Universitat de les Illes Balears

escola politècnica superior

Trabajo Fin de Grado

GRADO EN INGENIERÍA INFORMÁTICA

Utilización de técnicas de Machine learning en un videojuego de carreras

JAIME CRESPÍ VALERO

Tutores

José María Buades Rubio Gabriel Moyà Alcover

Escola Politècnica Superior
Universitat de les Illes Balears
Palma, 10 de julio de 2019

ÍNDICE GENERAL

ACRÓNIMOS

TFG Trabajo de Final de Grado

AM Aprendizaje máquina (en inglés *Machine learning*)

RN Red neuronal (en inglés *Neural network*)

FP Filtro de partículas (en inglés *Particle filter*)

IA Inteligencia artificial (en inglés *Artificial intelligence*)

AG Algoritmo genético (en inglés *Genetic algorithm*)

AP Apredizaje profundo (en inglés *Deep learning*)

PLN Procesamiento del lenguaje natural (en inglés *Natural language processing*)

VSC Valores Separados por Comas (en inglés *Comma-separated values*)

RESUMEN

En este Trabajo de Final de Grado (TFG) vamos a utilizar la técnica de las redes neuronales para dar distintos enfoques al entrenamiento de coches en un videojuego de carreras. El objetivo del mismo es conseguir mejorar los resultados obtenidos en el TFG antecesor [1], para ello vamos a utilizar nuevas maneras de entrenar los coches virtuales, para luego comprobar el rendimiento obtenido al recorrer un conjunto de circuitos completamente desconocidos.

Primero de todo haremos una análisis completo del proyecto. Especificaremos los requisitos de nuestro software y haremos una planificación de las diferentes fases. A continuación, estudiaremos los diferentes entornos de desarrollo y buscaremos el más adecuado para nuestra solución. Por último, explicaremos todos los conceptos teóricos necesarios para comprender el trabajo realizado.

Lo siguiente que haremos será describir la base que utilizaremos del trabajo previo [1], la cual está compuesta por su estructura interna, las pruebas realizadas y las conclusiones que se obtuvieron de los resultados. En cuanto a la estructura interna, especificaremos los elementos principales que utilizaremos, tales como los circuitos, los coches y la simulación, la cual engloba a estos dos últimos.

A continuación, vamos a hacer una extensa explicación de todo el trabajo realizado en el proyecto. Describiremos todas las mejoras realizadas sobre la base del trabajo anterior. Además, explicaremos los dos casos de estudio que hemos aplicado en el entrenamiento de los coches y el desarrollo para crear un sistema capaz de realizar multitud de pruebas y almacenar sus resultados. Por último, vamos a desarrollar un programa capaz de simular los casos de prueba, con capacidad de mostrarnos al mejor coche recorrer los circuitos de entrenamiento y de validación.

Para acabar, analizaremos con detenimiento cada caso de estudio. En cada uno, propondremos una serie de hipótesis y explicaremos la metodología utilizada para poder validarlas. Después, analizaremos los resultados y expondremos las conclusiones para cada caso de estudio y las del proyecto.

CAPÍTULO 1

INTRODUCCIÓN

Hoy en día, la Inteligencia artificial (en inglés *Artificial intelligence*) (IA) se ha ido extendiendo a lo largo del tiempo en una gran variedad de campos, algunos de los cuales son las finanzas, la cartografía, la medicina, la educación, la aviación, los videojuegos, etc. El concepto de IA consiste en dotar a las máquinas de la inteligencia humana [2], la cual imitará las funciones cognitivas de los humanos, tales como la capacidad de percibir, razonar, aprender y de resolver problemas [3].

En este TFG nos vamos a centrar en el desarrollo de una IA para el aprendizaje automático de la conducción de coches en un videojuego de carreras. Un videojuego se basa en un juego electrónico en el que una serie de personas interactúan, por medio de un controlador, con un dispositivo que muestra imágenes de vídeo [4]. Para lograr nuestras metas, vamos a utilizar el Aprendizaje máquina (en inglés *Machine learning*) (AM) el cual es una de las ramas más prometedoras de la IA. El AM tiene como objetivo el aprendizaje de las máquinas y hasta ahora ha demostrado tener un amplio rango de aplicaciones, como la detección de los diferentes tipos de cáncer, la detección del fraude de tarjetas de crédito o incluso la conducción automática por parte de coches inteligentes.

En este capítulo vamos a explorar el contexto histórico de la IA y haremos un pequeño hincapié en los logros en el mundo de los videojuegos de una de las técnicas más importantes del AM, la Red neuronal (en inglés *Neural network*) (RN). Además, explicaremos las técnicas utilizadas, los objetivos personales y del proyecto y, la estructura de la memoria.

Es importante recalcar que partimos del TFG de un alumno del grado de matemáticas [1], el cual desarrolló la base del proyecto que conforma todo el sistema de carreras incluyendo los circuitos y la creación de las RN en los coches.

1.1 Motivación

La IA ha demostrado a lo largo de los últimos años que es capaz de enfrentarse a todo tipo de retos, desde ganar partidas de juegos de mesa a los mejores jugadores del mundo, hasta casos más cotidianos como la conducción autónoma de un coche por la calle. Es innegable que la IA ya es un referente del futuro y el AM uno de sus campos más prometedores. A continuación, haremos un repaso por la historia de los logros más relevantes de los últimos tiempos.

En el año 1996, se celebró la primera partida de ajedrez entre el gran maestro Kasparov y una máquina desarrollada por el fabricante estadounidense IBM, conocido como DeepBlue. Sin embargo, la victoria fue para Kasparov con 3 partidas ganadas y 2 tablas de un total de 6 partidas. Este resultado no duró demasiado, ya que en el 1997 se propuso un nuevo enfrentamiento contra Kasparov, está vez con su nueva versión llamada Deeper Blue (azul más oscuro). La victoria fue para Deeper Blue ganando 3 partidas y dos en empate de un total de 6 partidas. La victoria de Deeper Blue supuso un punto de inflexión en el alcance al que podría llegar la IA [5].

Figura 1.1: Kasparov contra DeepBlue [6].

En el año 1999, la compañía Sony puso a la venta su *perro robot* llamado AIBO (es el acrónimo de Artificial Intelligence Robot, en español Robot de Inteligencia Artificial), el cual estaba dotado de una IA que reconocía una gran variedad de comandos voz y podía realizar un amplio conjunto de acciones tales como dar la pata, sentarse, hacerse el muerto, buscar una bola en el entorno, etc [7]. En la Figura 1.2 se puede ver cómo era AIBO.

En el 2009, la empresa Google desarrolló el primer prototipo de coche autónomo capaz de conducirse en las carreteras de las ciudades. El proyecto fue tan exitoso que se creo la compañía Waymo, la cual tenía planeado comercializar sus coches en el 2020 [9].

En el 2011 fue el nacimiento de Siri, una IA que se incluiría en el iPhone 4S y que tendría la capacidad para responder preguntas, hacer recomendaciones y muchas otras funcionalidades de asistente personal debido a la utilización del Procesamiento del lenguaje natural (en inglés *Natural language processing*) (PLN). El asistente personal obtuvo tanta fama que muchas otras empresas empezaron a crear los suyos propios,

Figura 1.2: AIBO, el perro robot[8].

como Google Now que fue incluido en el 2012 en su sistema operativo Android 4.1. En el 2011, Microsoft lanzaría Cortana, su propio asistente virtual creado para Windows 10 y Windows 10 Mobile [10].

El 26 de enero de 2014, Google anunció que había acordado adquirir Tecnologías DeepMind, la cual pasó a llamarse Google DeepMind [11]. El gigante de las tecnológicas había decidido introducirse en el mundo del AM y no necesitó mucho tiempo para demostrarnos sus avances. En octubre de 2015, DeepMind utilizó su programa AlphaGo para vencer al actual campeón europeo Fan Hui, ganando cinco a cero en el juego de mesa Go. Fue un hito histórico al ser la primera vez que una IA ganaba a un jugador profesional de Go [12]. El tablero de Go tiene un tamaño de 16 por 16 celdas, por lo que el número de posibilidades asciende notablemente dejando muy atrás el modelo del ajedrez, el cual utilizaba un tablero de 8 por 8 celdas y nos permitía utilizar programas de fuerza bruta para encontrar la solución más óptima. La solución a este problema consistía en utilizar una de las técnicas más interesantes del AM, las llamadas RN.

En el año 2019, DeepMind ha conseguido dos grandes logros en el tema de la IA en el mundo de los videojuegos. El primero de ellos, ocurrió a principios de año, cuando consiguió desarrollar una IA que era capaz de ganar a los humanos en el modo captura la bandera del videojuego competitivo Quake III [13]. El Quake III es un videojuego multijugador de disparos en primera persona; en él hay diversos modos de juego, entre ellos el de captura la bandera, en el cual hay dos equipos y cada uno debe intentar ir a la base del equipo contrario, quitarle su bandera y traerla a su base. Para conseguir esta hazaña se utilizaron complejos sistemas de RN que fueron capaces de aprender tácticas como esperar a que aparezca la bandera en campo enemigo, seguir a tus aliados para darles apoyo, crear emboscadas utilizando coberturas, etc.

El segundo logro fue pocos meses después con la victoria del programa AlphaStar contra dos de los mejores jugadores de StarCraft II. Los dos jugadores pertenecientes al equipo Liquid eran Grzegorz Komincz (LiquidMaNa) y Dario Wünsch (LiquidTLO). Hasta el momento, ninguna IA había sido capaz derrotar a los mejores jugadores mun-

diales debido a la complejidad del juego. La dificultad del juego radica en la toma de decisiones cuando se desconocen los movimientos del contrario debido a la niebla del juego [14]. En la Figura 1.3 se puede observar en enfrentamiento entre el programa AlphaStar y su contrincante humano Dario Wünsch, conocido como LiquidTLO.

Figura 1.3: AlphaStar contra el jugador humano Dario Wünsch (LiquidTLO) en StarCraft 2[14].

1.2 Técnicas utilizadas

Las técnicas utilizadas serán principalmente dos. La primera es la utilización de la técnica de RN como técnica de AM. La segunda técnica es la aplicación del Filtro de partículas (en inglés *Particle filter*) (FP) para la creación de un algoritmo genético que nos permita construir un sistema de aprendizaje con una fuerte base probabilística. Ambas técnicas pertenecen al campo de la IA.

El concepto de IA es llevar la inteligencia a cabo por máquinas. Una máquina «inteligente» ideal es un agente flexible que percibe su entorno y lleva a cabo acciones que maximicen sus posibilidades de éxito en algún objetivo o tarea [2]. Este campo es muy amplio y se puede dividir en diversas categorías que se observarán en la figura 1.4.

Una de las subcategorías de la IA es el AM. Este consiste en el conjunto de técnicas que proporciona a los ordenadores la habilidad de aprender sin ser explícitamente programados [15]. El AM explora el estudio y construcción de algoritmos, los cuales pueden aprender y hacer predicciones de los datos [16]. Dentro del AM utilizaremos las RN, las cuales tendrán un proceso de aprendizaje, donde primeramente se utilizarán un subconjunto de los datos para entrenar nuestra RN y el resto de los datos para hacer una validación y comprobar la eficiencia de la misma.

1. **Entrenamiento**. Consiste en hacer que la RN aprenda en base a modelos preestablecidos. Esta recibirá unos datos de entrada que le proporcionaremos y, utili-

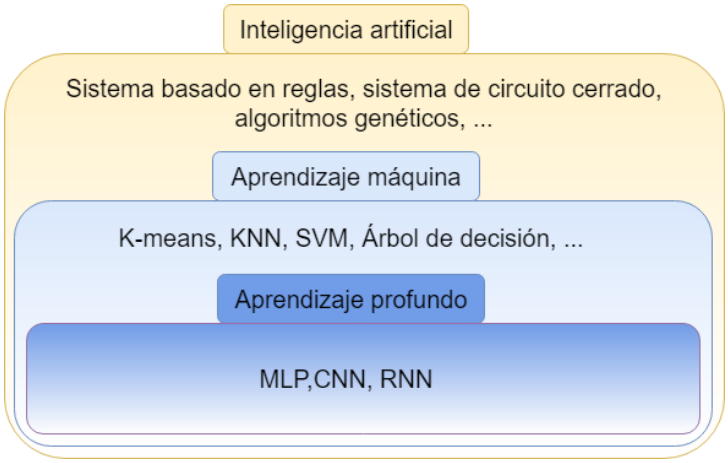

Figura 1.4: Campos de la IA.

zaremos los datos de salida para hacer que aprenda.

2. **Validación**. Una vez finalizado el proceso de entrenamiento procederemos a la validación. Utilizaremos el subconjunto de datos de validación, el cual no ha sido utilizado anteriormente y una vez calculado el resultado podremos medir la eficiencia del algoritmo.

El Algoritmo genético (en inglés *Genetic algorithm*) (AG) es una subcategoría de la IA. Este se basa en el proceso natural de selección de Darwin. El objetivo de esta técnica es seleccionar los mejores agentes para la reproducción y así obtener que la siguiente generación tenga mutaciones que le permitan ser mejor [17]. Se muestran las diferentes etapas en la Figura 1.5.

Las etapas del AG son las siguientes:

1. **Inicializar población**. Establecemos la población inicial de agentes con la asignación de sus parámetros iniciales de forma aleatoria.

2. **Cálculo del peso**. Calculamos la puntuación de cada agente según su grado de éxito realizando la tarea.

3. **Selección**. Utilizamos un criterio de selección para escoger los agentes que utilizaremos en la siguiente generación.

4. **Mutación**. Realizamos modificaciones (mutaciones) a los agentes que hemos seleccionado previamente. Se busca que gracias a estas modificaciones algún agente de la siguiente generación pueda completar con un mayor éxito la tarea.

