frac{\partial C}{\partial b_l}.$$

Figura 2.4: Formula del descenso del gradiente, para pesos y umbrales.

Por tanto, es importante seleccionar un ratio de aprendizaje adecuado, que normalmente suele ser un valor intermedio.

Finalmente, el algoritmo trata de repetir este proceso hasta que se cumple una de estas condiciones:

- Se cumplen un número de iteraciones (llamadas epochs en inglés)
- El valor resultante de la función E es inferior a cierto umbral predefinido.

Dicho algoritmo se aplica para cada dato de entrenamiento. Sin embargo, se dispone de una mayor cantidad de datos para poder asegurar un mejor entrenamiento, lo cual, implica un coste temporal muy elevado.

En el caso del programa ya desarrollado, para realizar el entrenamiento, se podía seleccionar el número de iteraciones, es decir, la cantidad de veces que se ejecutaba el descenso del gradiente, y el valor de η, que representa como ya se ha mencionado, el ratio de aprendizaje.

Como se ha podido ver, este tipo de aprendizaje necesita de una gran cantidad de datos de entrenamiento para ser posible, lo cual puede implicar un gran coste a la hora de realizarlo. Sin embargo, la ampliación realizada en este TFG, se basa mayormente en el otro tipo de aprendizaje, el cual se describirá a continuación.

2.8 Aprendizaje no supervisado

El aprendizaje no supervisado, como indica su nombre, se describe como un tipo de aprendizaje en el que no existe la figura del supervisor, y por tanto implica la no existencia de datos que permitan a las RN ajustarse en base a estos.

Por tanto, para realizar un aprendizaje sin datos, la única manera en que pueden aprender estas redes, es a partir de sí mismas. En base a esta idea, se explicarán los algoritmos que hacen posible este aprendizaje: los Algoritmos Genéticos.

2.8.1 Algoritmo Genético

Un Algoritmo Genético AG se define como un mecanismo de búsqueda basado en las leyes de selección natural y genética, el cual permite que los individuos mejor adaptados sean seleccionados, ademas de pasar por un proceso de mutación y cruce. Para poder decidir que individuo se ha adaptado mejor, se considerará un valor llamado *aptitud* o *fitness*, y quien tenga el mejor (puede ser el mayor o el menor), será entonces el elegido a iniciar la nueva generación. (19)

La idea principal de este tipo de algoritmos, trasladada al problema tratado en el trabajo previo es la siguiente:

- Se inicializan las redes de los coches con pesos y umbrales aleatorios y se comienza la simulación (adaptación).

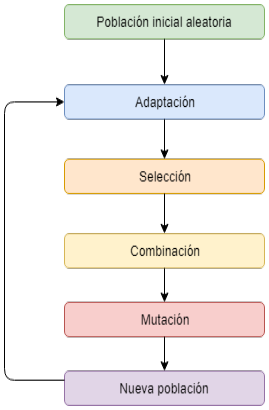

Figura 2.5: Esquema básico de un Algoritmo Genético.

- Una vez han terminado una simulacion (cuando los coches llegan a meta, o bien cuando colisionan todos), se realiza el paso de selección. En este, se escoge al mejor coche, bajo una función de ponderación, que puede ser la distancia recorrida, la velocidad, o bien una combinación de ambas.
- A partir del paso anterior, se puede crear la nueva generación, pero a partir de modificar ligeramente la RN seleccionada, se generaran las nuevas redes para los coches. Existen varias operaciones genéticas que se pueden realizar:
  - Combinación: consiste en combinar los valores de una RN con los de otra, de forma que se consigue un cruce de esa red.
  - Mutación: consiste en modificar algunos valores de la RN.
- Los pasos anteriores son necesarios puesto que replicar una RN implicaría un posible estancamiento puesto que al no haber ninguna variación, no existiría aprendizaje.
- Finalmente, una vez son realizadas las operaciones genéticas mencionadas, se crea una nueva generación de coches con nuevas las redes creadas a partir de dichas transformaciones.

2.8.2 Filtro de Partículas

El Filtro de Partículas se trata de un algoritmo para medir el estado de un sistema que cambia a lo largo del tiempo. Es un método secuencial de Montecarlo utilizado para varias aplicaciones, entre ellas el seguimiento de objetos en imágenes.

Para poder seguir un objeto en una imagen, dicho filtro de partículas lanza un determinado número N de puntos, (partículas). Dicha probabilidad se actualiza si las partículas se han situado cerca del objeto o no. Aplicando este paso sucesivamente, se consiguen obtener puntos que se situan más cerca del objeto, con lo cual se consigue una convergencia.

De forma similar funciona, en este caso, el aprendizaje de los coches. Los coches cuyo valor de ponderación o *fitness* sea mayor, serán los que tendrán más probabilidad de ser seleccionados como padres de la nueva generación de coches, creándose esta con una ligera variación o ruido para alterar los valores de la red y no replicar la misma red en todos los coches generados.

Si se observa la explicación anterior, pese a que son similares, existe una gran diferencia entre el AG y el filtro de partículas. En el primero, el coche con mejor valor de ponderación o *fitness* será siempre el que será elegido para generar la nueva generación, mientras que en el segundo, el que mejor *fitness* tenga no siempre será el elegido, pero sí que es más probable que lo sea, lo cual da la posibilidad de que otros no tan buenos puedan crear una nueva generación, y con una ligera variación puede ser que de un buenos resultados.

2.9 Resultados obtenidos

En el trabajo previo se realizaron diversos entrenamientos para conseguir que los coches aprendieran a conducir mediante aprendizaje no supervisado y supervisado. En cuanto al no supervisado, estos entrenamientos consistían en recorrer cada circuito dos veces, de forma ascendente o descendente en dificultad, de manera que después de recorrerlo una vez, se aplica el filtro de partículas. Estas pruebas fueron realizadas utilizando 30 coches por generación y con diferentes funciones de ponderación, como son la distancia recorrida, la distancia recorrida al cuadrado, la velocidad multiplicada por la distancia y la velocidad.

Los resultados obtenidos demostraron que las mejores funciones son la distancia y la distancia al cuadrado, y que era mejor entrenar de difíciles a fáciles, pudiéndose ver que el mejor coche, el cual era aquel que mas distancia recorría en el último circuito que entrenaba, acababa recorriendo los circuitos test 1 y 2, y recorría media vuelta del 3.

2.10 Consideraciones finales

A modo de conclusiones, el trabajo de Mateu Morro plantea una serie de cuestiones para una posible mejora del TFG. En este caso, se comentarán únicamente los aspectos relacionados con el aprendizaje no supervisado, puesto que este es el tipo de aprendizaje que se trabajará en esta ampliación. Aparentemente, los resultados obtenidos mejoraban, pero durante el desarrollo del trabajo, se plantearon ciertas cuestiones:

- Cambiar el número de coches y los sensores de cada uno.
- Cambiar la configuración de la red neuronal (neuronas y capas).
- Influencia del orden de circuitos a utilizar para el entrenamiento.
- Influencia de que circuitos utilizar para el entrenamiento.

Por tanto, en la siguiente sección se comentarán todos los aspectos realizados con la ampliación realizada.

CAPÍTULO 3

TRABAJO REALIZADO

3.1 Consideraciones previas

En este capítulo se comentarán con detalle todos los cambios y mejoras realizadas en el trabajo de Mateu Morro, que incluyen los cambios a nivel de código, y también los cambios realizados en el aspecto del aprendizaje de los coches.

Este capítulo se irá estructurando según los apartados del programa en los que se han realizado cambios y/o mejoras.

3.2 Circuitos

En cuanto a la parte de los circuitos, se han utilizado los mismos 12 circuitos de entrenamiento y los 3 circuitos de test. Cada circuito está enumerado, de manera que los circuitos están ordenados de manera creciente en dificultad, tanto los 12 utilizados para el entrenamiento, como los 3 utilizados en las pruebas, lo cual significa que el circuito 1 es el más fácil y el circuito 12 es el más difícil. En cuanto a los 3 circuitos de test, los circuitos 13,14 y 15, también están ordenados bajo el mismo criterio que los de entrenamiento. El motivo por el que se utilizan los mismos circuitos se debe a que se considera que todos abarcan una gran variedad de situaciones diferentes, con lo cual el aprendizaje resultante de recorrer estos circuitos se espera que sea muy completo.

3.2.1 Línea de meta

Un cambio realizado en todos los circuitos consiste en mostrar una linea que delimita la línea de meta, lo cual hace más sencillo el seguimiento de la carrera, con lo cual se puede saber que coches están más cerca de la meta.

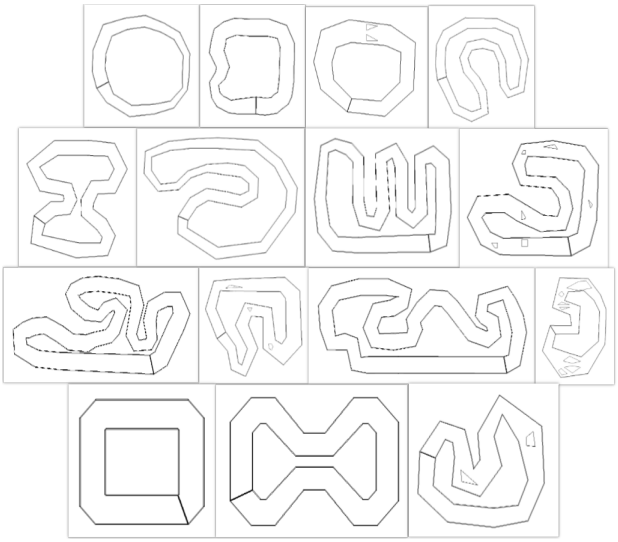

Figura 3.1: Circuitos utilizados ordenados de izquierda a derecha y de arriba a abajo. Los 3 circuitos de abajo son los circuitos que se usarán para probar a los coches entrenados.

En cuanto a los circuitos, no se ha realizado ningún cambio o mejora más, puesto que se considera que los mismos están bien para el objetivo deseado.

3.3 Coches

Los coches han sido modificados ligeramente para conseguir el propósito del trabajo.

3.3.1 Color de los coches

Para poder indicar visualmente cual es el coche que va en primer lugar, este siempre será pintado de color amarillo en todo momento, aunque no sea el coche que sea visualizado en ese momento. Si el coche visualizado no es el primero, este se pintará de verde para poder distinguirlo fácilmente de los demás.

