Universidad de Alcalá
Escuela Politécnica Superior

Grado en Ingeniería en Electrónica y Automática Industrial

Trabajo Fin de Grado

"Conducción autónoma de un vehículo en un simulador mediante CNN"

Autor: Javier del Egido Sierra

Tutor/es: Luis Miguel Bergasa Pascual

2018

UNIVERSIDAD DE ALCALÁ

Escuela Politécnica Superior

GRADO EN INGENIERÍA EN ELECTRÓNICA Y AUTOMÁTICA INDUSTRIAL

Trabajo Fin de Grado

"Conducción autónoma de un vehículo en un simulador mediante CNN"

Javier del Egido Sierra

2018

UNIVERSIDAD DE ALCALÁ

Escuela Politécnica Superior

GRADO EN INGENIERÍA EN ELECTRÓNICA Y AUTOMÁTICA INDUSTRIAL

Trabajo Fin de Grado

"Conducción autónoma de un vehículo en un simulador mediante CNN"

Autor: Javier del Egido Sierra

Tutor/es: Luis Miguel Bergasa Pascual

TRIBUNAL:

Presidente: Rafael Barea Navarro

Vocal 1º: Pablo Ramos Sainz

Vocal 2º: Luis Miguel Bergasa Pascual

Calificación: .....

Fecha: 17-07-2018

“El presente es suyo; el futuro, por el que tanto he trabajado, es mío.”

Nikola Tesla

Agradecimientos

La finalización de este Trabajo Fin de Grado supone el fin de una etapa maravillosa, llena de obstáculos y sorpresas, disfrute y sacrificio, que no hace de mí sino una mejor persona.

En primer lugar, me gustaría agradecer la oportunidad brindada por mi tutor Luis Miguel Bergasa de realizar un trabajo realmente interesante en un campo antes desconocido para mí y que ahora admiro.

A mi entorno, por apoyarme y ayudarme a lograr mis metas.

A mis amigos, por acompañarme en esta travesía y sacarme siempre una sonrisa en los momentos de mayor estrés.

A mi novia Rocío, el mejor regalo que me llevo de mi paso por la Universidad, por ser una gran ingeniera y mejor persona.

Y, por último, pero no menos importante, a mi familia, por crear el ambiente idóneo para lograr llegar hasta aquí, inculcarme el espíritu de la curiosidad e ilustrarme en mis decisiones.

Índice general

Índice de figuras

Índice de tablas

Resumen

Este trabajo elabora una comparativa entre distintos modelos de Redes Neuronales Convolucionales (CNN - *Convolutional Neural Network)*, comprobando su desempeño a la hora de controlar de forma autónoma un vehículo en un entorno simulado. Para ello se obtienen datos de conducción de dicho entorno mediante conducción manual como valor verdadero (*ground-truth*), se elaboran distintos modelos de red neuronal con diversos niveles de complejidad y se entrenan con los datos previamente obtenidos usando técnicas de aprendizaje profundo fin a fin (*End-to-End*).

Una vez entrenadas dichas redes se ponen a prueba en el simulador de conducción, comprobando la capacidad de mantener el vehículo próximo al centro del carril y su ángulo de cabeceo. Las redes neuronales serán evaluadas en función a dichos parámetros.

Finalmente, se extraerán conclusiones del funcionamiento de los distintos modelos en base a los parámetros indicados con el objeto de encontrar la CNN óptima para la aplicación desarrollada.

Conceptos clave

Convolutional Neural Network, conducción autónoma, red neuronal convolucional.

Abstract

This work makes a comparison between different Convolutional Neural Network models, testing its performance when it leads a self-driving car in a simulated environment. To do so, driving data has been obtained manually driving the simulator as ground truth and different network models with diverse complexity levels has been created and trained with the data previously obtained using end-to-end deep learning techniques.

Once this CNNs are trained, they are tested in the driving simulator, checking their ability of keeping the car near to the center of the road and its heading error. The neural networks will be evaluated according to these parameters.

Finally, conclusions will be drawn about the performance of the different models according to the parameters mentioned before in order to find the optimum CNN for the developed application.

Key concepts

Convolutional Neural Network (CNN), self-driving.

Resumen extendido

La tecnología de redes neuronales es relativamente nueva, ya que comenzó a desarrollarse a mediados del siglo XX. Sin embargo, la rápida evolución de la capacidad de procesamiento de los computadores modernos y la capacidad de obtener datos masivos (big data) ha hecho evolucionar rápidamente esta tecnología hacia lo que se conoce como redes profundas (Deep networks), logrando unos resultados superiores a otras técnicas del estado del arte y demostrando sus capacidades en multitud de áreas de trabajo.

A la hora de elaborar una red neuronal profunda para realizar una tarea, el abanico de posibilidades de configuración de la misma es muy amplio, tanto en la arquitectura de la red como en el método de entrenamiento.

En la elaboración de este Trabajo Fin de Grado se pretende establecer una comparación entre las diferentes alternativas posibles con el fin de obtener la configuración de red neuronal profunda que mejor controle un vehículo de forma autónoma.

Puesto que la información de entrada son imágenes, la red será de tipo convolucional, ya que las capas que conforman su arquitectura están optimizadas para el tratamiento de éstas.

El método de trabajo escogido es el entrenamiento supervisado, puesto que las redes aprenderán qué valores de salida producir ante determinadas entradas a partir de un conjunto de datos recabados previamente en el mismo simulador de forma manual (ground truth), lo que se conoce como un aprendizaje End-to-End.

Inicialmente, como primera aproximación, se elaborarán modelos de red neuronal convolucional (CNN) que controlen exclusivamente el ángulo de giro del volante, manteniendo la velocidad del vehículo constante en todo momento tomando como base la CNN desarrollada por NVIDIA en 2016 [1]. Se introducirán una serie de modificaciones sobre la arquitectura base para comprobar el efecto de las mismas y concluir qué cambios contribuyen a producir mejoras en el comportamiento y cuáles lo empeoran.

Posteriormente se probarán dos arquitecturas distintas para controlar el ángulo de volante y el valor de la aceleración del vehículo en cada momento, probando en ambos casos las modificaciones que resultaron exitosas en el control de volante.

La primera de ellas consiste en elaborar una red que controle el volante y otra red con la misma arquitectura que controle el acelerador, siendo sus valores de salida totalmente independientes.

La segunda arquitectura consiste en que una única CNN controle los valores de volante y acelerador simultáneamente, de forma que sus valores influyan entre sí al ser producidos conjuntamente.

Para evaluar la idoneidad de las redes se establecerán dos parámetros de calidad novedosos que permitirán establecer comparaciones cuantitativas.

El primero de ellos es la distancia que se produce entre el vehículo y el centro de la carretera por la que debe circular. Cuando un ser humano controla un vehículo intenta mantenerlo lo más centrado posible, por lo que se busca que esta CNN conduzca de forma similar.

El otro parámetro es el ángulo de cabeceo, que es el ángulo que se produce entre la dirección que lleva el vehículo y la dirección que sigue la carretera. Si el vehículo conduce siguiendo la carretera, aunque se presente una curva, el ángulo debería ser mínimo.

Finalmente se extraerán las conclusiones, determinando qué modificaciones contribuyen a mejorar la conducción del vehículo en pos de lograr una conducción lo más parecida a la conducción humana posible.

Capítulo 1.
Introducción. Estado del arte

En la antigüedad, pequeños desplazamientos locales suponían días de esfuerzo caminando. Posteriormente, con el invento de la rueda y el dominio de los animales se consiguió un vehículo capaz de desplazarse a mayor velocidad, reduciendo significativamente la duración de los viajes y transformando progresivamente las relaciones comerciales entre países.

Con la invención del automóvil de la mano de Karl Friedrich Benz a finales del siglo XIX y su producción en masa gracias a Henry Ford en los inicios del siglo XX la movilidad pasó a ser algo asequible para la población, que ahora podía recorrer distancias que antes supondrían meses de esfuerzo en un solo día.

Sin embargo, la muerte por accidentes de tráfico creció junto con el desarrollo del automóvil hasta convertirse en una de las principales causas de muerte de la actualidad.

Aunque parte de estos accidentes se producen debido a fallos mecánicos en el vehículo, la mayoría se deben a distracciones e imprudencias cometidas a diario por miles de conductores.

Con el objetivo de solventar este problema, disminuyendo la tasa de accidentes y aumentando la eficiencia en la conducción se desarrolla desde finales del siglo XX el vehículo autónomo.

El vehículo autónomo es un medio de transporte capaz de desplazarse sin la intervención humana, aprendiendo las normas de circulación y replicándolas. Para ello necesitan conocer el entorno en el que se desenvuelven, lo que se consigue empleando herramientas como radar, LIDAR, sistemas de posicionamiento GPS y cámaras.

En la actualidad diversas empresas compiten por conseguir desarrollar la tecnología capaz de desplazarse entre dos localizaciones haciendo frente a las situaciones de tráfico real que pudieran suceder frente a él, tales como otros vehículos circulando, peatones, bicicletas, actuaciones temerarias de otros conductores, etc.

Para clasificar los diferentes sistemas desarrollados hasta alcanzar la conducción totalmente autónoma se dispone del baremo elaborado por la Sociedad de Ingenieros Automotrices (SAE), que distingue los siguientes niveles:

- Nivel 0. Sin asistencia. El vehículo tradicional controlado por un conductor.
- Nivel 1. El vehículo controla bajo ciertas condiciones el movimiento longitudinal o lateral.
- Nivel 2. El vehículo puede controlar el movimiento longitudinal y lateral, pero no puede detectar eventualidades, que debe prevenir el conductor.
- Nivel 3. El vehículo se controla de manera autónoma en su totalidad y responde ante eventualidades, pero el conductor debe estar listo para intervenir si el sistema lo necesita.
- Nivel 4. El vehículo no necesita conductor, ya que se desplaza autónomamente y responde ante posibles fallos del sistema para minimizar el riesgo, pero el sistema solo funciona en ciertas condiciones.
- Nivel 5. El vehículo funciona autónomamente bajo todo tipo de situaciones, eliminando la necesidad de un conductor que supervise el funcionamiento incluso ante eventualidades.

Mientras se desarrolla la tecnología capaz de alcanzar la total autonomía de la conducción, ya se implementan en la actualidad sistemas de asistencia al conductor con distintos niveles de funcionamiento autónomo, alcanzando el nivel 3 de autonomía en vehículos como los fabricados por Tesla, que emplean la tecnología Autopilot [2].

Este sistema, como muchos otros desarrollados de forma puntera en la actualidad, está comandado por una red neuronal profunda, la cual recibe información del entorno mediante cámaras, radar, sonar o lidar. A partir de esos datos, la red produce unos valores de salida para actuar sobre los movimientos realizados por el vehículo.

Las decisiones realizadas por la red neuronal a la hora de comandar el vehículo vienen determinadas por los datos empleados durante el entrenamiento de la misma, por lo que cuanto mayor sea el banco de datos empleado, mejor desempeño se espera de la red, puesto que conocerá qué acciones realizar en base a situaciones parecidas vistas anteriormente.

La motivación de este Trabajo Fin de Grado es el análisis de la calidad de la conducción de estas CNN y concluir qué modificaciones producen que la conducción sea más parecida a la realizada por un humano.

Las modificaciones a analizar comprenden ámbitos desde diferentes arquitecturas de red, incluyendo diversos tipos de capas, hasta la modificación de la información de entrada de la CNN, estudiando distintos métodos de data augmentation.

El objetivo final del estudio es el de elaborar una red neuronal convolucional que sea capaz de imitar el comportamiento de un humano controlando un vehículo, comparando los resultados obtenidos con los de otras tecnologías del estado del arte.

Para la conclusión de estos objetivos se realizará un trabajo dividido en distintos capítulos que tratarán los temas que se indica a continuación.

- Capítulo 1. Introducción. Estado del arte.
- Capítulo 2. Redes neuronales convolucionales (CNN).
- Capítulo 3. Simulador.
- Capítulo 4. Arquitecturas de CNN implementadas.
- Capítulo 5. Métrica de evaluación.
- Capítulo 6. Resultados experimentales.
- Capítulo 7. Conclusiones generales y trabajos futuros.

Capítulo 2.
Redes neuronales convolucionales (CNN)

En este capítulo se presenta una introducción a las redes neuronales artificiales, su evolución y los principios teóricos en los que se basa su funcionamiento. El capítulo no pretende realizar un análisis exhaustivo de la temática debido a su gran complejidad, sino introducir al lector en el campo de trabajo que se utilizará en capítulos posteriores.

2.1 Historia de las redes neuronales convolucionales

Tradicionalmente la computación ha estado basada en la ejecución secuencial de una serie de órdenes previamente establecidas e inamovibles. Dichas órdenes debían contemplar todos los sucesos susceptibles de producirse para poder actuar frente a ellos.

Estos modelos convencionales de programación son demasiado rígidos y se pueden volver demasiado complejos a medida que el escenario de funcionamiento es mayor.

Sin embargo, esta no es la manera que tienen los seres vivos de desenvolverse en su entorno. En la naturaleza, los animales deciden sus acciones basándose en experiencias adquiridas previamente durante su vivencia.

Siguiendo esta filosofía, se desarrolla desde mediados del siglo XX el concepto de red neuronal artificial, intentando replicar el funcionamiento del cerebro animal a través de llamadas calculadoras en el Massachusetts Institute of Technology, definiendo la unidad básica que forman las redes neuronales, el perceptrón, equivalente a una neurona biológica.

El perceptrón recibe información de distintas entradas, equivalente a las dendritas, multiplicando dicha información de entrada por una variable llamada peso. El cuerpo del perceptrón suma toda la información recibida, y tras pasar por una función de activación en la salida, equivalente al axón de las neuronas, se produce un valor de salida, que normalmente se introducirá como entrada a otro perceptrón, creando una compleja red neuronal.

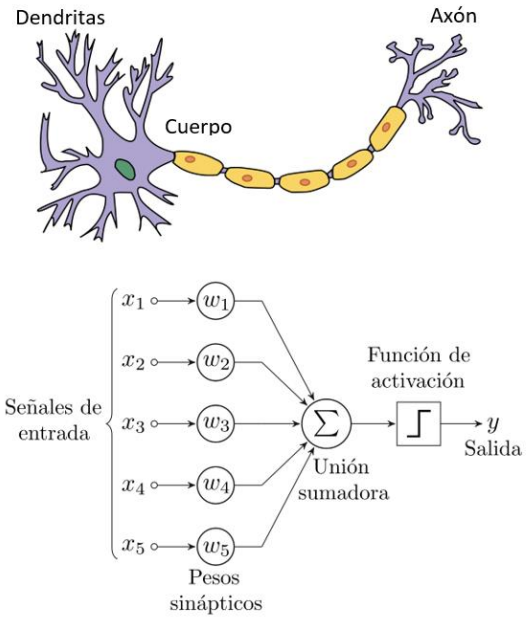

Ilustración 1 Comparación entre neurona biológica y perceptrón

Los valores de los pesos de los perceptrones son modificados durante un proceso de entrenamiento llamado propagación hacia atrás (backpropagation).

Este entrenamiento se realiza introduciendo un patrón de entrada y, tras pasar por el conjunto de la red, comparar la salida producida con la salida deseada para dicho patrón, reajustando los valores de los pesos de la red neuronal.

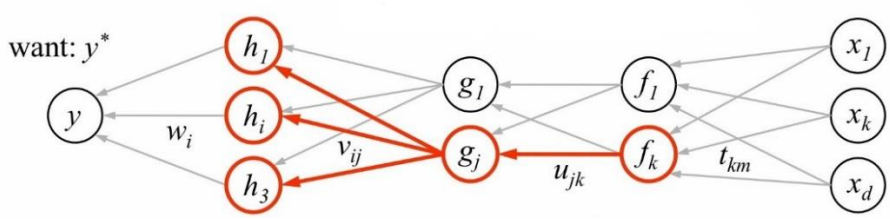

Ilustración 2 Esquema de cálculo de backpropagation

Repitiendo el proceso múltiples veces la red es capaz de autoajustarse para reconocer los valores que se desean para entradas similares a los patrones empleados.

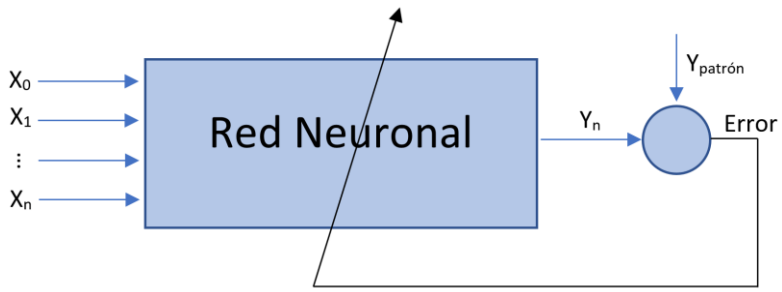

Ilustración 3 Entrenamiento de red neuronal mediante backpropagation

2.2 Aprendizaje profundo

Para conseguir mejores resultados se emplean redes de aprendizaje profundo (Deep Neural Network, DNN). Éstas son redes mucho más complejas, estando compuestas por diferentes capas ocultas entre la capa de entrada y la de salida, permitiendo modelar relaciones no lineales.

Su desarrollo se produce a partir de 2012 con el aumento de la potencia de procesamiento y la capacidad de obtener datos masivos (big data) para su entrenamiento, en el que se pueden emplear tarjetas de procesamiento gráfico o GPU para agilizar el proceso.

2.3 Redes neuronales convolucionales

Un tipo de redes de aprendizaje profundo son las redes neuronales convolucionales. Éstas son un tipo de red especializada en el procesamiento de imágenes como información de entrada, modelando no linealidades mediante sus diferentes capas.

La dimensión de la capa de entrada viene dada por la imagen que se transmite a la red, habitualmente $W \times H \times C_{in}$, siendo C_{in} el número de canales de la imagen. Siguiendo la evolución de la información a través de las distintas capas, el procedimiento más común produce una disminución de los valores $W \times H$, mientras que la profundidad de la red C_{in} aumenta. Por lo tanto, las primeras capas producen una información localizada (dónde), mientras que las capas finales indican información sobre el contenido de la imagen (qué), tal y como se observa en el trabajo [10].

Las imágenes de entrada suelen tener dimensiones $W \times H$, además de tres canales de color, o uno solo si se trata de escala de grises. Mediante entrenamiento end-to-end la CNN procesa las imágenes y aprende a obtener resultados similares a los proporcionados en el entrenamiento.

La arquitectura de estas redes está basada en el funcionamiento de la corteza visual del cerebro, y se encuentra formada principalmente por capas convolucionales, capas de submuestreo o pooling, capas completamente conectadas y de normalización.

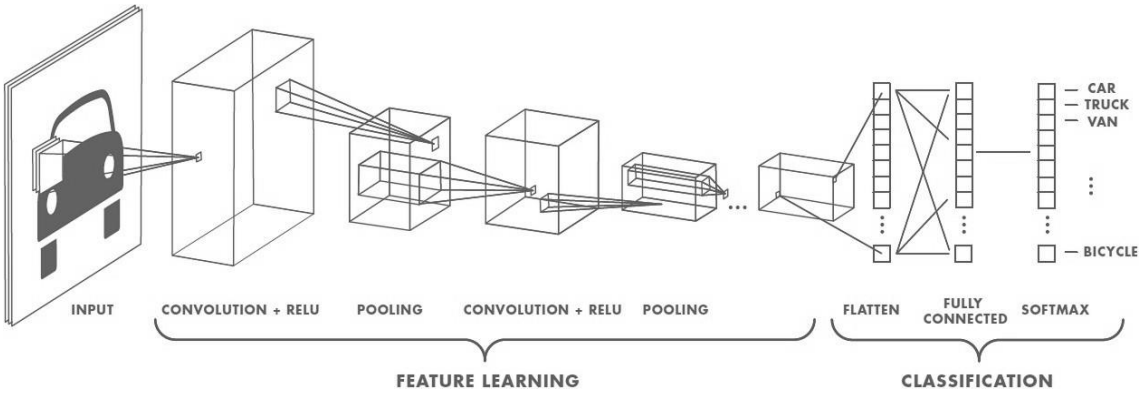

Ilustración 4 Red neuronal convolucional diseñada para distinguir vehículos

2.3.1 Capas convolucionales

Las capas convolucionales son la base de las redes neuronales convolucionales. En las redes neuronales clásicas cada neurona de una capa conecta con todas las neuronas de la capa anterior, y la salida se calcula como el producto escalar del vector de pesos por el vector de salidas de la capa anterior. Sin embargo, en las CNNs cada píxel de una capa conecta solo con un grupo reducido de píxeles de la capa anterior, y los pesos que afectan a estas conexiones se comparten por todos los píxeles de un mapa de características.

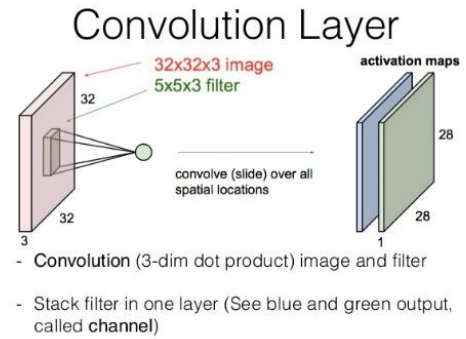

Ilustración 5 Ilustración de capa convolucional

El funcionamiento es similar a emplear un filtro, calculando el producto escalar de la ventana deslizante en toda la profundidad del tensor de la capa anterior. Estos filtros son rectangulares (normalmente cuadrados), y su profundidad viene determinada por la profundidad de la capa anterior. Las capas convolucionales vienen determinadas por los siguientes parámetros:

- Las dimensiones de los filtros de convolución, generalmente cuadrados, de tamaño *F x F*. Este será el campo de visión de la capa, ya que cada píxel de cada mapa de características conectará solo con los píxeles que se encuentren dentro del filtro.

- El número de filtros de convolución C_{out}, que determina la profundidad del tensor de salida. Cada filtro genera un canal de salida. Los canales también se denominan mapas de características puesto que cada filtro localiza diferentes características en la imagen.

- Stride, S, que determina cada cuántos píxeles se desliza la ventana sobre el tensor de entrada. El valor más común de stride es 1.

- Padding, P, indica cuántos píxeles de relleno se añaden en los bordes del tensor de entrada, rellenándose con ceros. El padding permite controlar las dimensiones del tensor de salida.

Las relaciones de estos parámetros indican que las dimensiones del vector de salida W_{out} x H_{out} x C_{out} se calcularían como $Wout = (Win - F + 2P)/S + 1$ y $Hout = (Hin - F + 2P)/S + 1$. Si el valor de $P = \lfloor (F-1)/2 \rfloor$ con S=1 se conservarán las dimensiones espaciales de la capa anterior. La profundidad de salida C_{out} es independiente de la de entrada C_{in}, pero ésta determina la profundidad de los filtros a emplear.

2.3.2 Capas de submuestreo

Las capas de submuestreo o pooling se emplean para reducir las dimensiones de los tensores de la red. Esto sirve para crear sistemas más robustos frente a desplazamientos de los objetos, ya que una vez detectada una característica no es necesario conocer su localización en la imagen. Éstas capas también producen una disminución del tamaño de los mapas de características, lo que deriva en un menor consumo de memoria durante el entrenamiento de las redes.

Al igual que las capas de convolución, el funcionamiento se basa en una ventana deslizante sobre el tensor de la capa anterior. Sin embargo, las capas de submuestreo no emplean padding, y el procesado se realiza independientemente en cada canal de entrada, por lo que la profundidad de salida coincide con la de entrada. La operación realizada por la ventana deslizante depende de el tipo de submuestreo, siendo los más empleados el average pooling y max pooling: el primerio calcula cada píxel de cada canal del tensor de salida realizando el promedio de los píxeles que se encuentran dentro de la ventana en el canal correspondiente del tensor de entrada, mientras que el segundo toma el máximo de los valores de la ventana.

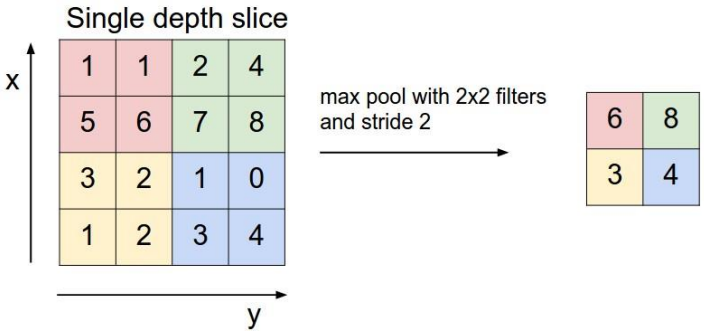

Ilustración 6 Ilustración de capa max pooling

2.3.3 Capas completamente conectadas

Estas capas derivan de la estructura tradicional de red neuronal artificial, conectando cada neurona de la capa de entrada con cada neurona de la capa de salida y asignando un peso a cada una de estas conexiones. El número de parámetros configurables de esta red es muy elevado debido a la gran cantidad de conexiones producidas.

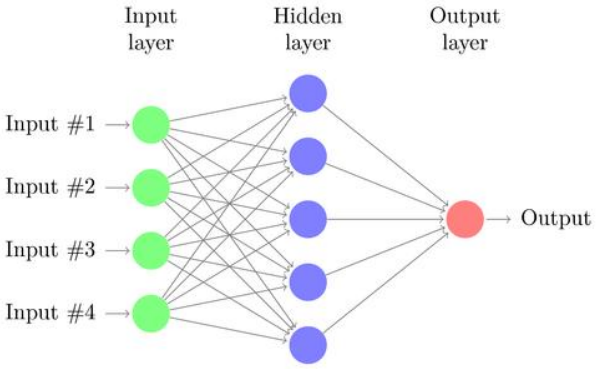

Ilustración 7 Ilustración de capa totalmente conectada

2.3.4 Capas LSTM

Para mejorar el funcionamiento de la red neuronal y aportar memoria a los datos de salida producidos con el objetivo de relacionar los valores futuros con los pasados se incorpora el uso de capas Long Short-Term Memory o LSTM.

Estas capas se incorporan en la parte final de la red. Durante el aprendizaje los datos son proporcionados de forma consecutiva, no aleatoria, para permitir la relación entre un valor y los aportados anteriormente. Durante la prueba, los datos que la red aporta como salida son función de la imagen de entrada y de los valores producidos con anterioridad, aportando estabilidad a estos valores.

En la ilustración siguiente se puede apreciar cómo la salida en el instante anterior $h_{(t-1)}$ influye junto con la entrada actual x_t para producir el valor de salida h_t.