1.3 Objetivos

El objetivo principal del proyecto es entrenar de diferentes maneras un conjunto de coches en una serie de circuitos, dónde utilizaremos las técnicas de RN y el FP para

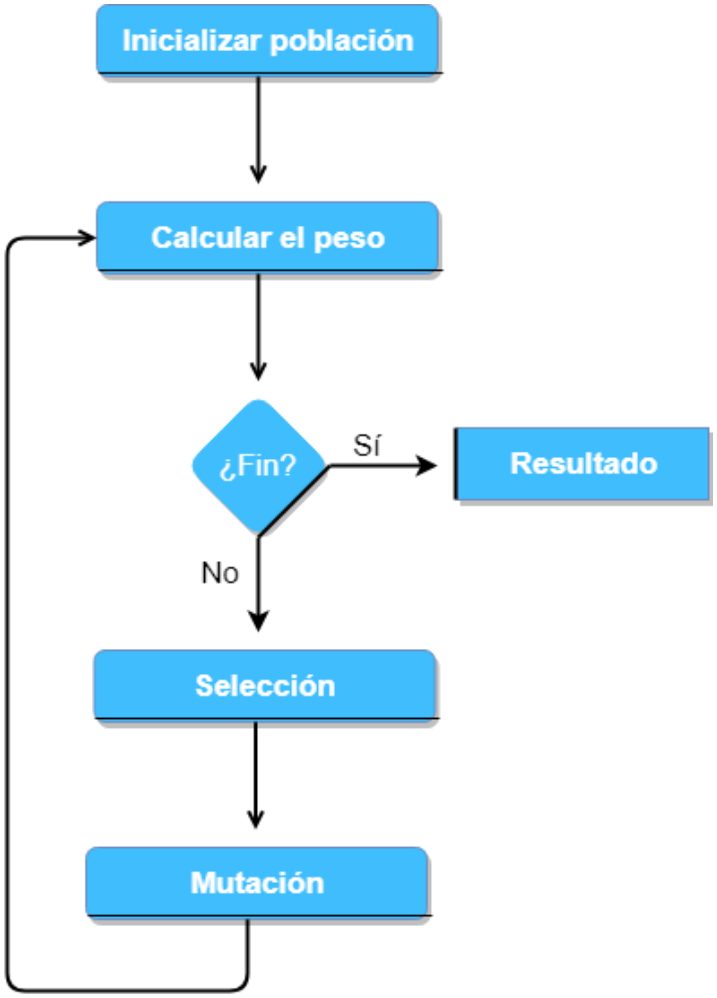

Figura 1.5: Etapas del AG.

conseguir el coche más efectivo y, así finalmente, poder comprobar su capacidad para completar circuitos desconocidos.

1.3.1 Objetivos específicos

Los objetivos son los siguientes:

1. Aplicar dos nuevos casos de estudio en el entrenamiento de la RN de los coches, cada uno utilizando su propio sistema de recorrido en el conjunto de circuitos de entrenamiento.

2. Construir un sistema automatizado capaz de realizar todo el conjunto de pruebas de ambos casos de estudio y almacenar en un fichero cada una de las RN del mejor coche y los resultados obtenidos.

3. Montar un sistema que sea capaz de replicar un caso de prueba a partir de la selección de circuitos del conjunto de entrenamiento. Una vez seleccionado los

circuitos del conjunto de entrenamiento se tiene que visualizar al mejor coche de ese caso de prueba recorriendo cada uno de los circuitos de entrenamiento y de validación. Los resultados obtenidos por esta simulación deben ser los mismos que se han obtenido en las métricas de rendimiento.

1.3.2 Objetivos personales

Los objetivos que más me han influido a la hora de elegir y realizar este TFG son los siguientes:

1. La posibilidad de profundizar en el concepto de RN y sus distintas aplicaciones.
2. Llevar a la práctica un caso real de aprendizaje automático en un videojuego.
3. Utilizar el TFG como un punto destacable en el currículum vítae.

1.4 Estructura de la memoria

La memoria está dividida en cinco capítulos:

En el capítulo 1 haremos una breve introducción del TFG, también dedicaremos un espacio a explicar el contexto de la IA que nos sirve de motivación. A continuación, explicaremos las técnicas que aplicaremos en un plano más general y los objetivos del proyecto. Por último, detallaremos la estructura de la memoria.

En el capítulo 2 vamos a desarrollar los requisitos y la planificación que seguiremos. Explicaremos detalladamente el entorno de programación, el fondo teórico y la explicación del problema.

En el capítulo 3 haremos hincapié en el trabajo previo realizado por el anterior TFG [1], el cual utilizaremos de base en nuestro proyecto.

En el capítulo 4 explicaremos todo el trabajo realizado en este TFG. El cual incluye la ampliación de funcionalidades, la creación del sistema automatizado de pruebas para los dos casos de estudio y el simulador de casos de prueba.

En el capítulo 5 explicaremos tanto las pruebas hechas como los resultados de las mismas. Discutiremos las métricas obtenidas y buscaremos la manera de dar unas conclusiones objetivas en base a los resultados.

En el capítulo 6 haremos una pequeña conclusión sobre lo que hemos conseguido en este proyecto.

CAPÍTULO 2

CONSIDERACIONES PREVIAS

En este capítulo vamos a definir los conocimientos necesarios que se tienen que comprender de nuestro proyecto. Primero de todo, vamos a hacer una definición de los requisitos y una explicación de la planificación de las diferentes etapas. A continuación, haremos una comparativa entre una serie de lenguajes de programación para evaluar cual es el que más se adecua a nuestra solución. Por último, explicaremos todos los conceptos teóricos necesarios para la comprensión del proyecto.

2.1 Requisitos

En este proyecto tenemos un número reducido de requisitos de software. Únicamente disponemos de un *stakeholder*, que es la persona que utiliza el software que hemos desarrollado.

La nomenclatura será la siguiente:

1. Utilizaremos **RFXX** para los requisitos funcionales, donde la **XX** significa el número de requisito.

2. Utilizaremos **RNFXX** para los requisitos no funcionales, donde la **XX** significa el número de requisito.

2.1.1 Requisitos funcionales

1. **RF01**: El programa debe ser capaz de automatizar la generación de casos de prueba de cada caso de estudio y el almacenamiento de sus resultados. Todos los casos de prueba tienen unas características comunes y se distinguen unos de otros por el conjunto de circuitos de entrenamiento.

2. **RF02**: El programa tiene que poder simular un determinado caso de prueba. La simulación consiste en escoger los circuitos de entrenamiento del caso de prueba

y visualizar el recorrido del mejor coche en los circuitos escogidos y los 3 de validación. Por último, se mostrarán los resultados obtenidos de la simulación.

2.1.2 Requisitos no funcionales

1. **RNF01**: El tiempo de ejecución es un factor secundario.

2. **RNF02**: Compatibilidad con los sistemas operativos Windows, Mac y Linux.

3. **RNF03**: Incluye un fichero que tiene la función de guía para la correcta instalación y puesta en funcionamiento del software.

2.2 Planificación

La solución a nuestro proyecto la dividiremos en tres etapas.

1. **Ampliación de funcionalidades**. Añadiremos y mejoraremos funcionalidades ya existentes para conseguir un programa más completo y sencillo de utilizar. Las mejoras se componen en diversas fases:

   a) Haremos cambios en la parte gráfica para mejorar su visualización y mejoras técnicas para mejorar la ejecución del programa.

   b) Procederemos a mejorar ciertas funcionalidades que no estaban correctamente implementadas.

   c) Añadiremos nuestro propio filtro de partículas con una serie de mejoras.

2. **Construcción de un sistema de entrenamiento de RN**. Vamos a crear un sistema automatizado que cumpliendo una serie de requisitos sea capaz de ejecutar y obtener los resultados de un conjunto de pruebas. En este caso tendremos dos casos de estudios, los cuales tendrán unas requisitos comunes y otros específicos de cada uno. Más adelante, en el capítulo 4, explicaremos más detalladamente cada requisito y la justificación de sus características.

   Los requisitos comunes que debemos cumplir son los siguientes:

   a) La función de ponderación que utilizaremos para obtener la métrica del resultado en un circuito será la de la distancia recorrida al cuadrado.

   b) Habrá un total de 100 coches y sus RN se generarán aleatoriamente.

   Los requisitos específicos del **primer caso de estudio** son los siguientes:

   a) En cada generación el conjunto de coches recorrerá el mismo circuito de entrenamiento hasta conseguir completarlo, es decir, no estableceremos un límite en el número de generaciones.

   Los requisitos específicos del **segundo caso de estudio** son los siguientes:

a) Estableceremos el número máximo de generaciones en 30. En caso de que algún coche acabe todos los circuitos del conjunto de entrenamiento antes de llegar al número máximo de generaciones, se procederá a pasar directamente al conjunto de validación.

b) En cada generación, el conjunto de coches recorrerá cada uno de los circuitos del conjunto de entrenamiento antes de pasar a la siguiente generación.

Cada una de las pruebas se divide en dos partes:

a) **Entrenamiento**. La primera parte consiste en el entrenamiento de las RN de un conjunto de coches en un conjunto de circuitos determinados.

b) **Validación y obtención de resultado**. La segunda parte consiste en seleccionar el coche con un mejor resultado en el entrenamiento y, utilizarlo para comprobar su eficacia resolviendo 3 circuitos completamente desconocidos. Una vez ha finalizado, se procederá a almacenar en un fichero el resultado del mejor coche en cada circuito (incluyendo a los 3 de validación) y además, almacenaremos la RN para poder simular su recorrido.

3. **Simulador de casos de prueba**. Consiste en el desarrollo de un programa que es capaz de simular el comportamiento del mejor coche de un determinado caso de prueba. Para ello, vamos a crear una ventana inicial en la cual se podrá seleccionar el conjunto de circuitos de entrenamiento. Una vez se han seleccionado, aparecerá una nueva ventana en la cual un coche recorrerá todos los circuitos previamente seleccionados y los 3 circuitos de validación. El coche que recorrerá dichos circuitos utilizará la mejor RN entrenada previamente.

En la Figura 2.1 se muestra la planificación del proyecto mediante un diagrama de Gantt.

2.3 Entorno de programación

En esta sección explicaremos las herramientas utilizadas para obtener una solución informática.

2.3.1 Lenguaje de programación

La elección de un lenguaje de programación es muy importante para desarrollar nuestra solución. En nuestro proyecto se tendrán que desarrollar dos aspectos muy bien diferenciados. El primer aspecto es la implementación gráfica, que tendrá que ser capaz de ocuparse de la gestión de las ventanas y del dibujado del escenario. El segundo aspecto consiste en librerías que permitan implementar adecuadamente las técnicas de AM necesarias para nuestro proyecto.

Los lenguajes que nos ofrecen una solución a nuestro problema de elección son tres: `Python`, R, C++ y Java. Cada lenguaje tendrá sus propias características que compararemos en la Tabla 2.1.

El lenguaje `Python` es el más efectivo para solucionar los aspectos más relevantes del proyecto. Otras características que nos serán de utilidad son:

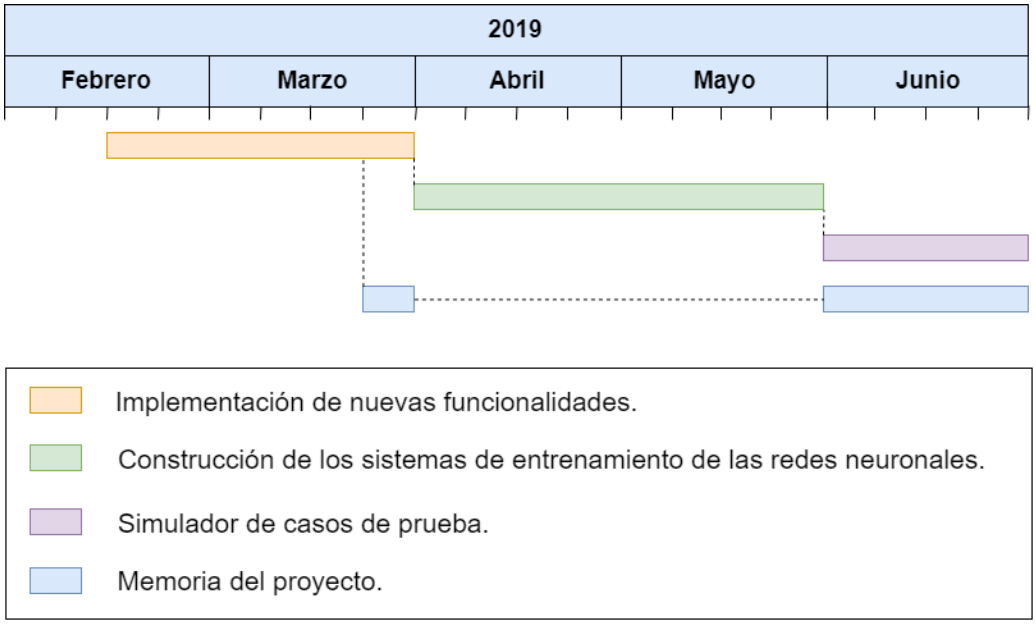

Figura 2.1: Planificación del proyecto.

	Python	**R**	**Java**	**C++**
Cantidad de código	Poca	Poca	Mucha	Mucha
Tiempo de ejecución	Lento	Lento	Rápido	Rápido
Rapidez de desarrollo	Rápido	Rápido	Lento	Lento
Librerías gráficas	Sí	No	Sí	Sí
Librerías de IA	Sí	Sí	Sí	Sí
Biblioteca común de ML	Sí	Sí	No	No

Tabla 2.1: Comparativa entre Python, R, Java y C++.

1. Es multiplataforma. Nos permite ejecutarlo en los tres sistemas operativos, Mac, Linux y Windows.

2. Gran variedad de librerías para solucionar todo tipo de problemas.

La desventaja más notable de `Python` es el tiempo de ejecución, el cual hemos considerado que es un factor secundario en comparación con las ventajas que nos ofrece las características del lenguaje.

2.3.2 Librerías

Las librerías o bibliotecas consisten en un conjunto de funcionalidades que podemos añadir a nuestros programas. Las librerías nos permiten abstraernos en el código y nos evitan la propia implementación de cálculos complejos o utilizar rápidamente funcionalidades muy complejas de programar.

OpenGL y GLUT

La librería `Open Graphics Library` (`OpenGL`) nos ofrece una API multilenguaje y multiplataforma para poder producir gráficos de 2 y 3 dimensiones [18]. Por otro lado, la librería GLUT se ocupa de toda la gestión de ventanas, manejo de eventos y inicialización de contexto de OpenGL [19]. Utilizaremos las API de ambas librerías principalmente para dos funcionalidades. La primera es para gestionar la ventana, las funciones de pintado y de actualización. Las segunda es para generar el pintado de los gráficos, lo cual incluye el pintado de los circuitos, los coches, los sensores de los coches, la interfaz que muestra la información de la carrera.

NumPy

Utilizaremos la librería `Numeric Python` (`NumPy`) debido a la potencia y rendimiento que ofrece en el cálculo de matrices y de números aleatorios [20]. Esto es debido a que la librería ha sido escrita mayormente en el lenguaje de programación `C`.

El cálculo de matrices es usado en las RN ya que debemos almacenar los pesos de las conexiones entre neuronas y los biases de cada neurona en matrices, sobre las cuales necesitaremos realizar operaciones, como aplicarle ruido para conseguir mutaciones en dichas neuronas. Por otro lado, utilizaremos el cálculo de números aleatorios para las mutaciones de neuronas en las RN y para el algoritmo del FP.