3.3.2 Coches erróneos

Debido al comportamiento de las RNs de algunos coches generados, hay veces en que los coches circulan de forma cíclica sin avanzar e incluso, avanzan en sentido opuesto al que deberían.

Este problema generaba que muchas simulaciones no pudiernan terminar adecuadamente, puesto que no todos los coches llegaban al final o no llegaban a colisionar. Para solucionar esto, se modificó una función que permitía detectar a aquellos coches que habían cruzado la línea de meta hacía atrás en la primera vuelta como coches colisionados, de forma que eran detenidos.

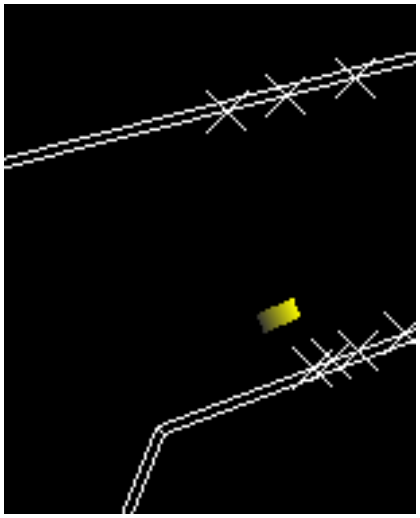

Figura 3.2: Vehículo que va en primer lugar.

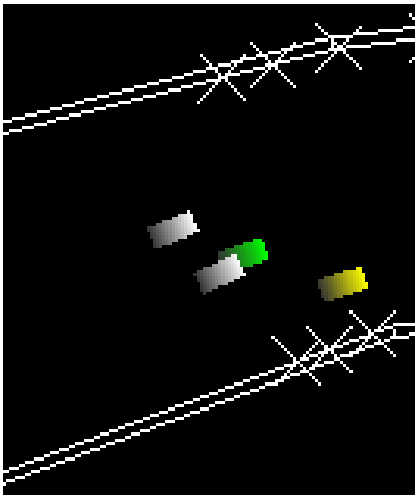

Figura 3.3: Vehículo visualizado en el momento actual.

Para los coches en bucle, la solución desarrollada consiste en finalizar la simulación cuando el tiempo de carrera exceda de 1000, de forma que se continuaría a la siguiente simulación sin que ningún usuario tenga que intervenir. El valor de 1000 ha sido elegido debido a que es un tiempo en el que es imposible que haya coches circulando con normalidad hacia la meta, ya que o todos habrán colisionado o llegado a meta, puesto que los circuitos no son lo suficientemente largos como para que se superen las 1000 unidades de tiempo en una carrera. La elección de este valor se ha obtenido a partir de extraer el doble del tiempo medio tardado por los coches que completan el circuito más largo , el número 6, de forma que se considera que los coches que van a llegar a la meta nunca se excenderán de esa cantidad.

Además, como se verá más adelante, se realizaron una serie de cambios en la simulación para poder modificar la visualización de cada vehículo.

3.4 Vueltas

En cuanto al número de vueltas a realizar para completar un circuito, se ha mantenido el número de 2 vueltas. Esta decisión se debe a que reducir a una sola vuelta implica suponer que el circuito va a ser recorrido por el mismo coche una segunda vez, cuando esto no tiene porque ser así. De hecho, a la hora de entrenar, había coches que no colisionaban en la primera vuelta, pero si que lo hacían en la segunda. Por otro lado, también se contemplo la posibilidad de aumentar a 3 vueltas, pero teniendo en cuenta el hecho de que hay circuitos bastante largos, con muchos coches en pista (el programa se ralentiza), esto haría el entrenamiento un proceso muy lento.

3.5 Red neuronal

En cuanto a las RNs que utilizan los coches, sí que se han realizado modificaciones. La primera de ellas, consiste en la modificación del número de neuronas en cada una de las capas.

A modo de recordatorio, hay que destacar que la RN tenía un total de 3 capas, con 21 neuronas en la capa de entrada, 25 neuronas en la capa intermedia y 2 neuronas en la capa de salida. El motivo por el que se utilizo está configuración para las neuronas está explicado en el capítulo 2.

La razón principal por la que se ve modificada la RN es la siguiente: durante las pruebas realizadas con el fin de conseguir que los coches aprendieran adecuadamente, se vió que la variación de velocidad, presente en el trabajo previo, podía resultar un problema a la hora de conseguir ese propósito, de forma que se optó por una solución en la que se prescindiese de dicho elemento. Esta simplifación, además de cambiar la estructura de la RN utilizada para cada coche, permite asentar unos objetivos diferentes a los del trabajo anterior:

- Antes, el objetivo no era únicamente que los coches terminarán los circuitos, sino que también se pretendía que los completarán en el menor tiempo posible, con lo que la velocidad era un factor importante,
- Por tanto, el objetivo en este nuevo trabajo se trata de mejorar el aprendizaje asegurando y priorizando el hecho de los coches terminen los circuitos, sin importar el tiempo que tarden.
- Dicho cambió se realizó con el objetivo de asegurar mejores resultados en el aprendizaje, pese a ser algo menos ambicioso que lo que se propuso en el anterior trabajo en el aspecto del tiempo tardado.

A raíz del cambio anterior, la red neuronal que regula a cada coche queda como se puede ver en la figura 3.4.

Esta nueva RN utilizada contiene las siguientes partes:

- **Capa de entrada**: En esta capa, se han eliminado 10 entradas, dando un total de 11 neuronas. Esas 10 entradas, se correspondían a cada una de las 10 lecturas que se tenían del sensor central en los últimos 10 instantes de tiempo, y como se vió antes, estas lecturas eran las que permitían saber si se entraba a una curva o si se salía de está, dando lugar a una aceleración o deceleración del coche. En este caso, como solo interesa conocer la dirección en la que realizar el giro, solo se tendrán en cuenta las lecturas de cada uno de los sensores en el mismo instante de tiempo.
- **Capa intermedia**: En esta capa, no hay ningún criterio definido por el momento para poder decidir el número de capas y de neuronas por capa. En el anterior trabajo se utilizo una sola capa con 25 neuronas. Debido a que ahora se utilizan

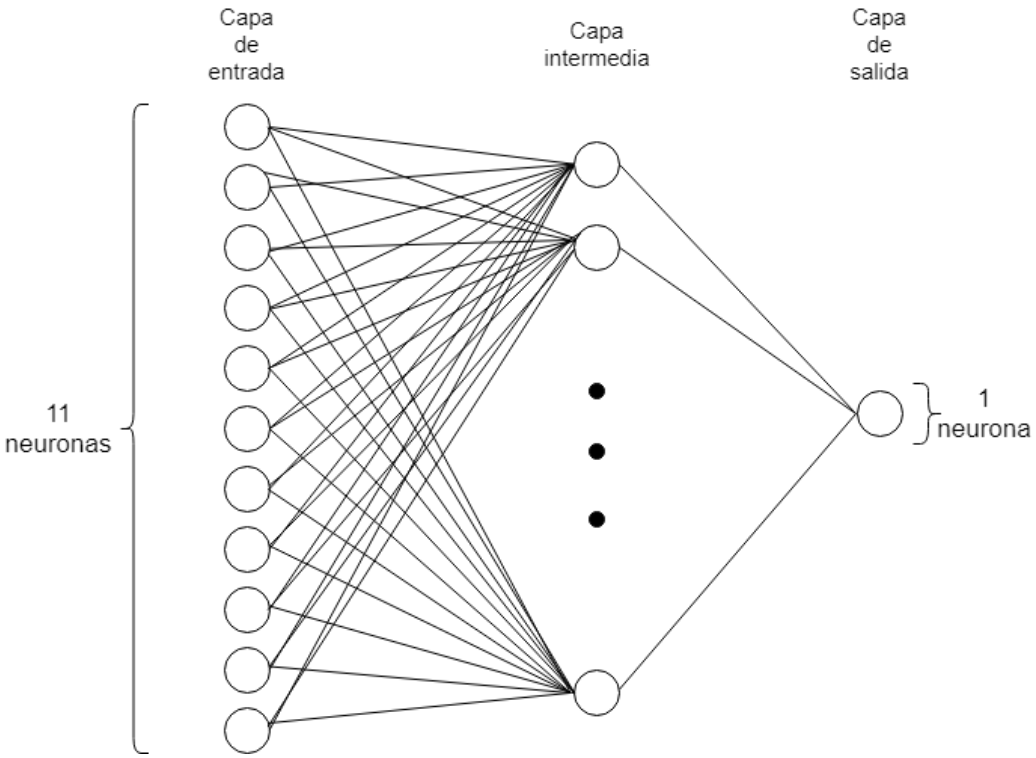

Figura 3.4: Red neuronal modificada en el nuevo trabajo

menos entradas, resultaba lógico pensar que también habría que reducir de forma proporcional el número de neuronas en esta capa, así que en un primer momento este se redujo a 13. Sin embargo, a la hora de probar diferentes entrenamientos se han utilizado diferentes números de neuronas en esta capa.

- **Capa de salida**: En esta última capa, se eliminó una de las neuronas. El motivo es sencillo, únicamente se tiene que controlar el giro del coche, puesto que la velocidad será constante, de forma que no tendría ningún sentido tener otra salida que se utilizaba para calcular el valor de la velocidad.

En cuanto a la regla de propagación, las RNs seguirán utilizando la suma ponderada de pesos, donde cada entrada estará asociada al peso de la conexión que llega hasta cada neurona.

Como función de transferencia, tanto en el trabajo anterior como en el trabajo, se utilizó principalmente la función sigmoide, la cual se puede ver en la figura 3.5.

Esta función se caracteriza por acotar los valores entre 0 y 1, de forma que valores muy altos llegarán a valer 1 o cerca de 1, y valores muy bajos llegarán a valer 0 o cerca 0. Es considerada la función más utilizada en las redes neuronales feedforward, aunque es también considerada una función de lenta convergencia.

Existen otras funciones, las cuales también serán utilizadas a modo de prueba:

- **Función tangente hiperbólica**: función similar a la sigmoide, pero esta vez acotada entre -1 y 1, lo cual da una mayor posibilidad de salidas, con lo que puede ser interesante. Sin embargo, al ser parecida a la sigmoide, también resulta tener una lenta convergencia.