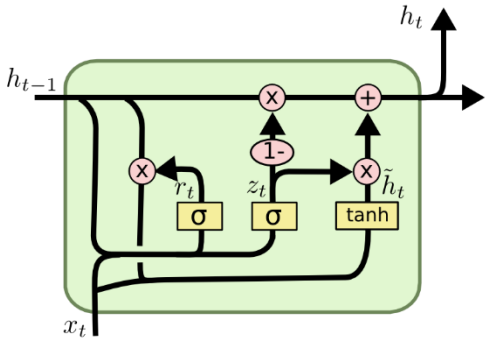

Ilustración 8 Ilustración de celda LSTM

2.3.5 Funciones de activación

Al igual que se realiza en las redes neuronales clásicas, es necesario procesar la salida de cada neurona mediante una función de activación no lineal para conseguir modelos no lineales. La función de activación actúa elemento a elemento del tensor de salida de la capa anterior. En el caso de las CNNs se emplea mayormente la función ReLU, que se implementa de forma sencilla como una operación máx $(0, xijk)$, para todo punto x_{ijk} con i, j, k ϵ $[0, W - 1] \times [0, H - 1] \times [0, Cin - 1]$ en un tensor.

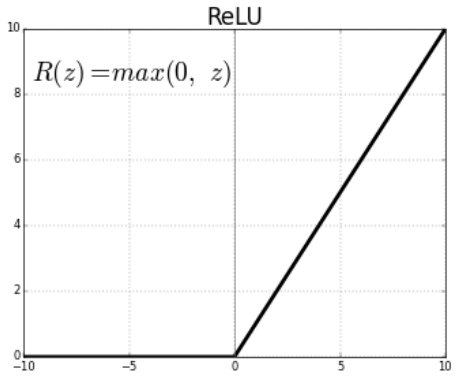

Ilustración 9 Representación gráfica de función de activación ReLU

2.3.6 Obtención de la salida

En las redes tradicionales creadas para clasificación, el mapa de características más profundo es aplanado en un vector utilizado para las últimas capas fully connected, que actúan como un MLP (perteptrón multicapa). También es posible utilizar las características extraídas como entrada para otro clasificador.

2.3.7 Entrenamiento de la red

Las redes neuronales se entrenan de forma supervisada minimizando una determinada función de pérdidas, que mide el error existente entre la estimación producida por la red y el ground truth.

La forma de entrenar redes neuronales profundas es mediante el descenso de gradiente estocástico (SGD), una variante del algoritmo de optimización de descenso de gradiente empleado en redes neuronales clásicas, en el cual se actualizan los pesos de la red iterativamente en la dirección de máxima pendiente de la función de pérdidas, para minimizarla. La simplicidad del método es una de las claves de su éxito, ya que emplear cálculos más complejos en el entrenamiento de redes profundas no sería viable.

$$Loss = \frac{1}{T}\sum_{t=1}^{T} \|\hat{y} - y\|^2$$

La actualización de los pesos de la red se realiza empleando backpropagation, una implementación eficiente de la regla de la cadena para calcular gradientes de funciones profundamente compuestas. Cada vuelta del entrenamiento tiene dos fases:

- Forward pass. Consiste en estimar una salida para la imagen de entrada actual, computando en cascada las activaciones intermedias, y obteniendo el valor de la función de pérdidas comparando el valor producido con el ground truth.

- Backward pass. Es la etapa de aprendizaje. Se computa el gradiente de la función de pérdidas respecto a la salida de la red, y se propaga hacia las capas anteriores para actualizar los pesos de las distintas convoluciones y otros parámetros configurables, empleando para ellos dos expresiones [11]:

    - Para obtener el gradiente asociado a los pesos de una capa d, se derivan sus salidas respecto cada uno de los pesos que participan en su cálculo, multiplicando este valor por el gradiente propagado desde la capa superior d+1.

    - Para propagar el gradiente hacia la capa inferior d-1 se derivan las salidas de d respecto a cada salida de d-1 que influya en su valor, multiplicando también por el gradiente propagado desde la capa d+1 para respetar la regla de la cadena. Cada salida d-1 acumula las aportaciones de todas las salidas de la capa d en las que ha influido, obteniendo el gradiente propagado.

En una CNN los pesos que calculan cada canal son compartidos, por lo que cuando un píxel de salida actualiza sus pesos, en realidad actualiza el mismo conjunto compartido, por lo que en lugar de realizar múltiples actualizaciones, se acumula el gradiente para todas las salidas y cada filtro tiene un único gradiente asociado.

Además, una iteración del algoritmo de descenso de gradiente suele comprender varias pasadas de entrenamiento según el tamaño del batch definido. Así se estima la esperanza matemática del gradiente global a partir de promediar los gradientes obtenidos para un subconjunto del dataset de entrenamiento que va cambiando aleatoriamente en cada epoch. Ésta es la principal novedad que aporta el SGD respecto al descenso de gradiente clásico, que promedia sobre todo el conjunto de entrenamiento.

2.4 Trabajo previo

Desde la aparición de las redes neuronales convolucionales, y especialmente tras su auge con la difusión de grandes bases de datos etiquetadas y las GPU modernas, y los excelentes resultados cosechados por las redes AlexNet [3] e ImageNet [4], las CNNs han sido objeto de investigación y continua mejora, elaborando progresivamente modelos cada vez más profundos y con mayor capacidad de aprendizaje.

2.4.1 Arquitecturas básicas para clasificación

AlexNet [3]. Supone el punto de partida de las CNNs actuales, aplicada a ImageNet [12], con tan solo 8 capas con pesos entrenables: 5 capas convolucionales y 3 fully connected para proporcional una salida de clasificación a nivel de imagen.

ZFNet [5]. La novedad que introduce este trabajo reside en el mecanismo que permite visualizar con capas inversas las regiones más destacadas por la red. Se observa cómo los fitros iniciales se especializan en detección de bordes y colores, mientras que los filtros más profundos se activan con características más semánticas.

VGG [6]. Con esta red se proponen cada vez arquitecturas más profundas, empleando filtros más reducidos en mayor número de capas, hasta las 19 entrenables. Esto permite un aumento del no linealidades entre filtros, mejorando la capacidad semántica de la red y reduciendo el número de pesos en proporción.

ResNet [7]. Como novedad introduce las shortcut connections en su arquitectura. Sigue la línea de VGG buscando redes con filtros de pequeño tamaño. Su arquitectura es periódica y modular, basada en un bloque sencillo con dos o tres capas convolucionales y una shortcut connection que puentea la entrada y la suma a la salida. Este bloque se repite varias veces dentro de una etapa.

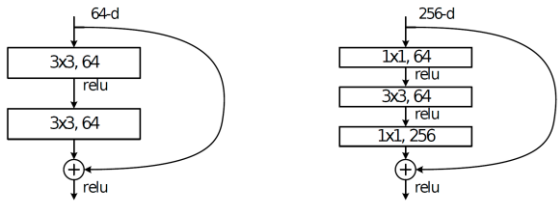

Ilustración 10 Bloques que conforman la arquitectura ResNet

2.4.2 Arquitecturas básicas para segmentación

Fully Convolutional Networks (FCNs) [8]. Convierte modelos de clasificación en modelos de segmentación semántica. Su nombre se debe a que las últimas capas de la red de clasificación, del tipo fully connected, se convierten a capas convolucionales con el mismo número de pesos que la capa original.

Full-Resolution Residual Networks (FRRNs) [9]. Desarrolla un arquitectura novedosa con dos flujos de datos: uno a resolución completa y otro con submuestreo para combinar características locales para detectar fronteras de objetos, con características globales para identificar los objetos presentes.

2.4.3 Herramientas software

Ligado al desarrollo de arquitecturas de red se produce la aparición de diversos mecanismos software que emplear para codificar dichas redes y entrenarlas. A continuación, se indican algunas de las más conocidas:

TensorFlow [10]. Biblioteca para machine learning desarrollada por Google, orientada al manejo de tensores. La arquitectura se define en forma de DAG.

MATLAB [11]. Herramienta para cálculo matemático matricial y lenguaje de programación. Contiene toolboxes dedicadas a redes neuronales, visión artificial y deep learning.

Caffe [12]. Framework escrito en C/C++ para implementaciones en CPU, o empleando CUDA para su uso en GPU. Posee una amplia biblioteca de funciones para definir redes, principalmente CNNs, entrenarlas y testearlas.

PyTorch [13]. Biblioteca para manejo de tensores empleando CPU y GPU. Proporciona una interfaz en Python al paquete torch, lo que dota de mayor funcionalidad y mayor universalidad al sistema.

En función de las necesidades específicas a desarrollar y de la capacidad del sistema de entrenamiento serán más adecuadas unas herramientas u otras. Por ejemplo, para un entrenamiento más rápido es más útil un sistema compatible con el uso de GPUs.

Capítulo 3.
Simulador

Para entrenar una CNN es necesario conseguir una gran cantidad de datos etiquetados. En el caso de este Trabajo Fin de Grado sería necesario conseguir imágenes vinculadas con el valor de ángulo de volante y de acelerador de forma instantánea. Debido a la dificultad para conseguir estos valores se decide emplear un simulador de conducción que permita recolectar estos datos de forma sencilla.

A la hora de elegir el simulador adecuado, un parámetro fundamental es que éste haya sido creado bajo licencia de código abierto, ya que será necesario adaptar el simulador elegido a la situación particular de la investigación.

Finalmente, el simulador empleado será el Udacity's Self-Driving Car Simulator, creado por la plataforma Udacity como medio para impartir su curso online sobre vehículos autónomos, que cumple el requisito de código abierto y además está enfocado al entrenamiento y puesta a prueba de algoritmos de conducción autónoma.

El simulador dispone de un circuito por el que transcurrirá el vehículo de forma manual y autónoma, que se recorre en sentido contrario a las agujas del reloj.

Ilustración 11 Imagen cenital del circuito en el simulador

Las diversas modificaciones que se han realizado tanto en la plataforma Unity, en la que se creó el proyecto originalmente, como en sus archivos de código, se indican en el apartado "Modificaciones introducidas en el simulador".

El simulador incluye una interfaz visual, en la que se incluyen distintos modos de conducción: manual o autónoma.

3.1 Conducción manual y modo de entrenamiento

En el modo de conducción manual es un usuario el que controla el vehículo a lo largo de una pista diseñada, pudiendo almacenar los datos de dicha conducción para el entrenamiento de la Convolutional Neural Network que posteriormente controlará el coche mediante el modo entrenamiento.

Ilustración 12 Interfaz gráfica de conducción manual

Los datos obtenidos en el modo de entrenamiento contienen información de tres cámaras ubicadas en el salpicadero del vehículo, concretamente en el lateral izquierdo, el centro y el lateral derecho. También se almacenan los datos de ángulo del volante, aceleración, freno, velocidad, posición y rotación.

center	left	right	steering	throttle	brake	speed
C:\Users\jds96\OneDriv	C:\Users\jds96\OneDriv	C:\Users\jds96\OneDriv	-0.2122216	0.8741225	0	21.06124
C:\Users\jds96\OneDriv	C:\Users\jds96\OneDriv	C:\Users\jds96\OneDriv	-0.2122216	0.8741225	0	21.72626
C:\Users\jds96\OneDriv	C:\Users\jds96\OneDriv	C:\Users\jds96\OneDriv	-0.2122216	0.8741225	0	22.38842
C:\Users\jds96\OneDriv	C:\Users\jds96\OneDriv	C:\Users\jds96\OneDriv	-0.2122216	0.8741225	0	22.83872
C:\Users\jds96\OneDriv	C:\Users\jds96\OneDriv	C:\Users\jds96\OneDriv	-0.2122216	0.8741225	0	23.50869
C:\Users\jds96\OneDriv	C:\Users\jds96\OneDriv	C:\Users\jds96\OneDriv	-0.2122216	0.8741225	0	24.1702
C:\Users\jds96\OneDriv	C:\Users\jds96\OneDriv	C:\Users\jds96\OneDriv	-0.2122216	0.8741225	0	24.82263

Tabla 1 Extracto de archivo .csv empleado para entrenamiento de las CNNs

3.2 Conducción autónoma

En el modo de conducción autónoma será una CNN, una vez entrenada, la que controlará el ángulo del volante y la aceleración del vehículo.

Ilustración 13 Interfaz gráfica de conducción autónoma

El simulador provee un archivo denominado drive.py encargado de establecer la comunicación entre el vehículo y la red neuronal, de forma que ésta recibe la imagen captada por la cámara central del vehículo y devuelve un valor de aceleración y ángulo de giro. También mostrará en la terminal en la que se ejecuta los valores de giro del volante, aceleración y velocidad que se produzcan en cada instante.

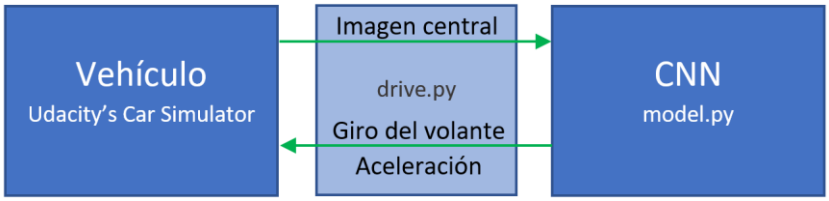

Ilustración 14 Comunicación entre simulador y CNN

3.3 Datos

Tanto en la extracción de datos mediante conducción manual como en la generación de datos en la conducción autónoma, los datos son expresados de una manera similar para corresponderse.

El ángulo del volante viene expresado en relación con el máximo giro posible (25º), obteniendo unos valores como los que se muestran en la tabla.

Datos de giro	Volante del vehículo
-1	-25 º
0	0 º
1	25 º

Tabla 2 Extracción de datos del volante

El valor del acelerador también se encuentra comprendido entre -1 y 1, siendo -1 la máxima potencia de frenado y 1 la máxima potencia de aceleración.

3.4 Modificaciones introducidas en el simulador

Para adaptar el simulador de conducción a nuestros requisitos ha sido necesario realizar una serie de modificaciones a través de la plataforma Unity.

- Velocidad. La velocidad era mostrada en millas por hora, por lo que se procedió a modificar el código para que fuera visualizada en kilómetros por hora, aportando una información más clara al usuario. Además, se ha aumentado el límite de velocidad de forma que el control del acelerador produzca cambios relevantes en el desempeño.

- Interacción. Para obtener datos de ángulo de giro más realistas se emplea un volante, Logitech Driving Force Pro, por lo que se configura en el simulador.

Ilustración 15 Volante Logitech Driving Force Pro

- Resolución. Determinados modelos de CNN requieren una resolución mayor, por lo que se modifica el valor de la resolución que capta el modo de entrenamiento. De esta forma obtenemos imágenes de 640*320 píxeles en formato JPEG.

- Extracción de datos en modo autónomo. Una vez se ha entrenado la CNN, ésta es puesta a prueba en el circuito, extrayendo los datos sobre su conducción mediante un archivo .csv (Comma-Separated Values) modificando el archivo drive.py, desarrollado en Python3.

3.5 Imágenes del simulador

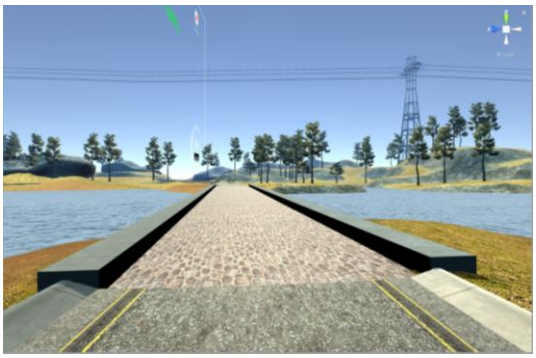

Ilustración 16 Imagen del puente del circuito

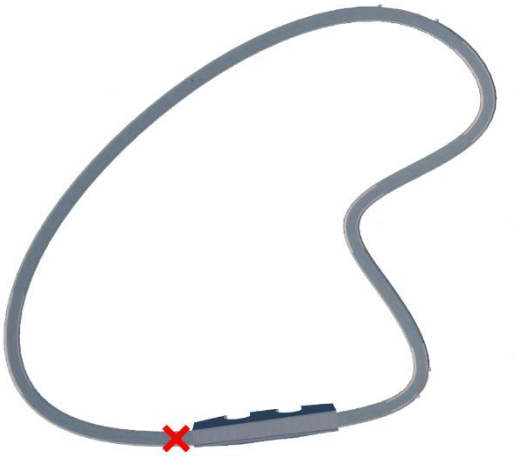

Ilustración 17 Posición de Ilustración 15 en el circuito

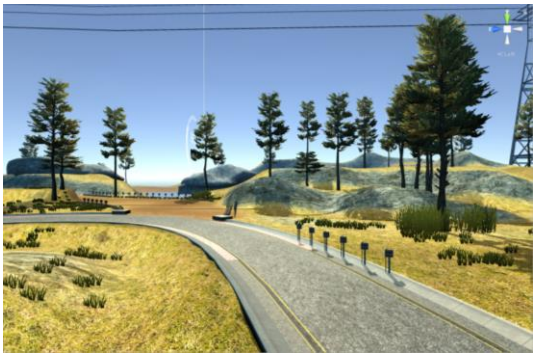

Ilustración 18 Imagen de la curva Zoom 1 del circuito

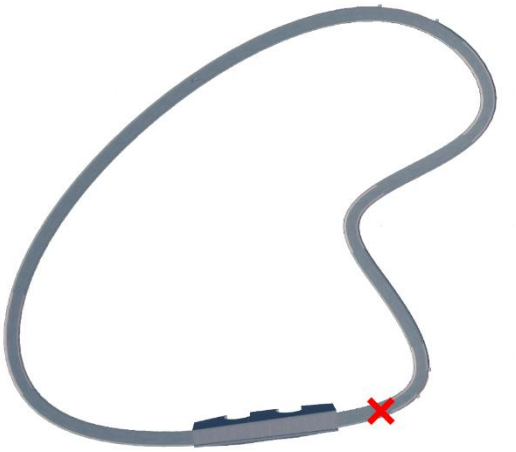

Ilustración 19 Posición de Ilustración 17 en el circuito

Ilustración 20 Imagen de la curva Zoom 2 del circuito

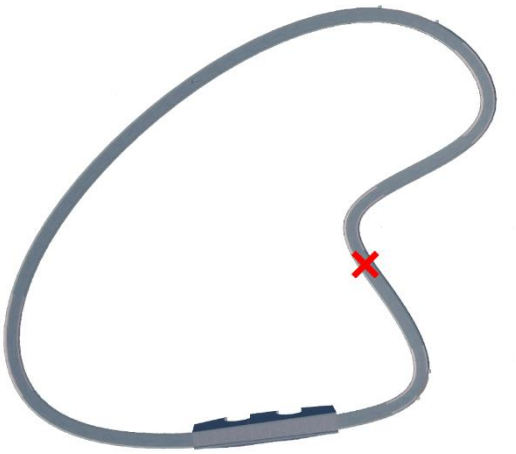

Ilustración 21 Posición de ilustración 19 en el circuito

Capítulo 4.
Arquitecturas de CNN implementadas.

Este Trabajo Fin de Grado pretende establecer comparaciones entre distintas arquitecturas de redes, añadiendo distintas complejidades y estudiando la mejora en el comportamiento ante estas modificaciones.

En este capítulo se exponen las diferentes arquitecturas y modificaciones sobre los datos de entrenamiento puestos a prueba.

Todas las redes neuronales empleadas son redes neuronales convolucionales construidas sobre Keras, una interfaz de programación de alto nivel especializada en redes neuronales [14].

4.1 Arquitecturas de red

En el desarrollo de esta investigación se han elaborado distintas arquitecturas de CNN, comprobando sus resultados sobre el simulador de conducción. En este punto se recogen las distintas arquitecturas implementadas.

4.1.1 TinyPilotNet

La arquitectura de esta red [15] deriva de la presentada por el equipo de NVIDIA, PilotNet [1], especializada en la conducción de un vehículo real mediante entrenamiento end-to-end.

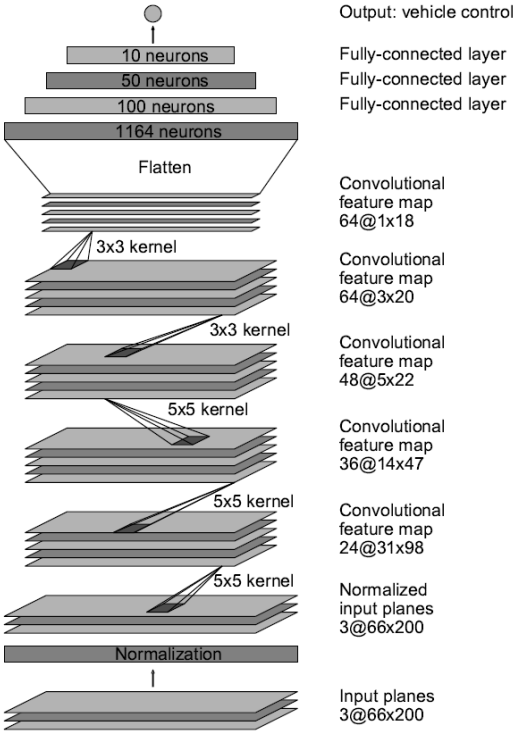

Ilustración 22 Arquitectura de PilotNet

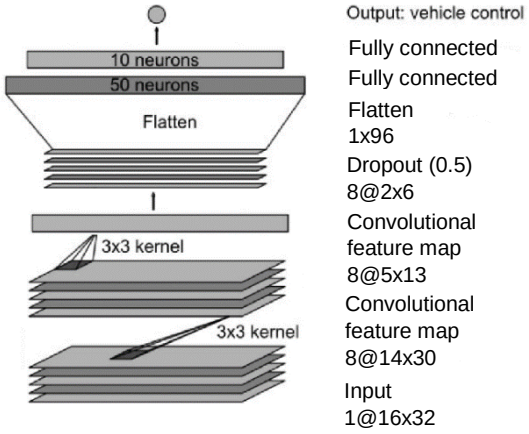

Ilustración 23 Arquitectura de TinyPilotNet

La arquitectura TinyPilotNet servirá de base para integrar las modificaciones realizadas en el estudio.

Se encuentra compuesta por una capa de entrada, en la que se introducirán imágenes de resolución 16x32 y un único canal, seguida por dos capas convolucionales de kernel 3x3, una capa dropout configurada al 50% de probabilidad para agilizar el entrenamiento. Finalmente, el tensor de información se convierte en un vector que es conectado a dos capas fully connected, produciendo la información de salida en una única neurona final.

4.1.2 LSTM-TinyPilotNet y derivadas

Partiendo de la arquitectura de TinyPilotNet se elaboran nuevas arquitecturas añadiendo diversas capas convolucionales y capas LSTM, obteniendo las arquitecturas descritas a continuación.

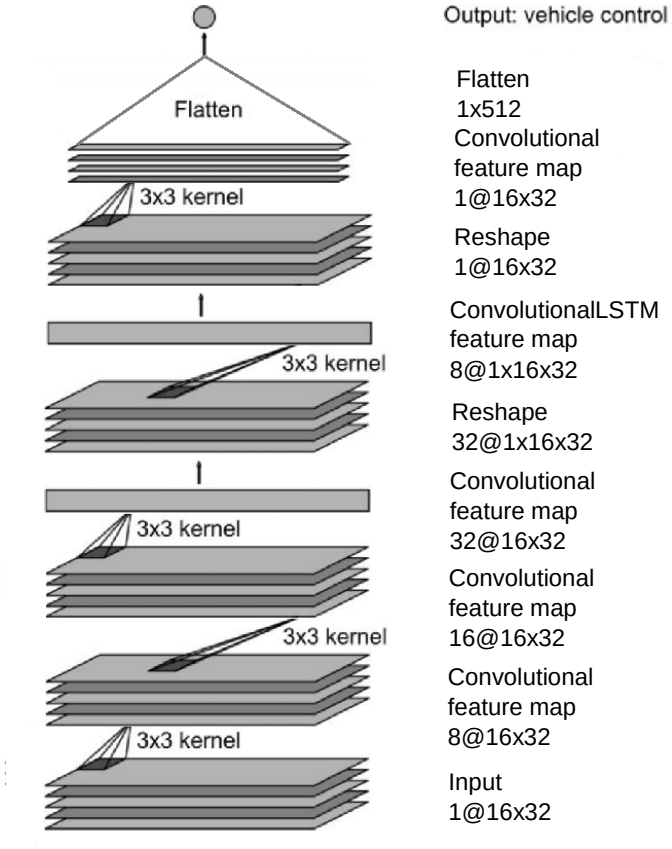

Ilustración 24 Arquitectura de LSTM-TinyPilotNet

4.1.2.1 DeeperLSTM-TinyPilotNet

También se elabora una segunda arquitectura, con un mayor número de capas ConvolutionalLSTM y mayor resolución de entrada, dando lugar a la red DeeperLSTM-TinyPilotNet.

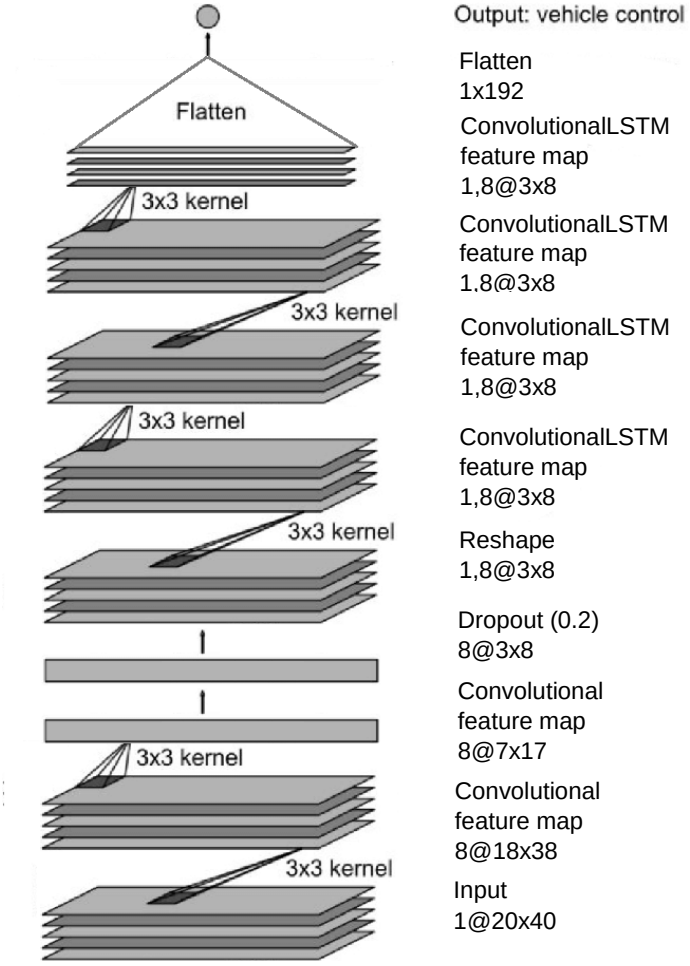

Ilustración 25 Arquitectura de DeeperLSTM-TinyPilotNet

4.1.2.2 DeepestLSTM-TinyPilotNet

Cuando se realice la conducción autónoma produciendo los valores de ángulo de volante y aceleración por una misma CNN, la red deberá de configurarse en función de ambos parámetros y no solo de uno de ellos.