Pickle

La funcionalidad principal de `Pickle` es su capacidad de serializar y deserializar la estructura de un objeto `Python` en un flujo de datos [21]. La utilizaremos para almacenar la RN del mejor coche de cada caso de prueba en un fichero, de manera que podamos reproducir su recorrido en su conjunto de circuitos de entrenamiento y en el de prueba.

2.3.3 Entorno de desarrollo integrado (IDE)

El entorno de desarrollo integrado que vamos a utilizar es el `PyCharm`, el cual está diseñado expresamente para programar en `Python` en multiples plataformas. La decisión de utilizarlo radica en la gran cantidad de herramientas que nos ofrece, como un muy completo depurador (en inglés, debugger), un sistema de atajos muy elaborado para desplazarnos y modificar el código de manera muy rápida y, un sistema de configuraciones que permite crear un entorno virtual y gestionar las librerías de `Python`. Otras funcionalidades que ofrece y no incluimos en nuestro proyecto son la posibilidad de tener soporte web con `Django` y de Data Science con `Anaconda` [22].

2.4 Fondo teórico

Primero de todo explicaremos en profundidad lo necesario para entender cómo funciona un modelo de RN y, para acabar explicaremos el funcionamiento del FP que aplicaremos para seleccionar los coches de la siguiente generación.

2.4.1 Funcionamiento de una red neuronal

Una RN consiste en un conjunto de neuronas artificiales que están conectadas entre sí y son capaces de transmitirse información. La idea principal se basa en el comportamiento de las neuronas biológicas y su funcionamiento interno de transmisión de información. Las neuronas biológicas conforman un conjunto de conexiones sinápticas con otras neuronas, lo cuál produce que tengamos una red neuronal biológica. Una conexión sináptica es la capacidad de una neurona de transmitir un impulso nervioso a otra neurona. Para conseguirlo, la primera neurona segregará sustancias químicas que se encargarán de excitar o inhibir a la otra neurona [23]. En la Figura 2.2 se puede apreciar un ejemplo de conexión sináptica.

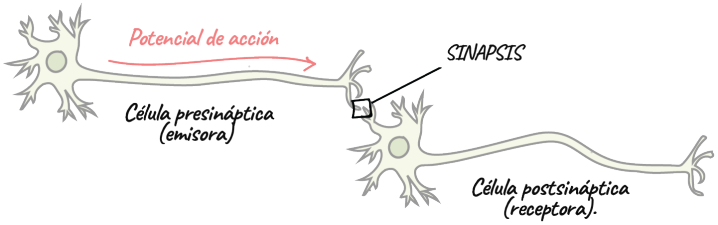

Figura 2.2: Conexión sináptica [24].

Las neuronas artificiales tienen un comportamiento parecido al de las neuronas biológicas. La idea que es que las neuronas artificiales estén conectadas entre sí formando diferentes capas, de manera que para pasar la señal de una neurona a otra se tenga que superar un umbral, muy parecido a la conexión sináptica mencionada anteriormente [23].

Una RN habitualmente está dividida en tres áreas. Tenemos una primera área que llamaremos la capa de entrada que contendrá los valores de entrada. A continuación, tendremos una segunda área que se compone de una o más capas ocultas y por último, tenemos una última área que está conformada por una capa la cual devuelve la salida de la RN. En la Figura 2.3 podremos apreciar un ejemplo de red neuronal.

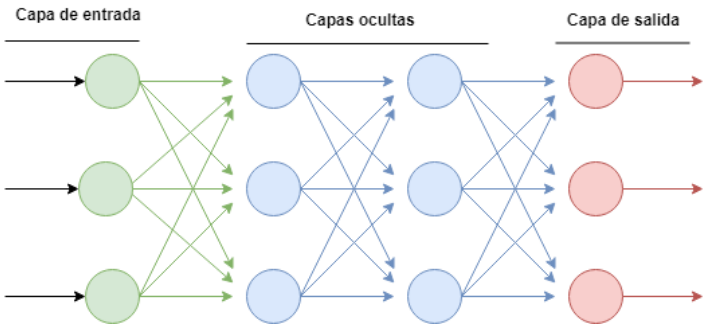

Figura 2.3: Ejemplo de red neuronal.

La figura 2.3 muestra una red neuronal, con tres neuronas como capa de entrada seguido de dos capas de 3 neuronas cada una en la capa oculta y, por último, 3 neuronas en la capa de salida. El número de neuronas de cada capa puede variar, al igual que la manera en que se conectan con la siguiente capa.

En las conexiones entre neuronas existirán unos valores denominados **pesos** que determinan la importancia de la conexión entre estas. Cada neurona de la capa oculta y de salida tendrá un umbral, el cual es un valor que determinará si tiene que transmitir la señal a la siguiente neurona. Se puede apreciar el gran parecido con las neuronas biológicas [23].

El caso más simple de RN sería el `Perceptrón`, el cual si el valor de entrada de la neurona, multiplicado por el peso de la conexión entre ambas neuronas supera el umbral, entonces la neurona tendrá como salida un 1, significando que debe transmitir la señal a la siguiente neurona. En el caso de que no supere el umbral devolverá 0 y por tanto no transmitirá la señal. Este sistema es muy sensible ya que un pequeño cambio en el umbral o los pesos puede provocar que la señal se transmita hacia otra neurona y cambie completamente el comportamiento de nuestra RN [25].
Para ello, utilizaremos en este caso **neuronas sigmoides**, las cuales permiten que los pequeños cambios afecten mínimamente la salida de las neuronas. Utilizaremos una **función sigmoidal**, la cual nos permite transmitir la señal a la siguiente neurona de una manera más natural y no tan abrupta como en el caso del `Perceptrón` [25]. Calculamos el valor de salida de la neurona i mediante la fórmula sigmoide:

$$y_i = \frac{1}{1+e^{-z}}.$$

En la Figura 2.4 vemos la representación gráfica de la función sigmoide.

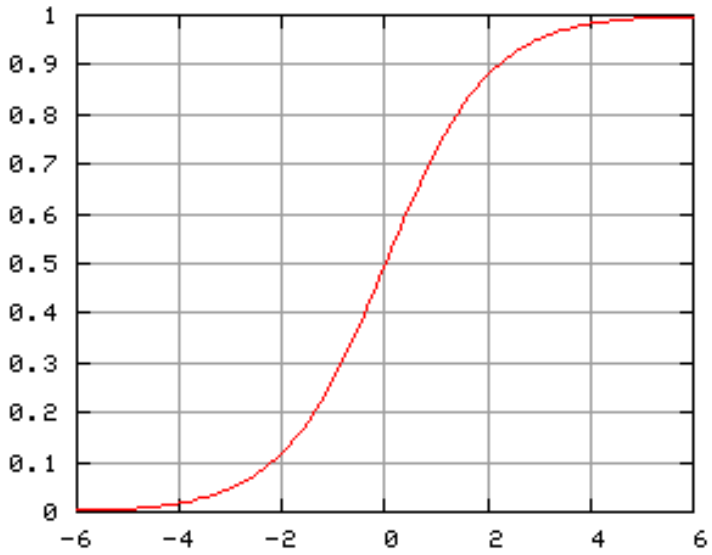

Figura 2.4: Representación gráfica de la función sigmoide [26].

Calculamos el valor de z mediante la siguiente fórmula que tenemos a continuación.

$$z = \sum_{j=1}^{n} x_j * w_{ji} - b_i.$$

Las neuronas j cumplen dos requisitos. El primero es que pertenecen a la capa anterior a la de la neurona i, la cual es la que queremos calcular su valor de salida o en

este caso conocido como *z*. El segundo es que esta neurona *j* tendrá una conexión con nuestra neurona *i*. La nomenclatura a la fórmula anterior es la siguiente:

1. $\mathbf{x_j}$: Valor de salida de la neurona j.
2. $\mathbf{w_{ji}}$: Peso de la conexión entre la neurona *j* y la *i*.
3. $\mathbf{b_i}$: El umbral de la neurona *i*.

Básicamente vemos que el valor de *z* es el sumatorio del valor de salida de la neurona de la capa anterior, multiplicado por el peso de la conexión entre nuestras dos neuronas y restando el umbral de nuestra neurona actual. A continuación, mostraremos la representación visual de la neurona en la Figura 2.5.

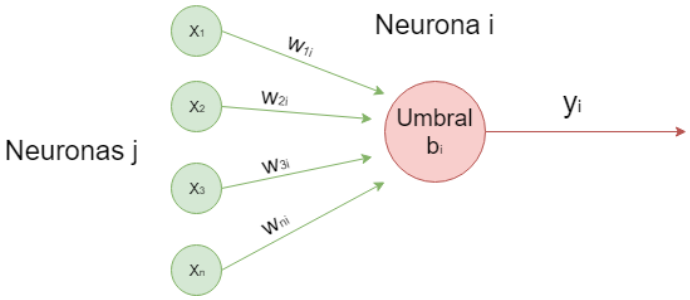

Figura 2.5: Representación de una neurona.

2.4.2 Filtro de partículas

Un FP modela el comportamiento de un sistema que varia a lo largo del tiempo [27]. En nuestro caso lo utilizamos para realizar una búsqueda por muestreo aleatorio de la solución. El FP está basado en el algoritmo de Montecarlo, el cual trata de resolver cálculos costosos mediante el uso de variables aleatorias [28].

El FP utiliza un conjunto de observaciones los cuales denominaremos partículas, que en un primer momento tendrán asociada una probabilidad uniforme. Dicha probabilidad se actualizará en base al modelo matemático que utilicemos. Usaremos el FP para crear de forma iterativa un nuevo conjunto de partículas que mejore la solución actual [27].

El algoritmo del FP tiene los siguientes pasos [27, 30]:

1. **Inicialización**. El primer paso consiste en inicializar las partículas aplicándoles una distribución uniforme para que tengan el mismo peso.
2. **Predicción**. Las partículas se redistribuyen al azar teniendo en cuenta la curva de probabilidad generada a partir de la observación de su rendimiento y se genera un nuevo conjunto de partículas añadiéndole ruido blanco.
3. **Medición**. Modificamos la curva de probabilidades de las partículas de acuerdo al modelo matemático que utilizamos. En la Figura 2.6 vemos la nueva distribución de las probabilidades de dicha curva.

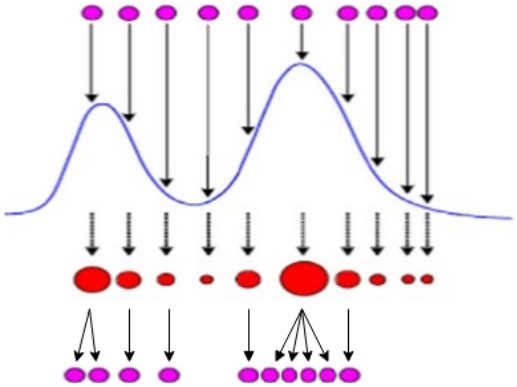

Figura 2.6: Filtro de partículas [29].

Los pasos de **medición** y **predicción** se realizan el número de veces necesario hasta obtener la solución con el margen de error deseado.

2.5 Conclusión

En este capítulo hemos definido todos los conocimientos necesarios para comprender nuestro proyecto. Hemos explicado detalladamente los requisitos y la planificación de las diferentes etapas. A continuación, hemos explicado porqué se ha elegido `Python` como el lenguaje de programación más adecuado para desarrollar una solución al problema y, por último, se ha explicado todos los conceptos teóricos como el funcionamiento de la RN y el FP.

CAPÍTULO 3

TRABAJO PREVIO

Para poder comprender mejor el trabajo que hemos desarrollado, tenemos que explicar la estructura, las pruebas y los resultados del TFG antecesor [1]. Los componentes principales que utilizaremos de la estructura son el **circuito**, el **coche** y la **simulación**. Por el otro lado, buscaremos los nuevos enfoques de entrenamiento para los coches a partir de las conclusiones que fueron obtenidas de los resultados de las pruebas del anterior trabajo.

3.1 Circuito

Un circuito consiste en un camino cerrado dónde los coches lo recorrerán en sentido horario o antihorario. La construcción del mismo se basa principalmente en un conjunto de cuadriláteros. Cada cuadrilátero consiste en cuatro puntos. Un circuito está compuesto de un conjunto de cuadriláteros, los cuales están unidos secuencialmente. La Figura 3.2 mostraremos visualmente la unión entre dos cuadriláteros.

Los circuitos pueden contener obstáculos, los cuales consisten en polígonos de 3 o más caras que estarán dentro de los cuadriláteros. En la Figura 3.1 podemos observar un circuito con obstáculos.

Para construir los cuadriláteros se utiliza un archivo de Valores Separados por Comas (en inglés *Comma-separated values*) (VSC), el cual almacena tres listas. La primera lista consiste en los puntos *x*, la segunda lista consiste en los puntos *y*, y, por último, una lista que contiene la anchura. Para construir el cuadrilátero se cogerá un par de puntos, se utilizará el ancho para obtener los cuatro puntos del cuadrilátero y unirlos.

Tenemos un total de 15 circuitos, 12 de los cuales son para entrenamiento y 3 para validación. Cada uno de los circuitos tendrá su propia dirección, la cual puede ser en sentido horario o antihorario. En la Figura 3.3 podemos observar la totalidad de los circuitos.

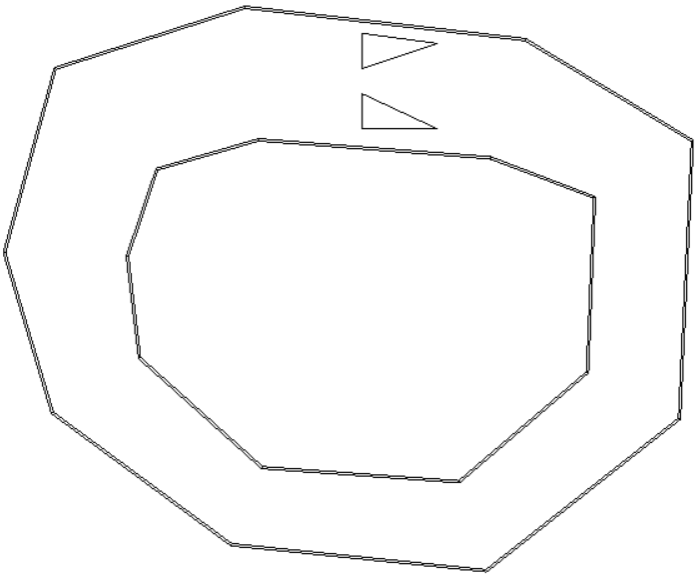

Figura 3.1: Representación de un circuito con obstáculos.

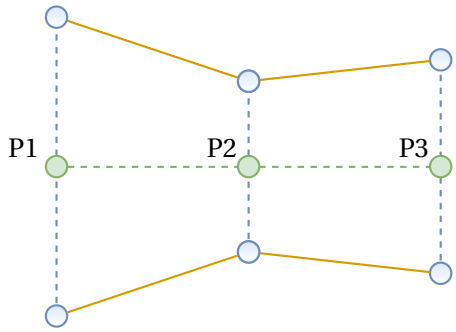

Figura 3.2: Representación la unión entre cuadriláteros del circuito.