- **Función arcotangente**: función similar a las anteriores, pero acotada entre $-\frac{\pi}{2}$ y $\frac{\pi}{2}$

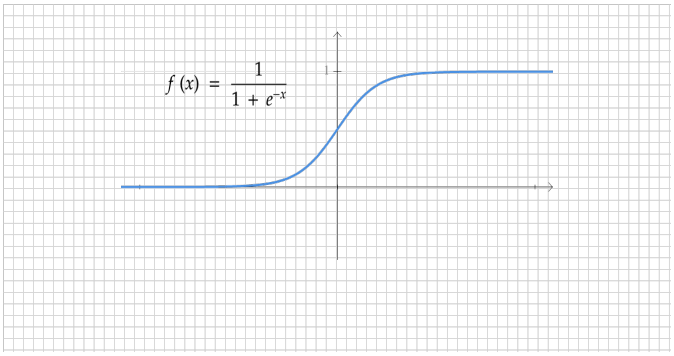

Figura 3.5: Función sigmoide

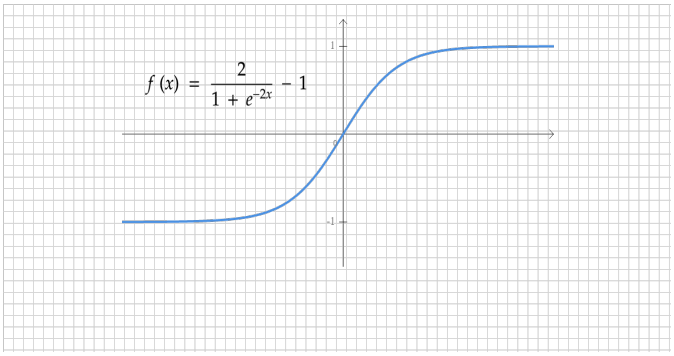

Figura 3.6: Función tangente hiperbólica

3.6 Simulación

En este apartado, se comentarán todos los aspectos modificados y añadidos en la simulación del programa, que se pueden ver a la hora de ejecutar el mismo. Hay muchos y diversos, con lo cual habrá diferentes apartados para cada uno.

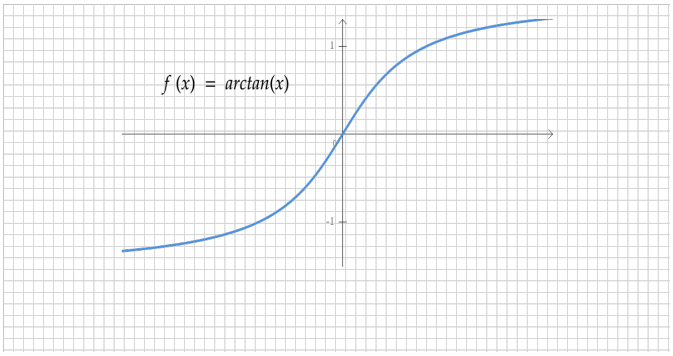

Figura 3.7: Función arcotangente

3.6.1 Presencia del usuario

En el anterior trabajo, se disponía de la posibilidad de poder manejar a uno de los coches y competir contra los coches que ya habían sido entrenados. Sin embargo, dicha opción se ha suprimido debido a que forma parte del aprendizaje supervisado, el cual no es el tipo de aprendizaje tratado en este TFG, de forma que toda la parte de código asociada a esta parte no ha sido reutilizada para el trabajo realizado.

3.6.2 Visualización de los coches

En el momento que empezaba la carrera, de forma automática se visualizaba al primer coche, junto a toda su información:

- Circuito en el que están corriendo los coches.
- Función de ponderación que utilizan los coches.
- Valores correspondientes a la velocidad y giro del coche mostrado actualmente.
- Número de coches vivos en la carrera (que no han colisionado).
- Número de vuelta actual del coche mostrado.
- Tiempo de carrera.
- Distancia recorrida por el coche actualmente mostrado.
- Número de coche mostrado actualmente.

Así pues, con el objetivo de aportar mayor información y de mejorar la experiencia de usuario durante la simulación, se han añadido las características que se verán a continuación.

Coche a mostrar

Anteriormente, solamente se mostraba el primer coche, lo cual impedía realizar un seguimiento a los demás coches, algo que para este trabajo es necesario para poder comparar a los coches que van entrenando, como más adelante se verá.

Por tanto, se modifico la función *display* de la clase principal (OpenGLWindow), con el objetivo de permitir mostrar los valores anteriores correspondientes al coche que esté actualmente mostrado en este momento, sin que necesariamente sea el primero.

Para poder mostrar cualquiera de los coches, se implementó una nueva función, la cual permite cambiar el coche que se visualizará en ese momento. Como los coches están enumerados, el cambio entre coche y coche podrá ser en orden decreciente y en orden creciente. Una vez realizado este cambio, no se seguirá al coche en primer lugar, sino al que se tenga seleccionado. Este nuevo sistema se controla de la siguiente manera:

- **Tecla de dirección izquierda**: Visualizar el anterior coche. Si es el primero, se volverá al último.
- **Tecla de dirección derecha**: Visualizar el siguiente coche. Si es el último, se volverá al primero.
- **Tecla de dirección arriba**: Visualizar el coche que vaya en primer lugar.
- **Tecla de dirección abajo**: Cambiar el modo de visualización. Existen dos modos cuando se permite cambiar de vehículo:
  - **Modo normal**: Este permitirá visualizar los coches en orden creciente o decreciente según se vaya indicando con las teclas ya mencionadas, con indenpendencia de que los coches hayan colisionado o no.
  - **Modo vivos**: Este permitirá visualizar únicamente a los coches que estén vivos, también en orden creciente o decreciente.

  Por defecto, estará activo el modo normal.

Sensores de los coches

En el anterior trabajo, siempre se mostraban los sensores de los coches, aunque solo fuera posible mostrar los datos de uno solo. Esto podía causar mucha confusión, ya que muchas veces se solapaban las marcas de los sensores causando un efecto visual incómodo. Para ello, se optó por solamente mostrar los sensores del coche que se está visualizando actualmente.

3.6.3 Opciones de entrenamiento y pruebas

En el anterior trabajo, a la hora de simular, inicialmente se le proponía al usuario qué coches, los cuales habían sido entrenados, deseaba visualizar. Cada entrenamiento se había realizado con 30 coches y con una serie de opciones, que son las siguientes:

- **Orden de entrenamiento**: El orden para entrenar podía ser el siguiente:
  - **De fácil a difícil**: En este caso, los coches han empezado a entrenar desde los circuitos más fáciles (enumerados con un menor número) hasta los más difíciles (enumerados con un mayor número, hasta 12)
  - **De difícil a fácil**: En este caso, los coches han empezado a entrenar desde los circuitos más difíciles hasta los más fáciles.
- **Función de ponderación**: Es la que permite definir cuál es el valor que hace que un coche sea mejor que otro. Las que se desarrollaron en el anterior trabajo fueron estas:
  - **Distancia**: el mejor coche es el que más distancia ha recorrido en el circuito.
  - **Distancia^2**: lo mismo que en el caso anterior, pero este se eleva al cuadrado.
  - **Velocidad**: el mejor coche es el que más distancia ha recorrido dividido el tiempo que ha tardado en colisionar o llegar meta.
  - **Velocidad * Distancia**: se trata de una combinación de distancia y velocidad, lo cual se traduce a la distancia^2 dividido por tiempo.

Sin embargo, en el anterior simulador no existía la posibilidad de realizar los entrenamientos a tiempo real. Por necesidad y objetivos de este TFG, en el simulador existe la opción de realizar el entrenamiento, aunque seguirá existiendo la opción de realizar un test con un entrenamiento realizado a poder elegir por el usuario, como más adelante se explicará. Además, en cuanto a las funciones de ponderación, en este trabajo únicamente se utilizarán las que valoran la distancia, debido a que en este trabajo los coches trabajan a velocidad constante, y no resulta demasiado útil tener en cuenta la velocidad para valorar a un coche. Finalmente, en cuanto al orden de entrenamiento, esta opción no existe, ya que el usuario podrá definir el orden de los circuitos a entrenar.

3.6.4 Finalización de una simulación

En el trabajo previo, se considera una simulación el hecho de lanzar a un conjunto de coches a que recorran un circuito una vez. Dicha simulación, finalizará en caso de cumplirse una condiciones:

- Si todos los coches colisionan.

- Si un coche llega la meta.

El problema que se puede observar aquí, es que cuando un coche llegue a la meta tras completar el número de vueltas necesario, obliga al programa a finalizar la simulación, sin que se pueda ver que como otros coches pueden llegar al final.

Para ello, se ha modificado la función *idle* del programa principal, la cual ahora también comprueba si todos los coches que quedan vivos han completado las dos vueltas, y no solamente uno de estos. De esta forma, es posible ver como los demás coches van circulando, lo cual puede dar un aspecto visual de como está yendo el entrenamiento en cada momento.

Sin embargo, un problema que surgió al implementar este cambio fue que los coches que ya habían completado el circuito seguían circulando, debido a que todavía quedaban coches vivos por llegar. La solución a este problema consistió en detener a los vehículos que completaron el circuito, y dejar de renderizarlos.

3.6.5 Versión sin interfaz gráfica de usuario

Se ha desarrollado una versión que realiza las mismas simulaciones, pero sin mostrar en ninguna ventana la situación actual de los coches, circuitos y demás aspectos de esta.

Con esta versión es posible ejecutar un entrenamiento cuando no se desea visualizar en cada momento la situación del mismo, con lo cuál, además, se ahorra en recursos temporales y espaciales (memoria), de manera que se pueden realizar entrenamientos de una forma más eficiente.

3.6.6 Representación gráfica de la Red Neuronal de un coche

Durante el entrenamiento de los coches, uno de los problemas o necesidades, era el de ver como estaban funcionando los coches que se estaban entrenando, de forma que resultaba difícil ver en que situación se encontraban las RNs entrenadas. Sí es cierto que se podían ver los coches que realizaban correctamente el circuito, pero si que se echaba en falta mayor información de la situación.
Es por este motivo, que se optó por desarrollar una implementación de un algoritmo que permitierá visualizar un dibujo de la RN del coche que se está mostrando actualmente, la cual se puede ver en la figura 3.8

A continuación se explicarán los significados de cada uno de los elementos que componen el gráfico.