Para aumentar el número de parámetros configurables de la red y por ello obtener un mejor aprendizaje de los datos de entrada se elabora una red aún mayor derivada de DeeperLSTM-TinyPilotNet, la red DeepestLSTM-TinyPilotNet.

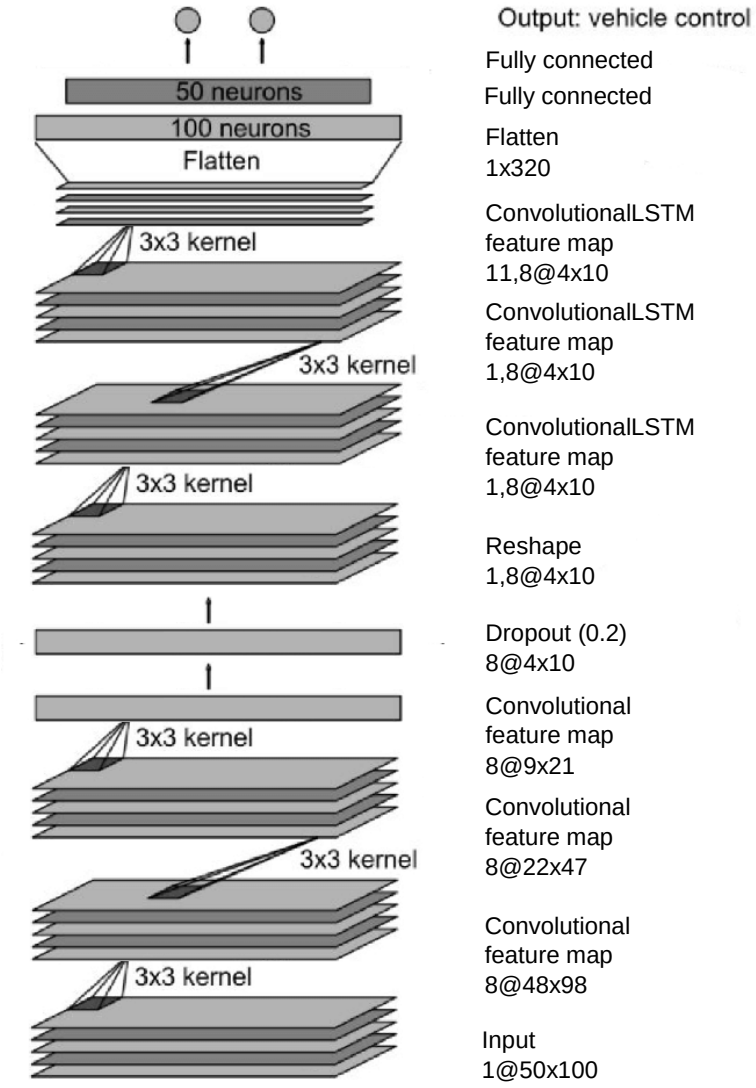

Ilustración 26 Arquitectura de DeepestLSTM-TinyPilotNet

4.2 Tratamiento de datos

Como se introduce en el capítulo 3, los datos de entrenamiento son una parte fundamental para un correcto desempeño por parte de la red neuronal convolucional. Sin embargo, extraer una cantidad de datos importante supone una gran dificultad.

El data augmentation, o aumento de datos, permite modificar o aumentar la información de entrenamiento de la red a partir de un banco de datos previamente obtenido de forma que la CNN comprenda con mayor facilidad qué datos extraer para obtener los resultados esperados.

Los efectos de tratamiento de datos de entrenamiento llevados a cabo en el estudio se muestran a continuación.

4.2.1 Imagen RGB

Mediante este procedimiento se modifica la imagen de entrada de la red, que anteriormente era una imagen que tomaba únicamente la saturación del espacio de color HSV (matiz, saturación, brillo), por una imagen de 3 canales a color RGB.

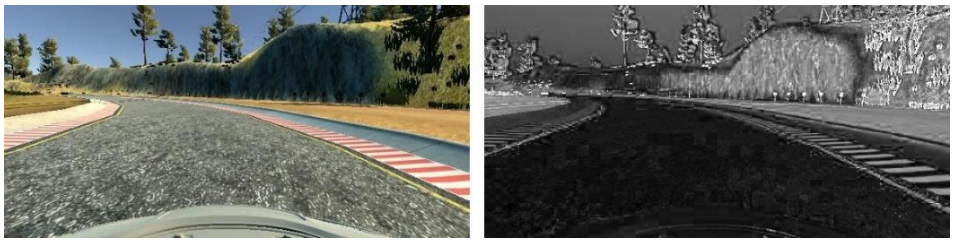

Ilustración 27 Imagen de simulador RGB frente a canal saturación HSV

Para ser implementada en una CNN simplemente es necesario modificar la primera capa de la red, por lo que no se considera una arquitectura nueva, sino un método de tratamiento de datos.

Las redes entrenadas y probadas con este procedimiento irán precedidas por las siglas RGB.

4.2.2 Aumento de resolución

Algunas de las pruebas realizadas consisten en el aumento de la resolución de la imagen de entrada. Esta modificación, al igual que la anterior, simplemente supone una modificación de la dimensión de la capa de entrada, por lo que no se puede considerar una arquitectura distinta.

4.2.3 Recorte de imagen

El recorte de imagen, o image cropping, consiste en extraer una zona concreta de la imagen en la que se considera que se concentra el grueso de la información relevante para la CNN.

En este caso, la imagen que analizará la CNN solo contiene información de la carretera, eliminando la parte de paisaje del fotograma.

Ilustración 28 Imagen original de simulador frente a imagen recortada

Las redes entrenadas y probadas con este procedimiento irán precedidas por la palabra Cropping.

4.2.4 Filtro detector de bordes

Otro tratamiento de imagen llevado a cabo consiste en extraer los bordes de la imagen de entrada para destacarlos sobre la imagen original. Esto se consigue empleando un filtro Canny de la librería cv2.

Las redes entrenadas y probadas con este procedimiento irán precedidas por la palabra Edge.

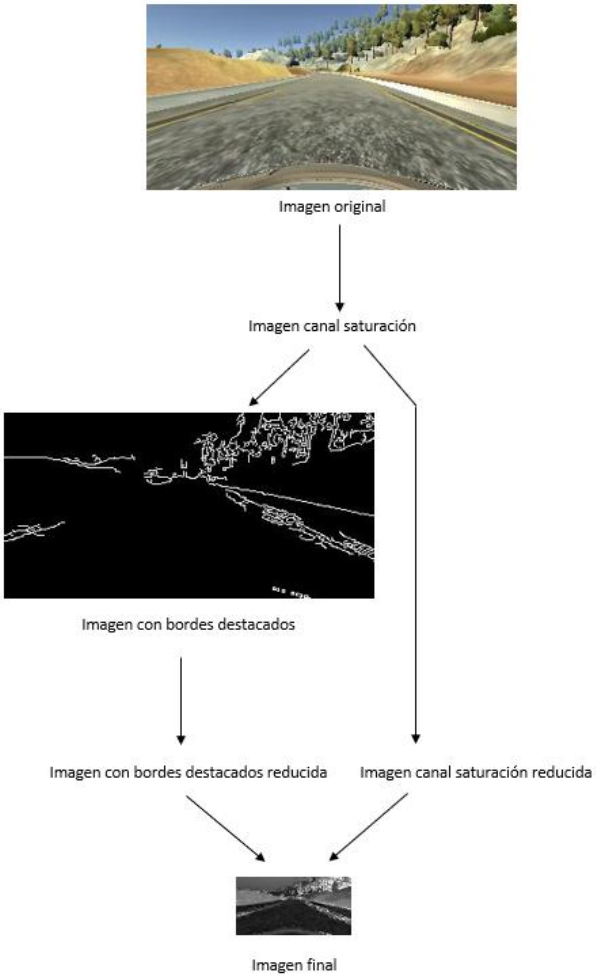

Ilustración 29 Esquema de implementación de la detección de bordes

Capítulo 5.
Métrica de evaluación.

Para evaluar la calidad de la conducción es necesario obtener una cuantificación de la calidad de la conducción producida por la red.

A través de diversos métodos que se exponen en este capítulo se obtendrán diversas formas de cuantificar la correcta conducción de las CNN para poder comparar en capítulos posteriores los resultados obtenidos.

5.1 Métricas de evaluación estándar.

Una de las métricas de evaluación más comunes establece como referencia de una correcta conducción la realizada por un ser humano [16, 17], comparando los datos producidos por la CNN entrenada y los producidos por un conductor humano en cada imagen, tratando estos últimos como ground truth. El valor numérico se realiza mediante Root-Mean Square Error.

Los valores de conducción obtenidos por la red y el conductor humano pueden expresarse mediante gráficas o el cálculo del error producido mediante la diferencia de valores.

Ilustración 30 Gráfica comparativa entre ground truth producido por humano y valores de dos CNNs. Extraída de [27]

Sin embargo, estos valores no parecen muy significativos, ya que la conducción humana no tiene por qué ser perfecta, por lo que se establecen otros parámetros novedosos, calculados a partir de los datos extraídos del simulador y procesados mediante código en MATLAB.

5.2 Nuevas métricas de evaluación.

Para comparar cuantitativamente el desempeño de los distintos modelos de CNN se desarrolla un código en MATLAB que, partiendo de los datos extraídos durante la prueba en el simulador, calcula distintos parámetros.

Como base para su cálculo primero es necesario definir los puntos de la carretera dentro de MATLAB, haciendo uso de los *waypoints* del simulador, mostrados en la siguiente ilustración.

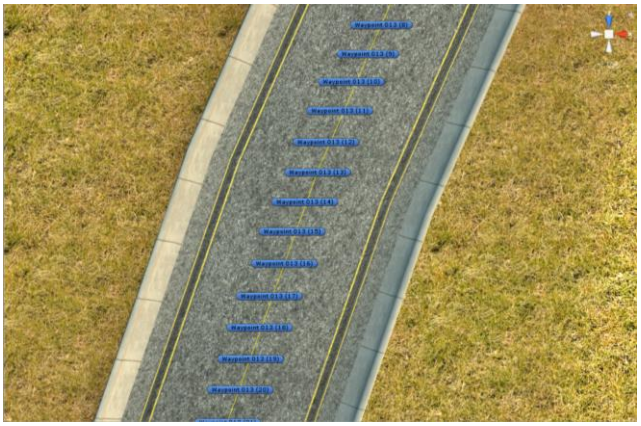

Ilustración 31 Diferentes waypoints en una zona del circuito

Sobre cada waypoint se obtiene una línea perpendicular a la dirección de avance de la carretera en dicho punto. Se calculan los puntos extremos de un segmento de 3 metros centrado en el waypoint y perpendicular a la dirección del carril teniendo en cuenta que el ancho total de la vía es de 6 metros.

5.2.1 Desviación respecto al centro del carril

El primer parámetro de calidad se establece en función de la desviación respecto al centro del carril en cada momento, calculando después el valor medio y máximo de esta desviación. Para ello busca el *waypoint* más próximo al coche y calcula la distancia que existe entre el coche y el segmento que une dicho punto con el anterior y posterior. Estas distancias son almacenadas en un vector para ser representadas y analizadas posteriormente.

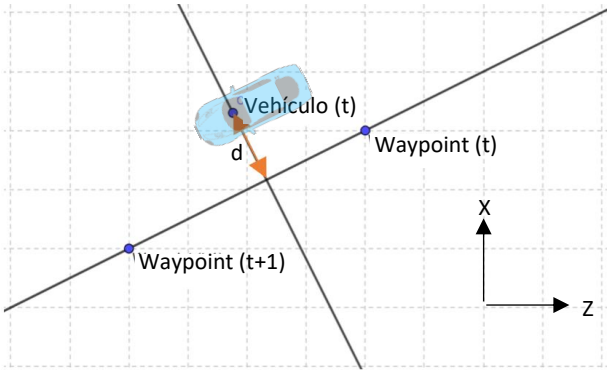

Ilustración 32 Cálculo de la desviación respecto al centro del carril

El waypoint más próximo al vehículo será el que menor norma tenga entre la posición del vehículo y todos los waypoints del circuito.

$$Waypoint = mín\{norm(A, C)\}$$

Siendo A el punto del vehículo en t y C cada uno de los waypoints.

La desviación respecto al centro del carril se calculará como la distancia entre el vehículo y el segmento que une el waypoint más próximo con el siguiente en el caso de que el vehículo esté por delante del waypoint (Ilustración 32), o con el segmento entre el waypoint anterior si el waypoint está adelantado respecto del vehículo.

$$Distancia = \frac{abs[det(A)]}{norm(Waypoint(t+1) - Waypoint(t))}$$

Siendo A:

$$A = \begin{bmatrix} Waypoint(t+1)_x & Waypoint(t+1)_z & 1 \\ Waypoint(t)_x & Waypoint(t)_z & 1 \\ Vehículo(t)_x & Vehículo(t)_z & 1 \end{bmatrix}$$

5.2.2 Ángulo de cabeceo

El segundo parámetro de calidad se establece en función del *heading angle* o ángulo de cabeceo del vehículo. Éste se calcula como el ángulo entre el vehículo y la dirección que sigue la carretera. Para obtenerlo se calcula el ángulo entre el vector definido por el vehículo entre los puntos actual y anterior, y el vector definido por el waypoint más próximo y el anterior. Al igual que los valores de distancia al centro del carril, el resultado se guarda en un vector para poder representarlos posteriormente.

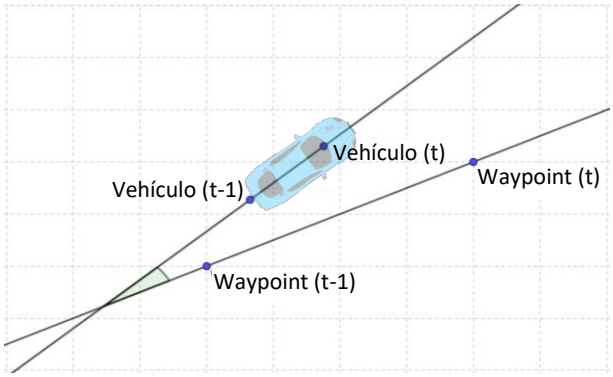

Ilustración 33 Cálculo del ángulo de cabeceo

Para obtener este parámetro se realiza el siguiente procedimiento

1. Se definen los vectores veh_vector somo Vehículo(t) – Vehículo (t-1) y road_vector como Waypoint(t) – Waypoint(t-1).
2. $heading\ angle = \text{acos}\left(mín\left\{1, máx\left\{-1, \frac{veh\_vector * road\_vector}{norm(veh\_vector) * norm(road\_vector)}\right\}\right\}\right) * \frac{180}{\pi}$

5.3 Representación genérica

Para que la representación de todas las simulaciones quede referida de forma similar a los mismos puntos del circuito el eje de abscisas se ubica entre 0 % y 100 % del recorrido.

Para calcular el total recorrido se calcula el ángulo existente entre dos rectas.

1. Recta vehículo – circuito. Une el vehículo con un punto central del circuito
2. Recta meta – circuito. Une el punto de inicio del circuito y el punto central del circuito.

El ángulo, que tomará valores entre 0 y 360º se normaliza entre 0 y 100 para adecuarlo a la representación.

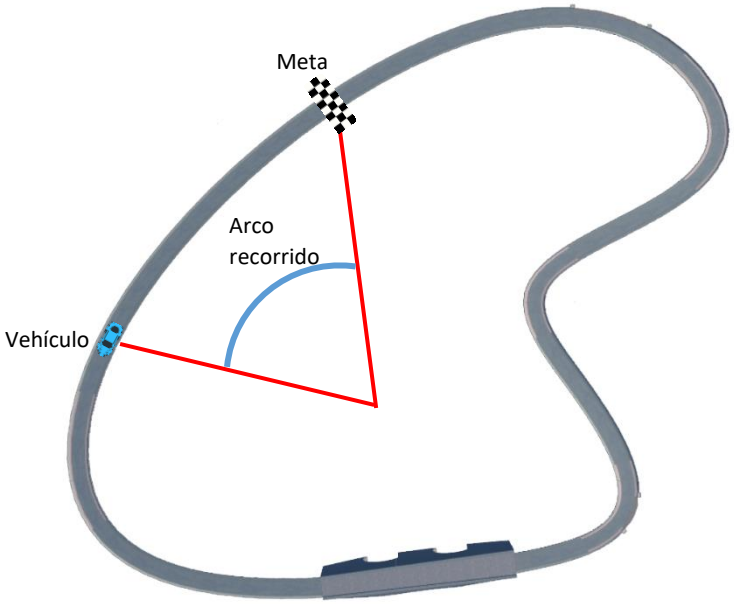

Ilustración 34 Total recorrido por el vehículo

Capítulo 6.
Resultados experimentales.

Una vez definidas las arquitecturas de CNN, los métodos de tratamiento de datos y las métricas de evaluación en los capítulos anteriores, este capítulo tratará acerca de los resultados obtenidos mediante las pruebas de conducción de las distintas arquitecturas de CNN en el simulador.

Para entrenar la red se realiza una recolección de datos mediante conducción manual en el simulador, recorriendo varias vueltas en el circuito para obtener un gran volumen de información. Esta información contendrá grabaciones de imágenes del salpicadero recogidas por una cámara central y dos laterales ubicadas a ambos lados del vehículo a 14 FPS, además de datos de ángulo de volante, aceleración, freno y velocidad absoluta vinculados a dichas imágenes.

El parámetro RMSE de métrica de evaluación para las diferentes CNNs ha sido obtenido empleando el mismo conjunto de datos (*dataset*) empleado para el entrenamiento.

Ilustración 35 Imágenes izquierda, central y derecha recogidas para entrenamiento

Todas las CNN han sido entrenadas empleando las imágenes izquierda, central y derecha del vehículo aplicando un desplazamiento de 5º al ángulo de volante. Además, se ha aumentado el conjunto de datos de entrenamiento realizando un volteo de imagen horizontal (*mirroring*) a las imágenes, invirtiendo también el valor de ángulo de volante, pero manteniendo el valor de aceleración.

El entrenamiento de redes con capas LSTM debe realizarse sin aleatorizar las imágenes de entrada para poder vincular los datos actuales con los pasados en una secuencia.

6.1 Control de ángulo del volante

En primer lugar, se analizarán los resultados de dichas modificaciones sobre el control del ángulo de giro de volante, fijando una aceleración del 5% que hace que el vehículo circule a su velocidad máxima sobre el circuito.

6.1.1 TinyPilotNet

Según los datos extraídos durante la prueba, los resultados son los mostrados en la siguiente tabla, todos ellos en grados.

CNN	RMSE
TinyPilotNet	0.083478

Tabla 3 RMSE de TinyPilotNet controlando volante

CNN	Error medio	Error máx.	Cabeceo medio	Cabeceo máx.
TinyPilotNet	0.54	2.41	2.44	18.91

Tabla 4 Datos extraídos de TinyPilotNet controlando volante para métricas nuevas

Los valores instantáneos de ángulo de giro de volante, ángulo de cabeceo y error o distancia al centro se muestran en la siguiente ilustración.

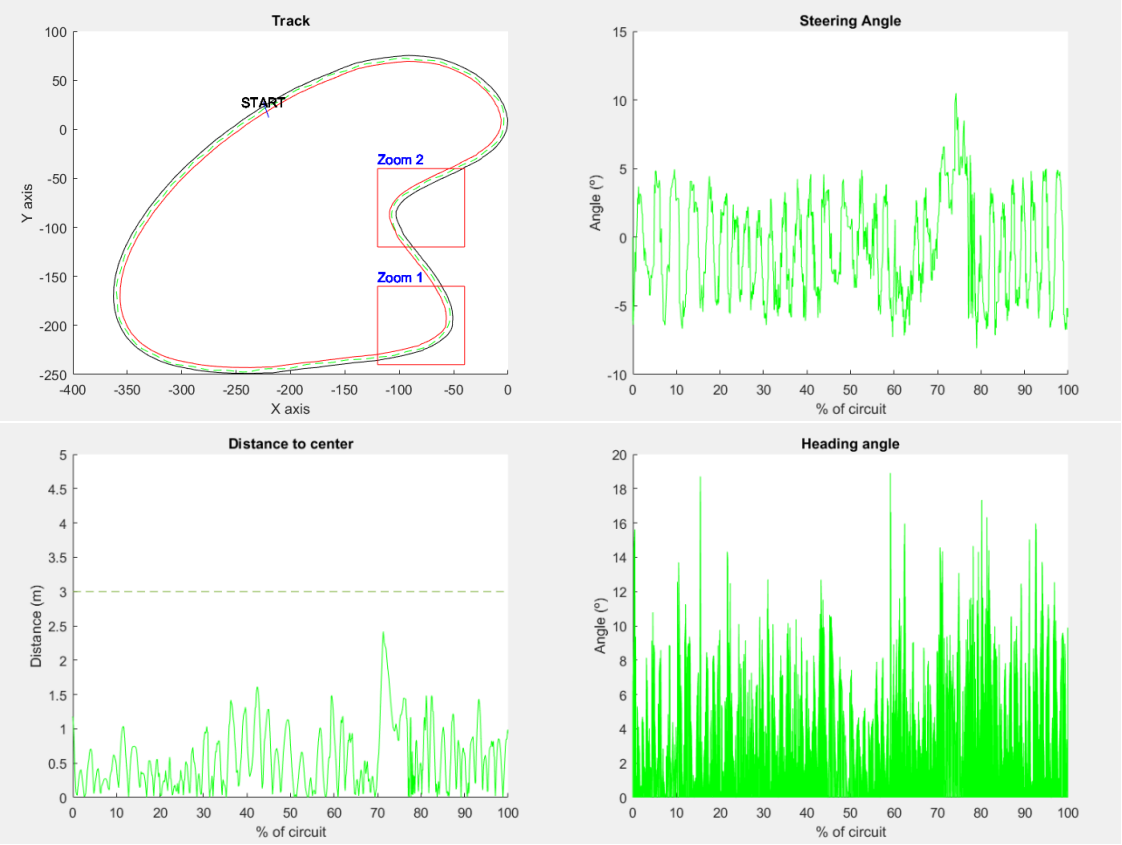

Ilustración 36 Gráficas de resultados de TinyPilotNet controlando volante

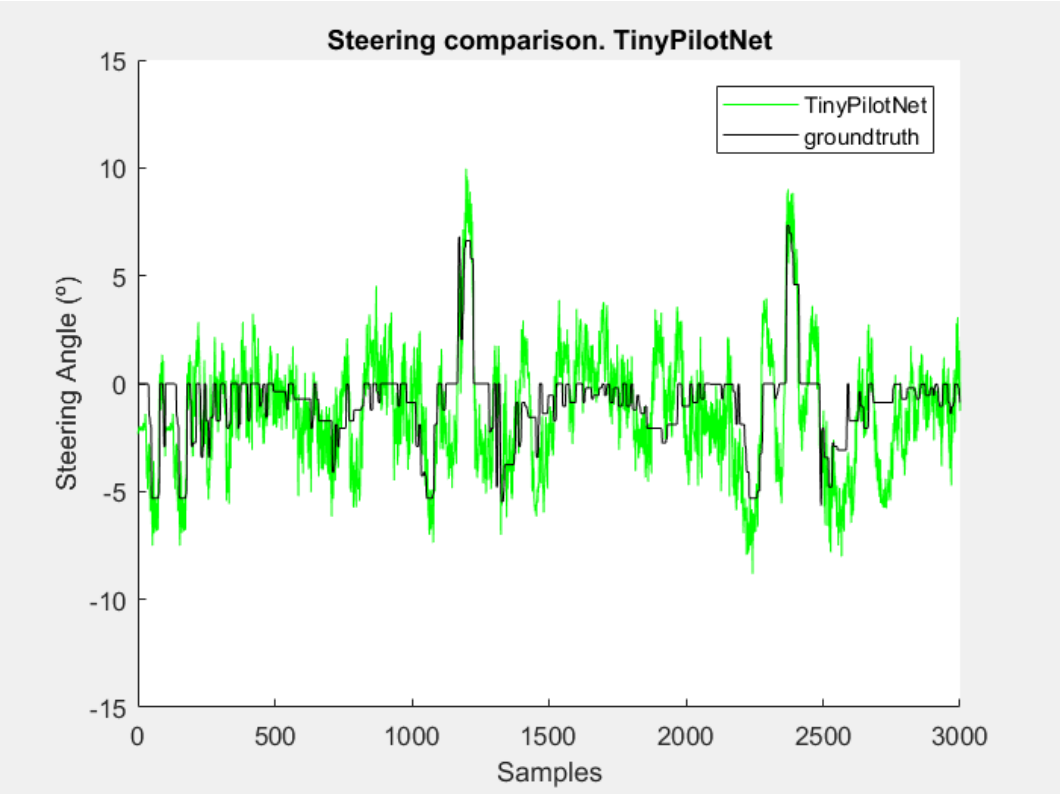

Ilustración 37 Comparativa frame-to-frame de ángulo de volante TinyPilotNet

6.1.2 TinyPilotNet de mayor resolución

Este modelo de red neuronal sigue la misma arquitectura que la red TinyPilotNet, solo que tiene a su entrada una imagen de 20 x 40 píxeles en lugar de 16 x 32.

Los resultados se muestran en la siguiente tabla.

CNN	RMSE
TinyPilotNet de mayor resolución	0.081172

Tabla 5 RMSE de TinyPilotNet de mayor resolución controlando volante

CNN	Error medio	Error máx.	Cabeceo medio	Cabeceo máx.
TinyPilotNet de mayor resolución	0.70	3.39	1.71	23.06

Tabla 6 Datos extraídos de TinyPilotNet de mayor resolución controlando volante

Las gráficas que muestran los datos recogidos del simulador durante la prueba se muestran a continuación.