3.2 Coche

El coche es el elemento que utilizaremos para recorrer los circuitos. Para realizar dicha tarea tendrá la capacidad de girar hacia ambos lados y de modificar su velocidad.

Un coche está formado por un cuadrilátero, el cual se forma a partir de cuatro puntos. Utilizaremos los dos puntos traseros del cuadrilátero, que denominaremos P1 y P2, para calcular el punto central. Utilizaremos dicho punto para los siguientes casos :

1. Es el lugar desde donde salen los sensores de colisión.
2. Nos informará de la dirección en la que se mueve el coche.
3. Es el punto de referencia para saber cuando la distancia que recorremos en los cuadriláteros del circuito.

La fórmula para calcular el punto central de la parte trasera del coche es la siguiente:

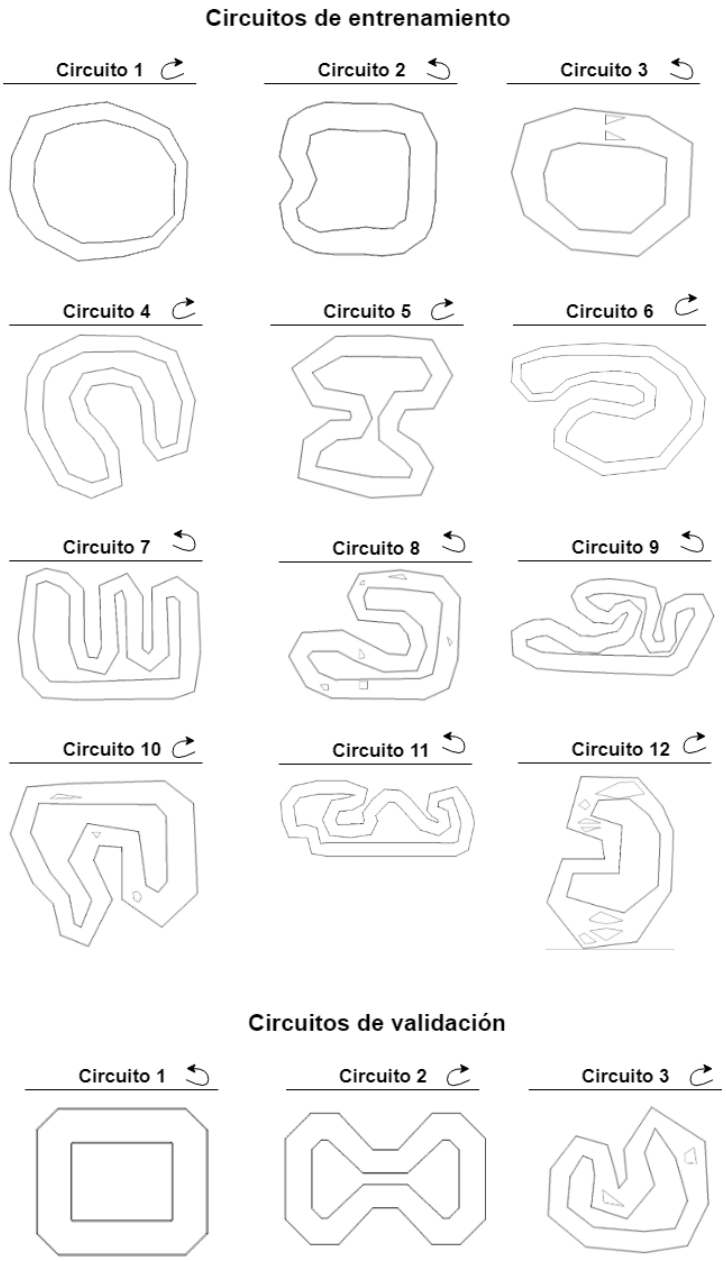

Figura 3.3: Representación la unión entre cuadriláteros del circuito.

$$P_{central} = \frac{P1 + P2}{2}.$$

Cada coche tendrá un total de 11 sensores de colisiones que calculan la distancia de colisión con el entorno, es decir, las paredes del circuito y los obstáculos. Los ángulos de los sensores son 10°, 26°, 42°, 58°, 74°, 90°, 106°, 122°, 138°, 154° y 170°. Estos sensores están colocados en el punto central del coche. La Figura 3.4 mostrará visualmente un coche, donde el sensor central lo mostramos en azul.

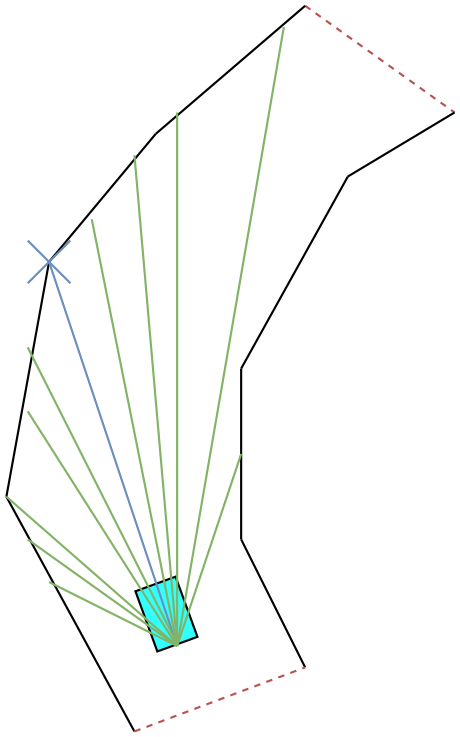

Figura 3.4: Representación de un coche donde el sensor central es mostrado de color azul.

3.2.1 Arquitectura de la red neuronal de los coches

La RN de cada coche está conformada por 3 capas. Una capa de entrada con 21 neuronas, una única capa oculta con 25 neuronas y en la última capa tendremos 2 neuronas. Cada neurona de una capa está conectada con la siguiente capa, es decir, la capa de entrada conecta todas sus neuronas con las de la capa oculta y esta, conectaría todas sus neuronas con las de salida.

La capa de entrada tendrá un total de 21 neuronas, de las cuales las primeras 11 neuronas están destinadas a los sensores de colisión con el escenario, es decir, calcula las distancias entre el coche y las paredes del circuito o los obstáculos. Cada una de estas 11 neuronas equivale a un sensor con un determinado ángulo. Los ángulos que vamos a utilizar son 10°, 26°, 42°, 58°, 74°, 90°, 106°, 122°, 138°, 154° y 170°.

El segundo conjunto de neuronas de la primera capa son un total de 10 neuronas, las cuales equivalen a los valores del sensor de central que se han ido almacenando. El sensor central consiste en el sensor de colisión del coche que tiene un ángulo de 90°. Un concepto que tenemos que definir es la unidad de tiempo, la cual es la definición que utilizaremos para la propia medición del tiempo. Los valores de entrada de estas 10 neuronas dependerán directamente de las unidades de tiempo:

1. **Las unidades de tiempo transcurridas es inferior a 10 unidades**: El valor de entrada de cada neurona será la distancia calculada por el sensor central en cada una de las unidades de tiempo. La primera neurona obtendrá la distancia de la unidad de tiempo más reciente y así, mientras haya unidades de tiempo. En

el momento que se acaben las unidades de tiempo, se asignará a las neuronas restantes el mismo valor del sensor central de la última neurona que tenía unidad de tiempo.

2. **Las unidades de tiempo transcurridas se encuentran entre 10 y 100 unidades**: Los valores de entrada de las neuronas serán las distancias del sensor central de las últimas 10 unidades de tiempo.

3. **Las unidades de tiempo transcurridas son superior a 100 unidades**: Los valores de entrada serán las distancias del sensor central obtenidas de 10 en 10 unidades de tiempo hacía atrás.

Hay que recalcar que las unidades de tiempo se incrementan rápidamente y por lo tanto, llegaremos en poco tiempo al tercer caso.

La importancia de las neuronas del sensor de colisión radica en la decisión de giro de nuestro coche. Dependiendo de las distancias con el entorno, el coche decidirá girar hacía el lado izquierdo o el derecho. Por otro lado, la importancia de las neuronas con los últimos valores del sensor central se ocuparán de ser un factor decisivo para la elección de la velocidad del coche.

En la capa oculta tendremos un total de 25 neuronas, las cuales tienen la función de realizar los cálculos que se transmitirán hacía la salida. El número de neuronas que hemos escogido nos permite obtener buenos resultados sin tener un coste computacional muy elevado. Por supuesto, se podría modificar el número de neuronas y acabaría provocando resultados diferentes.

En la capa de salida tenemos un total de 2 neuronas. La primera nos devolverá la velocidad angular, es decir, decidirá el giro de nuestro coche. La segunda neurona nos devolverá la velocidad lineal, es decir, su velocidad. Hay que tener en cuenta que nuestra neurona nos devolverá un valor que está comprendido en un rango entre 0 y 1 y tiene que adaptarse al rango de la velocidad. El rango de velocidades comprende un valor mínima de 3 unidades y uno máximo de 6 unidades. A continuación, mostraremos la red neuronal de un coche en la Figura 3.5.

3.3 Simulación

Una simulación consiste básicamente en un conjunto de coches que compiten en un determinado circuito, lo cual es conocido como una carrera. Cada simulación tendrá las siguientes características:

1. Dispone de información sobre la cantidad total de coches que hay en la simulación y la cantidad de coches que siguen compitiendo.

2. Gestiona el control del tiempo transcurrido en la competición.

3. Especifica el número de vueltas máximo de la simulación.

Cada simulación tiene asignada una **función de ponderación**, la cual se utiliza como métrica para saber el rendimiento de cada coche en ese circuito. Las funciones de ponderación son las siguientes:

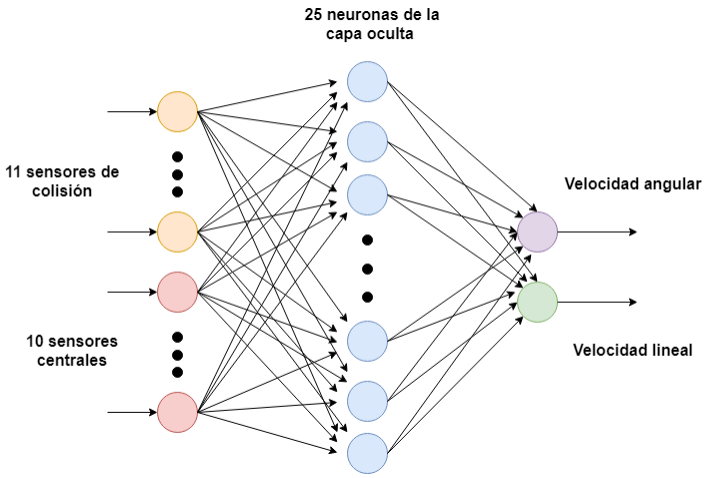

Figura 3.5: Red neuronal de un coche.

1. **Distancia recorrida**: Distancia total recorrida por un coche en un circuito.
2. **Distancia recorrida al cuadrado**: Distancia total al cuadrado recorrida por un coche en un circuito.
3. **Velocidad**: Velocidad media de los coches.
4. **Distancia recorrida por la velocidad**: La distancia recorrida multiplicado por la velocidad.

Se considera que una simulación ha finalizado cuando un coche ha acabado el número de vueltas que se ha establecido o todos los coches han colisionado con el entorno.

3.4 Pruebas y resultados

Las pruebas se realizaron en dos sistemas de entrenamiento para las RN de los coches. El primer sistema consiste en entrenar los coches en los 12 circuitos de entrenamiento de manera ascendente, es decir, del circuito fácil al difícil. Por el otro lado, el segundo sistema consiste en recorrer esos 12 circuitos de manera descendente, es decir, del circuito difícil al fácil. En ambos sistemas se utilizaron 30 coches que recorrían cada circuito dos veces, es decir, se generaban dos generaciones antes de pasar al siguiente circuito. Por último, se realizará una segunda vuelta en ambos sistemas, donde se volverán a recorrer todos los circuitos.

El conjunto de pruebas consiste en aplicar cada una de las 4 funciones de ponderación previamente explicadas, en los dos sistemas de entrenamiento. Una vez se han hecho todos los entrenamiento, se utilizará el mejor coche de cada sistema de entrenamiento y se utilizará para recorrer los 3 circuitos de validación, los cuales son totalmente desconocidos para estos coches.

Las conclusiones que se han obtenido de los resultados del conjunto de pruebas son las siguientes:

1. En ambos sistemas de entrenamiento, la función de ponderación de la distancia recorrida al cuadrado es la que proporcionaba unos mejores resultados en los entrenamientos.

2. El sistema de entrenamiento que recorre los circuitos de difícil a fácil, ha obtenido mejores resultados y ha sido capaz de superar 2 de los circuitos de validación.

3.5 Conclusión

En este capítulo hemos explicado toda la estructura que utilizaremos de base, el conjunto de pruebas y las conclusiones obtenidas de los resultados del TFG antecesor. Partiendo de este punto, vamos a realizar una serie de mejoras gráficas y técnicas en la estructura que utilizamos de base para proporcionarnos un proyecto más eficiente y completo.

Utilizaremos las lecciones aprendidas en las conclusiones de los resultados para enfocarnos en realizar 2 nuevos casos de estudio dónde utilizaremos en cada uno un nuevo sistema de entrenamiento. Utilizaremos la función de ponderación de la distancia recorrida al cuadrado para ambos entrenamientos, ya que se ha demostrado que proporciona mejores resultados. Por último, en ambos casos de estudio se busca conseguir eliminar la dependencia que existía en el orden en que se entrenaba el conjunto de circuitos y mejorar los resultados del anterior trabajo, es decir, que algún coche consiga recorrer los 3 circuitos de validación satisfactoriamente.

CAPÍTULO 4

TRABAJO REALIZADO

El trabajo realizado lo dividimos en 3 etapas que explicaremos a continuación:

1. **Primer etapa**. Aplicaremos una serie de ampliaciones y mejoras en el código relacionadas con aspectos visuales y técnicos.

2. **Segunda etapa**. Construiremos un sistema automatizado de generación de pruebas que utilizaremos para dos enfoques de entrenamiento diferentes. Este sistema será capaz de ejecutar dichas pruebas y almacenar los resultados obtenidos.

3. **Tercera etapa**. Creamos un sistema de simulación de pruebas en el que un coche ya entrenado recorrerá los circuitos de entrenamiento y los de validación.

Antes de explicar el trabajo realizado, se tiene que hacer un pequeño hincapié en la técnica de aprendizaje no supervisado y como la hemos utilizado en nuestro proyecto.