Entradas de la RN

Como se ha mencionado en diferentes ocasiones, las entradas de la red neuronal representan las distancias de colisiones de cada uno de los sensores del coche. En esta representación gráfica, como se puede ver en la figura 3.9, la codificación utilizada,

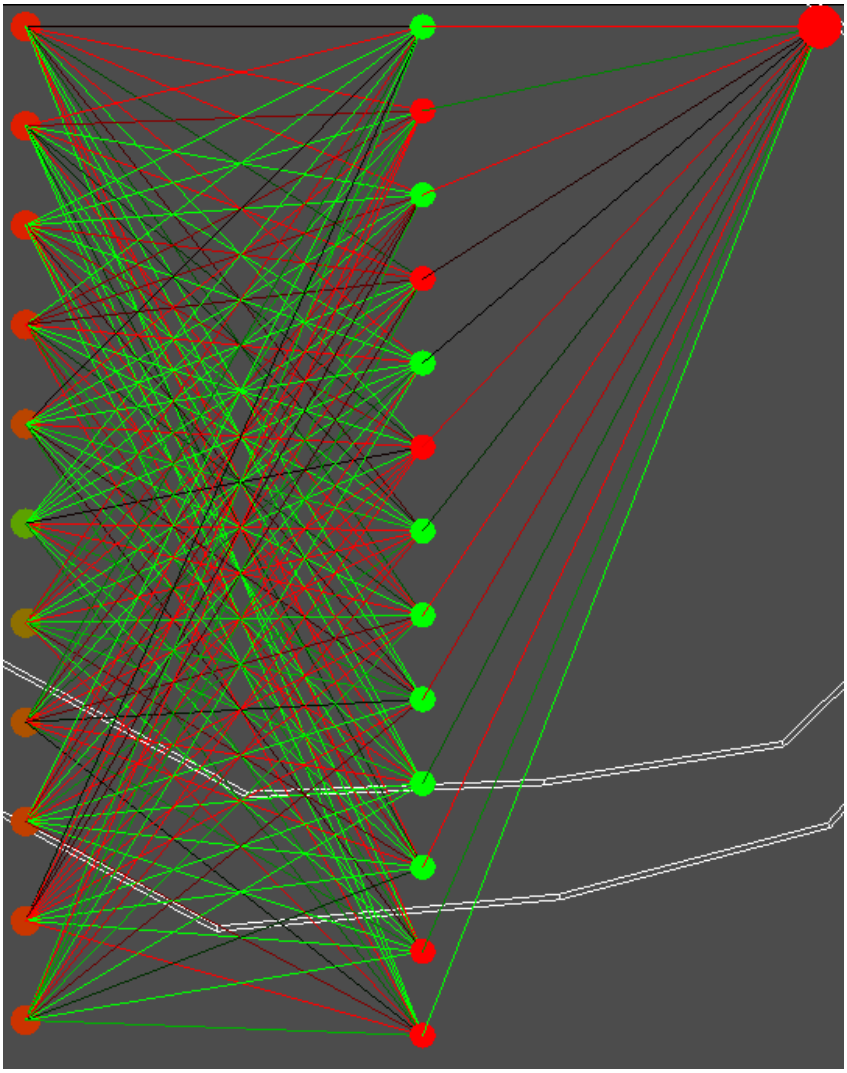

Figura 3.8: Representación gráfica de la RN del coche mostrado actualmente. Esta se visualiza en la parte derecha de la ventana de simulación.

para cada instante de tiempo es la siguiente:

- Si la distancia de colisión de un sensor es menor a 5 unidades, el color de la neurona de entrada correspondiente será rojo.
- Si la distancia de colisión de un sensor se sitúa entre 5 y 50 unidades (una distancia intermedia), el color visualizado será una tonalidad entre rojo y verde, con lo cual se verá un tono naranjizo o amarillo.
- Si la distancia de colisión de un sensor se sitúa a más de 50 unidades, el color utilizado en este caso será el verde.

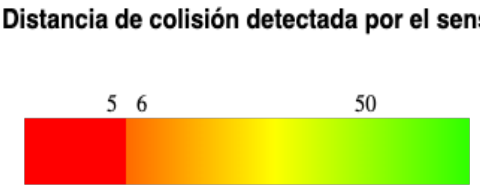

Figura 3.9: Representación colorimétrica de cada neurona de entrada según la distancia detectada por el sensor.

Conexiones de la RN

Para codificar las conexiones de la RN, cuyo valor viene a ser representado por los pesos de estas, inicilamente se optó por simple criterio colorimétrico:

- Si el peso es positivo, color verde.
- Si el peso es negativo, color rojo.

Sin embargo, el problema visto en esta situación es que no se puede comprender ninguna diferencia entre valores positivos grandes y valores positivos más pequeños, ni tampoco entre valores negativos más pequeños y más grandes negativamente.

Por tanto, para solucionar dicho problema, se optó por utilizar un criterio similar ya visto en las entradas de la RN: utilizar un grado de color para representar la magnitud de cada peso, sea positivo o negativo, como se puede ver en la figura 3.10:

- Para los pesos positivos, se utilizará un verde más intenso, y un verde menos intenso para valores más pequeños positivamente.
- Para los pesos negativos, se utilizará un rojo más intenso, mientras que se utilizará un rojo menos intenso para valores más pequeños negativamente.

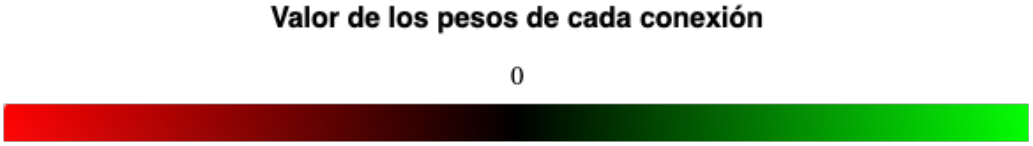

Figura 3.10: Representación colorimétrica de cada conexión según el valor del peso asociado.

Neuronas intermedias de la RN

Para las neuronas que conforman las capas intermedias, también se ha utilizado un criterio colorimétrico para representar su valor. En este caso, el criterio es algo diferente. Para este caso, se utilizará el valor del umbral de cada neurona para computar el

resultado:

- Si el valor de la salida computada es mayor que 0, el color utilizado será el verde.
- Si el valor de la salida computada es menor que 0, entonces se utilizará el rojo.

De está forma, cada neurona de este tipo tendrá un color u otro. El problema es que hay funciones que siempre dan valor positivo (la sigmoide) con lo cual el color visualizado siempre sería el verde, aunque para otras funciones (la tangente hiperbólica) si que se visualizarían ambos colores.

Salidas de la RN

Como ya se ha descrito en anteriores ocasiones, la capa de salida está formada por neuronas que dan lugar a la salida completa de la red, y en este caso, la única neurona utilizada como salida dará lugar al valor del giro del coche, tal y como se puede ver en la figura 3.11:

- Si el valor de la salida se sitúa por encima de un pequeño umbral por encima de 0, quiere decir que el coche ha de girar a la izquierda.
- Si el valor de la salida se sitúa por debajo de un pequeño umbral por debajo de 0, quiere decir que el coche ha de girar a la derecha.
- Sin embargo, si el valor se sitúa dentro de esos umbrales, significa que el coche irá en dirección recta.

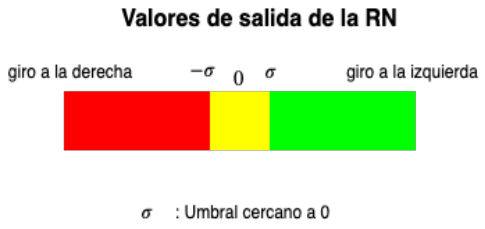

Figura 3.11: Representación colorimétrica de cada neurona de salida.

Finalmente, es importante saber que esta mejora fue desarrollada con el propósito de no solamente añadir información visual atractiva para el usuario, sino que además permite:

- Junto a la funcionalidad desarrollada que permite cambiar de coche a visualizar, es posible poder comparar visualmente el aspecto de las RNs de cada uno de los coches.

- Se puede visualizar el comportamiento de la RN ante diferentes entradas.
- Visualizando las conexiones en cada simulación, se puede ver que aspecto van adquieriendo las RNs que van aprendiendo a recorrer los circuitos.

3.6.7 Aprendizaje de los coches

En esta sección se abarcarán todos los aspectos tratados para conseguir el aprendizaje de los coches, así como todos los tipos de entrenamientos que se han tratado de realizar.

En el trabajo previo, para realizar el entrenamiento, los coches recorrían cada circuito dos veces, de la siguiente forma:

1. Para k $\in$ [1,2]
    a) Para i $\in$ [1,12]
        i. Los coches recorren el circuito i una primera vez
        ii. Se aplica el filtro de partículas para aprender
        iii. Los coches recorren el circuito una segunda vez
        iv. Se aplica el filtro de partículas para aprender

El problema que se puede apreciar aquí, es que no se puede asegurar que los coches hayan aprendido a correr en todos los circuitos, puesto que es posible que un coche que ha completado uno de los circuitos más díficiles no sepa completar alguno de los fáciles.

Por tanto, una vez se realizó este mismo entrenamiento, se planteó la posibilidad de crear un entrenamiento que pudiera permitir el aprendizaje de los coches en todos los circuitos, de manera que se pudiera asegurar que al final del entrenamiento existe una RN que recorre todos los circuitos de entrenamiento correctamente, de forma que posteriormente se le pueda someter a un test en 3 circuitos.

Entrenamiento progresivo

En este entrenamiento, se propone una metodología que permite el entrenamiento de cada coche en diferentes rondas.

Estas rondas consisten en grupos de N circuitos, de forma que la siguiente ronda añadirá un circuito, con N + 1 circuitos. Si se consigue que al menos un mismo coche complete todos los circuitos de una determinada ronda, entonces se podrá pasar a la siguiente, mientras que en caso contrario, deberá repetirse la ronda hasta que no se cumpla lo anteriormente descrito. Por ejemplo, si se está realizando la ronda 3, el objetivo es conseguir una RN que permita a un coche completar los circuitos 1,2 y 3. Si al simular los 3 circuitos esto se consigue, la pasará a la siguiente ronda, donde se intentará conseguir lo mismo, esta vez para los circuitos 1,2,3 y 4, y en caso contrario, se volverá a intentar que un coche complete los circuitos 1,2 y 3.