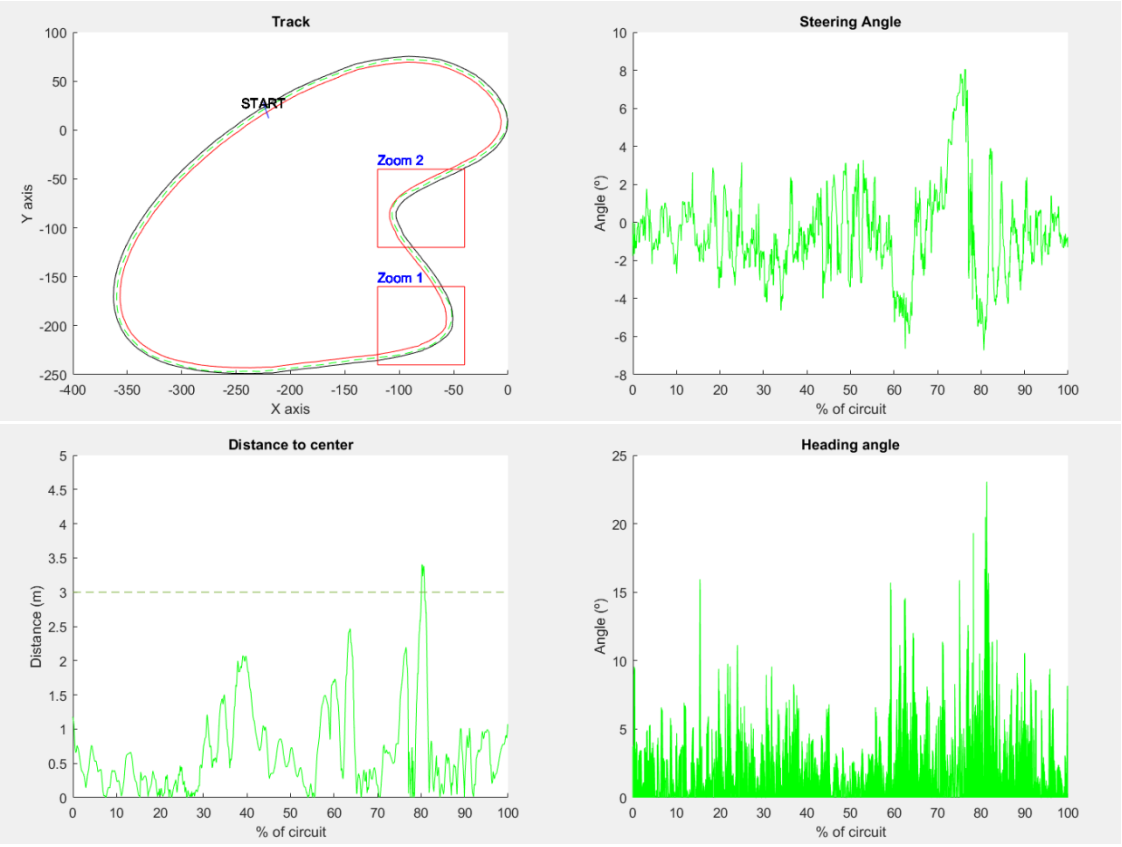

Ilustración 38 Gráficas de resultados de TinyPilotNet de mayor resolución controlando volante

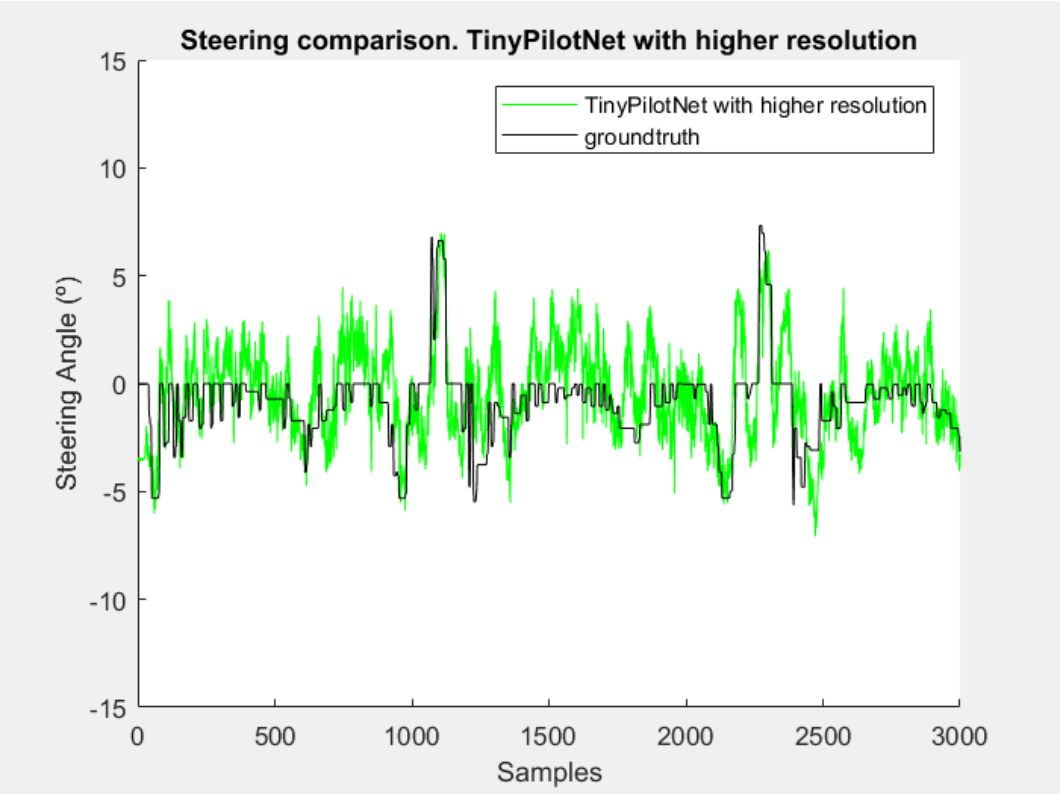

Ilustración 39 Comparativa frame-to-frame de ángulo de volante TinyPilotNet de mayor resolución

6.1.3 HD-TinyPilotNet

Al igual que la red TinyPilotNet de mayor resolución, esta red emplea la misma arquitectura, pero la imagen de entrada tiene una resolución de 64 x 128 píxeles en lugar de 16 x 32.

Los resultados se muestran en la siguiente tabla.

CNN	RMSE
HD-TinyPilotNet	0.083799

Tabla 7 RMSE de HD-TinyPilotNet controlando volante

CNN	Error medio	Error máx.	Cabeceo medio	Cabeceo máx.
HD-TinyPilotNet	2.07	9.87	2.73	40.33

Tabla 8 Datos extraídos de HD-TinyPilotNet controlando volante

Las gráficas que muestran los datos recogidos del simulador durante la prueba se muestran a continuación.

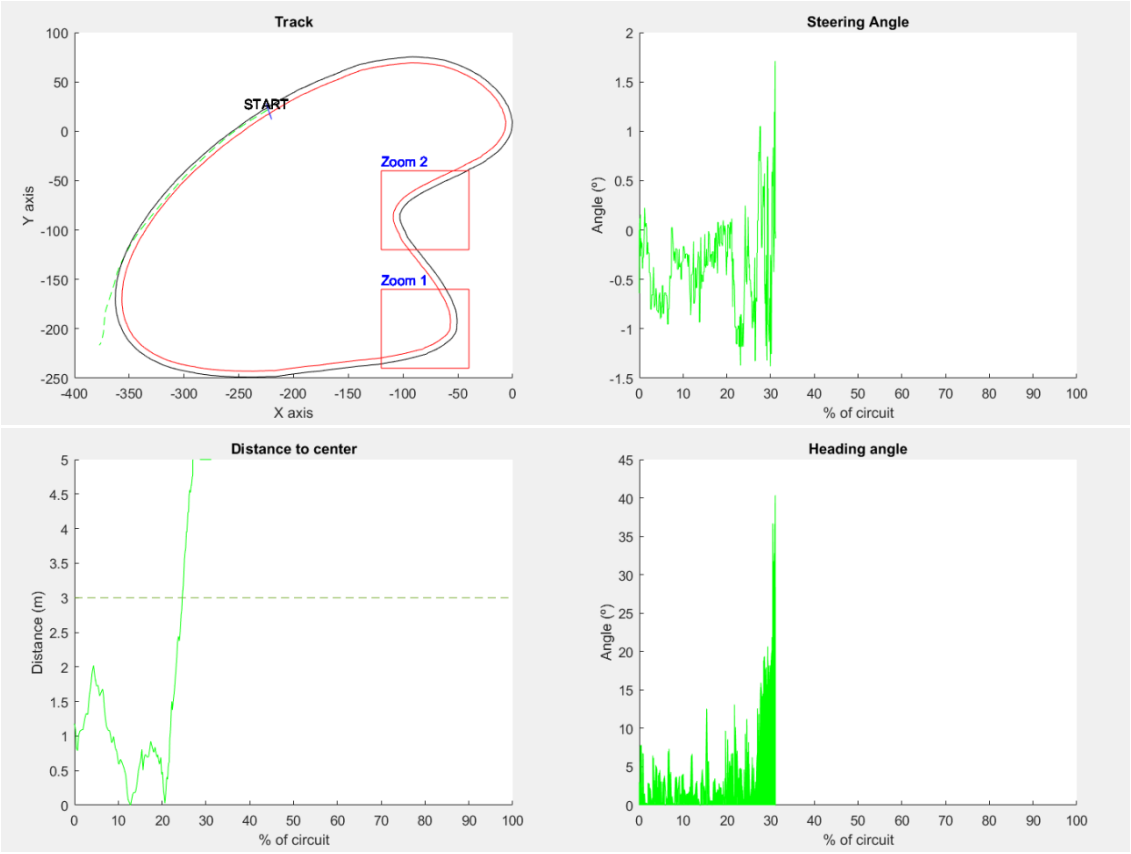

Ilustración 40 Gráficas de resultados de HD-TinyPilotNet controlando volante

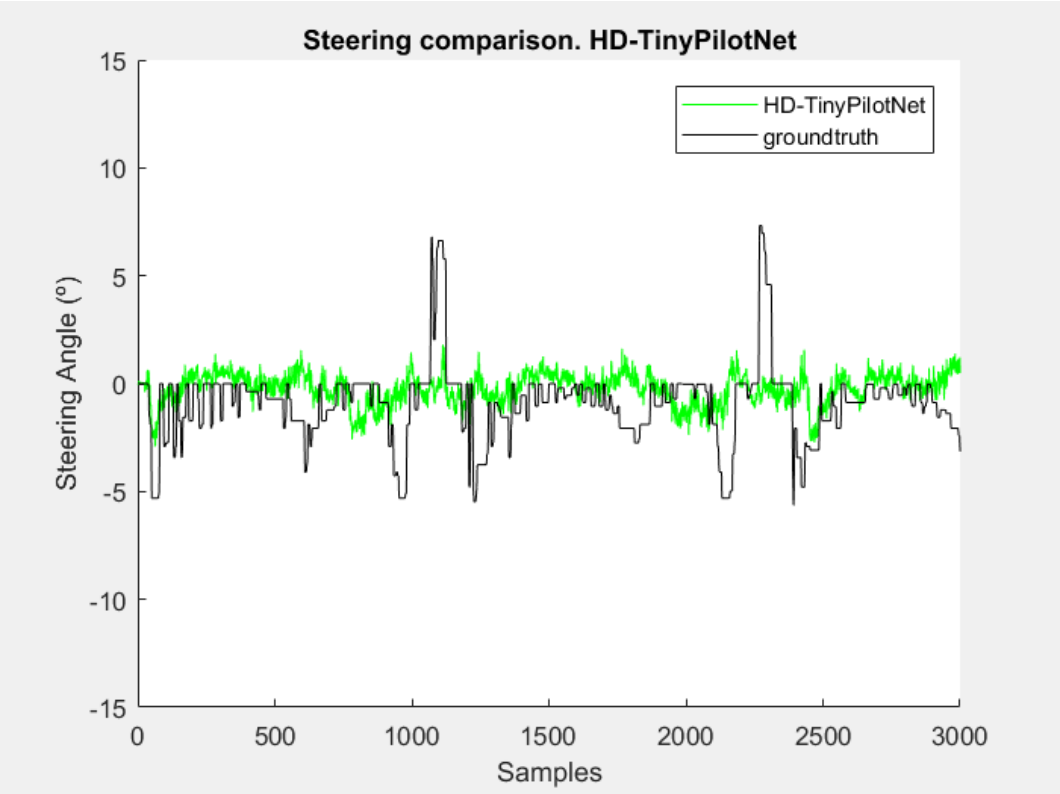

Ilustración 41 Comparativa frame-to-frame de ángulo de volante HD-TinyPilotNet

6.1.4 RGB-TinyPilotNet

La arquitectura de esta red es similar a la de las redes puestas a prueba anteriormente.

La imagen de entrada en este caso contiene 3 canales (RGB) en lugar de un solo canal.

Los resultados del circuito para la red con entrada RGB se recogen en la siguiente tabla.

CNN	RMSE
RGB-TinyPilotNet	0.104

Tabla 9 RMSE de RGB-TinyPilotNet controlando volante

CNN	Error medio	Error máx.	Cabeceo medio	Cabeceo máx.
RGB-TinyPilotNet	2.86	9.71	3.69	44.98

Tabla 10 Datos extraídos de RGB-TinyPilotNet controlando volante

Las gráficas de los datos recogidos por el vehículo comandado por esta red son las siguientes.

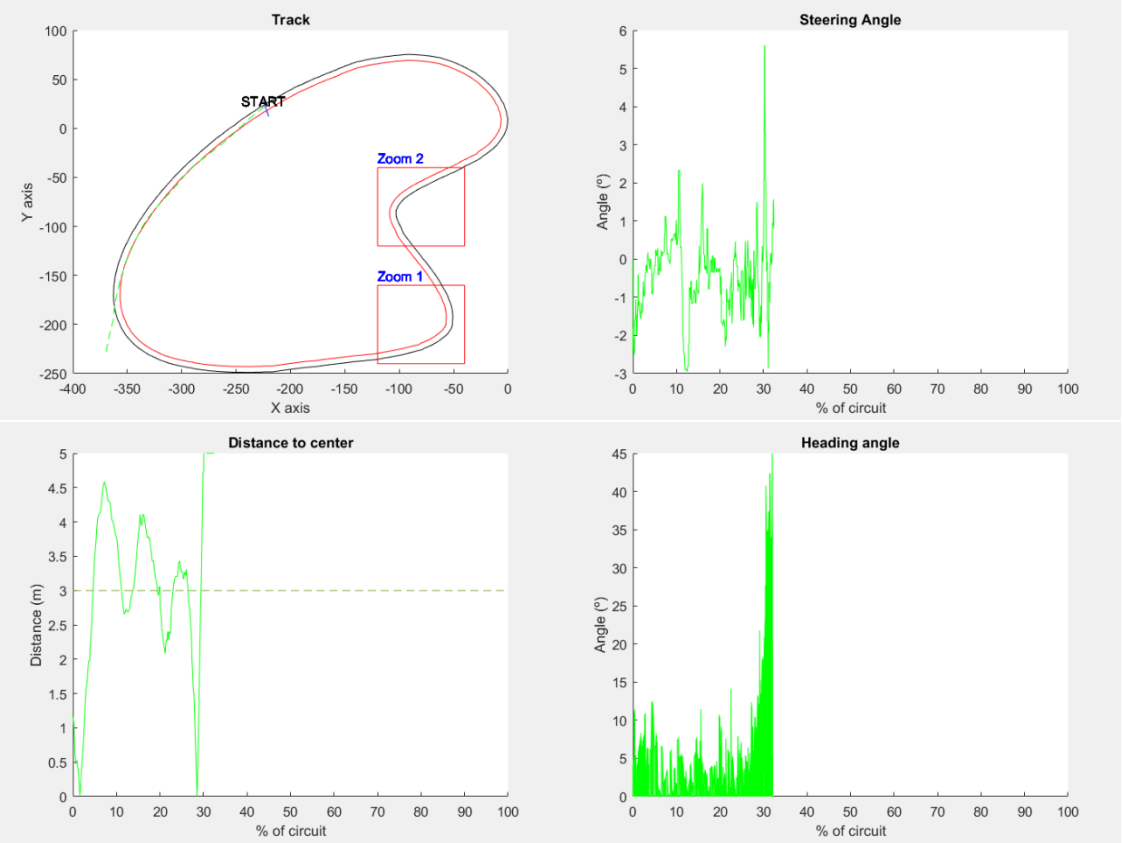

Ilustración 42 Gráficas de resultados de RGB-TinyPilotNet controlando volante

Como se puede observar tanto en la ilustración anterior como en el valor de error máximo de la tabla, el vehículo no es capaz de seguir la curva del circuito y se sale de él siguiendo una trayectoria rectilínea, concluyendo que el empleo de imágenes RGB es más sensible a variaciones de la iluminación del circuito que las imágenes en espacio de color HSV.

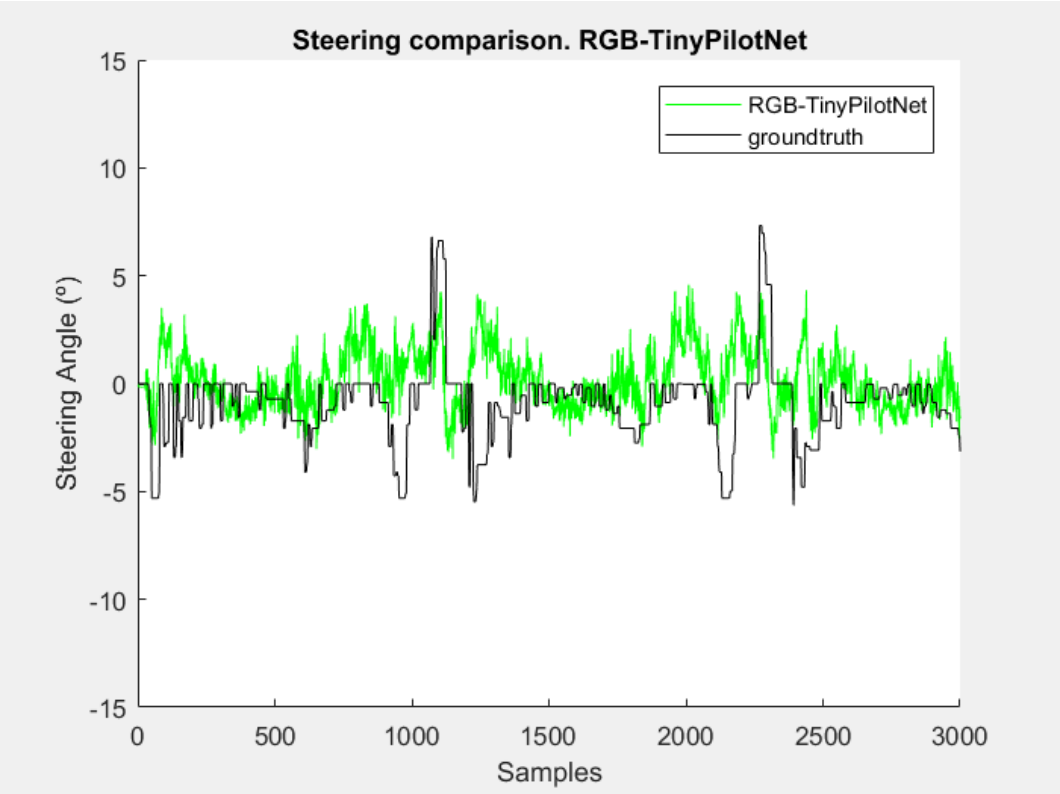

Ilustración 43 Comparativa frame-to-frame de ángulo de volante RGB-TinyPilotNet

6.1.5 LSTM-TinyPilotNet

Para introducir el efecto de memoria en la red para que esta tenga en cuenta los valores de salida producidos anteriormente se añaden capas LSTM a la salida de esta.

Siguiendo el manual de Keras, lenguaje de alto nivel en el que están programadas las diversas redes, se puede encontrar una capa que mezcla LSTM con un efecto convolucional, llamada ConvLSTM2D.

Es esta capa la que se añadirá entre las capas convolucionales y la neurona de salida para aportar el efecto de memoria como se recoge en el esquema de arquitectura de la red.

Los resultados conseguidos con esta red son los siguientes.

CNN	RMSE
LSTM-TinyPilotNet	0.08725

Tabla 11 RMSE de LSTM-TinyPilotNet controlando volante

CNN	Error medio	Error máx.	Cabeceo medio	Cabeceo máx.
LSTM-TinyPilotNet	0.58	2.35	2.10	27.03

Tabla 12 Datos extraídos de LSTM-TinyPilotNet controlando volante

Las gráficas de datos del vehículo empleando esta CNN son las siguientes.

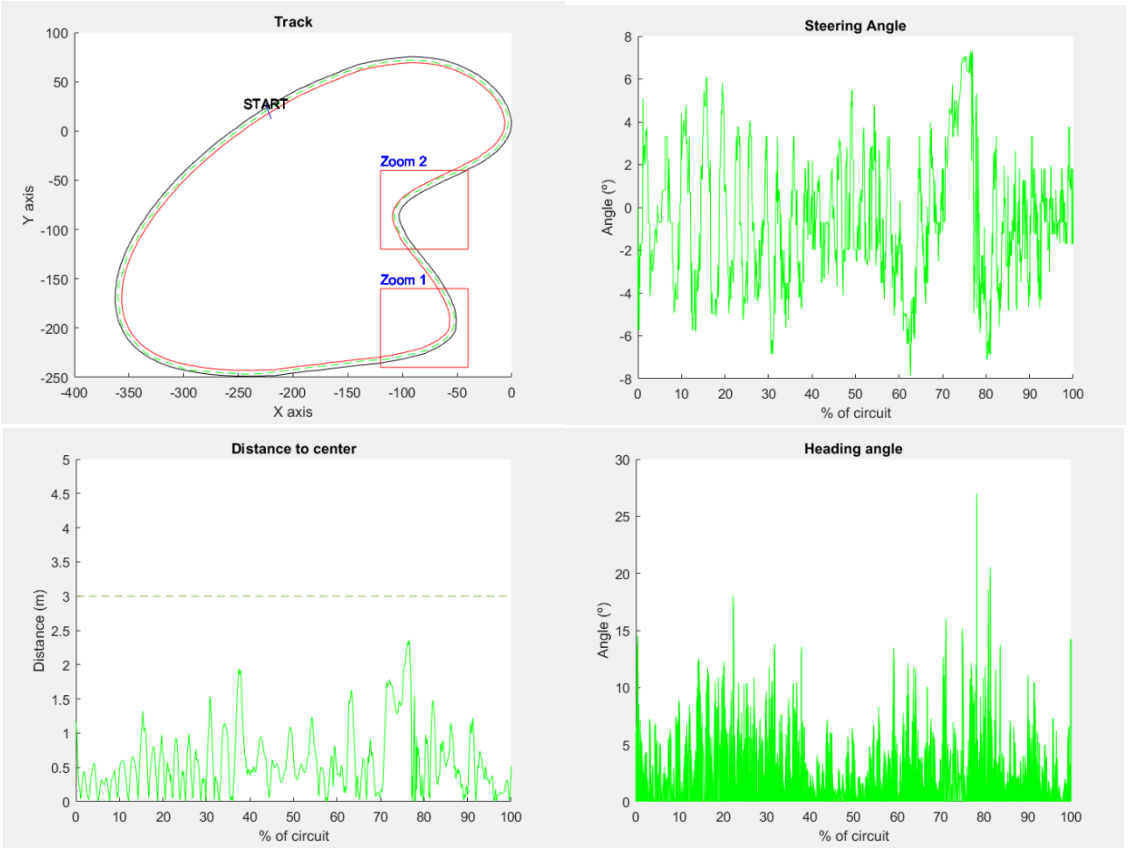

Ilustración 44 Gráficas de resultados de LSTM-TinyPilotNet controlando volante

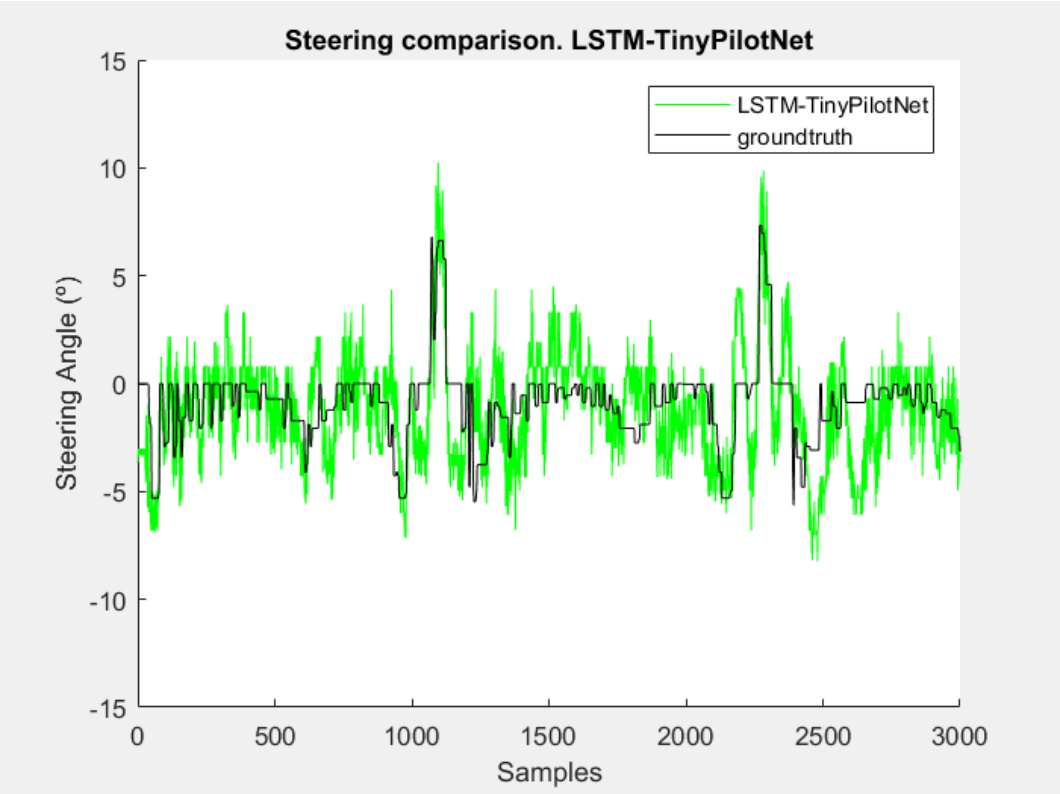

Ilustración 45 Comparativa frame-to-frame de ángulo de volante LSTM-TinyPilotNet

Se observa un comportamiento bastante similar al producido por la red TinyPilotNet, mejorando el parámetro de ángulo de cabeceo al aportar el efecto de memoria de los datos de ángulo de volante producidos con anterioridad.