4.1 Aprendizaje no supervisado

El aprendizaje no supervisado es un modelo de aprendizaje automático en el cual no sabemos a priori los datos de salida que debería tener nuestra red y, por lo tanto, no sabemos si las decisiones que ha tomado son correctas. En el caso del proyecto no sabemos si la RN de cada coche está decidiendo correctamente la velocidad y la dirección del mismo. Para solucionar este problema deberemos complementar la RN con un sistema capaz de obtener una métrica del rendimiento y que nos permita modificarla para aproximarnos más a nuestro objetivo. Este sistema consistirá en el FP, el cual está basado en los AG.

Una vez una carrera ha finalizado, es decir, cuando todos los coches han acabado el número de vueltas total del circuito o se han estrellado con el entorno, entonces aplicaremos el FP para poder obtener la siguiente generación de coches para la siguiente carrera.

Este proceso se aplicará reiteradamente hasta que se acabe el número de generaciones máximo en caso de que se haya determinado o que uno de los coches haya conseguido acabar todos los circuitos satisfactoriamente.

4.2 Ampliación de funcionalidades

En esta sección se van a explicar una serie de modificaciones en el funcionamiento de nuestro proyecto. Algunas de las modificaciones consistirán en añadir nuevas funcionalidades, mientras otras consistirán en arreglar errores que dificultan la correcta ejecución y entendimiento del programa. Principalmente tendremos 3 partes, las cuales serán las mejoras gráficas, las técnicas y por último, la explicación de la nueva implementación del FP.

4.2.1 Mejoras gráficas

Se han implementado 2 nuevas mejoras y se ha arreglado un error en el visionado del sensor central. La primera mejora que hemos implementado es utilizar la cámara para que tenga el foco siempre en el coche vivo que más distancia ha recorrido en el circuito. En el anterior proyecto, la cámara únicamente seguía al coche que más distancia había recorrido y en caso que colisionará con el entorno, la cámara se quedaba estática y bloqueada mientras había coches en la carrera que aún no habían colisionado. Para solucionar este problema se ha diseñado un sistema que mantiene la cámara en el coche vivo que ha recorrido más distancia.

La segunda mejora ha sido la implementación de color a todos los coches de una carrera y la aplicación de dicho color a la información de estado de la pantalla. Se le asignará un color distintivo a cada coche que será su marca de identidad en los circuitos de entrenamiento y de validación. El color de la información de estado de la pantalla será el del coche que está siguiendo la cámara. En la Figura 4.1 se puede apreciar los distintos colores.

La última mejora gráfica consiste en cambiar la implementación del anterior proyecto, el cual mostraba las barras del sensor central del último coche vivo que se había calculado y no del que tenía actualmente el foco de la cámara. Para dicha modificación se tiene que calcular primeramente las distancias del sensor central de cada coche e implementar en la parte visual la del coche que está siguiendo la cámara. En la Figura 4.2 podemos observar las barras de los sensores en la pantalla.

4.2.2 Mejoras técnicas

Se han implementado 3 nuevas mejoras técnicas con el objetivo de mejorar el entrenamiento de los coches en los circuitos. La primera nueva funcionalidad consiste en no hacer sobreentrenamiento, es decir, no se seguirá entrenando el conjunto de coches en los circuitos de entrenamiento cuando uno de ellos ha sido capaz de realizar satisfactoriamente todos los circuitos. La segunda nueva funcionalidad consiste en inhabilitar los coches que son ineficientes a la hora de recorrer un circuito. Existen dos tipos de acciones que determinan a un coche ineficiente:

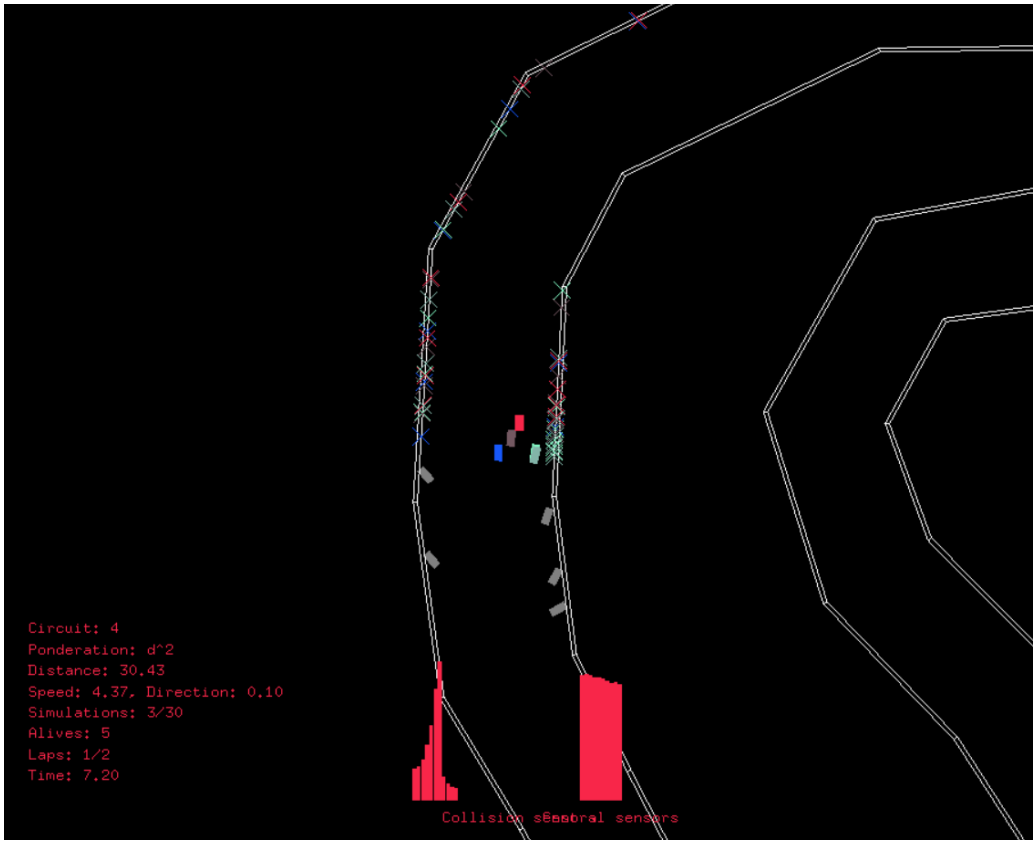

Figura 4.1: Aplicación de color a los coches y a la información de estado de la pantalla.

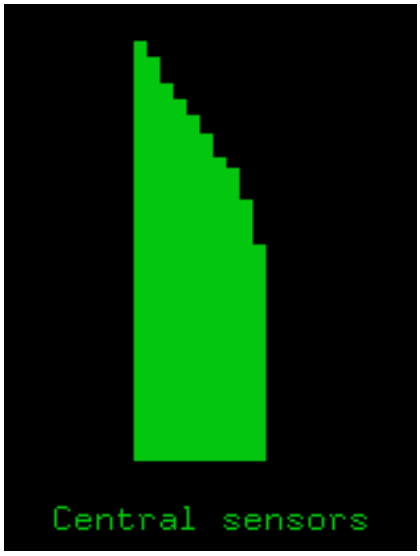

Figura 4.2: Sensores centrales del coche con el foco de la cámara.

1. La RN del coche se ha configurado de tal manera que se queda dando vueltas indeterminadamente en la misma posición.

2. La RN tiene una mala configuración para un conjunto de entradas y determina que el coche debe dar la vuelta e ir en sentido contrario en el circuito.

La solución más adecuada para solucionar este problema es la de inhabilitar los coches donde el tiempo de carrera sea superior a la distancia que han recorrido. El concepto de inhabilitar un coche es el equivalente a cuando un coche ha colisionado con el entorno, es decir, deja de participar en el recorrido del circuito. Para dotar a los coches de un cierto grado de libertad para poder rectificar en caso de un mal inicio de la carrera, se ha establecido que se tendrá en cuenta la regla de inhabilitar el coche a partir de las 10 unidades de tiempo.
La última funcionalidad técnica consiste en permitir a todos los coches finalizar un circuito. La finalidad de esta mejora consiste en que nosotros determinamos el rendimiento de un coche por la distancia recorrida sin tener en cuenta el tiempo que ha tardado en finalizar la carrera. Hay que tomar en consideración que un coche a la velocidad mínima siempre superará el tiempo transcurrido y, por lo tanto, no será inhabilitado por la mejora anterior.

4.2.3 Filtro de partículas

Se ha reestructurado y mejorado el FP que hemos obtenido del anterior TFG. En el anterior TFG, el FP únicamente se podía usar en la misma carrera e implementaba la funcionalidad de crear los nuevos coches para la siguiente generación. En la reestructuración que hemos implementado, la utilización es totalmente reusable y no necesita depender de las carreras, ni tiene que crear los coches. Su funcionalidad consistirá en utilizar el rendimiento obtenido por los coches en varios circuitos, aplicar el FP y se obtendrá un nuevo conjunto de RN que han sido modificadas para la siguiente generación.
Por otro lado, la nueva funcionalidad implementada consiste en mantener un determinado número de las RN que han obtenido un mejor resultado. Cuando aplicamos el FP se obtiene un nuevo conjunto de RN a las cuales se les aplica un ruido aleatorio, el cual modifica los pesos de las conexiones entre neuronas y sus umbrales de manera aleatoria. Este ruido implica que las nuevas generaciones pueden ser mejores o peores que sus antecesores.
Para mejorar el aprendizaje, hemos implementado que se mantengan el 5 % de las mejores RN sin aplicarles ruidos para la siguiente generación. Las restantes RN de la siguiente generación las escogemos mediante el FP y les aplicaremos un ruido el cual modificará minimamente sus pesos y umbrales. Con esta mejora conseguimos que en el caso de que las RN a las que se les ha aplicado el ruido, sean peores que el 5 % que mantenemos, entonces las RN sin ruido seguirán siendo las mejores y las mantendremos. Por otro lado, en el caso de que las nuevas RN con ruido sean mejores, es muy probable que el FP las escoja para futuras generaciones.

4.3 Sistema automatizado de generación de casos de prueba

En el capítulo 3, explicamos el conjunto de pruebas que se utilizó en el TFG antecesor. Partiendo de este punto, vamos a realizar dos nuevos casos de estudio a la hora de

entrenar un conjunto de circuitos. Estos nuevos enfoques buscarán eliminar la dependencia que existía en el orden en que se entrenaba el conjunto de circuitos y mejorar los resultados del anterior trabajo, es decir, que algún coche consiga recorrer los 3 circuitos de validación satisfactoriamente.

4.3.1 Objetivos

Los objetivos principales de nuestro proyecto son los siguientes:

1. Aplicar dos nuevos casos de estudio en el entrenamiento de la RN de los coches, cada uno utilizando su propio sistema de recorrido en el conjunto de circuitos de entrenamiento.
2. Construir un sistema automatizado capaz de realizar todo el conjunto de pruebas de ambos casos de estudio y almacenar en un fichero cada una de las RN del mejor coche y los resultados obtenidos.
3. Montar un sistema que sea capaz de replicar un caso de prueba a partir de la selección de circuitos del conjunto de entrenamiento. Una vez seleccionado los circuitos del conjunto de entrenamiento se tiene que visualizar al mejor coche de ese caso de prueba recorriendo cada uno de los circuitos de entrenamiento y de validación. Los resultados obtenidos por esta simulación deben ser los mismos que se han obtenido en las métricas de rendimiento.

4.3.2 Caso de estudio 1

En el primer caso buscamos ver la influencia que tiene cada circuito de entrenamiento en los de validación. Para ello, vamos a entrenar un conjunto de coches en cada uno de los circuitos individualmente hasta que consigan completarlos satisfactoriamente para luego utilizar el mejor coche de cada circuito en los tres de validación.

Los objetivos de este caso de estudio son los siguientes:

1. Determinar si los coches que han sido entrenados en circuitos que se recorren en el mismo sentido que los de validación, obtendrán mejores resultados.
2. Determinar si los coches que han sido entrenados en circuitos con obstáculos deberían obtener un mejor rendimiento al recorrer los circuitos de validación que también tienen esa dificultad añadida.
3. Determinar si el mejor coche que ha sido capaz de completar el circuito de entrenamiento, es capaz de recorrer los 3 circuitos de validación satisfactoriamente.

Las características de este caso de estudio son las siguientes:

1. Utilizaremos un total de 100 coches y sus RN se generarán de forma aleatoria. Hemos realizado varios conjuntos de pruebas con diversas cantidades de coches y hemos obtenido que esta cantidad de coches era la ideal para conseguir resultados en un tiempo razonable y con mayor probabilidad de obtener una RN que fuera muy eficiente.

2. La función de ponderación utilizada es la distancia recorrida al cuadrado. En los resultados obtenidos del TFG antecesor se demuestra que esta función de ponderación nos ofrece los mejores resultados.

3. Tendremos un número de generaciones ilimitado. Por lo tanto, cada prueba finalizará cuando al menos uno de los coches finalice el circuito.

La función de ponderación será nuestra métrica para calcular el rendimiento de un coche en un circuito determinado. La función de ponderación que vamos a utilizar para nuestras pruebas será la de la distancia que ha recorrido un coche elevado al cuadrado. Esta función de ponderación favorecerá a los coches que han recorrido más distancia ya que el rendimiento será cuadrático y no lineal. En la Figura 4.3 se muestra la comparativa del rendimiento entre una función de ponderación de distancia lineal y una cuadrática.

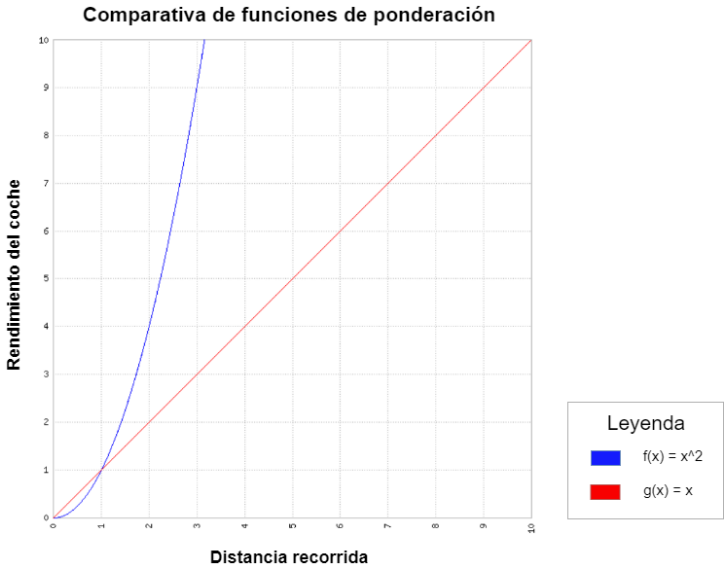

Figura 4.3: Comparativa de las funciones de ponderación.