La idea de entrenamiento inicial, fue la de ir incluyendo los circuitos en cada ronda de manera consecutiva, de orden creciente, y por tanto de fácil a difícil, como se puede observar en la figura 3.12:

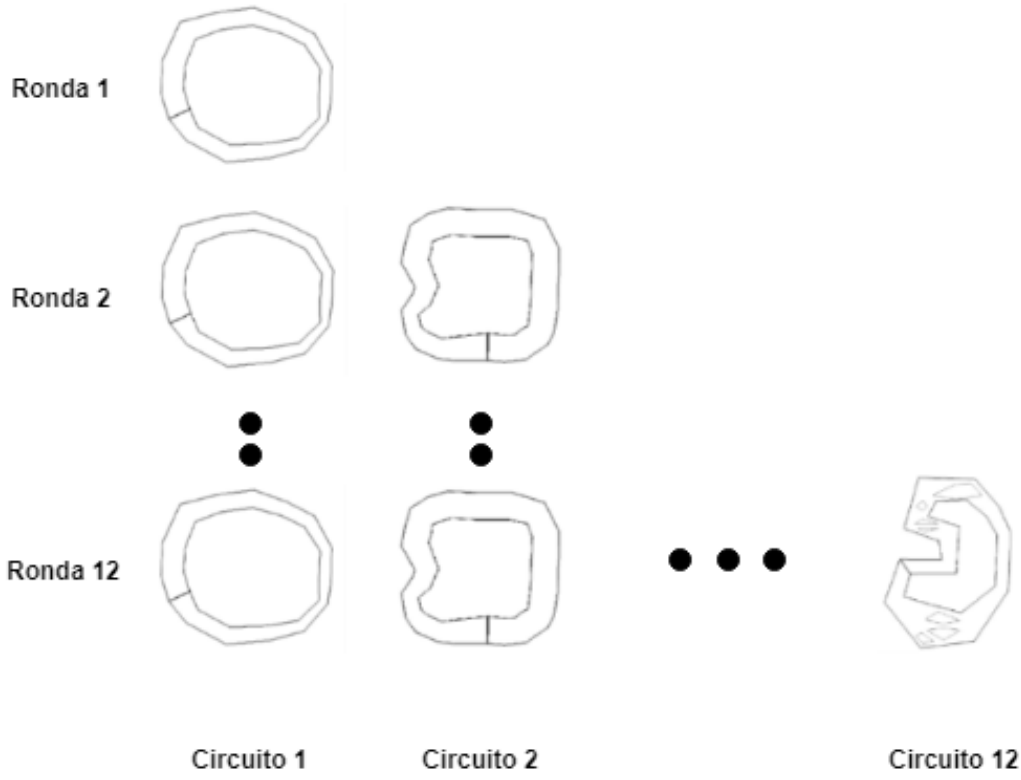

Figura 3.12: Representación gráfica del sistema de rondas del entrenamiento propuesto.

En un principio, esta era la idea inicial de entrenamiento, de manera que el entrenamiento iría de los circuitos fáciles hasta los difíciles. Se realizó este entrenamiento, con unos resultados que se verán en el siguiente capítulo. Es decir, el algoritmo realizado estaba preparado para simular los circuitos del 1 al 12, pero este se modificó para poder simular los circuitos deseados en el orden deseado, como se podrá ver a continuación.

Sin embargo, como ya se vió en el trabajo previo, el orden de entrenamiento de los circuitos hace que varíen los resultados y así mismo el camino para llegar a completarlo.

De todas formas, las pruebas que se realizarán en este trabajo, van algo más allá y tratarán de probar la influencia de otros parámetros en el entrenamiento:

- **La cantidad de coches utilizados**. Se harán pruebas con 30 y 100 coches.
- **La cantidad de circuitos utilizados**. De esta forma, se tendrán entrenamientos con más o menos circuitos, y con algunos en concreto, los cuales tengan cierto tipo de características a querer mejorar.

- **La función de ponderación**. Se probará con la distancia y la distancia al cuadrado. En este caso, el valor de la distancia será representado como un porcentaje de distancia completada sobre el total de cada circuito. Por tanto, un 100 % de puntuación indica que se ha completado el circuito.
- **El número de neuronas en la capa intermedia**. Se probarán diferentes valores: 7,13 y 21.
- **Cambiando la función de transferencia de la red**. Habrá 3 posibles: sigmoide, tangente hiperbólica y arcotangente.

Por tanto, para poder realizar los diferentes entrenamientos, se propondrá al usuario un conjunto de opciones a poder seleccionar para realizar el entrenamiento.

Una vez se tienen en mente todos estos tipos de entrenamiento, es necesario adaptar el algoritmo que se ejecuta durante la simulación de cada uno, el cual se puede ver en la figura 3.13.

```
while rondaActual < totalRondas do
    if simulacionActual >totalSimulaciones then
        terminarSinExito
    end if
    if cochesHanTerminadoCircuito or cochesVivos == 0  then
        if cochesVivos >0 then
            comprobarCochesQueHanTerminadoLosCircuitos
            if circuitoActualEsElUltimoDeLaRonda and algunoTerminoTodos then
                realizarAprendizaje
                siguienteRonda
            else if !algunoTerminadoTodos then
                // Ningún coche terminó todos los circuitos de la ronda hasta el actual
                realizarAprendizaje
                repetirRonda
            else
                siguienteCircuitoDeLaRonda
            end if
        else
            realizarAprendizaje // Ningún coche terminó el circuito actual
            repetirRonda
        end if
    end if
end while
```

Figura 3.13: Algoritmo de entrenamiento

A continuación se describirá el significado de cada una de las sentencias:

- **cochesHanTerminadoCircuitos**: Esta función permite saber que coches han alcanzado la línea de meta, es decir, que han completado 2 vueltas en el circuito.

- **cochesVivos**: Se trata de una variable que permite saber cual es la cantidad de coches que no han colisionado.
- **comprobarCochesQueHanTerminado**: Esta función se encarga de saber que coches han terminado todos los circuitos de la ronda hasta el momento, de forma que habrá una variable que indicará que coches si han cumplido esa condición.
- **circuitoActualEsElUltimoRonda**: Esta función permite saber si el circuito actualmente simulado es el último de la ronda, de forma que se puede saber si ha de pasar a la siguiente ronda o no.
- **algunoTerminoTodos**: Esta función permite conocer que al menos existe un mismo coche que ha terminado todos los circuitos hasta ahora.
- **realizarAprendizaje**: Se realiza el proceso de aprendizaje, en el cual se aplica el algoritmo del filtro de partículas, teniendo en cuenta una función que permite dar las puntuaciones de cada coche, las cuales dependen de la distancia recorrida por este, según dos posibles soluciones:
  - *Suma de puntuaciones*: Se realiza la suma de todas las puntuaciones en cada circuito para cada coche, de forma que al final, el coche que mejor lo ha hecho tendrá la puntuación máxima posible, haciendo que con el filtro de partículas este sea el más probable a ser seleccionado.
  - *Mínimo de puntuaciones*: Se computa la puntuación mínima de cada coche en todos los circuitos. Como resultado, se tiene que el coche que haya completado todos los circuitos, tendrá como puntuación el máximo de un circuito, mientras que los demás tendrán valores muy pequeños, pues serán sus peores puntuaciones, dando aún mucha más ventaja al coche que los ha terminado todos. Por tanto, está función parece dar mejores resultados, porque beneficia mucho más a los que completen todos los circuitos y perjudica a los que no lo hacen.

  Las puntuaciones de cada circuito se asignan de la siguiente manera: se calcula la división entre la distancia recorrida por el coche y la distancia total del circuito en 2 vueltas y se multiplica por 100, de forma que cada puntuación indica el porcentaje de completitud de un circuito.
- **siguienteRonda**: Se vuelve al primer circuito y a la siguiente ronda, lo cual añade un circuito más al entrenamiento.
- **repetirRonda**: Se vuelve al primer circuito, pero se mantiene la misma ronda, puesto que ningún coche ha completado todos los circuitos de esa ronda.

Una vez definido el algoritmo que se utilizará para realizar los entrenamientos, se procederá a comentar todos y cada uno de los diferentes entrenamientos que se intentarán realizar, con el fin de poder sacar conclusiones interesantes:

- **Entrenamiento total (de fácil a difícil)**: Este entrenamiento consiste en completar los 12 circuitos en orden creciente.

	Circuito	1	2	3
Nº de coche				
1		100	100	100
2		100	50	30
3		100	10	15

	Puntuación final (mínimo)	Puntuación final (suma)
Nº de coche		
1	100	300
2	30	180
3	10	125

Figura 3.14: Ejemplo del sistema de puntos a utilizar, en este caso, en los circuitos 1,2 y 3. Como se puede observar, el coche nº 1 ha completado los 3 circuitos, de forma que a la hora de calcular de puntuación final, la función mínimo penaliza más a los coches que no completan los circuitos y beneficia más a los que sí lo consiguen.

- **Entrenamiento total (de difícil a fácil)**: Este entrenamiento consiste en completar los 12 circuitos en orden decreciente.
- **Entrenamiento en solo circuitos fáciles**: El objetivo es aprender únicamente a conducir en circuitos más sencillos. Son los circuitos del 1 al 6.
- **Entrenamiento en solo circuitos difíciles**: El objetivo es aprender únicamente a conducir en circuitos más difíciles Son los circuitos del 7 al 12.
- **Entrenamiento en circuitos con obstáculos**: El objetivo es aprender a recorrer circuitos de diversas dificultades, los cuales tienen obstáculos. Son los circuitos 3,8,10 y 12.
- **Entrenamiento en circuitos con muchas curvas**: El objetivo es aprender a recorrer circuitos con la presencia de muchas curvas y curvas muy seguidas. Son los circuitos 4,7,9 y 11.

Funcionamiento del simulador desarrollado

Para poder realizar los entrenamientos adecuadamente, es necesario indicar inicilamente que parámetros deseamos seleccionar para este.

Por ello, a continuación se explicará el uso del simulador. Como se puede ver en la figura 3.15, esta primera ventana ofrece un par de opciones:

- **Elegir los circuitos en los que se realizará el entrenamiento o test**. Se deben escribir los números de cada circuito separados por coma, de forma que en el orden que estén puestos será el orden en el que se irán entrenando.