6.1.6 DeeperLSTM-TinyPilotNet

Esta CNN combina los efectos de la red LSTM y la red de mayor resolución, aumentando el tamaño de la imagen de entrada hasta 20x40 píxeles.

La arquitectura considerada para esta red es la mostrada en la ilustración 25 de este trabajo.

Los resultados producidos por esta red son los siguientes.

CNN	RMSE
DeeperLSTM-TinyPilotNet	0.095224

Tabla 13 RMSE de DeeperLSTM-TinyPilotNet controlando volante

CNN	Error medio	Error máx.	Cabeceo medio	Cabeceo máx.
DeeperLSTM-TinyPilotNet	0.453	1.98	1.98	18.37

Tabla 14 Datos extraídos de DeeperLSTM-TinyPilotNet controlando volante

Las gráficas que muestran los datos del vehículo en el simulador empleando esta red neuronal son las siguientes.

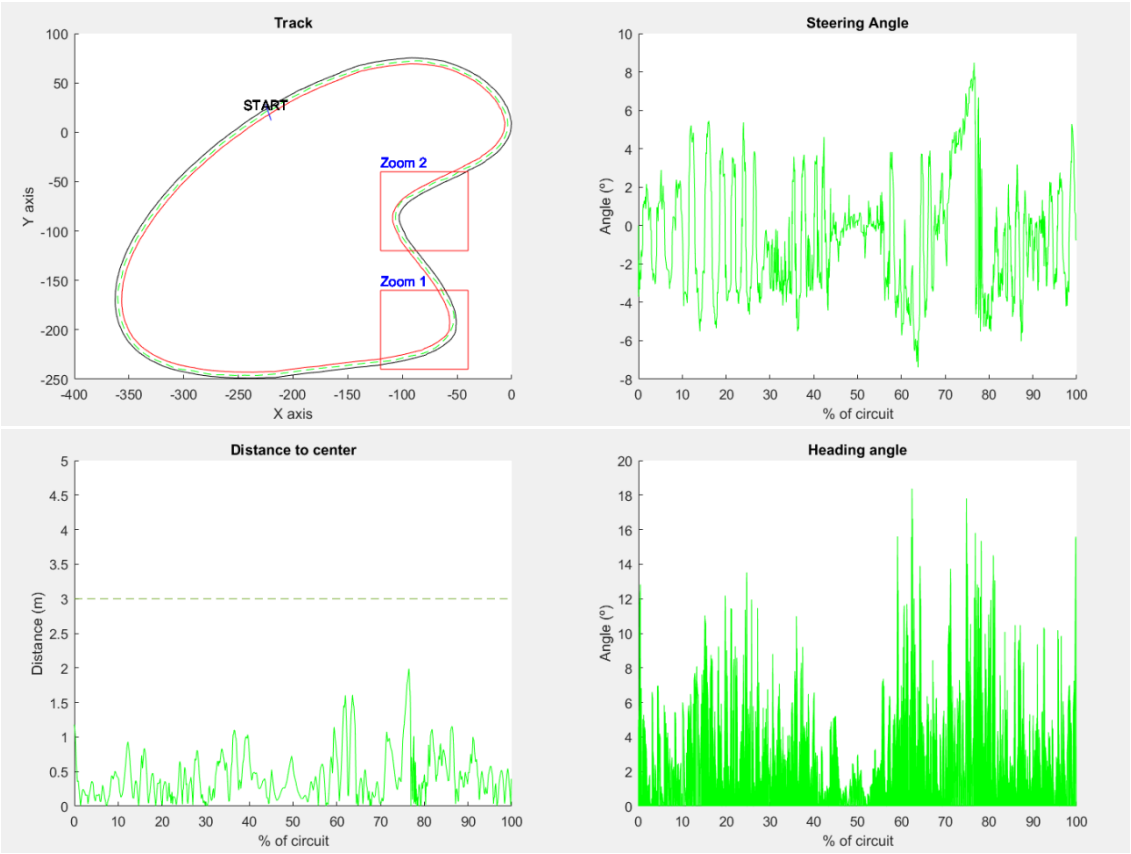

Ilustración 46 Gráficas de resultados de DeeperLSTM-TinyPilotNet controlando volante

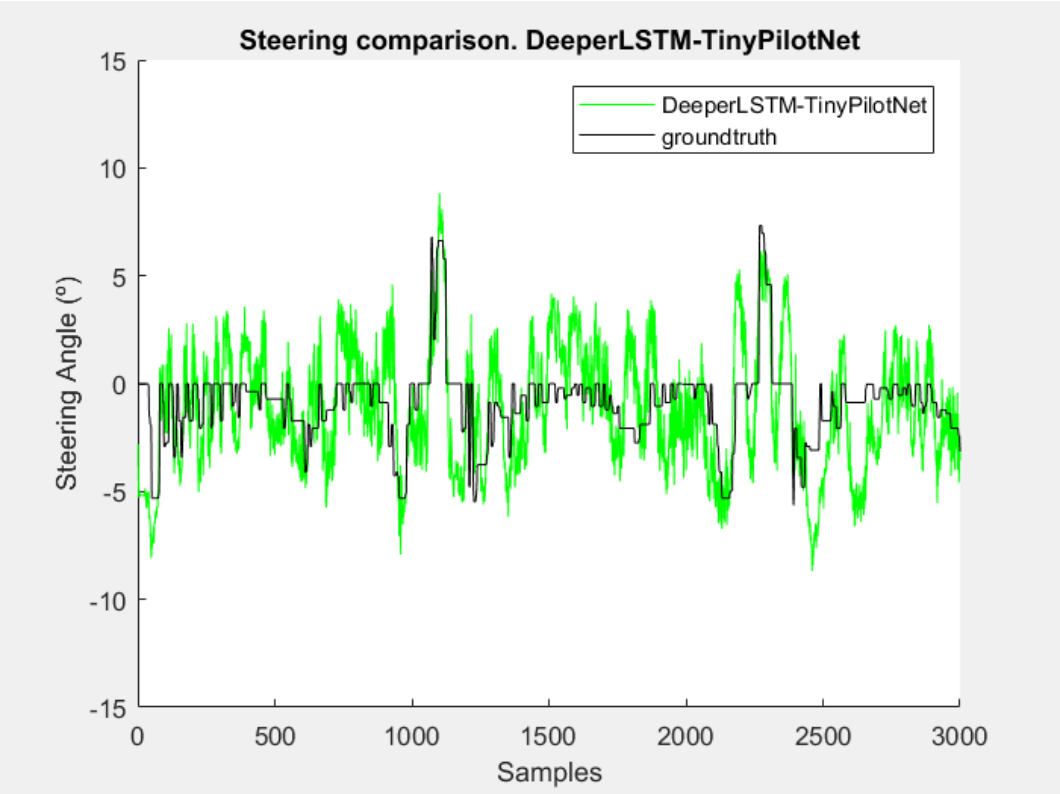

Ilustración 47 Comparativa frame-to-frame de ángulo de volante DeeperLSTM-TinyPilotNet

Se puede comprobar que los resultados de la tabla de esta red mejoran los resultados obtenidos con TinyPilotNet en cuanto al parámetro de cabeceo o heading error.

CNN	Error medio	Error máx.	Cabeceo medio	Cabeceo máx.
TinyPilotNet	0.54	2.41	2.44	18.91
DeeperLSTM-TinyPilotNet	0.453	1.98	1.98	18.37

Tabla 15 Comparación entre TinyPilotNet y DeeperLSTM-TinyPilotNet controlando volante

Sin embargo, el valor de RMSE aumenta debido a que la red es más eficiente disponiendo de una secuencia de datos anteriores, mientras que este parámetro se calcula solo fotograma a fotograma.

CNN	RMSE
TinyPilotNet	0.083478
DeeperLSTM-TinyPilotNet	0.095224

Tabla 16 Comparación de RMSE entre TinyPilotNet y DeeperLSTM-TinyPilotNet controlando volante

6.1.7 Cropping-DeeperLSTM-TinyPilotNet

Esta red es similar a la red DeeperLSTM-TinyPilotNet desarrollada anteriormente, pero a su entrada se aplica el efecto de image cropping, o recorte de imagen, para eliminar la información relativa al paisaje y el entorno del circuito, analizando solo la parte de la pista.

Los resultados obtenidos por esta red son los siguientes.

CNN	RMSE
Cropping-DeeperLSTM-TinyPilotNet	0.094075

Tabla 17 RMSE de Cropping-DeeperLSTM-TinyPilotNet controlando volante

CNN	Error medio	Error máx.	Cabeceo medio	Cabeceo máx.
Cropping-DeeperLSTM-TinyPilotNet	0.48	2.53	2.43	21.68

Tabla 18 Datos extraídos de Cropping-DeeperLSTM-TinyPilotNet controlando volante

Las gráficas que muestran los datos del vehículo en el simulador empleando esta red neuronal son las siguientes.

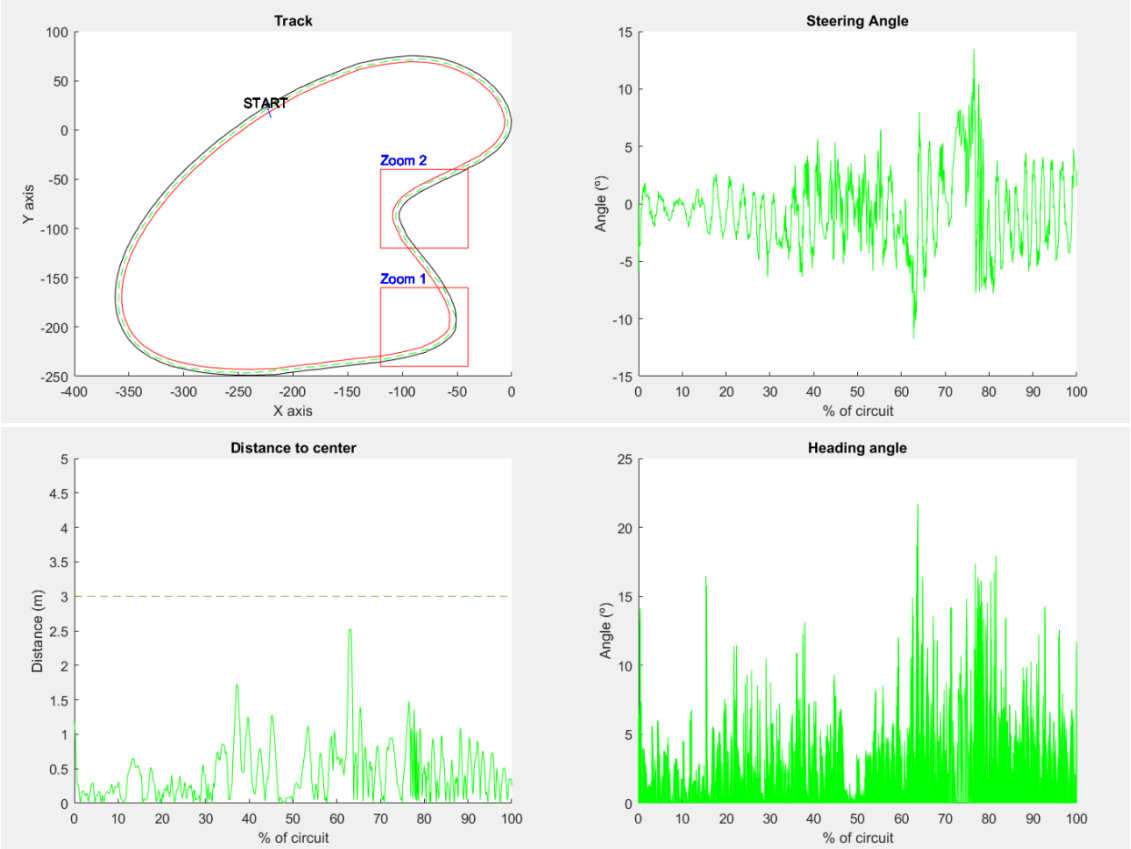

Ilustración 48 Gráficas de Cropping-DeeperLSTM-TinyPilotNet controlando volante

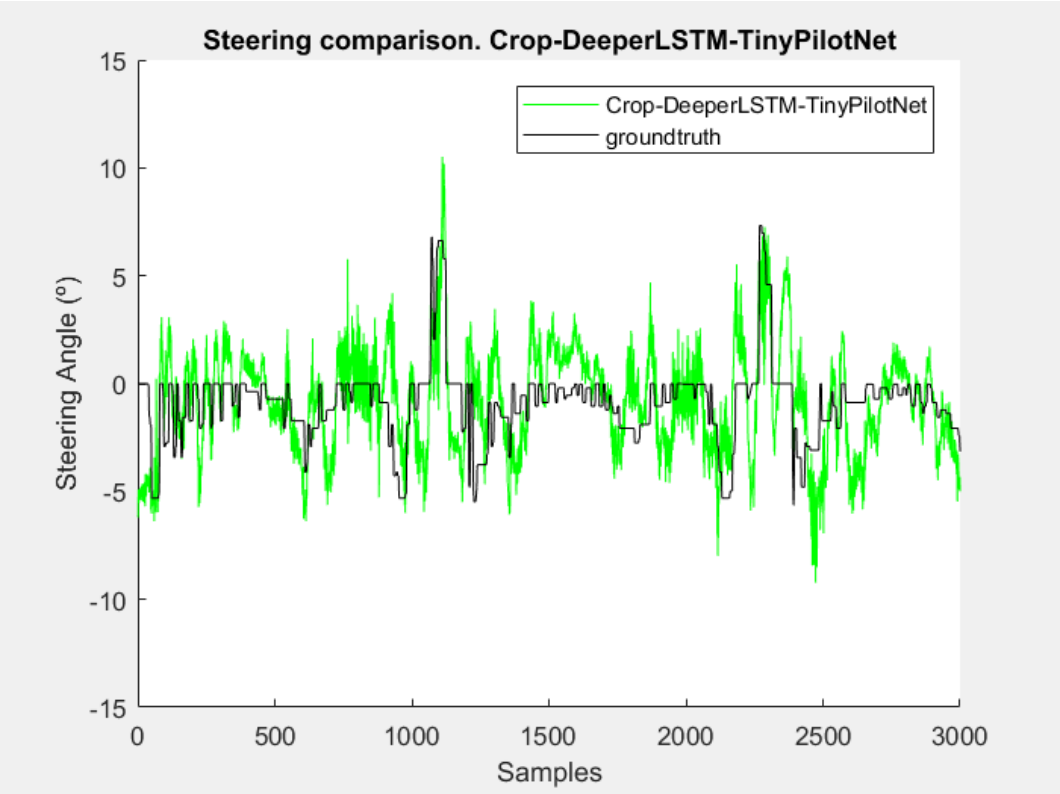

Ilustración 49 Comparativa frame-to-frame de ángulo de volante Cropping-DeeperLSTM-TinyPilotNet

6.1.8 Edge-DeeperLSTM-TinyPilotNet

En esta última red se pretende aportar unos bordes del circuito más definidos de forma que la red los interprete con mayor calidad y sea capaz de seguirlos aun cuando no queden demasiado claros por la imagen, ya que hay zonas del circuito en las que este se funde con una pista de tierra en lugar de tener un borde metálico claro que resalte la información.

La red empleada es similar a la DeeperLSTM-TinyPilotNet debido a sus buenos resultados.

La detección de bordes se realiza sobre la imagen tras convertirla al espacio de color HSV y emplear solo la saturación. Tras obtener los bordes de la imagen, las imágenes son reducidas hasta obtener la resolución de entrada de la red (20 x 40 píxeles), para aplicarles finalmente una fusión tipo suma. Todo este proceso se recoge en el siguiente esquema.

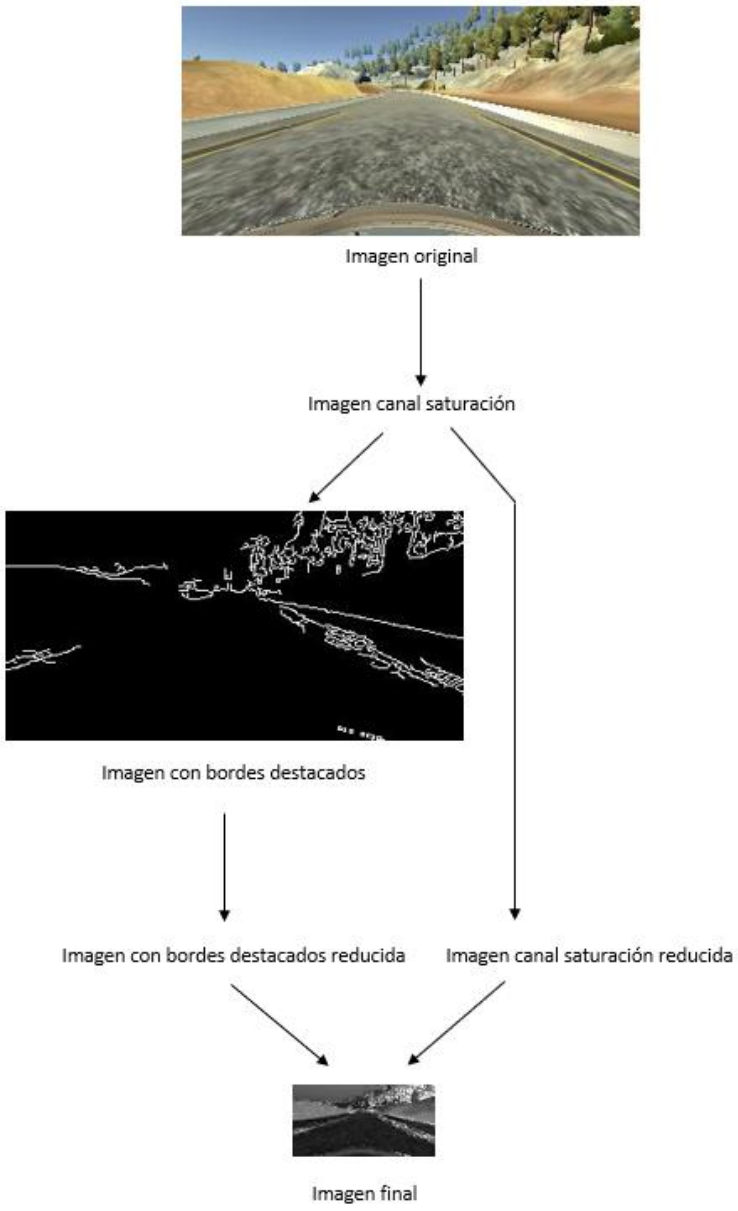

Ilustración 50 Esquema de procesado de imagen para detección de bordes

Los resultados obtenidos son los siguientes:

CNN	RMSE
Edge-DeeperLSTM-TinyPilotNet	0.11852

Tabla 19 RMSE de Edge-DeeperLSTM-TinyPilotNet controlando volante

CNN	Error medio	Error máx.	Cabeceo medio	Cabeceo máx.
Edge-DeeperLSTM-TinyPilotNet	0.71	2.37	2.12	22.69

Tabla 20 Datos extraídos de Edge-DeeperLSTM-TinyPilotNet controlando volante

Las gráficas del vehículo en el simulador empleando esta red son las siguientes.

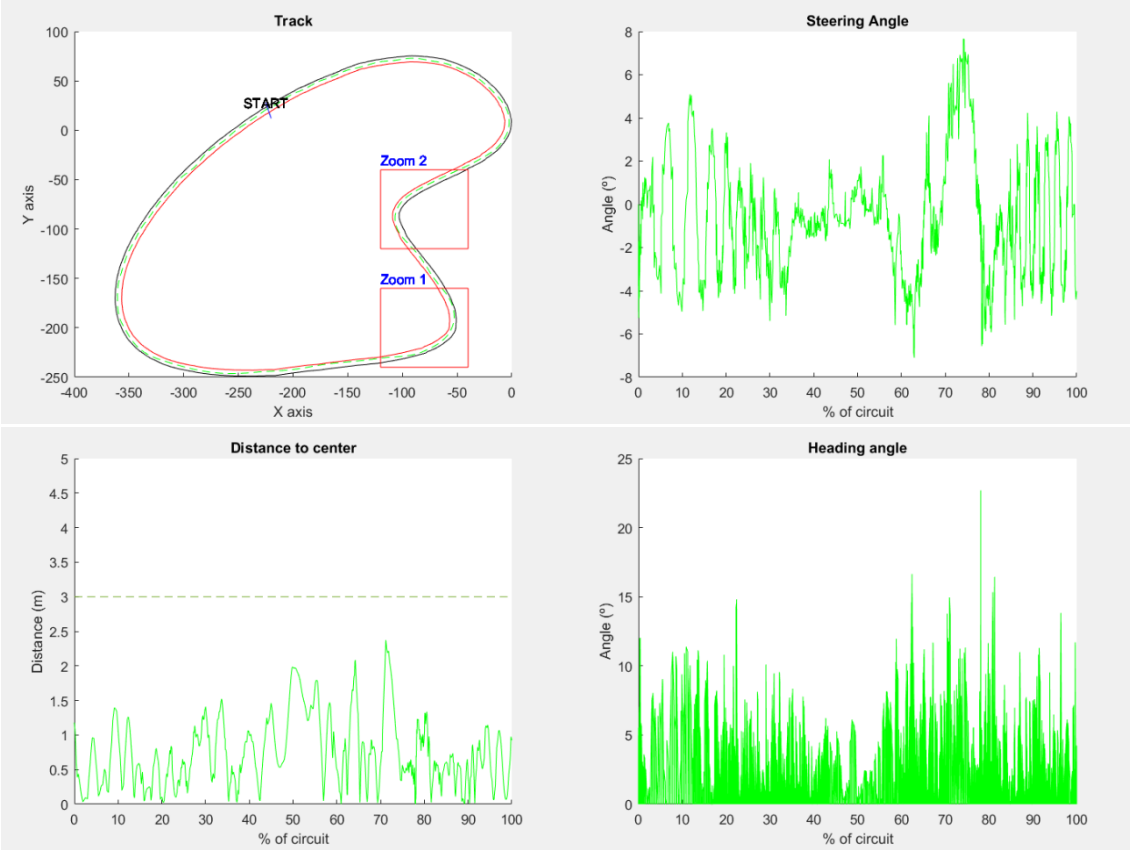

Ilustración 51 Gráfica de resultados Edge-DeeperLSTM-TinyPilotNet controlando volante

Como podemos observar, los resultados mejoran los valores de la red TinyPilotNet, pero empeoran los obtenidos por la red DeeperLSTM-TinyPilotNet.

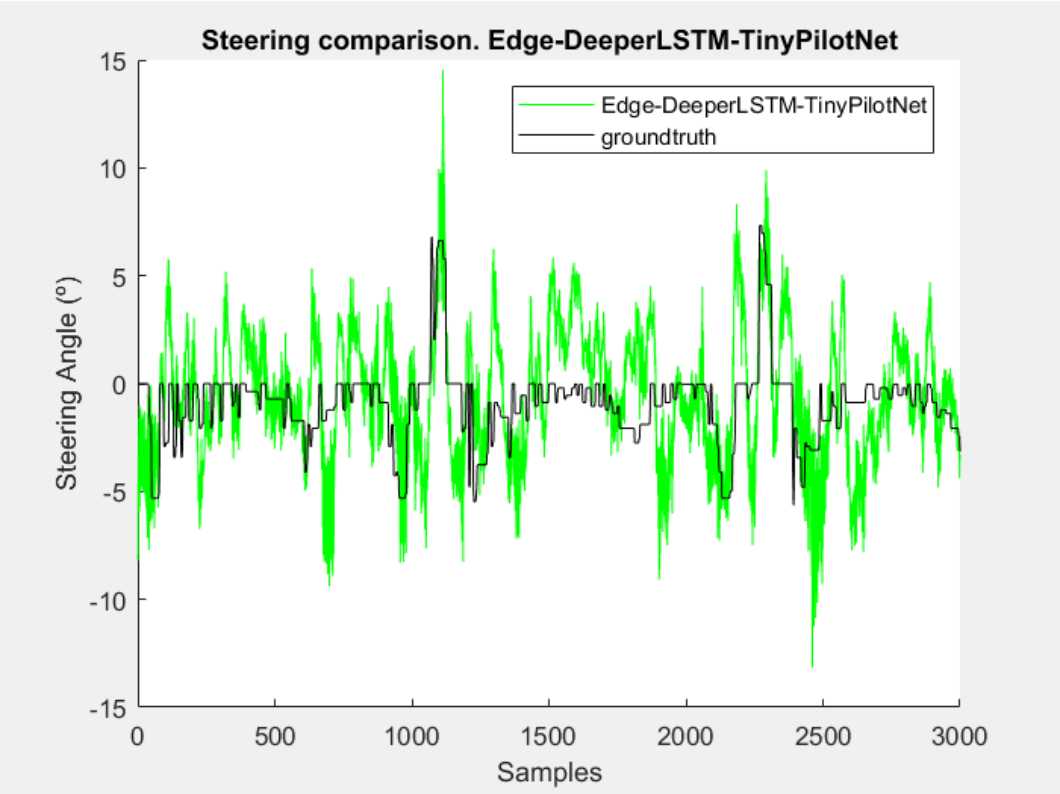

Ilustración 52 Comparativa frame-to-frame de ángulo de volante Edge-DeeperLSTM-TinyPilotNet

6.1.9 Comparativa sobre el control de volante con una CNN

A continuación, se establecerá una comparación entre las redes neuronales convolucionales puestas a prueba en el apartado anterior respecto a la red TinyPilotNet, concluyendo qué elementos mejoran el comportamiento del vehículo y cuáles lo empeoran a la hora de controlar el ángulo del volante.

6.1.9.1 Root-Mean Square Error

La siguiente tabla muestra los valores de error obtenidos para una comparativa fotograma a fotograma realizada para cada una de las CNN con una base de datos del circuito.

CNN	RMSE	Mejora
TinyPilotNet	0.083478	0%
TinyPilotNet de mayor resolución	0.081172	**3%**
HD-TinyPilotNet	0.083799	0%
RGB- TinyPilotNet	0.104	-25%
LSTM- TinyPilotNet	0.08725	-5%
DeeperLSTM- TinyPilotNet	0.095224	-14%
Cropping-DeeperLSTM-TinyPilotNet	0.094075	-13%
Edge-DeeperLSTM-TinyPilotNet	0.11852	-42%

Tabla 21 Comparación de RMSE entre distintas CNNs controlando volante

Se observa que la única red que mejora este parámetro de calidad es la TinyPilotNet de mayor resolución.

Sin embargo, visualmente en el simulador se aprecia que la conducción es mucho mejor mediante las CNN que contienen capas LSTM, como se aprecia en los nuevas métricas de evaluación integradas en este Trabajo Fin de Grado, mucho más realistas.