Los circuitos que utilizaremos para nuestro entrenamiento serán los primeros cuatro. Para la validación utilizaremos los tres existentes. La decisión de entrenar únicamente estos circuitos es debido a la falta de tiempo para la realización de pruebas y la gran cantidad de generaciones que son necesarias para obtener un caso de prueba en el resto de circuitos. En la Figura 4.4 podemos ver todos los circuitos que disponemos para entrenamiento y validación.

4.3.3 Caso de estudio 2

Vamos a realizar un nuevo sistema de entrenamiento para las RN de los coches que busca eliminar la dependencia que existía en el orden en que se entrenaba el conjunto de circuitos del TFG antecesor. En el trabajo anterior, los coches recorrían un circuito, realizaban todas las generaciones en ese mismo circuito y luego pasaban al siguiente

realizando mismo proceso. Esto provocaba que una vez habíamos acabado el entrenamiento, las RN de los coches habían perdido el aprendizaje de los primero circuitos. En este sistema, nos enfocamos en que los coches deberán recorrer todos los circuitos antes de pasar a la siguiente generación, de esta manera nos aseguramos que no se olviden de los circuitos ya aprendidos.

Para poder aplicar este sistema hay que tener en cuenta dos conceptos, el primero es la función de ponderación, la cual determina el rendimiento de un coche en el circuito y, el segundo, es un nuevo concepto que le llamaremos función de rendimiento de circuitos.

La función de rendimiento de circuitos se ocupa de la tarea de calcular el rendimiento de un coche en la totalidad de los circuitos. Para obtener el rendimiento mencionado es necesario calcular el desempeño de los coches en un determinado circuito utilizando una función de ponderación. La primera función de rendimiento de circuitos que se ha utilizado ha sido la de hacer el sumatorio del resultado de la función ponderación de cada circuito para cada coche. Esta función consigue que podamos implementar nuestro nuevo enfoque de entrenamiento, pero sigue teniendo el fallo principal, el cual es la dependencia de los circuitos.

Un caso de ejemplo sería en el que hay dos circuitos y dos coches. En la primera generación, el primer coche hace excepcionalmente bien el primer circuito y muy mal el segundo circuito. Por otro lado, el segundo coche realiza medianamente bien ambos circuitos. Como resultado de aplicar esta función obtenemos que las futuras generaciones se enfocarán en realizar lo mejor posible el primer circuito, dejando de lado el segundo circuito. Este fallo de consistencia lo solucionaremos con la siguiente y definitiva función de rendimiento de circuitos.

La versión mejorada de esta función consiste en aplicar el sumatorio de todas las funciones de ponderación para cada circuito de cada coche pero una vez calculado el valor, lo multiplicaremos por el resultado más bajo que nos habrá dado las funciones de ponderación. Aplicando esta medida nos aseguramos que las siguientes generaciones de coches no dependan de los circuitos que mejor recorren sino del equilibrio entre circuitos. Cogiendo como referencia el ejemplo anterior, en este caso el coche que ha recorrido ambos circuitos de manera aceptable tendrá más posibilidades de pertenecer a la siguiente generación que el coche que estaba desbalanceado.

Una vez haya finalizado una generación y obtenido el rendimiento de cada uno de los coches gracias a la función de rendimiento de circuitos, se comprobará si ha acabado el entrenamiento. La finalización del entrenamiento ocurre cuando un coche ha logrado acabar satisfactoriamente todos los circuitos o ya hemos realizado el número de generaciones que habíamos establecido. En el caso de que no hayamos concluido deberemos aplicar el filtro de partículas, el cual nos proporcionará la siguiente generación de coches. En cada generación de coches se volverá a calcular la función de rendimiento de circuito de nuevo.

Una vez acabado el entrenamiento del conjunto de coches, se seleccionará el coche

cuya RN ha demostrado tener un mayor rendimiento y lo utilizaremos para recorrer los 3 circuitos de validación, los cuales son totalmente desconocidos para él. Una vez finalizado todo el proceso, se procede a almacenar en un fichero la RN del mejor coche y en otro fichero el rendimiento en cada uno de los circuitos del conjunto de entrenamiento y de validación.

El conjunto de pruebas lo vamos a crear a partir de combinar todas las posibilidades que podemos generar sin repeticiones a partir de los 7 primeros circuitos de entrenamiento. Por lo tanto, tendremos un total de $(2^7 - 1) = 127$ casos de prueba. La representación de las combinaciones de todos los circuitos se puede observar en el conjunto {{1}, {2}, ..., {7}, {1, 2}, {1, 3}, ..., {6, 7}, {1, 2, 3}, {1, 2, 4}, ..., {5, 6, 7}, {1, 2, 3, 4}, {1, 2, 3, 5}, ..., {4, 5, 6, 7}, {1, 2, 3, 4, 5, 6}, ..., {2, 3, 4, 5, 6, 7}, {1, 2, 3, 4, 5, 6, 7}}. La cantidad de circuitos la hemos elegido teniendo en cuenta la limitación del tiempo para realizar las pruebas y que cada ampliación con un nuevo circuito supone duplicar el número de casos de pruebas. En la Figura 4.4 podemos visualizar todos los circuitos que usamos en los casos de prueba.

Los objetivos de este caso de estudio son los siguientes:

1. Suponemos que hay coches que han entrenado en un conjunto circuitos y han podido completar uno o más circuitos de validación. Queremos determinar que en el caso de entrenar en los circuitos que fueron completados y, además, añadir más circuitos al entrenamiento, nos hará conseguir el mismo o un mejor resultado en los circuitos de validación.

2. Determinar si alguna combinación de circuitos de entrenamiento nos proporciona un coche capaz de completar satisfactoriamente los 3 circuitos de validación.

Todas las pruebas deben cumplir una serie de requisitos:

1. La función de ponderación que utilizaremos para obtener la métrica del resultado en un circuito será la de la distancia recorrida al cuadrado. En los resultados obtenidos del TFG antecesor se demuestra que esta función de ponderación nos ofrece los mejores resultados.

2. Habrá un total de 100 coches y sus RN iniciales se generarán aleatoriamente. Hemos realizado varios conjuntos de pruebas con diversas cantidades de coches y obtuvimos que esta cantidad de coches era la ideal para conseguir resultados en un tiempo razonable y con mayor probabilidad de obtener una RN que fuera muy eficiente.

3. Se establece el número máximo de generaciones en 30. En caso de que algún coche acabe todos los circuitos del conjunto de circuitos de entrenamiento antes de llegar al número máximo de generaciones, se procederá a pasar directamente a los circuitos de validación. Hemos realizado varias pruebas para determinar el número de generaciones y como conclusión, hemos obtenido que 30 generaciones nos permite obtener coches capaces de aprender a acabar los distintos circuitos en un tiempo razonable.

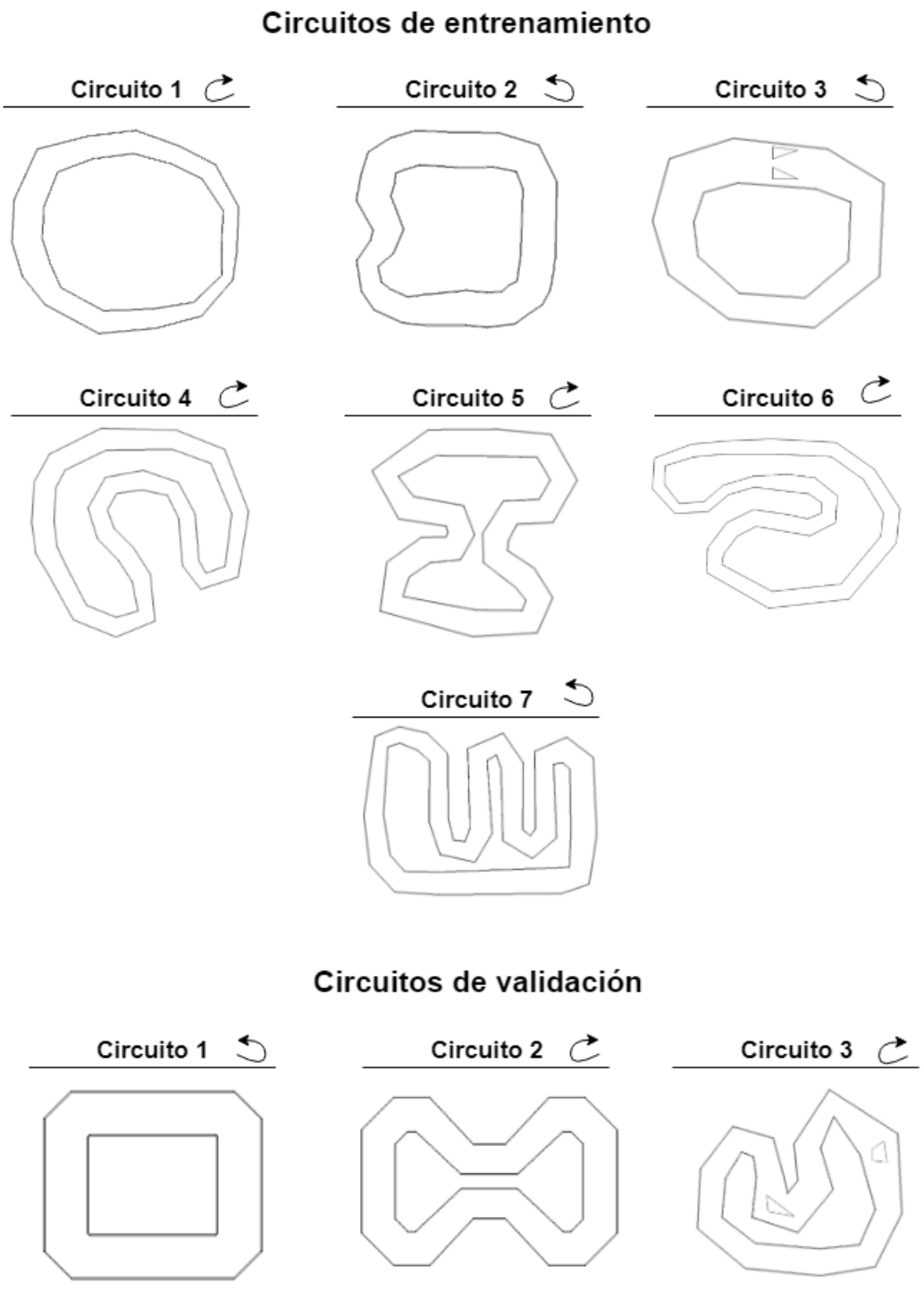

Figura 4.4: Circuitos de entrenamiento y de validación.

4.3.4 Sistema automático de pruebas

Una vez concluido con el desarrollo de ambos casos de estudio, vamos a proceder a crear un sistema automatizado que se encargará de realizar el conjunto de pruebas.

Cada prueba consiste en aplicar el proceso que ya hemos mencionado en los apartados anteriores, aplicándole distintos circuitos de entrenamiento. El objetivo es obtener métricas suficientes para poder hacer más adelante una comparativa de la influencia de los circuitos de entrenamiento escogidos y los resultados obtenidos en los circuitos de validación. En la Figura 4.5 se muestra el ciclo de vida de nuestro enfoque.

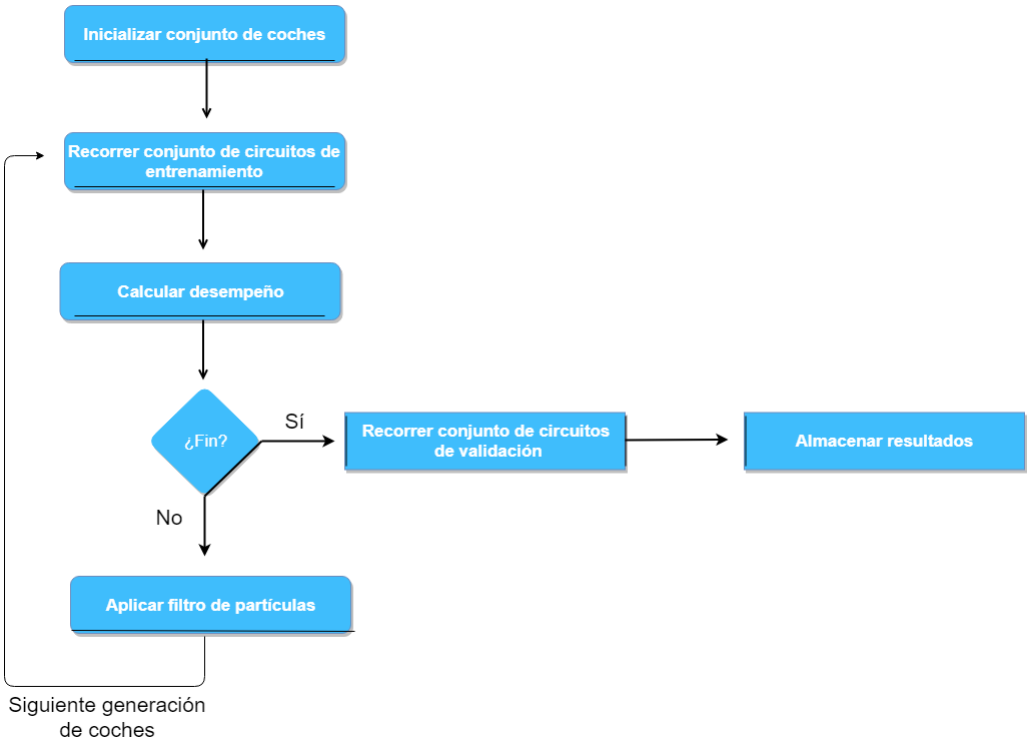

Figura 4.5: Diagrama de flujo del programa.

4.3.5 Simulador de casos de prueba

El simulador tiene como objetivo buscar una manera más visual de representar el comportamiento de la RN del mejor coche de un determinado caso de prueba. La parte gráfica se compone de 2 ventanas.

La primera ventana que nos mostrará el simulador es la de elegir los circuitos de entrenamiento que queremos utilizar. Recordemos que cada caso de prueba tiene unas características comunes, pero cada uno tendrá su propio conjunto de circuitos de entrenamiento. En la Figura 4.6 vemos la ventana de selección de circuitos de entrenamiento.