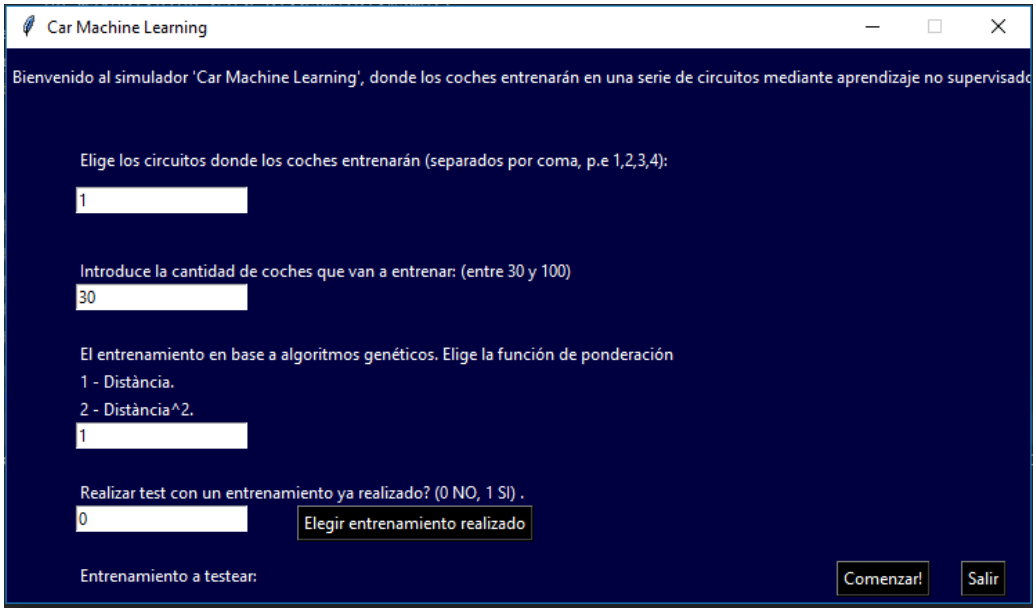

Figura 3.15: Ventana incial del simulador

- **Elegir la cantidad de vehículos que se entrenarán o testearán**. Se debe escribir el número de coches.
- **Elegir la función de ponderación**. Existen dos funciones, la distancia (1) y la distancia al cuadrado (2).
- **Elegir el entrenamiento a testear**. Si se escribe 0, no se tendrá en cuenta nada y se realizará un entrenamiento con los parámetros escogidos, pero si se escribe 1, se ha de seleccionar una carpeta pulsando el botón 'Elegir entrenamiento realizado', de forma que los coches entrenados con cierto entrenamiento serán testeados.

Respecto a la última cuestión, hay que destacar que cada entrenamiento realizado genera un conjunto de archivos que son almacenados en un directorio cuyo formato es el siguiente:

$$Circ\_(circuitos)C\_(coches)P\_(ponderacion)$$

, donde *circuitos* son los números de los circuitos separados por guión, *coches* es el número de coches que han hecho el test y *ponderacion* es el número correspondiente a la función de ponderación utilizada. Por ejemplo si se tiene Circ_1-2-3-4-5-6-C_30_P_2, quiere decir que se ha realizado un entrenamiento en los circuitos 1,2,3,4,5 y 6, utilizándose 30 coches y con la función de ponderación Distancia al cuadrado.

Con este formato, se permite cononcer que parametros ha utilizado cada entrenamiento. Cada directorio contiene un número determinado de archivos .csv, donde hay tantos como coches se hayan utilizado, correspondientes a las RNs de cada uno de

estos, y un archivo "log.txt", donde se pueden comprobar cada una de las simulaciones y rondas con los resultados obtenidos (puntuaciones de los coches en cada ronda y en cada circuito, distancia recorrida y tiempo tardado)3.16,3.17 3.18.

```
Circuito :  1
Ronda:  1
Simulacion :  1
Tiempo tardado:  157.68000000000225
Máxima distancia recorrida:  672.0386362685324
Número de coches vivos:  1
```

Figura 3.16: Información general sobre una simulación.

```
Car nº 1, : 22.03 , Distancia: 31.52 , Tiempo: 8.48
Car nº 2, : 778.85 , Distancia: 187.44 , Tiempo: 47.28
Car nº 3, : 11.68 , Distancia: 22.96 , Tiempo: 6.32
Car nº 4, : 20.62 , Distancia: 30.49 , Tiempo: 7.28
Car nº 5, : 0.45 , Distancia: 4.48 , Tiempo: 2.40
Car nº 6, : 0.25 , Distancia: 3.35 , Tiempo: 2.32
Car nº 7, : 35.56 , Distancia: 40.05 , Tiempo: 10.88
Car nº 8, : 0.92 , Distancia: 6.43 , Tiempo: 2.56
Car nº 9, : 625.41 , Distancia: 167.96 , Tiempo: 42.24
Car nº 10, : 0.56 , Distancia: 5.05 , Tiempo: 2.48
```

Figura 3.17: Resultados de cada coche: puntos obtenidos, distancia recorrida y tiempo tardado.

```
APRENDO
Coche nº 1, : 22.03
Coche nº 2, : 778.85
Coche nº 3, : 11.68
Coche nº 4, : 20.62
Coche nº 5, : 0.45
Coche nº 6, : 0.25
Coche nº 7, : 35.56
Coche nº 8, : 0.92
Coche nº 9, : 625.41
Coche nº 10, : 0.56
```

Figura 3.18: Valores de ponderación o *fitness* de cada coche al realizar aplicar el algoritmo de aprendizaje.

Todo lo anterior se centra en la parte de los entrenamientos. Sin embargo, el simulador también permite realizar tests con los coches ya entrenados. Para elegir el entrenamiento que se debe realizar, hay que seleccionar la carpeta del entrenamiento, con lo cual el programa cargará los coches entrenados en los circuitos seleccionados.

Para probarlos en los circuitos de test, hay que indicar los circuitos 13,14 y 15, aunque en realidad pueden ser probados en cualquiera de estos. De una forma similar que con los entrenamientos, los resultados de simular los tests serán guardados en un

directorio con el formato siguiente:

$$Test\_Circ\_(circuitos)C\_(coches)P\_(ponderacion)$$

, donde *circuitos* son los números de los circuitos separados por coma, *coches* es el número de coches que han hecho el test y *ponderacion* es el número correspondiente a la función de ponderación utilizada. Para poder realizar el test de un entrenamiento ya realizado, es importante seleccionar en la ventana del simulador los mismos parámetros que los fueron usados en el entrenamiento (número de coches y función de ponderación).

3.6.8 Consideraciones finales del entrenamiento

Finalmente, se ha concluido con la explicación del funcionamiento del sistema de entrenamientos y tests desarrollado. Sin embargo, para finalizar es necesario tener en cuenta una serie de consideraciones necesarias para entender los resultados que se explicarán en el siguiente capítulo:

- En cuanto a la estructura de las RN, se ha optado por una red con 11 neuronas de entrada, 13 neuronas en la capa intermedia y 1 de salida. Las razones para dicho valor de neuronas intermedias son las siguientes:
  - Un valor de neuronas mayor que 20 generaba un mayor espacio de posibles soluciones, lo cual significa que puede converger, el único problema es que dicha convergencia es muy lenta.
  - Un valor de neuronas menor que 10 generaba un menor espacio de posibles soluciones, de forma que no siempre podía llegar a converger si se entrenaba utilizando diferentes circuitos.
  - Añadiendo más capas se consigue una red más compleja, de forma que también tarda más en entrenar.
- Por tanto, ese valor de neuronas resulta en un entrenamiento realizable y sin una excesiva lentitud.
- En cuanto a la función de activación, se ha utilizado la función sigmoide, una de las más populares para este tipo de redes, la redes *feedfordward*, ya que es sencilla de computar y tiene buen desempeño con redes neuronales más simples. (20)

3.7 Tecnología utilizada

Para el desarrollo del programa se ha utilizado el lenguaje Python 3.6.4, el cual incorpora las siguientes librerias:

- La librería Numpy 1.4, la cual ha sido utilizada para gestionar la clase Network, la cual representa la RN de cada coche.

- La librería OpenGL 1.4 para poder renderizar la escena correspondiente a las simulaciones de los coches.
- La librería Tkinter, de la cual hace uso la clase Inici, con lo que se permite gestionar una ventana con diferentes opciones para los entrenamientos y tests.
- La librería Pickle 4.0, que permite guardar las RN de los coches una vez finalizadas las simulaciones, de forma que también se puede cargar, como archivos ".csv"

En cuanto al desarrollo de esta memoria, se han utilizado las siguientes herramientas:

- Para el desarrollo de la memoria se ha utilizado Latex, con el editor Overleaf, utilizando las plantillas ofrecidas por la EPS.
- Para generar las figuras correspondientes a las RNs y a otras figuras se ha utilizado el programa Draw.io.
- Para generar las figuras correspondientes a las funciones de transferencia y expresiones matemáticas se ha utilizado el programa Mathcha.io.

El programa realizado está incluido junto a esta memoria, de forma que se pueden consultar el código fuente, como así ver las instrucciones para instalar las librerías de OpenGL necesarias para ejecutar el mismo. Además, con el programa se adjuntan varias carpetas con entrenamientos, tests y resultados de las pruebas realizadas.

CAPÍTULO 4

RESULTADOS

En este capítulo se mostrarán los resultados de los entrenamientos mencionados previamente. Una vez se realizaron los entrenamientos mencionados, los coches entrenados en cada uno de estos fueron probados en los circuitos de test, enumerados como 13, 14 y 15.

Para cada uno de estos tests, se mostrarán los resultados correspondientes, con la siguiente información:

- Tipo de entrenamiento realizado. Esto incluye:
  - Circuitos utilizados para el entrenamiento.
  - Cantidad de coches utilizados en el entrenamiento.
  - Función de ponderación utilizada.
- Cantidad de coches utilizados
- Cantidad de coches vivos.
- Para los coches que completaron todos los circuitos entrenados:
  - Distancia recorrida en cada circuito de test.
  - Tiempo tardado en recorrer cada circuito de test.

4.1 Primeras pruebas y resultados

Inicialmente, se trató de ejecutar un entrenamiento utilizando el sistema desarrollado, con 30 coches en todos los circuitos de entrenamiento, del 1 al 12, para conseguir un coche que los hiciera todos bien. Esto se consiguió, pero no se pudieron guardar estas RNs, aunque si se guardó el log con todo el entrenamiento descrito.

Entonces, una vez se modificó el simulador, se volvió a intentar realizar el mismo entrenamiento, pero el proceso se extendió durante más de un día y no fue posible concluirlo. Es de suponer que, si en una primera vez funcionó, aunque en menos tiempo, es muy probable que el entrenamiento hubiera acabado finalizando exitosamente.

Por tanto, para poder realizar una mayor cantidad de estos entrenamientos, se decidió entrenar en una menor cantidad de circuitos, para ver si era suficiente para poder aprender a conducir y superar los circuitos de test en los que nunca había conducido.