6.1.9.2 Desviación respecto al centro del carril

En la siguiente tabla se pueden ver los valores medios y máximos para el parámetro de calidad de desviación respecto al centro del carril para cada CNN.

CNN	Error medio	Mejora media	Error máx.	Mejora máx.
TinyPilotNet	0.54	0%	2.41	0%
TinyPilotNet de mayor resolución	0.70	-30%	3.39	-41%
HD-TinyPilotNet	2.07	-196%	9.87	-191%
RGB- TinyPilotNet	2.86	-430%	9.71	-303%
LSTM- TinyPilotNet	0.58	-7%	2.35	2%
DeeperLSTM- TinyPilotNet	0.45	**16%**	1.98	**18%**
Cropping-DeeperLSTM- TinyPilotNet	0.48	11%	2.53	-5%
Edge-DeeperLSTM- TinyPilotNet	0.71	-31%	2.37	2%

Tabla 22 Comparación de desviación al centro del carril entre distintas CNNs controlando volante

En primer lugar, se descarta el empleo de RGB-TinyPilotNet, así como de HD-TinyPilotNet, pues estas redes no son capaces de guiar el vehículo sin salirse de la vía.

Siguiendo el criterio de la mejora del error medio respecto al centro del carril para el resto de las redes, el orden de desempeño de mejor a peor es el siguiente:

1. DeeperLSTM-TinyPilotNet
2. Cropping-DeeperLSTM- TinyPilotNet
3. TinyPilotNet
4. LSTM- TinyPilotNet
5. TinyPilotNet de mayor resolución
6. Edge-DeeperLSTM- TinyPilotNet

Por lo tanto, podemos concluir que la inclusión de capas Long Short-Term Memory en la arquitectura es un factor importante que mejora este parámetro de calidad asociado a un aumento en la resolución de la imagen de entrada.

La red TinyPilotNet de mayor resolución sí supone una mejora respecto a TinyPilotNet, pero si se continúa aumentando la resolución, como en HD-TinyPilotNet, observamos que la red se hace inestable y conduce al vehículo fuera de la vía, por lo que un aumento en la resolución es bueno en cierta medida.

El efecto de cropping sobre la imagen de entrada aporta menor error respecto al centro del carril, mientras que el detector de bordes probado en Edge-DeeperLSTM-TinyPilotNet empeora considerablemente el funcionamiento.

6.1.9.3 Ángulo de cabeceo

En la siguiente tabla se pueden ver los valores medios y máximos para el parámetro de calidad de ángulo de cabeceo para cada CNN.

CNN	Cabeceo medio	Mejora media	Cabeceo máx.	Mejora máx.
TinyPilotNet	2.44	0%	18.91	0%
TinyPilotNet de mayor resolución	1.71	**30%**	23.06	-22%
HD-TinyPilotNet	2.73	-12%	40.33	-113%
RGB- TinyPilotNet	3.69	-51%	44.98	-138%
LSTM- TinyPilotNet	2.10	14%	27.03	-43%
DeeperLSTM- TinyPilotNet	1.98	19%	18.37	**3%**
Cropping-DeeperLSTM- TinyPilotNet	2.43	0%	21.68	-15%
Edge-DeeperLSTM- TinyPilotNet	2.12	13%	22.69	-20%

Tabla 23 Comparación de ángulo de cabeceo entre distintas CNNs controlando volante

En base a este parámetro, una vez descartada la red RGB, el orden de las distintas redes ordenadas de mejor a peor desempeño es el siguiente:

1. TinyPilotNet de mayor resolución
2. DeeperLSTM- TinyPilotNet
3. Edge-DeeperLSTM-TinyPilotNet
4. LSTM-TinyPilotNet
5. Cropping-DeeperLSTM-TinyPilotNet
6. TinyPilotNet

En consecuencia, para la mejora de este parámetro de calidad las modificaciones más adecuadas son la inclusión de capas LSTM (en ningún caso supone una desmejora) y el aumento de la resolución.

El ángulo de cabeceo no mejora realizando un cropping de la imagen de entrada, como en Cropping-DeeperLSTM-TinyPilotNet, mientras que sí se produce una mejora del cabeceo medio mediante la detección de bordes, ya que la red observa mejor dónde se encuentran los límites de la carretera.

En conclusión, se produce una mejora de ambos factores de calidad mediante la inclusión de capas LSTM en la arquitectura y un aumento en la resolución de la imagen de entrada mediante data augmentation, resultando ineficaces los otros métodos de tratamiento de datos.

6.2 Control del ángulo del volante y aceleración mediante CNNs desacopladas

Esta forma de control pretende agregar un grado de dificultad a lo experimentado anteriormente, alcanzando un nivel 2 de autonomía, puesto que ahora se controlarán simultáneamente el ángulo del volante (desplazamiento lateral) y la aceleración del vehículo en cada momento (desplazamiento longitudinal)

Para ello se configurarán dos redes neuronales convolucionales desacopladas, cada una especializada en un tipo de desplazamiento.

En base a los resultados obtenidos en el apartado de control de volante a continuación se testearán las siguientes redes:

- TinyPilotNet
- TinyPilotNet de mayor resolución
- DeeperLSTM-TinyPilotNet
- Edge-DeeperLSTM-TinyPilotNet

Las arquitecturas de estas redes serán similares a las de las utilizadas en el caso de control de ángulo de volante solamente.

6.2.1 TinyPilotNet

La red que controla el ángulo del volante será la misma usada anteriormente, pues su cometido es similar. Por tanto, solamente es necesario entrenar la red que controla la aceleración del vehículo.

Los resultados obtenidos para este nivel de autonomía del vehículo son los siguientes.

CNN	RMSE
TinyPilotNet volante	0.082959
TinyPilotNet acelerador	0.099855

Tabla 24 RMSE para TinyPilotNet controlando volante y aceleración mediante CNNs desacopladas

CNN	Error medio	Error máx.	Cabeceo medio	Cabeceo máx.
TinyPilotNet	0.85	3.35	3.08	26.25

Tabla 25 Datos extraídos de TinyPilotNets controlando volante y acelerador por separado

La gráfica que muestra el recorrido del circuito del que se extraen los datos de la tabla es la siguiente.

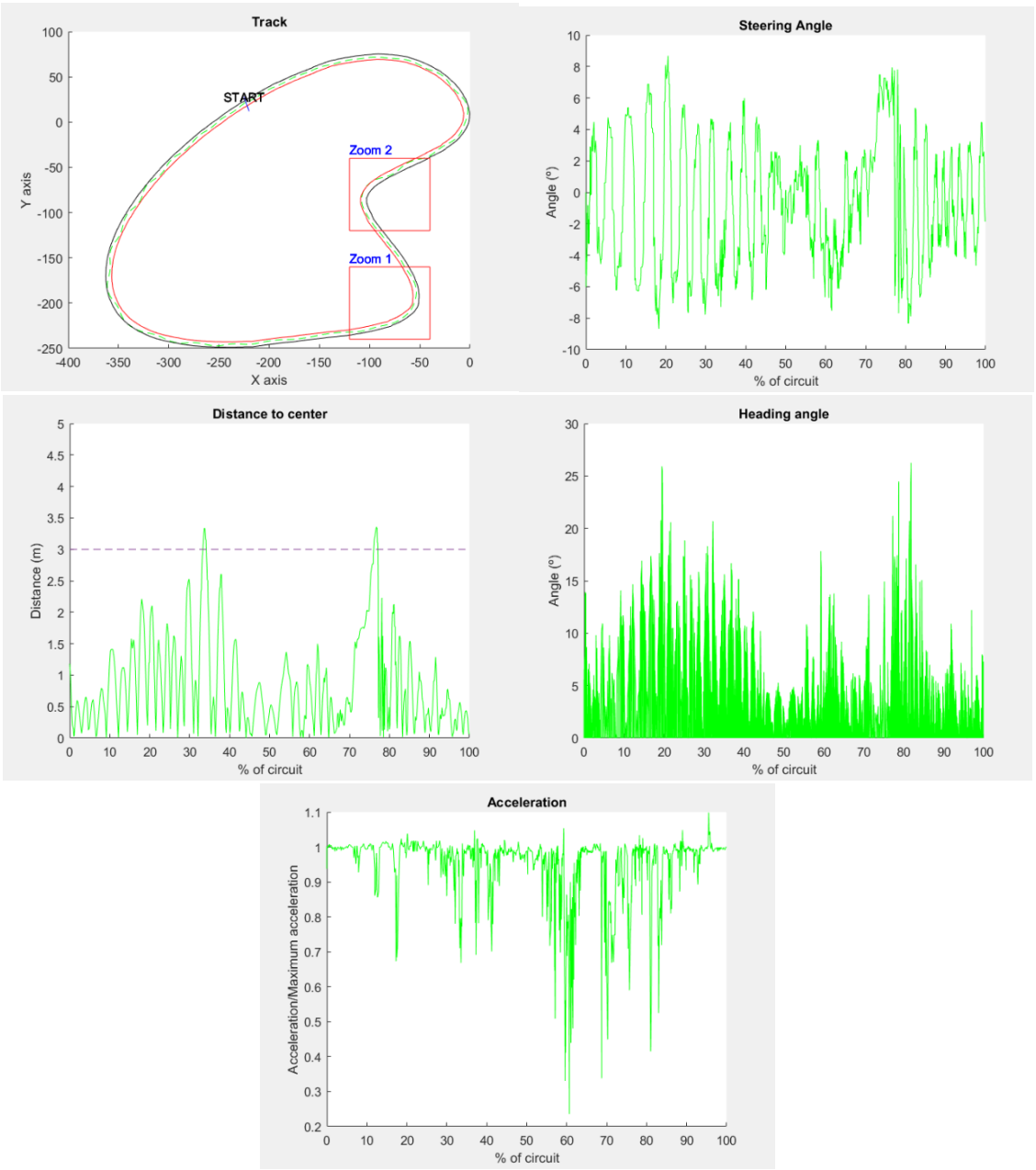

Ilustración 53 Gráfica de TinyPilotNets controlando volante y acelerador por separado

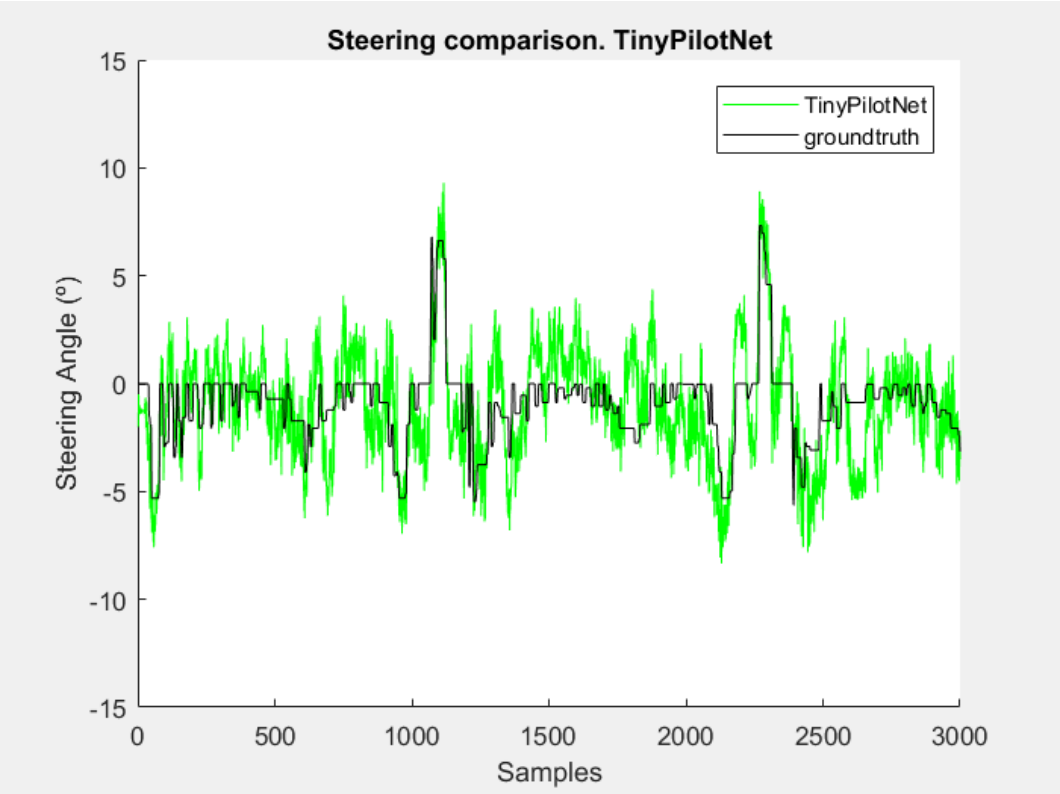

Ilustración 54 Comparativa frame-to-frame de ángulo de volante TinyPilotNet desacoplada

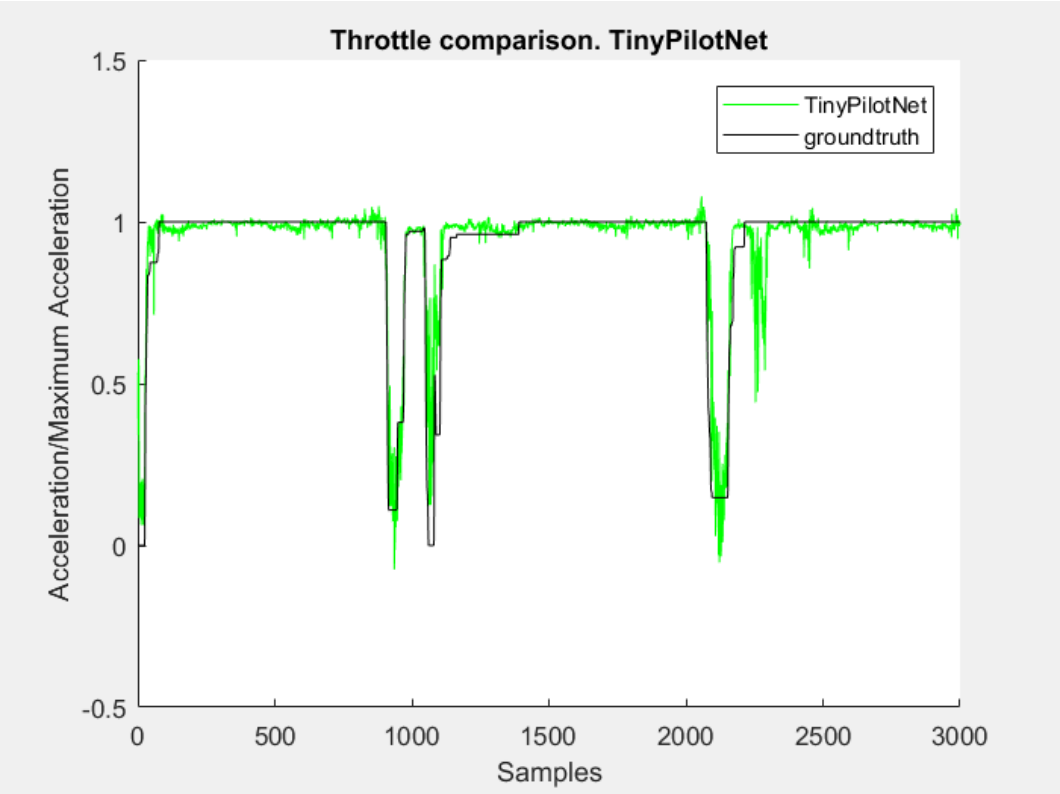

Ilustración 55 Comparativa frame-to-frame de aceleración TinyPilotNet desacoplada

6.2.2 TinyPilotNet de mayor resolución

Al igual que en el caso de la red sencilla, se utiliza la red ya entrenada para el control del volante, entrenando únicamente la red que controla la aceleración.

Los resultados obtenidos son los siguientes.

CNN	RMSE
TinyPilotNet de mayor resolución volante	0.072945
TinyPilotNet de mayor resolución acelerador	0.072789

Tabla 26 RMSE para TinyPilotNet de mayor resolución controlando volante y aceleración mediante CNNs desacopladas

CNN	Error medio	Error máx.	Cabeceo medio	Cabeceo máx.
TinyPilotNet de mayor resolución	0.85	3.39	3.61	29.30

Tabla 27 Datos extraídos de TinyPilotNets de mayor resolución controlando volante y acelerador por separado

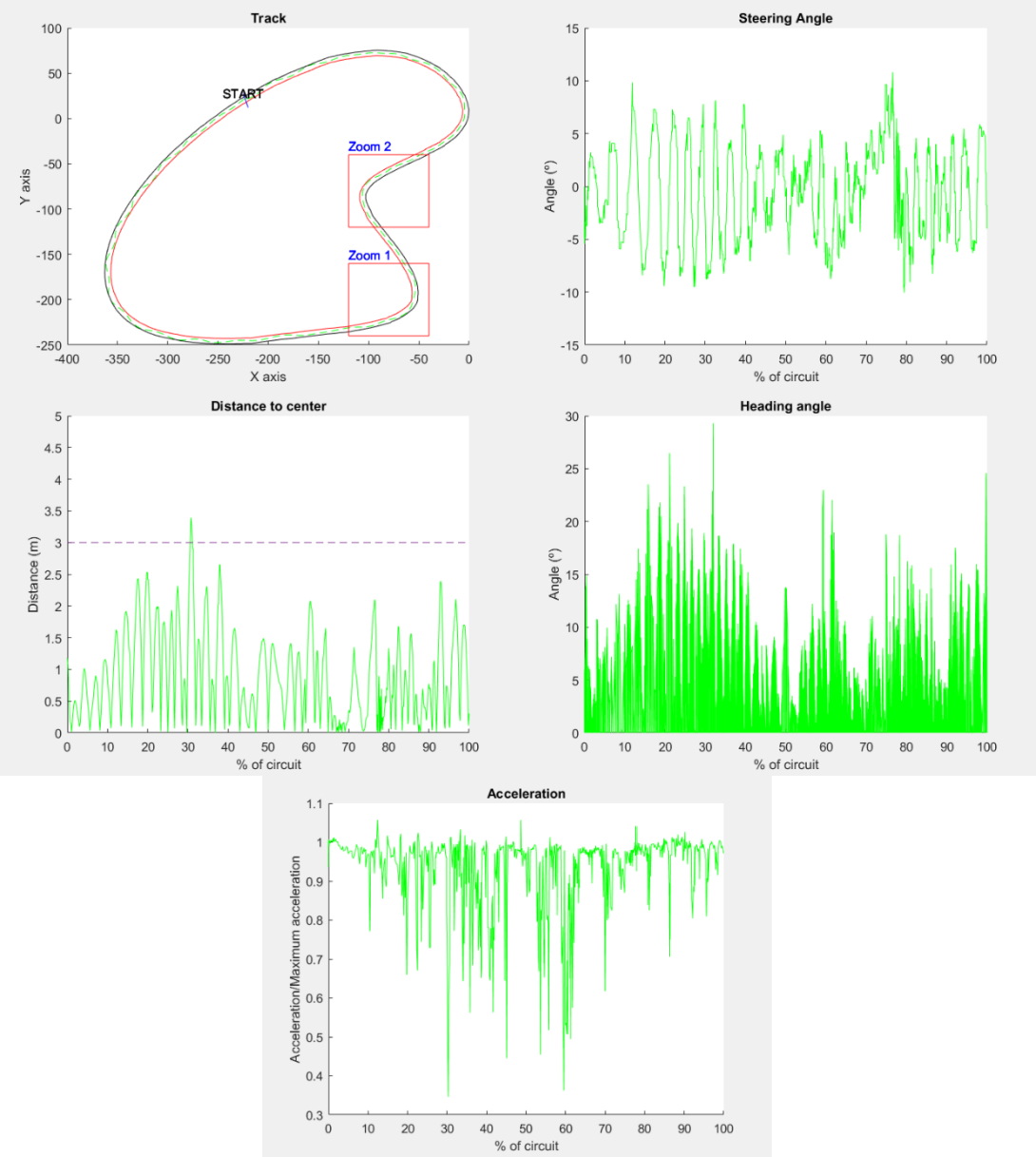

Ilustración 56 Gráfica de TinyPilotNets de mayor resolución controlando volante y acelerador por separado

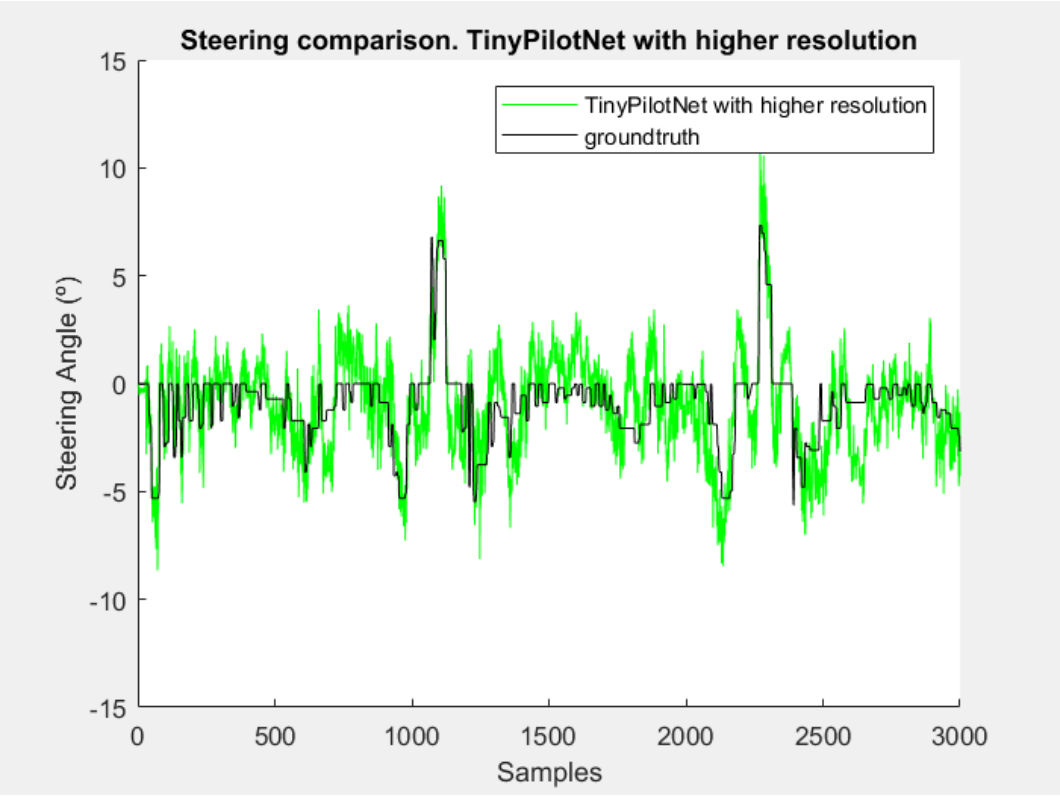

Ilustración 57 Comparativa frame-to-frame de ángulo de volante para TinyPilotNet de mayor resolución desacoplada

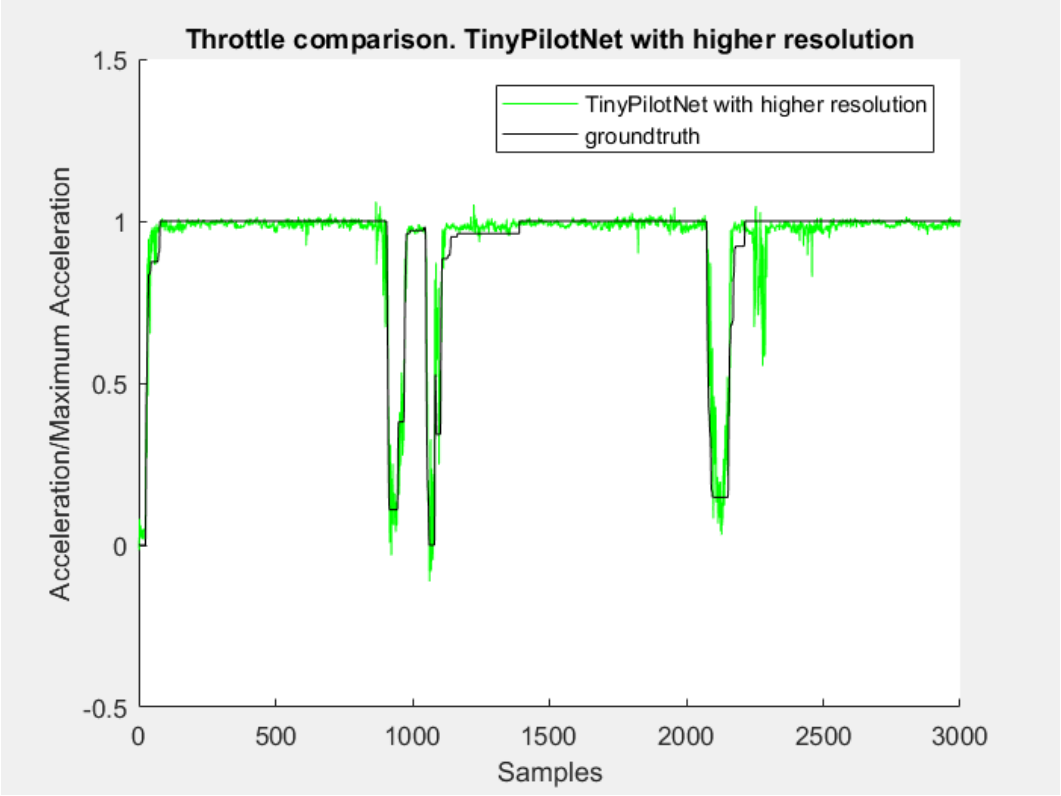

Ilustración 58 Comparativa frame-to-frame de aceleración para TinyPilotNet de mayor resolución desacoplada

6.2.3 DeeperLSTM-TinyPilotNet

Empleando la red con un aumento de resolución en la imagen de entrada y con capas LSTM del apartado de control del ángulo del volante obtenemos los siguientes resultados.