Una vez se ha seleccionado el conjunto de entrenamiento, procederemos a preparar el escenario. Para ello, vamos a utilizar la RN del mejor coche para ese conjunto de circuitos que fue almacenada cuando hicimos el **sistema automatizado de pruebas**. En la simulación podremos observar el recorrido del coche en cada uno de los circuitos de entrenamiento y más adelante los de validación. Al finalizar la ejecución se mostrará por consola el caso de prueba al cual pertenece y el rendimiento en cada uno de los circuitos. Recordemos que el rendimiento en nuestro caso es el resultado de aplicar

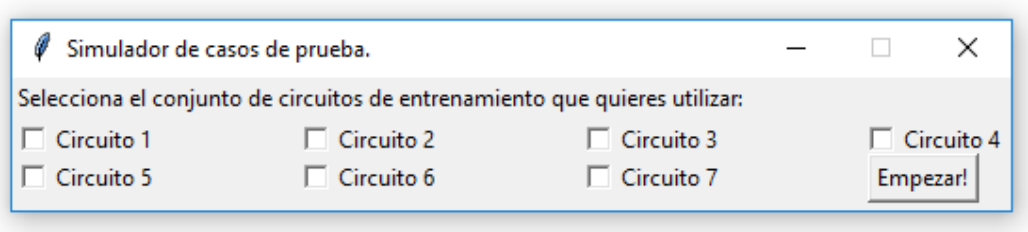

Figura 4.6: Ventana inicial del simulador de casos de prueba.

la función de ponderación, la cual es la distancia recorrida por el coche en el circuito elevado al cuadrado.

4.4 Conclusión

En este capítulo hemos explicado como el trabajo que hemos realizado ha sido dividido en 3 etapas. En la primera etapa hemos explicado todas las nuevas funcionalidades y las mejoras que hemos aplicado en la base de nuestro proyecto. En la segunda etapa hemos explicado como ha sido desarrollado un sistema automatizado que cumpliendo una serie de requisitos sería capaz de ejecutar y obtener los resultados de un conjunto de pruebas. Por último, en la etapa final se ha construido un sistema de simulación de pruebas en el que un coche ya entrenado recorrerá los circuitos de entrenamiento y los de validación.

CAPÍTULO 5

RESULTADOS

En este capítulo vamos explicar con detenimiento los resultados obtenidos en los dos casos de estudios. Cada uno de los casos tendrá las siguientes secciones:

1. **Hipótesis**. Explicaremos las suposiciones que nos han llevado a hacer el estudio.
2. **Metodología experimental**. Es el proceso que hemos utilizado para obtener lograr responder la hipótesis.
3. **Resultados**. La exposición de los resultados de manera clara y concisa.
4. **Análisis de los resultados**. Analizamos los resultados obtenidos y verificamos las hipótesis.

5.1 Caso de estudio 1

El objetivo principal de este caso de estudio es dar una explicación a por qué algunos circuitos de entrenamiento influyen más que otros cuando recorremos los de validación. Para ello, buscamos conocer la capacidad que tendrá un coche de resolver un conjunto de circuitos desconocido cuando previamente ha sido entrenado en un circuito de entrenamiento hasta que consigue resolverlo.

5.1.1 Hipótesis

Nuestras hipótesis van a ser las siguientes:

1. Los coches que han sido entrenados en circuitos que se recorren en el mismo sentido que los de validación, obtendrán mejores resultados.
2. Los coches que han sido entrenados en circuitos con obstáculos deberían obtener un mejor rendimiento al recorrer los circuitos de validación que también tienen esa dificultad añadida.

3. Determinar si el mejor coche de cada circuito de entrenamiento es capaz de recorrer los 3 circuitos de validación satisfactoriamente.

5.1.2 Metodología experimental

Para realizar nuestro experimento, vamos a utilizar los 4 primeros circuitos de entrenamiento y los 3 de validación de la Figura 4.4. Los motivos de esta elección son la limitación de tiempo a la hora de realizar las pruebas y que este conjunto de circuitos es tiene todos las características para intentar validar nuestras hipótesis.

En este caso de estudio hemos entrenado los coches en cada uno de los circuitos de entrenamiento individualmente, hasta que conseguían completarlos. Una vez hemos acabado el entrenamiento, escogemos el coche que ha conseguido completar el circuito y lo utilizaremos para recorrer los 3 circuitos de validación. Este experimento lo vamos a realizar un total de 30 veces para uno de los circuitos de entrenamiento.

Las características que utilizaremos fueron explicadas en cada caso de prueba, son las siguientes:

1. Utilizaremos un total de 100 coches. Esta cantidad ha sido elegida para tener mayor probabilidad de tener uno o más coches que sean mejores a la hora de recorrer el circuito.

2. La función de ponderación utilizada para calcular el rendimiento del coche es la distancia recorrida al cuadrado.

3. Tendremos un número de generaciones ilimitado. Por lo tanto, cada experimento finalizará cuando al menos uno de los coches finalice el circuito.

5.1.3 Resultados

Los resultados obtenidos en todas las pruebas han sido almacenados en un archivo llamado *Resultados_caso_de_estudio_1.ods*, que se encuentra en el repositorio del proyecto [31].

Una vez obtenidos los resultados, debemos tratarlos para permitir hacer un correcto análisis. Para ello, calcularemos la media, la varianza y el porcentaje de completitud en base a la media de las 30 muestras de cada circuito de validación. Hay que tener en cuenta que la distancia de cada circuitos está elevada al cuadrado para facilitar la comprensión de los resultados. En las Tablas 5.1, 5.2, 5.3 se puede apreciar el tratamiento que le hemos aplicado a los resultados. Por último, en la Tabla 5.4 podemos observar el porcentaje medio obtenido de los tres circuitos de validación para cada circuito de entrenamiento.

5.1.4 Análisis de resultados

Vamos a estudiar el caso de cada circuito de entrenamiento de manera individual para verificar nuestras hipótesis. Por último, estudiaremos los resultados en los circuitos de

	Circuito de validación 1 Distancia total: 174000		
Circuitos de entrenamiento	**Media**	**Varianza**	**Porcentaje de la media**
1	15766,66	44129,55	9 %
2	126511,32	74372,15	73 %
3	114547,39	79719,6	66 %
4	88560,86	84449,34	51 %

Tabla 5.1: Tratamiento de los resultados del circuito de validación 1.

	Circuito de validación 2 Distancia total: 434000		
Circuitos de entrenamiento	**Media**	**Varianza**	**Porcentaje de la media**
1	49502,07	110502,27	11 %
2	145008,46	197958,51	33 %
3	52905,31	132365	12 %
4	270530,02	201576,38	62 %

Tabla 5.2: Tratamiento de los resultados del circuito de validación 2.

	Circuito de validación 3 Distancia total: 684000		
Circuitos de entrenamiento	**Media**	**Varianza**	**Porcentaje de la media**
1	11231,31	12064,20	2 %
2	21875,57	40804,15	3 %
3	5087,77	8436,93	1 %
4	60375,69	170354,09	9 %

Tabla 5.3: Tratamiento de los resultados del circuito de validación 3.

Circuitos de entrenamiento	**Porcentaje medio de los circuitos de validación**
1	7 %
2	36 %
3	26 %
4	41 %

Tabla 5.4: Tratamiento de los resultados de todos los circuitos de validación. Para ello, calcularemos la media de los porcentajes recorridos en los 3 circuitos.

validación de forma conjunta.

Antes de empezar el análisis hay que explicar las características de cada circuito de validación (ver Figura 4.4):

1. **Primer circuito de validación**. Este circuito se recorre en sentido antihorario y carece de obstáculos.

2. **Segundo circuito de validación**. Este circuito se recorre en sentido horario y carece de obstáculos.

3. **Tercer circuito de validación**. Este circuito se recorre en sentido horario y tiene dos obstáculos.

Circuito de entrenamiento 1

Las características de este circuito son que se recorre en sentido horario y que no tiene obstáculos. Si tenemos en cuenta nuestras hipótesis, debería dar un mejor resultado para resolver el circuito de validación 2, ya que este se recorre en el mismo sentido y no tiene obstáculos. Debemos tener en cuenta que el primer circuito de validación es más fácil de recorrer que este y el último es mucho más complicado.

El resultado que hemos obtenido confirma la hipótesis de que recorrer el circuito en el mismo sentido es un factor importante a la hora del aprendizaje de los coches ya que hemos obtenido de media un mayor rendimiento a la hora de completar el circuito de validación 2.

En este caso podemos apreciar que alguna vez ha completado primeros dos circuitos de validación aunque en la mayoría haya sido bastante ineficiente. Esto lo podemos observar con su media que es muy baja y su varianza que es bastante elevada. En el caso del circuito de validación 3, podemos observar que ha sido incapaz de resolverlo debido a las dificultades del sentido en el que se recorre, que hay obstáculos y que es el circuito más complicado de resolver.

En el caso de la tercera hipótesis, en ninguna de las pruebas se ha conseguido que un coche sea capaz de recorrer satisfactoriamente los 3 circuitos de validación.

Circuito de entrenamiento 2

Las características de este circuito son que se recorre en sentido antihorario y que no tiene obstáculos. Si tenemos en cuenta nuestras hipótesis, debería dar un mejor resultado para resolver el circuito de validación 1, ya que este se recorre en el mismo sentido y no tiene obstáculos.

El resultado que hemos obtenido sigue confirmando la hipótesis de que el sentido en que se recorre el circuito es un factor muy decisivo ya que ha tenido un gran éxito en el primer circuito de validación. En este caso obtenemos una media muy alta con una varianza más baja, lo cual implica que en muchas ocasiones se ha conseguido recorrer

satisfactoriamente dicho circuito. En cuanto al resto de circuitos, podemos observar que le ha influido mucho el sentido en que se recorre y, por lo tanto, se han obtenido pésimos resultados.

En el caso de la tercera hipótesis, en ninguna de las pruebas se ha conseguido que un coche sea capaz de recorrer satisfactoriamente los 3 circuitos de validación.

Circuito de entrenamiento 3

Las características de este circuito son que se recorre en sentido antihorario y que tiene obstáculos. En este caso ninguno de los circuitos de validación cumple coincide completamente con las características de este circuito.

En este caso vemos que influye mucho el sentido en que se recorre, ya que en el primer circuito de validación ha obtenido de media buenos resultado. Por otro lado, la capacidad de esquivar obstáculos no le ha funcionado en el tercer circuito de validación ya que ha obtenido un resultado muy malo. Para este caso, deducimos que ha influido mucho el sentido en que se recorre el circuito y que el tercer circuito es mucho más complicado.

En el caso de la tercera hipótesis, en ninguna de las pruebas se ha conseguido que un coche sea capaz de recorrer satisfactoriamente los 3 circuitos de validación.

Circuito de entrenamiento 4

Las características de este circuito son que se recorre en sentido horario y que no tiene obstáculos. En este caso los coches aprenderán a hacer curvas mucho más cerradas y analizaremos su implicación en el resto de circuitos.

Como podemos observar, los circuitos donde debería obtener una mejor puntuación son el segundo y tercer circuito de validación. En el segundo circuito se han obtenido unos resultados aceptables pero en el tercero no se ha conseguido. Por otro lado, el primer circuito a pesar de que no hubiera aprendido a recorrerlo en el mismo sentido ha obtenido también unos resultados aceptables. Esto nos hace replantearnos hasta que punto nos influencia el sentido del circuito.

En el caso de la tercera hipótesis, en la prueba número 10, se ha obtenido un coche que ha sido capaz de realizar los 3 circuitos de validación.

Conclusión

Para concluir, podemos afirmar que la hipótesis del sentido en que se recorren los circuitos es correcta. Hemos podido comprobar como se cumplía en cada uno de los diferentes análisis que hemos hecho. Por otro lado, no podemos confirmar la influencia de los circuitos de entrenamiento que tenían obstáculos a la hora de superar circuitos de validación con el mismo problema. Esto se debe a la falta de obstáculos en los circuitos de entrenamiento y de validación que hemos utilizado. Por último, una única prueba del circuito 4 nos ha proporcionado un coche capaz de realizar los 3 circuitos

de validación, por lo cual no podemos validar la tercera hipótesis ya que ha sido un resultado demasiado atípico.

En la Tabla 4.4 podemos observar como en general el primer circuito de entrenamiento ha sido muy ineficiente. Esto es debido a que su facilidad a la hora de resolverlo hace que los coches aprendan muy poco a resolver factores tan importantes como las curvas y los obstáculos. El resto de circuitos tiene un rendimiento poco eficiente debido al poco entrenamiento que supone únicamente un circuito.

5.2 Caso de estudio 2

El objetivo principal de este caso de estudio es comprobar el aprendizaje simultaneo en varios circuitos (entrenamiento paralelo). En el trabajo previo se realizó el aprendizaje de varios circuitos pero no de forma simultanea (entrenamiento en serie).

5.2.1 Hipótesis

Nuestras hipótesis van a ser las siguientes:

1. Suponemos que hay coches que han entrenado en un conjunto circuitos y han podido completar uno o más circuitos de validación. La primera hipótesis que vamos a utilizar es que en el caso de entrenar en los circuitos que fueron completados y, además, añadir más circuitos al entrenamiento, nos hará conseguir el mismo o un mejor resultado en los circuitos de validación.

2. Determinar si alguna combinación de circuitos de entrenamiento nos proporciona un coche capaz de completar satisfactoriamente los 3 circuitos de validación.

5.2.2 Metodología experimental

Para realizar nuestro experimento, vamos a utilizar los 7 circuitos de entrenamiento y los 3 de validación de la Figura 4.4. Los motivos de esta elección se basa en la limitación del tiempo para hacer la mayor cantidad de pruebas posible.

Cada una de las pruebas que haremos consistirá en una combinación diferente de los 7 circuitos, las cuales pueden incluir los circuitos individuales o la combinación de 2 o más circuitos. Con la combinación de todos los circuitos obtendremos un total de 127 casos de prueba.

Las características que utilizaremos en nuestros casos de prueba son los siguientes:

1. La función de ponderación que utilizaremos para obtener la métrica del resultado en un circuito será la de la distancia recorrida al cuadrado. Para un conjunto de circuitos se calcula el sumatorio de dichas funciones para cada circuito, y se multiplica por el mínimo de dichas funciones.

2. Habrá un total de 100 coches y sus RN iniciales se generarán aleatoriamente.

	Porcentaje de completitud		
Circuitos de entrenamiento	**Circuito de validación 1**	**Circuito de validación 2**	**Circuito de validación 3**
1	0%	2%	1%
2	5%	6%	4%
3	1%	0%	3%
4	7%	5%	0%
5	3%	1%	1%
6	100%	56%	6%
7	100%	0%	3%

Tabla 5.5: Resultados de cada circuitos de manera individual.

3. Establecemos el número máximo de generaciones en 30. A diferencia del caso de estudio 1, al llegar a las 30 simulaciones no tiene por qué existir un coche que finalice el conjunto de circuitos. En caso de que algún coche acabe todos los circuitos del conjunto de entrenamiento antes de llegar al número máximo de generaciones, se procederá a pasar directamente a los circuitos de validación.