Cada entrenamiento pues, se realizará repetidas veces, puesto que hay factores aleatorios que pueden influir y dar resultados diferentes. Estos factores son los siguientes:

- **RNs iniciales**. Este hecho provoca que la situación inicial de cada entrenamiento sea diferente, y que por tanto el aprendizaje tome un rumbo diferente.
- **Algoritmo de aprendizaje**. Como ya se ha visto en el capítulo 3, el algoritmo utilizado para aprender se basa en el filtro de partículas, el cual permite, con una mayor probabilidad, seleccionar al mejor coche para crear la nueva generación. Por tanto, esto implica que no siempre será elegido el mejor coche, y por tanto las generaciones resultantes pueden ser en ocasiones peores que las anteriores, con lo cual se tardará más en converger. Además, como suele ser habitual en las técnicas basados en Algoritmo Genéticos (AGs), se aplica una ligera modificación o ruido para no replicar, en este caso, al coche seleccionado.

Para las diferentes muestras obtenidas de cada tipo de entrenamiento, se ha obtenido cual es la mayor distancia del mejor coche obtenido en el entrenamiento. Se define como mejor coche a aquel coche que ha completado los N circuitos en los que ha sido entrenado. El motivo por el que se muestran los resultados de ese tipo de coche es debido a que el interés de estas pruebas es demostrar que con el entrenamiento en determinados circuitos, realizado hasta que se consigue al menos un coche que los complete todos, permite obtener buenos resultados en cualquier circuito al que sea sometido.

Todas estas pruebas están guardadas en un conjunto de directorios, tanto los entrenamientos como los tests asociados a estos.

4.2 Entrenamientos en los 6 primeros circuitos

El entrenamiento realizado en este caso incluye los 6 primeros circuitos, en orden creciente, y por tanto de fácil a difícil, de forma que se obtienen RNs que son capaces de completar los 6 circuitos. Se han realizado diferentes entrenamientos modificando la cantidad de coches y la función de ponderación utilizada. Por tanto se ha realizado este entrenamiento varias veces:

- 5 veces para 30 coches y función de ponderación d.

- 5 veces para 30 coches y función de ponderacion d^2.
- 2 veces para 100 coches y función de ponderación d.
- 2 veces para 100 coches y función de ponderación d^2.

Para cada circuito de los 3 circuitos de test, se mostrarán los mejores resultados conseguidos por un coche que ha completado satisfactoriamente el entrenamiento. Además, se mostrarán la media y la desviación típica obtenida en cada uno de los circuitos.

Circuito Test 1			
Entrenando en los circuitos 1, 2, 3, 4, 5 y 6			
Ponderación	**Coches**	**Distancia**	**Tiempo**
d	30	417,71	88.48
d	100	417,52	89,36
d^2	30	417,62	88,72
d^2	100	417,47	89,36

Cuadro 4.1: Mejor resultado para cada uno de los entrenamientos realizados en los 6 primeros circuitos, para el circuito test 1

En el cuadro 4.1 se puede ver que no existen apenas diferencias notables utilizando diferentes parámetros, además de que hay un coche mínimo que que ha superado el primer circuito de test.

Circuito Test 1		
Entrenando en los circuitos 1, 2, 3, 4, 5 y 6		
Nº de veces completado por un mejor coche: 14/14		
	Distancia	Tiempo
Media	417,58	92,54
Desviación típica	0,26	4,98

Cuadro 4.2: Resultados generales obtenidos en todos los entrenamientos en el circuito test 1, después de entrenar en los circuitos 1,2,3,4,5 y 6.

Como se puede observar en el cuadro 4.2, los resultados obtenidos indican que se ha conseguido por lo menos que un coche que ha completado los 6 primeros circuitos con éxito haya podido completar siempre el circuito de test 1.

Como se puede ver en el cuadro 4.3, en cada tipo de entrenamiento realizado se ha conseguido un coche que ha completado este circuito, aunque no se observan diferencias notables usando unos parámetros u otros.

Del mismo modo que en el circuito test 1, como se ha visto en el cuadro 4.2, se puede ver en 4.4 que todos los mejores coches que han terminado los circuitos del 1 al

Circuito Test 2			
Entrenando en los circuitos 1, 2, 3, 4, 5 y 6			
Ponderación	**Coches**	**Distancia**	**Tiempo**
d	30	659,86	141,2
d	100	660,04	140,08
d^2	30	659,81	136,88
d^2	100	659,85	144,72

Cuadro 4.3: Mejor resultado para cada uno de los entrenamientos realizados en los 6 primeros circuitos, para el circuito test 2.

Circuito Test 2		
Entrenando en los circuitos 1, 2, 3, 4, 5 y 6		
Nº de veces completado por un mejor coche: 14/14		
	Distancia	Tiempo
Media	659,95	142,9
Desviación típica	0,12	3,69

Cuadro 4.4: Resultados generales obtenidos en todos los entrenamientos en el circuito test 2, después de entrenar en los circuitos 1,2,3,4,5 y 6.

6 han completado el circuito de test 2.

Circuito Test 3			
Entrenando en los circuitos 1, 2, 3, 4, 5 y 6			
Ponderación	**Coches**	**Distancia**	**Tiempo**
d	30	827,8	175,2
d	100	827,74	187,44
d^2	30	827,93	174
d^2	100	827,99	181,44

Cuadro 4.5: Mejor resultado para cada uno de los entrenamientos realizados en los 6 primeros circuitos, para el circuito test 3.

Como se puede ver en el cuadro 4.5, se ha podido conseguir que un coche entrenado complete el circuito de test 3, aunque con algunas ligeras variaciones de tiempo.

Sin embargo, como se puede observar en el cuadro 4.6, la media obtenida resulta ser inferior a la distancia total del circuito, lo cual significa que no se ha conseguido un coche que complete el circuito test 3. De hecho, de los 14 entrenamientos realizados en este caso, en 3 de estos no se ha conseguido completar el circuito test 3, lo cual radica

Circuito Test 3		
Entrenando en los circuitos 1, 2, 3, 4, 5 y 6		
Nº de veces completado por un mejor coche: 11/14		
	Distancia	Tiempo
Media	684,75	149,65
Desviación típica	286,18	63,06

Cuadro 4.6: Resultados generales obtenidos en todos los entrenamientos en el circuito test 3, después de entrenar en los circuitos 1,2,3,4,5 y 6.

en una precisión del 78,58 % para este circuito.

Por tanto, se puede ver que con el entrenamiento de los circuitos 1,2,3,4,5 y 6, los circuitos de test 1 y 2 son completados con exito, mientras que en el circuito de test 3, no es posible asegurar su completitud con este entrenamiento, aunque si que ha sido completado en la mayoría de ocasiones.

4.3 Entrenamientos en los circuitos más difíciles

En vista del anterior entrenamiento realizado, se ha propuesto la posibilidad de realizar el entrenamiento en los 6 últimos circuitos: 7,8,9,10,11 y 12. Sin embargo, esto no ha sido posible debido a que se ha visto que se trata de un entrenamiento con lenta convergencia, debido a la gran dificultad de los circuitos. Por ello, el objetivo será entrenar en algunos de los circuitos más difíciles para averiguar como de bien resolverán los circuitos test, y si son capaces de mejorar los resultados del anterior entrenamiento.

4.3.1 Entrenamiento en el circuito 7

Para este entrenamiento, se han obtenido un total de 20 muestras, es decir, 20 entrenamientos de este tipo, para poder generalizar y comprobar los resultados al recorrer los circuitos de test. En esta ocasión todos los entrenamientos se han realizado con 30 coches, utilizando la función ponderación d^2.

En el cuadro 4.7 se puede ver que para el circuito test 1 se ha conseguido un valor medio que se corresponde a la distancia total del circuito, lo cual indica que para los 20 entrenamientos realizados de este tipo se ha conseguido un mejor coche que complete este primer circuito.

Para el circuito test 2, como se puede ver en el cuadro 4.8, la distancia media obtenida es inferior a la distancia correspondiente a completar dos vueltas en el circuito, lo cual indica que para los 20 entrenamientos realizados no siempre se ha encontrado un mejor coche que haya completado siempre este circuito, de hecho, hay 2 casos en los que no ha sido finalizado, de forma que no se puede asegurar que un coche que

Circuito Test 1		
Entrenando en el circuito 7		
Nº de veces completado por un mejor coche: 20/20		
	Distancia	Tiempo
Media	417,61	94,66
Desviación típica	0,34	4,81

Cuadro 4.7: Resultados generales obtenidos en todos los entrenamientos en el circuito test 1, después de entrenar en el circuito 7.

Circuito Test 2		
Entrenando en el circuito 7		
Nº de veces completado por un mejor coche: 18/20		
	Distancia	Tiempo
Media	604,19	134,68
Desviación típica	172,05	38,72

Cuadro 4.8: Resultados generales obtenidos en todos los entrenamientos en el circuito test 2, después de entrenar en el circuito 7.

completa el circuito 7 también pueda completar el circuito test 2.

Circuito Test 3		
Entrenando en el circuito 7		
Nº de veces completado por un mejor coche: 6/20		
	Distancia	Tiempo
Media	365,07	79,39
Desviación típica	315,71	69,63

Cuadro 4.9: Resultados generales obtenidos en todos los entrenamientos en el circuito test 3, después de entrenar en el circuito 7.

Finalmente, para el circuito de test más díficil, el número 3, se ha conseguido que la distancia media recorrida por los mejores coches 365,07, lo que equivale a recorrer casi la mitad del circuito, y una desviación típica de 315,71 , lo cual se deriva en el hecho de que solamente se ha conseguido que este circuito sea completado en 6 ocasiones, lo cual demuestra que no es superado satisfactoriamente.

4.3.2 Entrenamiento en el circuito 8

A continuación se verán los resultados de los tests realizados después del entrenamiento realizado en el circuito 8, considerado de los más difíciles. Debido a que este circuito es más difícil, haciendo que la convergencia sea más lenta, se ha optado por realizar menos pruebas, un total de 10.

Circuito Test 1		
Entrenando en el circuito 8		
Nº de veces completado por un mejor coche: 9/10		
	Distancia	Tiempo
Media	382,38	84,06
Desviación típica	111,46	24,33

Cuadro 4.10: Resultados generales obtenidos en todos los entrenamientos en el circuito test 1, después de entrenar en el circuito 8

Como se puede ver en el cuadro 4.10, el circuito test 1, el más fácil de los utilizados para probar, sigue siendo completado prácticamente siempre. Sin embargo, en este caso, hay un prueba en la que no se completa, lo cual parece tratarse de un caso atípico.