CNN	RMSE
DeeperLSTM-TinyPilotNet volante	0.097846
DeeperLSTM-TinyPilotNet acelerador	0.14497

Tabla 28 RMSE de DeeperLSTM-TinyPilotNet controlando volante y acelerador mediante CNNs desacopladas

CNN	Error medio	Error máx.	Cabeceo medio	Cabeceo máx.
DeeperLSTM-TinyPilotNet	0.95	9.54	3.32	37.23

Tabla 29 Datos extraídos de DeeperLSTM-TinyPilotNet controlando volante y acelerador por separado

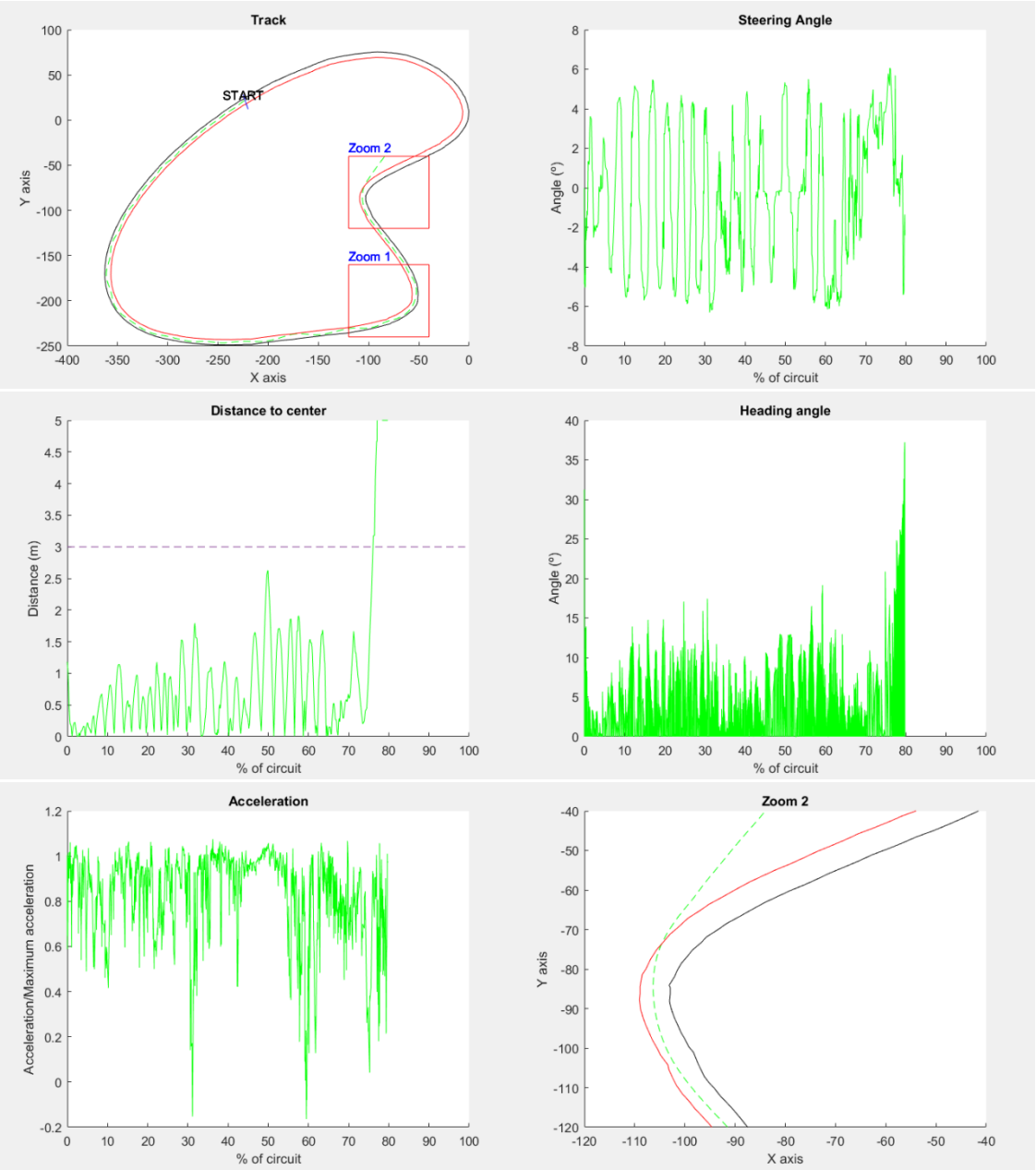

Ilustración 59 Graficas de DeeperLSTM-TinyPilotNet controlando volante y acelerador por separado

Como se observa en la ilustración 59, el vehículo se sale del circuito en la curva Zoom2.

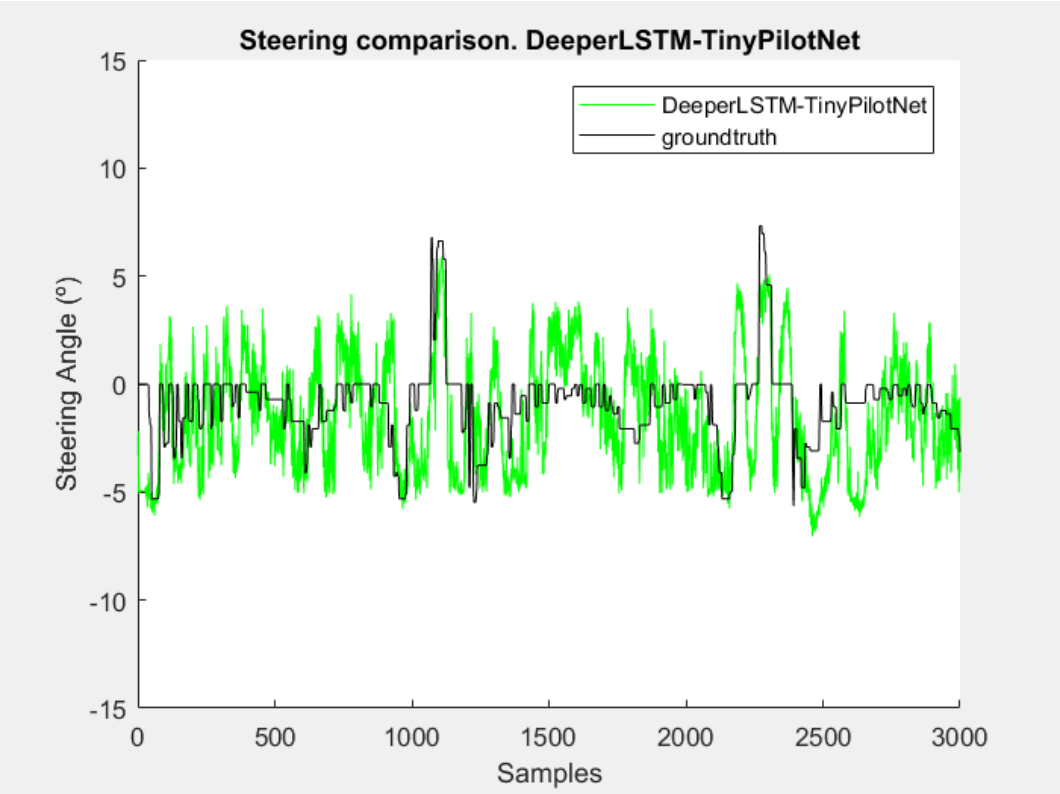

Ilustración 60 Comparativa frame-to-frame de ángulo de volante DeeperLSTM-TinyPilotNet desacoplada

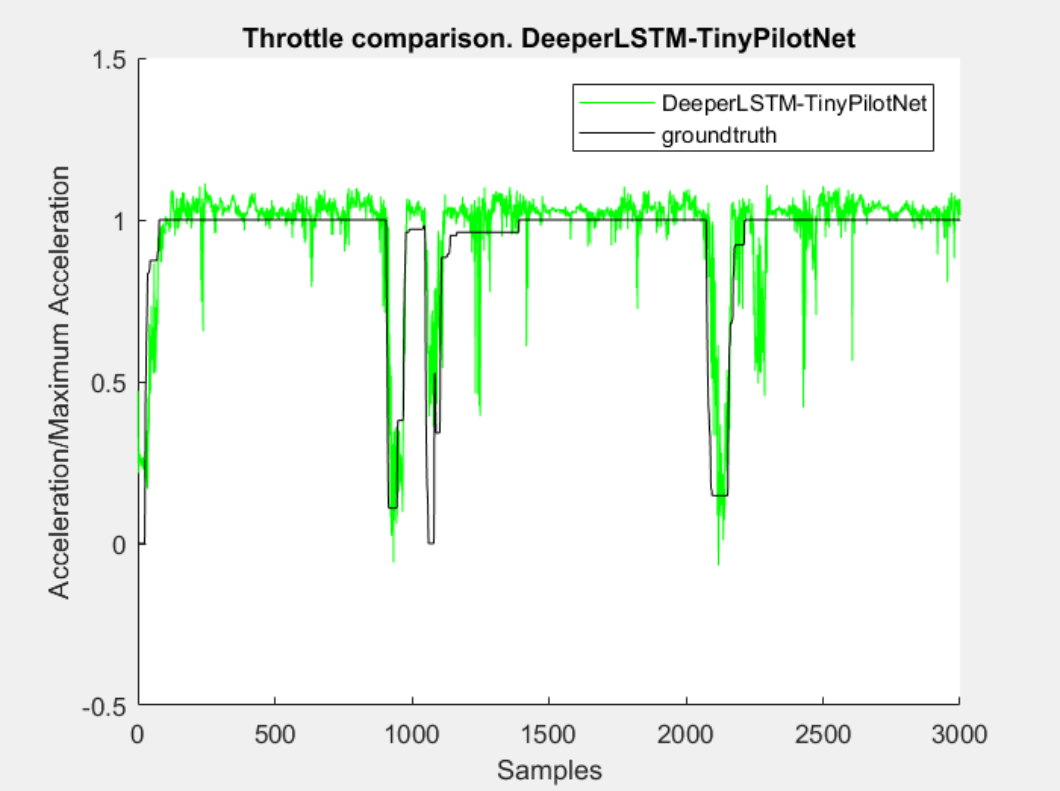

Ilustración 61 Comparativa frame-to-frame de aceleración DeeperLSTM-TinyPilotNet desacoplada

6.2.4 Edge-DeeperLSTM-TinyPilotNet

Para realizar esta prueba se ha empleado el método Canny de detección de bordes descrito en el apartado 4.2.4 sobre la arquitectura DeeperLSTM-TinyPilotNet. A continuación, se muestran los datos de conducción obtenidos.

CNN	RMSE
Edge-DeeperLSTM-TinyPilotNet volante	0.098896
Edge-DeeperLSTM-TinyPilotNet acelerador	0.1621

Tabla 30 RMSE de Edge-DeeperLSTM-TinyPilotNet controlando volante y acelerador mediante CNNs desacopladas

CNN	Error medio	Error máx.	Cabeceo medio	Cabeceo máx.
Edge-DeeperLSTM-TinyPilotNet	2.14	9.12	3.78	35.32

Tabla 31 Datos extraídos de Edge-DeeperLSTM-TinyPilotNet controlando volante y acelerador por separado

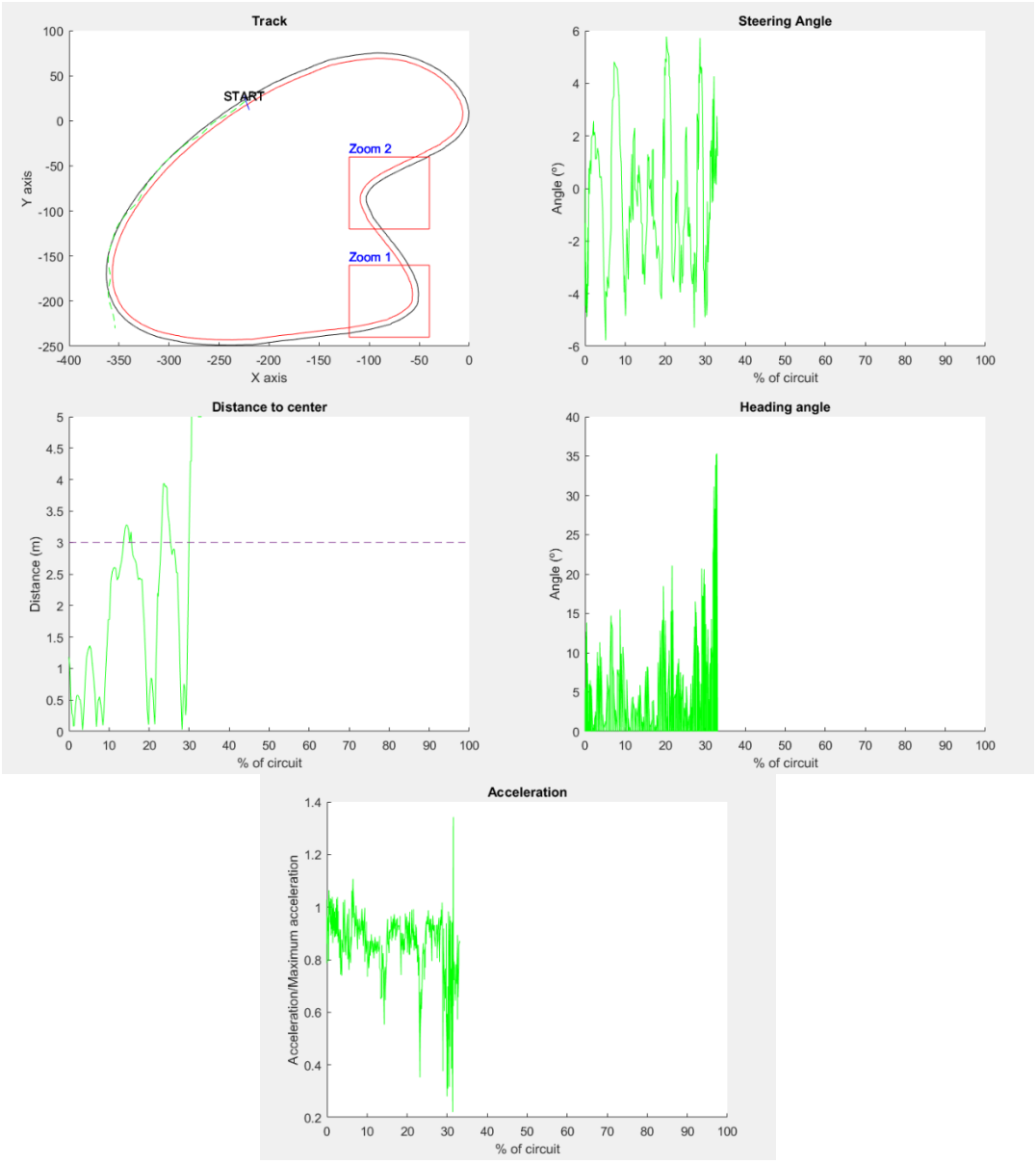

Ilustración 62 Gráfica de Edge-DeeperLSTM-TinyPilotNet controlando volante y acelerador por separado

Como se puede observar, el control de volante y acelerador a través de esta red no es tan eficaz como en el caso en el que únicamente se controla el volante, llegando a producirse una salida de la vía en la primera de las curvas, mientras que en el otro caso el circuito era completado con éxito.

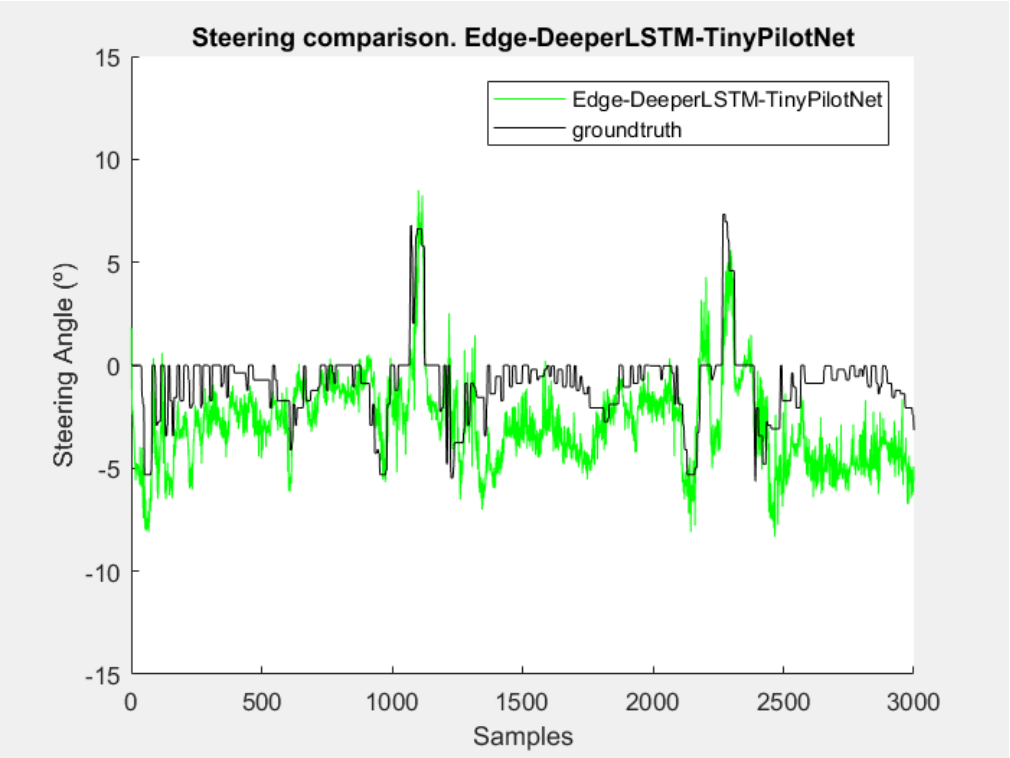

Ilustración 63 Comparativa frame-to-frame de ángulo de volante Edge-DeeperLSTM-TinyPilotNet desacoplada

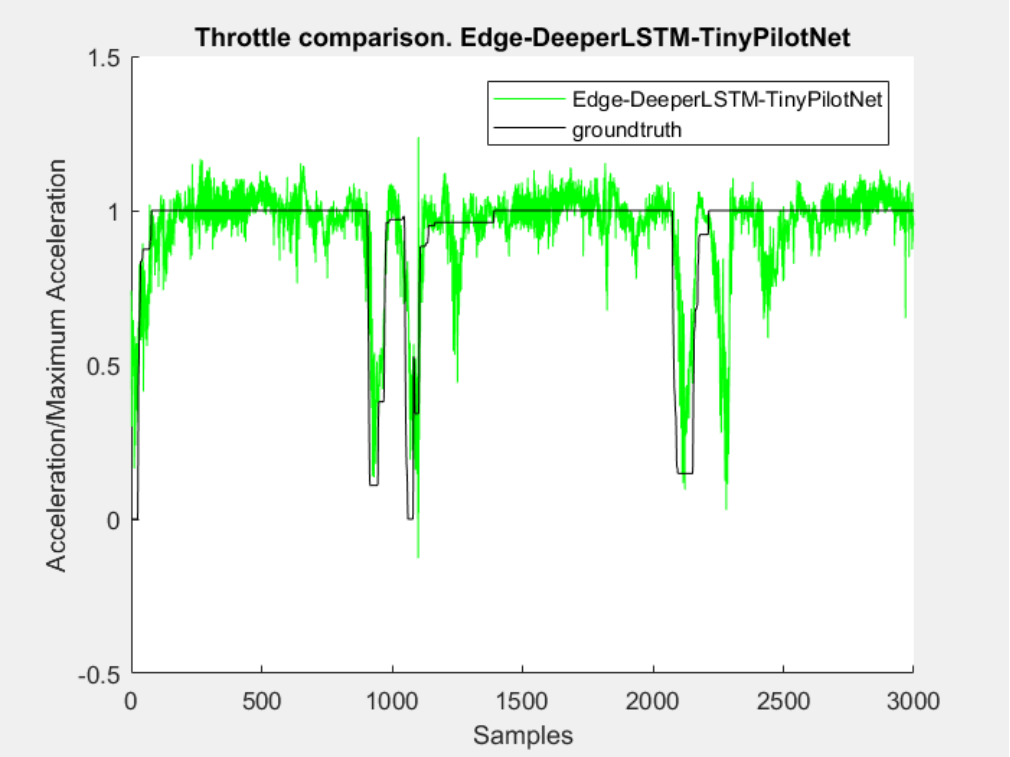

Ilustración 64 Comparativa frame-to-frame de aceleración Edge-DeeperLSTM-TinyPilotNet desacoplada

6.2.5 Comparativa sobre el uso de distintas CNNs para control de volante y acelerador

A continuación, se establecerá una comparación entre las redes neuronales convolucionales puestas a prueba en el apartado anterior respecto a TinyPilotNet, concluyendo qué elementos mejoran el comportamiento del vehículo y cuáles lo empeoran a la hora de controlar el ángulo del volante y la aceleración del vehículo empleando dos CNNs desacopladas. Una de las redes controlará el ángulo de volante mientras que otra proporcionará los valores de aceleración.

6.2.5.1 Root-Mean Square Error

La siguiente tabla muestra los valores de error obtenidos para una comparativa fotograma a fotograma realizada para cada una de las CNN con una base de datos del circuito.

CNN	RMSE	Mejora
TinyPilotNet volante	0.082959	0%
TinyPilotNet acelerador	0.099855	-20%
TinyPilotNet de mayor resolución volante	0.072945	**12%**
TinyPilotNet de mayor resolución acelerador	0.072789	**12%**
DeeperLSTM-TinyPilotNet volante	0.097846	-18%
DeeperLSTM-TinyPilotNet acelerador	0.14497	-75%
Edge-DeeperLSTM-TinyPilotNet volante	0.098896	-19%
Edge-DeeperLSTM-TinyPilotNet acelerador	0.1621	-95%

Tabla 32 Comparación de RMSE entre distintas CNNs controlando volante y acelerador acoplados

Se observa que, al igual que en el caso de control de ángulo de volante, la única red que mejora este parámetro de calidad es la TinyPilotNet de mayor resolución, mientras que las redes LSTM salen peor paradas

6.2.5.2 Desviación respecto al centro del carril

En la siguiente tabla se pueden ver los valores medios y máximos para el parámetro de calidad de desviación respecto al centro del carril para cada CNN.

CNN	Error medio	Mejora media	Error máx.	Mejora máx.
TinyPilotNet	0.85	0%	3.35	**0%**
TinyPilotNet de mayor resolución	0.85	**0%**	3.39	-1%
DeeperLSTM-TinyPilotNet	0.95	-12%	9.54	-185%
Edge-DeeperLSTM-TinyPilotNet	2.14	-152%	9.12	-172%

Tabla 33 Comparación de desviación al centro del carril para distintas CNNs separadas controlando volante y acelerador

Se observa que las redes que incluyen capas LSTM no son capaces de mantener al vehículo en el interior del circuito.

La única red que no se desvía iguala en funcionamiento a la red sencilla respecto a este parámetro de calidad.

6.2.5.3 Ángulo de cabeceo

En la siguiente tabla se pueden ver los valores medios y máximos para el parámetro de calidad de ángulo de cabeceo para cada CNN.

CNN	Cabeceo medio	Mejora media	Cabeceo máx.	Mejora máx.
TinyPilotNet	3.08	**0%**	26.25	**0%**
TinyPilotNet de mayor resolución	3.61	-17%	29.30	-12%
DeeperLSTM-TinyPilotNet	3.32	-8%	37.23	-42%
Edge-DeeperLSTM-TinyPilotNet	3.78	-23%	35.32	-35%

Tabla 34 Comparación de ángulo de cabeceo para distintas CNNs separadas controlando volante y acelerador

Ninguna de las CNN entrenadas es capaz de mejorar el factor de ángulo de cabeceo de la red sencilla.

En conclusión, el entrenamiento de redes separadas para controlar el desplazamiento longitudinal y lateral no es un método fiable.

6.2.5.4 Valores de aceleración

Para comparar el desempeño en la aceleración de las diferentes CNN se muestran a continuación los valores de salida aportados por las redes encargadas de controlar este parámetro.

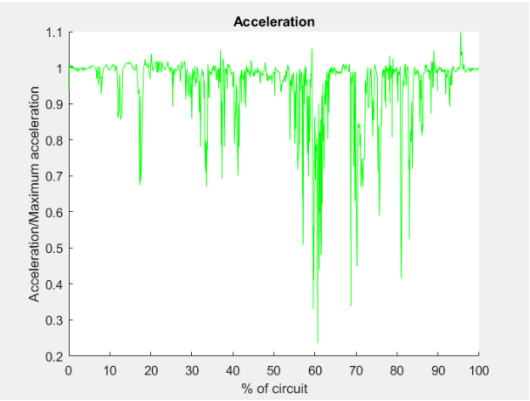

Ilustración 65 Gráfica de aceleración de CNN sencilla

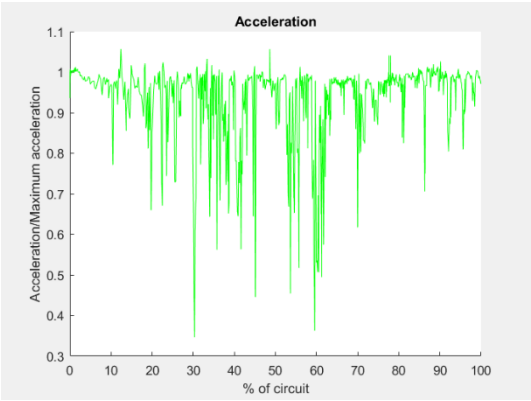

Ilustración 66 Gráfica de aceleración de CNN con mayor resolución

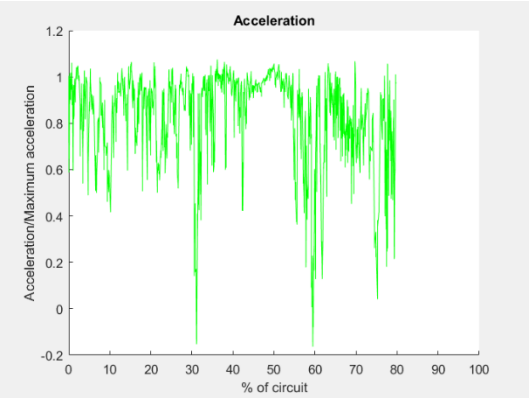

Ilustración 67 Gráfica de aceleración de CNN con LSTM

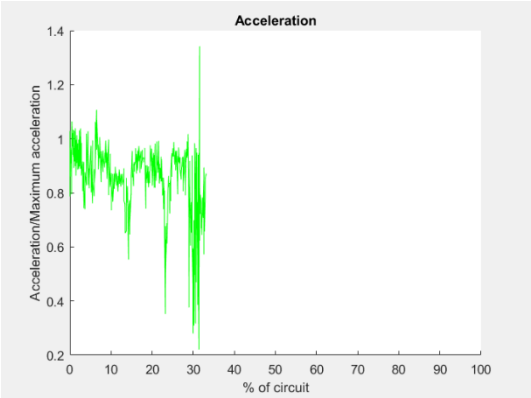

Ilustración 68 Gráfica de aceleración de CNN con LSTM y detección de bordes

Los valores de aceleración pueden oscilar entre -1 (frenado total) y 1 (aceleración máxima).