5.2.3 Resultados

Los resultados obtenidos de todas las pruebas han sido almacenados en un archivo llamado *Resultados_caso_de_estudio_2.ods*, que se encuentra en el repositorio del proyecto [31]. Hemos escogido una pequeña muestra de todos los resultados, dicha muestra la podemos observar en las Tablas 5.5 y 5.6.

5.2.4 Análisis de resultados

Lo primero que vamos a hacer para poder validar nuestra primera hipótesis, es dividir los resultados obtenidos en las categorías de circuitos individuales, combinación de 2 circuitos, etc.

Los resultados obtenidos de los circuitos individuales nos muestra claramente que el circuito 6 y 7 de entrenamiento han sido muy influyentes a la hora de completar el primer circuito de validación. Además, podemos observar que el circuito 6 ha tenido un mayor impacto en el segundo circuito de validación comparado con el resto los circuitos de entrenamiento. En el caso de que nuestra hipótesis fuera cierta, la combinación de los circuitos 6 o 7 con otros circuitos nos ofrecerá unos resultados iguales o superiores en los circuitos de validación.

En el caso de la combinación de 2 circuitos podemos observar que hay muchos casos en los que no se cumple nuestra hipótesis y por lo tanto queda invalidada. En el resto de combinaciones ocurre exactamente la misma situación.

En el caso de la segunda hipótesis, hemos obtenido que distintas combinaciones de circuitos de entrenamiento nos proporciona un coche capaz de realizar los 3 circuitos

Circuitos de entrenamiento							Porcentaje de completitud de los circuitos de validación		
1	**2**	**3**	**4**	**5**	**6**	**7**	**1**	**2**	**3**
X	X						5%	5%	4%
X		X					100%	100%	4%
X			X				100%	100%	100%
X				X			100%	100%	5%
X					X		2%	36%	2%
X						X	3%	2%	2%
	X	X					100%	0%	0%
	X		X				100%	73%	5%
	X			X			3%	100%	6%
	X				X		0%	0%	2%
	X					X	100%	100%	5%
		X	X				100%	100%	6%
		X		X			100%	100%	0%
		X			X		100%	100%	6%
		X				X	19%	3%	1%
			X	X			0%	4%	1%
			X		X		2%	1%	2%
			X			X	100%	5%	1%
				X	X		13%	100%	57%
				X		X	100%	3%	100%
					X	X	100%	23%	64%

Tabla 5.6: Resultados de la combinación de dos circuitos de entrenamiento.

de validación. Por otro lado, no hay una clara conexión entre la capacidad de finalizar los circuitos de validación y el conjunto escogido para el entrenamiento.

La conclusión que hemos obtenido es que nuestro experimento no es generalizable, lo que implica que no se obtendrán los mismos resultados si se vuelven a realizar todas las pruebas. Esto es debido a que cada prueba se ha realizado una única vez. Para solucionar este problema deberíamos realizar múltiples veces cada una de las pruebas y obtener la media y varianza de cada una. Por lo tanto, se necesitan hacer muchas más pruebas para comprobar correctamente las hipótesis.

CAPÍTULO 6

CONCLUSIONES

En este TFG hemos utilizado la técnica de redes neuronales para dar un nuevo enfoque al entrenamiento de coches en un videojuego de carreras. El objetivo del mismo era conseguir mejorar los resultados obtenidos en el TFG antecesor, para ello hemos utilizado dos nuevas maneras de entrenar los coches virtuales, para luego comprobar el rendimiento obtenido al recorrer un conjunto de circuitos completamente desconocidos.

Lo primero que hicimos fue un análisis completo del proyecto. Hemos conseguido cumplir todos los requisitos funcionales y no funcionales que ya habíamos especificado, en los cuales creábamos un software capaz de automatizar la generación de los casos de prueba y una vez obtenido los resultados, podíamos simular el recorrido del mejor coche en los circuitos de entrenamiento y validación. En cuanto a la planificación del proyecto, se han cumplido las 3 etapas en el plazo que habíamos establecido. La primera etapa tenía un plazo de un mes y medio y consistía en mejorar las funcionalidades de la estructura que utilizábamos de base del TFG antecesor.

Las siguientes etapas trataban de la creación de un software para generar y simular los casos de prueba y su duración total era de 3 meses. Por último, La elección de `Python` como lenguaje de programación ha sido muy acertada ya que nos ha sido de gran utilidad las librerías de `OpenGL` y de `NumPy`. Por otro lado, hemos aprovechado la portabilidad que nos ofrece para generar pruebas en el sistema operativo Windows y Linux.

Al acabar el análisis del proyecto, hemos empezado la etapa de desarrollo. Lo primero que hemos hecho ha sido mejorar la parte gráfica y técnica de la base del proyecto. A continuación, hemos desarrollado los dos casos de estudio a la hora de entrenar los coches de carreras. Una vez desarrollados los dos casos, hemos implementado un sistema automatizado que es capaz de realizar todo el conjunto de pruebas. Por último, hemos creado un simulador de los casos de prueba para hacer una simulación del mejor coche que recorrerá los circuitos de entrenamiento y de validación.

6.1 Resultados

En el primer caso de estudio, hemos podido confirmar la hipótesis de que influye el sentido en que se recorren los circuitos de entrenamiento para luego completar los circuitos de validación. Por otro lado, no hemos podido confirmar la segunda hipótesis, las cual eran que los circuitos de entrenamiento con obstáculos influían cuando se recorría los circuitos de validación que también tuviera obstáculos. La tercera hipótesis no la hemos podido validar, la cual era determinar si el mejor coche de cada circuito de entrenamiento es capaz de recorrer los 3 circuitos de validación satisfactoriamente. Como podemos observar, la falta de tiempo para realizar pruebas en este caso de estudio nos ha afectado significativamente.

En el segundo caso de estudio no hemos podido hacer un análisis completo debido a que no podemos generalizar las pruebas obtenidas por la falta de repetición de las mismas.

6.1.1 Mejoras

El primer caso estudio se podría mejorar considerablemente si aumentamos el número de circuitos de entrenamiento para obtener métricas más completas para confirmar las hipótesis.

En cuanto al segundo caso de estudio, se podría repetir la experimentación de todas las combinaciones de circuitos para obtener la generalización de los resultados.

6.2 Cumplimiento de objetivos

Se han cumplido satisfactoriamente todos los objetivos principales que consistían en implementar los dos casos de estudio, el desarrollo de un sistema automatizado de pruebas y un simulador de dichas pruebas. Además, queda comprobado que el aprendizaje paralelo obtiene mejores resultados que el aprendizaje en serie.

6.3 Futuras mejoras

Este TFG puede utilizarse como base para posibles mejoras en el futuro. La mejora más relevante consistiría en profundizar en las pruebas del caso de estudio 2, para determinar los circuitos que aportan mayor aprendizaje. Aquí se presentan otras posibles mejoras:

1. El número de generaciones que hemos utilizado para entrenar los coches no ha sido suficiente para que consiga completar los 7 circuitos de entrenamiento y, eso ha repercutido a la hora de superar los circuitos de validación. Una notable mejora sería incrementar el número de generaciones para que los coches puedan finalizar todos los circuitos.

2. Se podría utilizar técnicas de Apredizaje profundo (en inglés *Deep learning*) (AP) con múltiples capas ocultas de la RN para mejorar la precisión en el momento de decidir la velocidad angular y lineal.

3. Ampliar la cantidad de circuitos de entrenamiento nos ofrecería coches que son capaces de resolver un mayor abanico de situaciones. Esta mejora podría incluir un extenso conjunto de circuitos de validación para obtener mejores métricas a la hora de completar un circuito desconocido.

4. Las probabilidades de generar una RN de un coche que sea capaz de recorrer de manera más eficientemente los circuitos aumenta con la cantidad de coches que se utilicen.

5. En este TFG se ha utilizado únicamente una función de ponderación para hacer las pruebas. Se podría considerar utilizar otras que tengan en cuenta la distancia con la velocidad o el tiempo.

6. Una manera de mejorar la eficiencia de los coches a la hora de girar y de modificar la velocidad es aumentando la cantidad de sensores de colisión y de sensores centrales almacenados. Esta mejora permitirá modificar la capa inicial de la RN y aumentar el número de neuronas destinada a los sensores.

BIBLIOGRAFÍA

[1] M. Morro, "Machine Learning Playing Videogames," https://github.com/MateuMorro/TFG, 2019, [Online; accessed 01-July-2019]. (document), 1, 1.4, 3

[2] D. L. Poole, A. K. Mackworth, and R. Goebel, *Computational intelligence: a logical approach*. Oxford University Press New York, 1998, vol. 1. 1, 1.2

[3] S. J. Russell and P. Norvig, *Artificial intelligence: a modern approach*. Malaysia; Pearson Education Limited,, 2016. 1

[4] N. R. W. T. H. N. H. W. L. N. N. Baer, Ralph H. (Manchester, "Television gaming apparatus and method," patentus 3 659 285, April, 1972. 1

[5] M. Campbell, A. J. Hoane Jr, and F.-h. Hsu, "Deep blue," *Artificial intelligence*, vol. 134, no. 1-2, pp. 57–83, 2002. 1.1

[6] M. Illescas, "El día en que un ordenador ganó al campeón del mundo de ajedrez," https://www.lavanguardia.com/deportes/20160210/302037419496/dia-ordenador-gano-campeon-mundo-ajedrez.html, 2016, [Online; accessed 01-July-2019]. 1.1

[7] L. Hohl, R. Tellez, O. Michel, and A. J. Ijspeert, "Aibo and webots: Simulation, wireless remote control and controller transfer," *Robotics and Autonomous Systems*, vol. 54, no. 6, pp. 472–485, 2006. 1.1

[8] "Aibo (1999) - ROBOTS: Your Guide to the World of Robotics," https://robots.ieee.org/robots/aibo/, [Online; accessed 01-July-2019]. 1.2

[9] S. Gibbs, "Google sibling waymo launches fully autonomous ride-hailing service," *The Guardian*, vol. 7, 2017. 1.1

[10] K. Payares Quiñones and J. Romero Torres, "Asistente virtual para el sistema de identificación de potenciales beneficiarios de programas sociales-sisbén," 2019. 1.1

[11] S. Gibbs, "Google buys uk artificial intelligence startup deepmind for£ 400m," *The Guardian*, vol. 27, 2014. 1.1

[12] D. Silver, A. Huang, C. J. Maddison, A. Guez, L. Sifre, G. Van Den Driessche, J. Schrittwieser, I. Antonoglou, V. Panneershelvam, M. Lanctot *et al.*, "Mastering the game of go with deep neural networks and tree search," *nature*, vol. 529, no. 7587, p. 484, 2016. 1.1

[13] M. Jaderberg, W. M. Czarnecki, I. Dunning, L. Marris, G. Lever, A. G. Castaneda, C. Beattie, N. C. Rabinowitz, A. S. Morcos, A. Ruderman *et al.*, "Human-level performance in first-person multiplayer games with population-based deep reinforcement learning," *arXiv preprint arXiv:1807.01281*, 2018. 1.1

[14] O. Vinyals, I. Babuschkin, J. Chung, M. Mathieu, M. Jaderberg, W. M. Czarnecki, A. Dudzik, A. Huang, P. Georgiev, R. Powell, T. Ewalds, D. Horgan, M. Kroiss, I. Danihelka, J. Agapiou, J. Oh, V. Dalibard, D. Choi, L. Sifre, Y. Sulsky, S. Vezhnevets, J. Molloy, T. Cai, D. Budden, T. Paine, C. Gulcehre, Z. Wang, T. Pfaff, T. Pohlen, Y. Wu, D. Yogatama, J. Cohen, K. McKinney, O. Smith, T. Schaul, T. Lillicrap, C. Apps, K. Kavukcuoglu, D. Hassabis, and D. Silver, "AlphaStar: Mastering the Real-Time Strategy Game StarCraft II," https://deepmind.com/blog/alphastar-mastering-real-time-strategy-game-starcraft-ii/, 2019. 1.1, 1.3

[15] J. McCarthy and E. A. Feigenbaum, "In memoriam: Arthur samuel: Pioneer in machine learning," *AI Magazine*, vol. 11, no. 3, pp. 10–10, 1990. 1.2

[16] C. M. Bishop *et al.*, *Neural networks for pattern recognition*. Oxford university press, 1995. 1.2

[17] L. Davis, "Handbook of genetic algorithms," 1991. 1.2

[18] "OpenGL API Documentation Overview," https://www.opengl.org/, [Online; accessed 01-July-2019]. 2.3.2

[19] "The freeglut Project," http://freeglut.sourceforge.net/, [Online; accessed 01-July-2019]. 2.3.2

[20] "NumPy," https://www.numpy.org/, [Online; accessed 01-July-2019]. 2.3.2

[21] "pickle — Python object serialization — Python 3.7.4rc2 documentation," https://docs.python.org/3/library/pickle.html, [Online; accessed 01-July-2019]. 2.3.2

[22] "PyCharm: the Python IDE for Professional Developers by JetBrains," https://www.jetbrains.com/pycharm/, [Online; accessed 01-July-2019]. 2.3.3

[23] T. Rashid, *Make your own neural network*. CreateSpace Independent Publishing Platform, 2016. 2.4.1, 2.4.1, 2.4.1

[24] "La sinapsis," https://es.khanacademy.org/science/biology/human-biology/neuron-nervous-system/a/the-synapse, [Online; accessed 01-July-2019]. 2.2

[25] M. Nielsen, "Using neural nets to recognize handwritten digits," *Neural Networks and Deep Learning*, 2015. 2.4.1

[26] M. Agirregabiria, "Curva sigmoidea: El secreto de la vida y la felicidad," https://blog.agirregabiria.net/2012/02/curva-sigmoidea-el-secreto-de-la-vida-y.html, 2012, [Online; accessed 01-July-2019]. 2.4

[27] M. Isard and A. Blake, "Condensation—conditional density propagation for visual tracking," *International journal of computer vision*, vol. 29, no. 1, pp. 5–28, 1998. 2.4.2, 2.4.2

[28] L. J. Rodrıguez-Aragón, “Simulación, método de montecarlo,” 2011. 2.4.2

[29] M. S. Sharifian, A. Rahimi, and N. Pariz, “Classifying the weights of particle filters in nonlinear systems,” *Communications in Nonlinear Science and Numerical Simulation*, vol. 31, no. 1-3, pp. 69–75, 2016. 2.6

[30] F. Kaelin, “Ecse-626 project: An adaptive color-based particle filter.” 2.4.2

[31] J. Crespí, “Utilización de técnicas de Machine learning en un videojuego de carreras,” https://github.com/Jaicomp/TFG-Race-Circuit-Neural-Networks, 2019, [Online; accessed 01-July-2019]. 5.1.3, 5.2.3