Circuito Test 2		
Entrenando en el circuito 8		
Nº de veces completado por un mejor coche: 10/10		
	Distancia	Tiempo
Media	659,97	146,04
Desviación típica	0,11	10,23

Cuadro 4.11: Resultados generales obtenidos en todos los entrenamientos en el circuito test 2, después de entrenar en el circuito 8

Para el circuito test 2, como se puede ver en el cuadro 4.12, en todos los entrenamientos se ha conseguido un mejor coche que lo completa siempre.

Finalmente, en el circuito test 3, de forma similar a los entrenamientos en el circuito 7, no se consigue que en más de un 30 % de los entrenamientos realizados se consiga un mejor coche que lo complete.

En definitiva, con los entrenamientos realizados, y sobretodo con el primero, se ha podido ver que se han obtenido mejores coches que han completado los 3 circuitos test, lo cual significa una mejora respecto a los resultados obtenidos en el trabajo previo.

Circuito Test 3		
Entrenando en el circuito 8		
Nº de veces completado por un mejor coche: 3/10		
	Distancia	Tiempo
Media	372,65	80,94
Desviación típica	319,3	71,14

Cuadro 4.12: Resultados generales obtenidos en todos los entrenamientos en el circuito test 3, después de entrenar en el circuito 8

CAPÍTULO 5

CONCLUSIONES

El objetivo de este TFG ha sido el de mejorar un juego de carreras en el que unos coches debían aprender a conducir, cuya base fue realizada por el alumno Mateu Morro Ribot, de manera que los cambios y mejoras realizadas abarcan diferentes aspectos del juego.

El tipo aprendizaje en el que se ha enfocado este TFG ha sido el no supervisado, en el cual los coches debían aprender por si mismos, sin ningún tipo de información o ayuda adicional. La manera en que esto ha sido posible es mediante las RNs, las cuales se encargan de gobernar el manejo de los coches, y el uso de un filtro de partículas basado en AGs, los cuales permiten que los mejores coches de cada generación o simulación fueran seleccionados para poder crear la siguiente generación, pudiendo mejorarse las RNs de los coches entrenados hasta llegar a completar los circuitos entrenados.

Para hacer posible este aprendizaje y mejorarlo, se ha optado por desarrollar un sistema que permite a los coches completar los circuitos en los que se les entrene, de forma que se acaben conseguiendo RNs preparadas para afrontar cualquier tipo de circuito.

Además, junto al sistema anteriormente mencionado, se han incorporado funcionalidades, las cuales permiten mejorar la visualización del entrenamiento y test, dando la posibilidad de visualizar gráficamente las RN de los coches entrenados y poder comparar las de cada coche y así observar que tendencia van adquieriendo en el aprendizaje.

5.1 Posibles mejoras y cambios

En las pruebas realizadas se ha podido ver que el entrenamiento más completo ha resultado ser el que abarca mayor cantidad de circuitos, puesto que los coches entrenados en los circuitos 1,2,3,4,5 y 6, los cuales son más sencillos, han dado mejores

resultados en los circuitos de test que los circuitos 7 y 8 por separado. Sin embargo, sería interesante poder realizar una mayor cantidad de pruebas teniendo en cuenta una serie de posibles mejoras:

- Añadir una mayor cantidad de senores, de forma que la lectura sea más precisa, lo cual resultaría en una RN más compleja, y que por tanto podría dar resultados mejores.
- Añadir más circuitos, tanto de entrenamiento como de test, lo cual haría más fiables y generalizables los resultados, y probablemente se obtendrían coches mejores.
- Junto al aspecto anterior, también realizar mayor cantidad de pruebas para cada entrenamiento y test, lo cual daría resultados más generalizables con los que extraer conclusiones más completas.

5.2 Competencias y conocimientos adquiridos

Ha sido interesante poder trabajar el concepto de RN en un videojuego, lo cual ha permitido poner en práctica conocimientos adquiridos en el Grado de Ingenieria Informática sobre Machine Learning (ML) e Inteligencia Artificial (IA), de manera que se han podido comprender mejor los mecanismos y operaciones que se realizan en el procesamiento de la información dentro de una RN, como así su importancia y relevancia en campos como el de los videojuegos.

También ha sido interesante el uso de Algoritmos Genéticos, AGs para poder realizar el aprendizaje de los coches en este TFG de una manera no supervisada e independiente, de forma que se ha comprendido la utilidad de estos algoritmos como así sus ventajas e incovenientes.

Junto a los aspectos anteriores, también hay que destacar el uso de los conocimientos sobre la programación, necesarios para poder interpretar un código ya realizado como era el caso de este TFG, así como también para poder desarrollar nuevas mejoras. El uso de Python como lenguaje de programación ha permitido adquirir conceptos sobre el propio lenguaje, como también conceptos sobre las diferentes librerías que se han utilizado, como OpenGL, Tkinter, Pickle y Numpy, las cuales han hecho posible todo lo siguiente:

- La librería Pickle ha permitido el almacenamiento y carga de las RNs de los coches, de forma que ha sido posible conservar cada entrenamiento realizado.
- La librería OpenGL ha permitido el desarrollo de nuevas funcionalidades para este TFG, concretamente la representación gráfica de la RN de cada coche.
- La librería Tkinter ha permitido desarrollar la interfaz gráfica de usuario necesaria para poder seleccionar el tipo de entrenamiento que se desee, así como también seleccionar un entrenamiento entre los que se tienen guardados para poder realizar los tests deseados.

- La librería Numpy ha permitido la creación de nuevas estructuras para poder gestionar la representación gráfica de las RNs.

BIBLIOGRAFÍA

[1] J. T. P. M. y Roque Marín Morales, *INTELIGENCIA ARTIFICIAL Métodos, técnicas y aplicaciones*. McGraw-Hill, 2008, ch. Prólogo. 1.1

[2] D. la Torre, "¿cómo nació la inteligencia artifical?" 2019. [Online]. Available: https://invertia.elperiodico.com/es/noticias/tecnologia/20190112/turing-deep-blue-como-nacio-inteligencia-artificial-232510 1.1

[3] S. R. Cid, "Historia y evolución de la inteligencia artificial," 2018. [Online]. Available: https://www.cice.es/noticia/historia-evolucion-la-inteligencia-artificial/ 1.1

[4] R. M. G. y. J. J. G. R. Julio Villena Román, "Inteligencia en redes de comunicaciones. tema 1 historia de la inteligencia artificial," 2012. [Online]. Available: http://ocw.uc3m.es/ingenieria-telematica/inteligencia-en-redes-de-comunicaciones/material-de-clase-1/01-historia-de-la-inteligencia-artificial 1.1

[5] L. H. Rotger, "Alan m. turing y claude e. shannon: matemáticas para la informática," 2013. [Online]. Available: https://blogs.elpais.com/turing/2013/03/alan-m-turing-y-claude-e-shannon-matematicas-para-la-informatica.html 1.2

[6] D. Cires, "Inteligencia artificial en los videojuegos," 2012. [Online]. Available: https://imascono.com/es/magazine/inteligencia-artificial-videojuegos-origen/ 1.2

[7] J. Alcalá, "Inteligencia artificial en videojuegos," 2012. [Online]. Available: http://www.flasentertainment.com/blog/ia.pdf 1.2

[8] ——, "¿cuál es la diferencia entre inteligencia artificial y machine learning?" 2017. [Online]. Available: https://www.analytics10.com/blog/cual-es-la-diferencia-entre-inteligencia-artificial-ai-y-machine-learning-ml/ 1.3

[9] C. Mansilla, "Así juega a super mario kart una inteligencia artificial," 2017. [Online]. Available: https://www.3djuegos.com/noticias-ver/175370/asi-juega-a-super-mario-kart-una-inteligencia-artificial/ 1.3

[10] M. Martínez, "Inteligencia artificial aprende a superar super mario world," 2015. [Online]. Available: https://computerhoy.com/noticias/software/inteligencia-artificial-aprende-superar-super-mario-world-29979 1.3

[11] SINC, "Un ordenador aprende a jugar a videojuegos por sí mismo gracias a una red neuronal artificial," 2015. [Online]. Available: https://www.agenciasinc.es/Noticias/

Un-ordenador-aprende-a-jugar-a-videojuegos-por-si-mismo-gracias-a-una-red-neuronal-artificial 1.3

[12] J. Pastor, "Google adquiere deepmind technologies, una empresa especializada en ia," 2015. [Online]. Available: https://www.xataka.com/robotica-e-ia/google-adquiere-deepmind-technologies-una-empresa-especializada-en-ia 1.3

[13] A. Triglia, "¿cuántas neuronas tiene el cerebro humano?" 2015. [Online]. Available: https://psicologiaymente.com/neurociencias/cuantas-neuronas-tiene-cerebro-humano 2.6

[14] F. S. Caparrini, "Redes neuronales: una visión superficial," 2018. [Online]. Available: http://www.cs.us.es/~fsancho/?e=72 2.6

[15] M. A. Nielsen, "Using neural nets to recognize handwritten digits," 2015. [Online]. Available: http://neuralnetworksanddeeplearning.com/chap1.html 2.6

[16] A. B. y Enrique Domínguez. Universidad de Málaga, "Aprendizaje supervisado en redes neuronales," 2015. [Online]. Available: http://www.redes-neuronales.com.es/tutorial-redes-neuronales/aprendizaje-supervisado-en-redes-neuronales.htm 2.7.1

[17] A. M. U. d. P. V. H. U. Pedro Larranaga, ~ Inaki ~ Inza, "Tema 8. redes neuronales," pp. 12–14, 1997. [Online]. Available: http://www.sc.ehu.es/ccwbayes/docencia/mmcc/docs/t8neuronales.pdf 2.7.1

[18] F. S. Caparrini, "Entrenamiento de redes neuronales: mejorando el gradiente descendiente," pp. 12–14, 1997. [Online]. Available: http://www.cs.us.es/~fsancho/?e=165 2.7.1

[19] J. C. López, "Introducción a los algoritmos genéticos: como implementar un algoritmo genético en java," 2010. [Online]. Available: https://www.adictosaltrabajo.com/2010/10/07/jgap/ 2.8.1

[20] F. A. J. V. Luis Llano, Andrés Hoyos, "Comparación del desempeño de funciones de activación en redes feedforward para aproximar funciones de datos con y sin ruido , gidia: Grupo de investigación y desarrollo en inteligencia artificial grupo de finanzas computacionales escuela de ingeniería de sistemas, facultad de minas," 2007. [Online]. Available: https://pdfs.semanticscholar.org/faa7/68d4f072337638a757041896956610705a78.pdf 3.6.8