Las redes que completan el circuito con éxito (red sencilla y con mayor resolución) tienen la mayoría de sus valores próximos a la unidad, y en ciertas zonas como en las curvas del circuito muestran una aceleración menor, adquiriendo los valores del circuito de aprendizaje realizado previamente.

Por otro lado, las redes que no completan el circuito muestran unos valores comprendidos en un margen mayor, llegando a frenar en dos puntos del circuito la red LSTM.

6.3 Control del ángulo del volante y aceleración mediante una misma CNN

Cuando un vehículo es dirigido manualmente, el conductor controla el volante y el acelerador de forma conjunta, llegando a producirse situaciones que correlacionan ambas órdenes, como el decelerar antes de llegar a una curva.

Es por esto por lo que se propone también el control del vehículo por una única red neuronal convolucional, de forma que ésta tenga dos salidas en vez de una, que den los valores de ángulo y aceleración.

Para ello es necesario modificar ligeramente la arquitectura de las redes antes empleadas, cambiando la neurona de salida por un par de neuronas.

Las redes entrenadas siguiendo este método son las siguientes:

- TinyPilotNet.
- TinyPilotNet de mayor resolución.
- DeeperLSTM-TinyPilotNet.
- Edge-DeeperLSTM-TinyPilotNet.
- DeepestLSTM-TinyPilotNet.
- Edge-DeepestLSTM-TinyPilotNet.

6.3.1 TinyPilotNet

La arquitectura de esta red es similar a la TinyPilotNet empleada en los casos anteriores. Únicamente recibe una modificación en la capa de salida, ubicando dos neuronas en lugar de una para ser capaz de proporcionar dos datos de salida.

La siguiente tabla recoge los datos obtenidos en la simulación.

CNN	RMSE
TinyPilotNet	0.0912475

Tabla 35 RMSE de TinyPilotNet controlando volante y acelerador acoplados

CNN	Error medio	Error máx.	Cabeceo medio	Cabeceo máx.
TinyPilotNet	1.07	7.17	3.38	44.91

Tabla 36 Datos extraídos de TinyPilotNet controlando volante y acelerador acoplados

Las gráficas disponibles a continuación muestran el recorrido seguido por el vehículo.

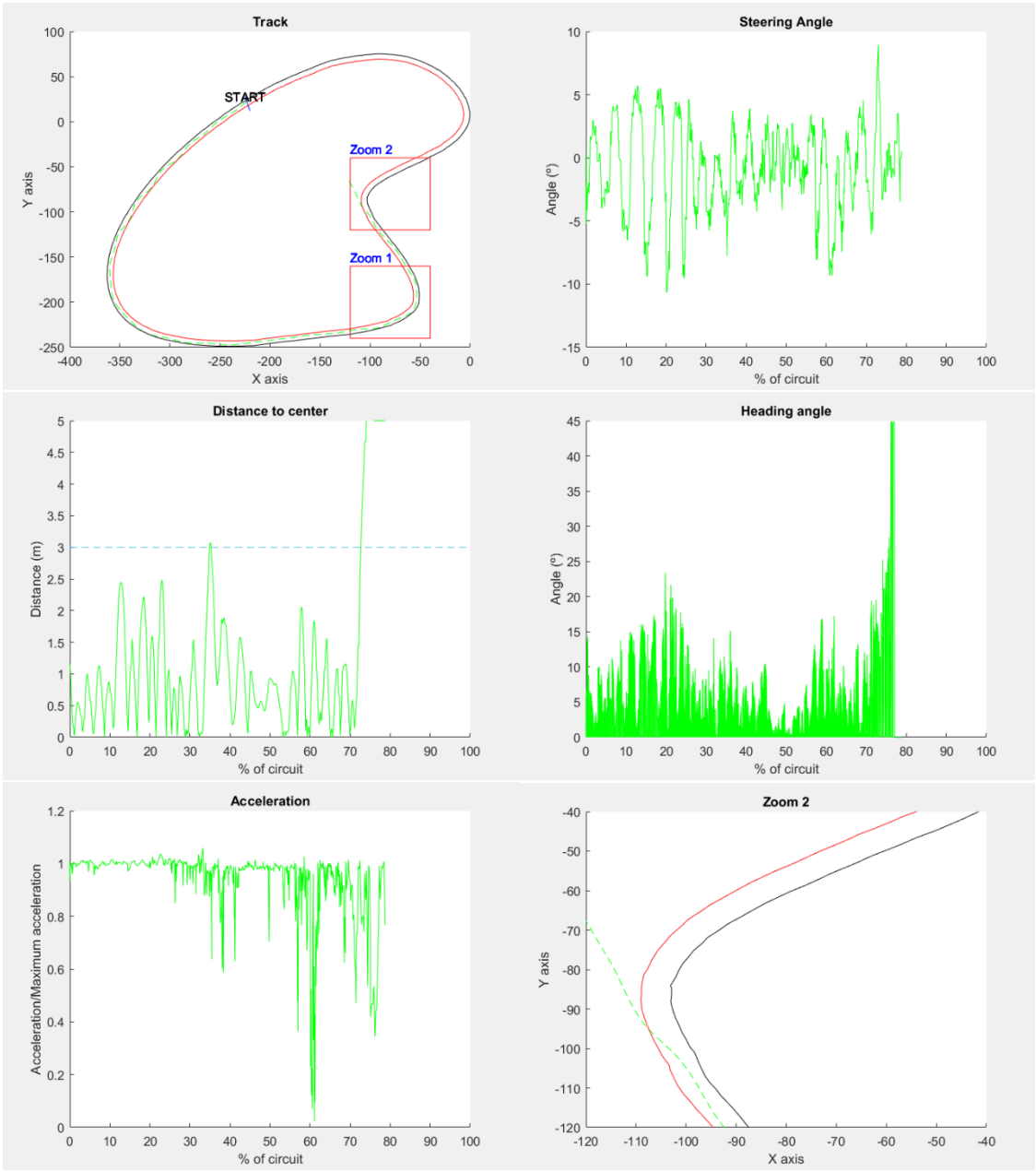

Ilustración 69 Gráficas de resultados de TinyPilotNet controlando volante y acelerador acoplados

Como se puede observar, el vehículo se sale de la vía en la curva remarcada Zoom 2, manteniendo un comportamiento oscilante durante el recorrido del circuito.

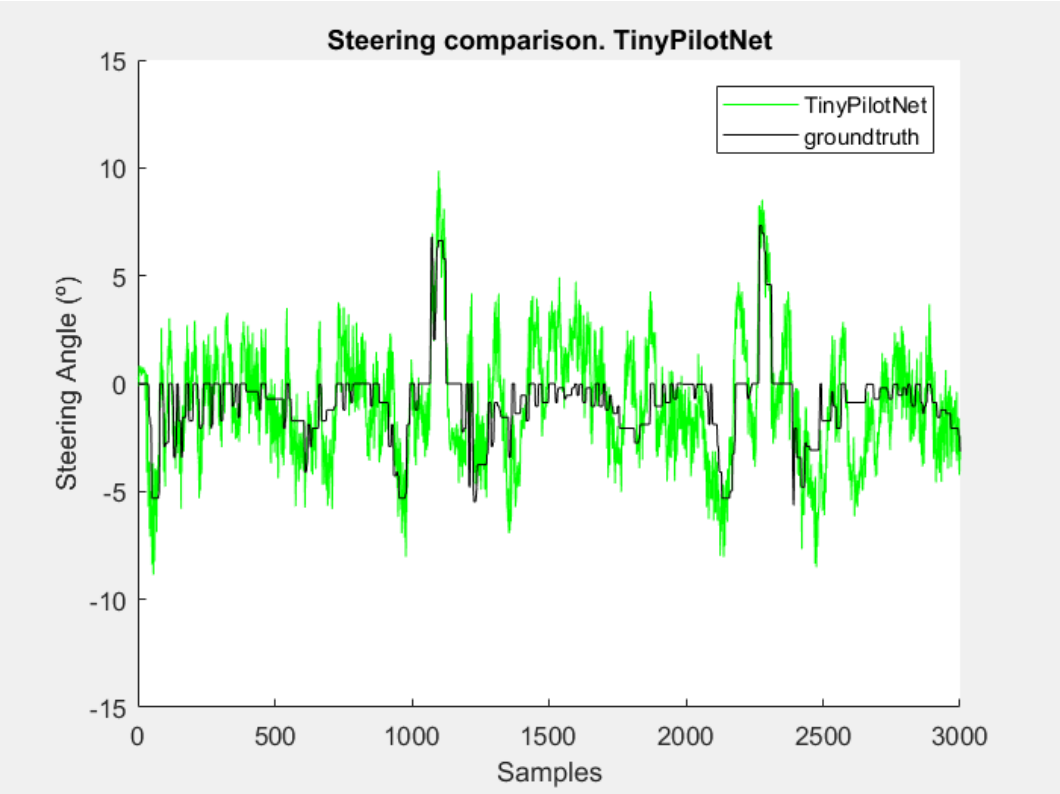

Ilustración 70 Comparativa frame-to-frame de volante TinyPilotNet acoplada

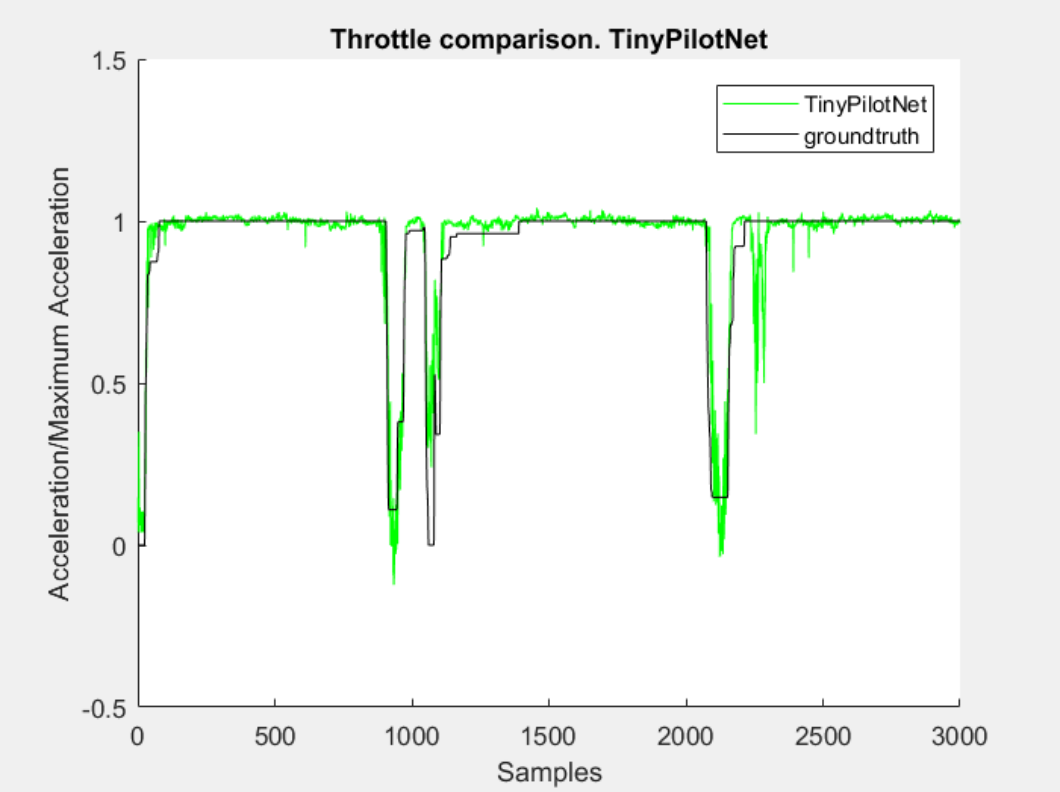

Ilustración 71 Comparativa frame-to-frame de aceleración para TinyPilotNet acoplada

6.3.2 TinyPilotNet de mayor resolución

Al igual que en el apartado anterior, en esta prueba se emplea la red TinyPilotNet de mayor resolución, modificando su salida para obtener dos valores.

Los resultados extraídos de la simulación se muestran en la siguiente tabla y gráficas.

CNN	RMSE
TinyPilotNet de mayor resolución	0.083217

Tabla 37 RMSE de TinyPilotNet de mayor resolución controlando volante y acelerador acoplados

CNN	Error medio	Error máx.	Cabeceo medio	Cabeceo máx.
TinyPilotNet de mayor resolución	0.90	3.62	3.46	31.24

Tabla 38 Datos extraídos de TinyPilotNet de mayor resolución controlando volante y acelerador acoplados

Las gráficas disponibles a continuación muestran el recorrido seguido por el vehículo.

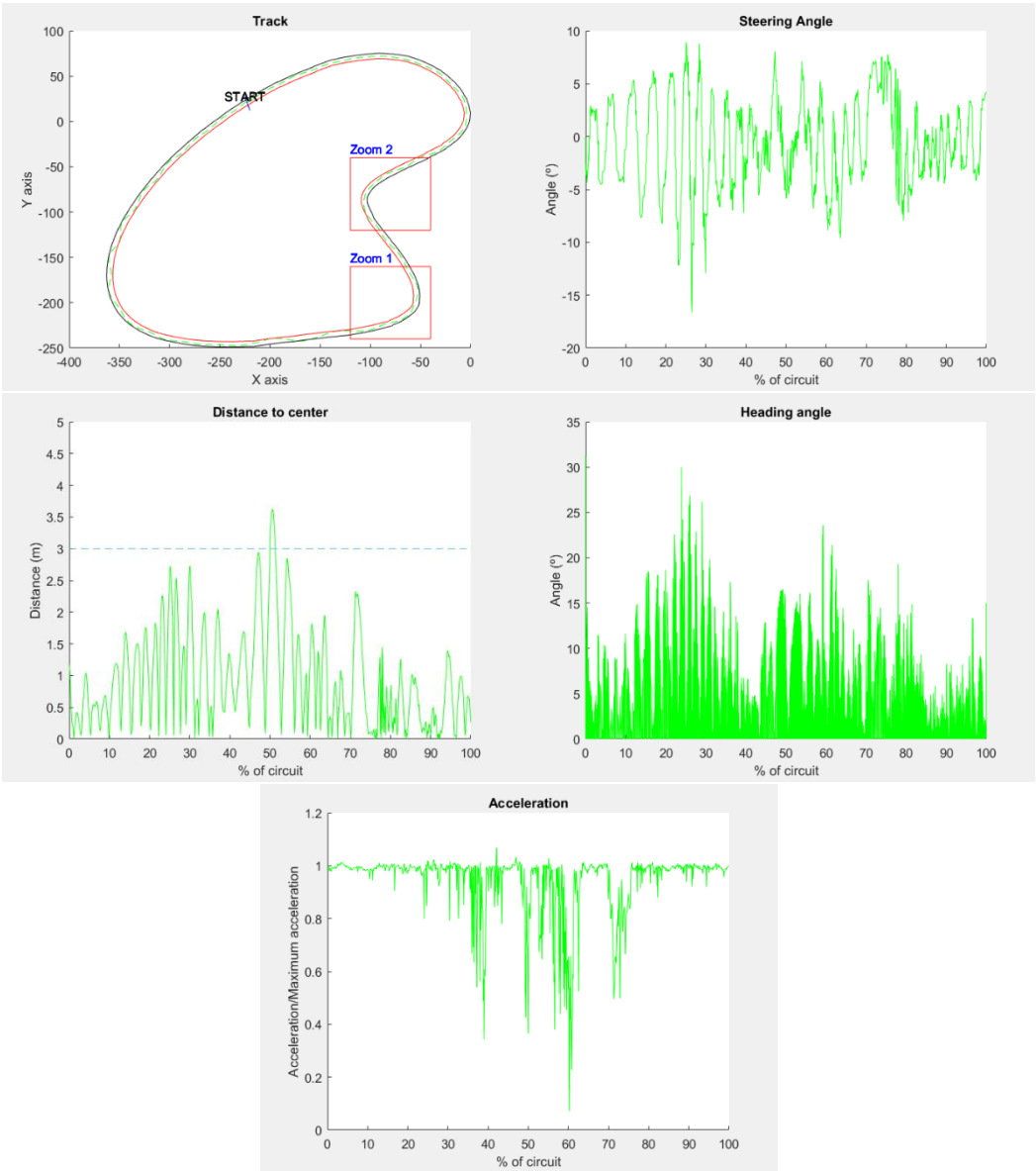

Ilustración 72 Gráfica de TinyPilotNet de mayor resolución controlando volante y acelerador acoplados

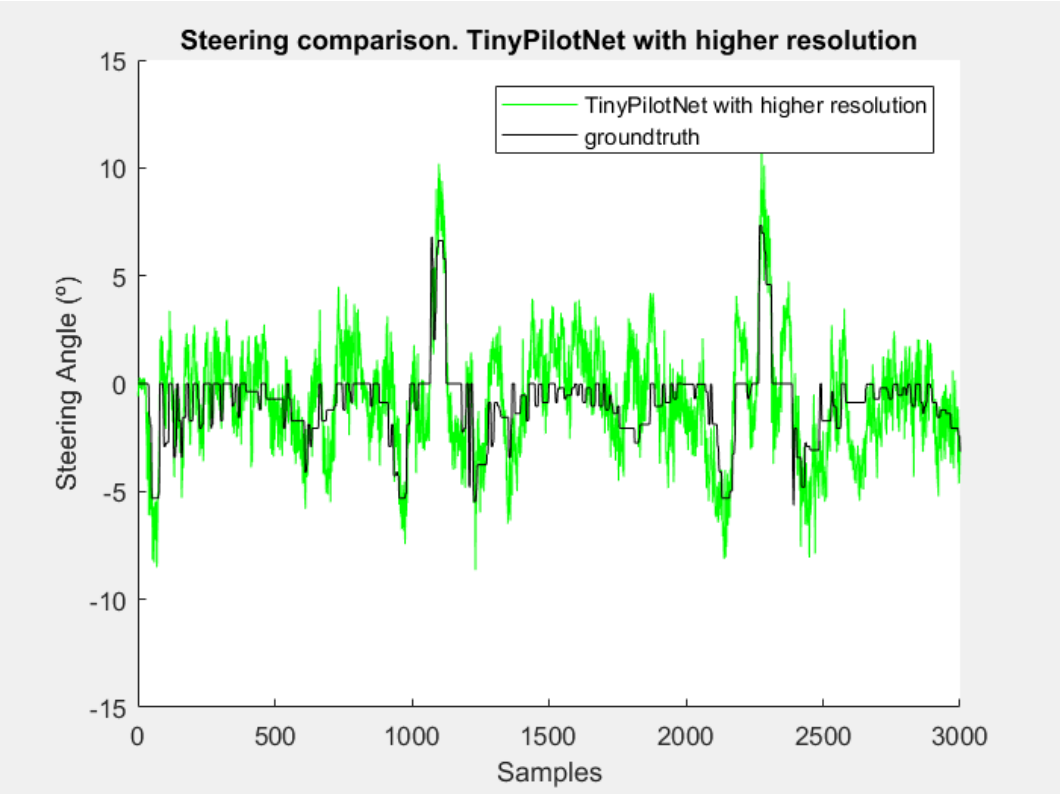

Ilustración 73 Comparativa frame-to-frame de volante TinyPilotNet de mayor resolución acoplada

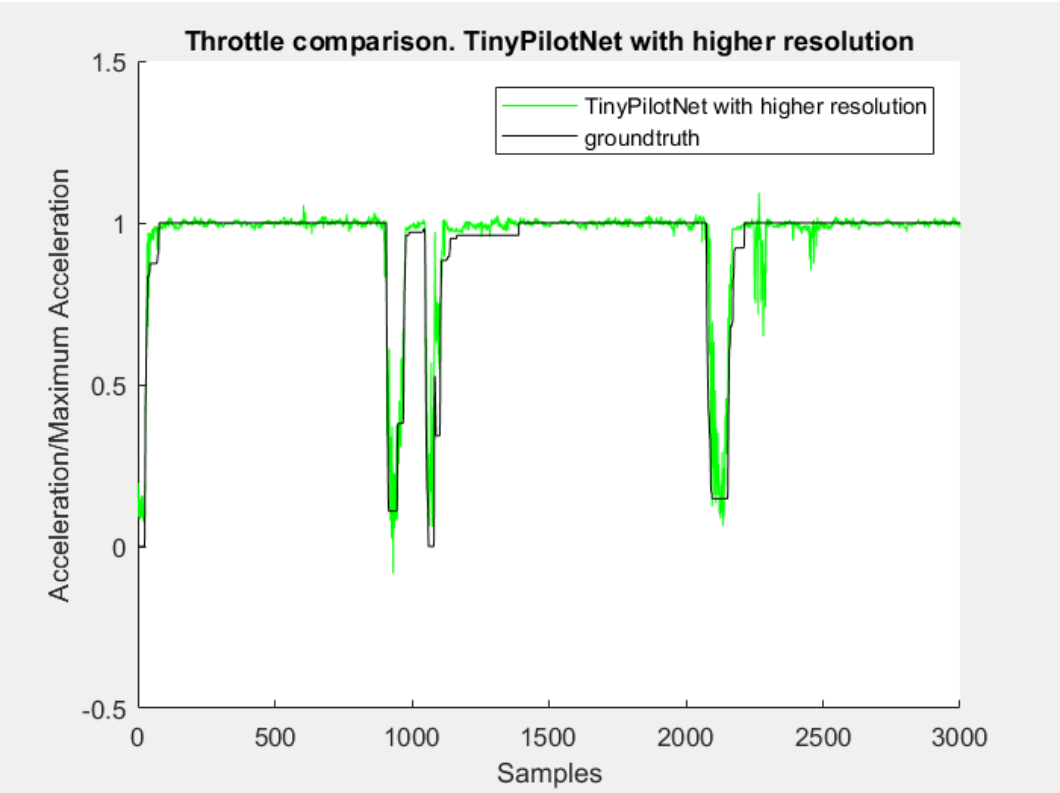

Ilustración 74 Comparativa frame-to-frame de aceleración TinyPilotNet de mayor resolución acoplada

6.3.3 DeeperLSTM-TinyPilotNet

Empleando la misma red DeeperLSTM-TinyPilotNet, modificando su salida, se obtienen los siguientes resultados.

CNN	RMSE
DeeperLSTM-TinyPilotNet	0.16482

Tabla 39 RMSE de DeeperLSTM-TinyPilotNet controlando volante y acelerador acoplados

CNN	Error medio	Error máx.	Cabeceo medio	Cabeceo máx.
DeeperLSTM-TinyPilotNet	1.43	9.71	4.01	35.92

Tabla 40 Datos extraídos de DeeperLSTM-TinyPilotNet controlando volante y acelerador acoplados

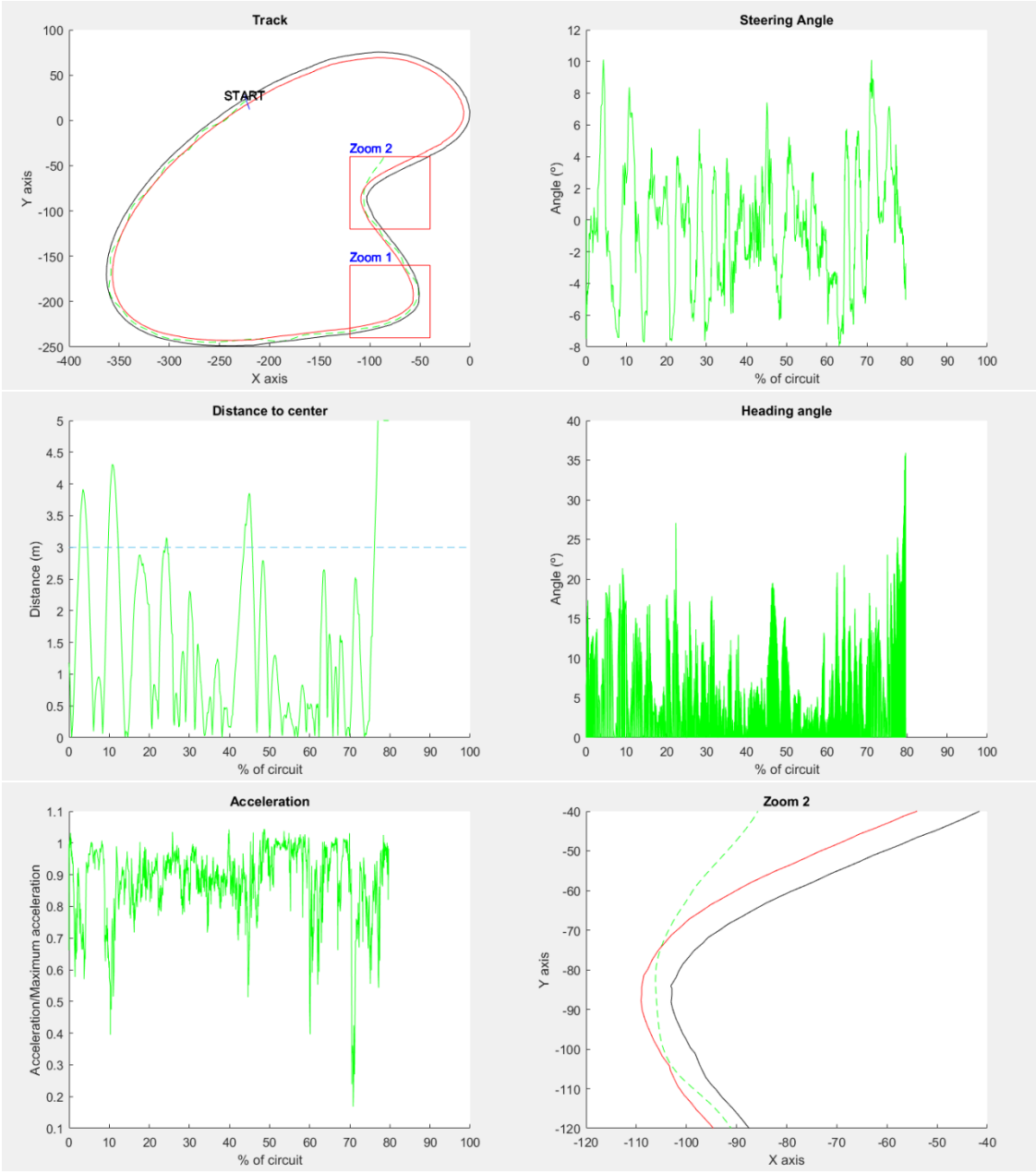

Ilustración 75 Gráfica de DeeperLSTM-TinyPilotNet controlando volante y acelerador acoplados

Como se observa en la gráfica anterior, el vehículo abandona el circuito en la curva Zoom2, produciendo un comportamiento peor que el conseguido en la red que solo controla el volante.

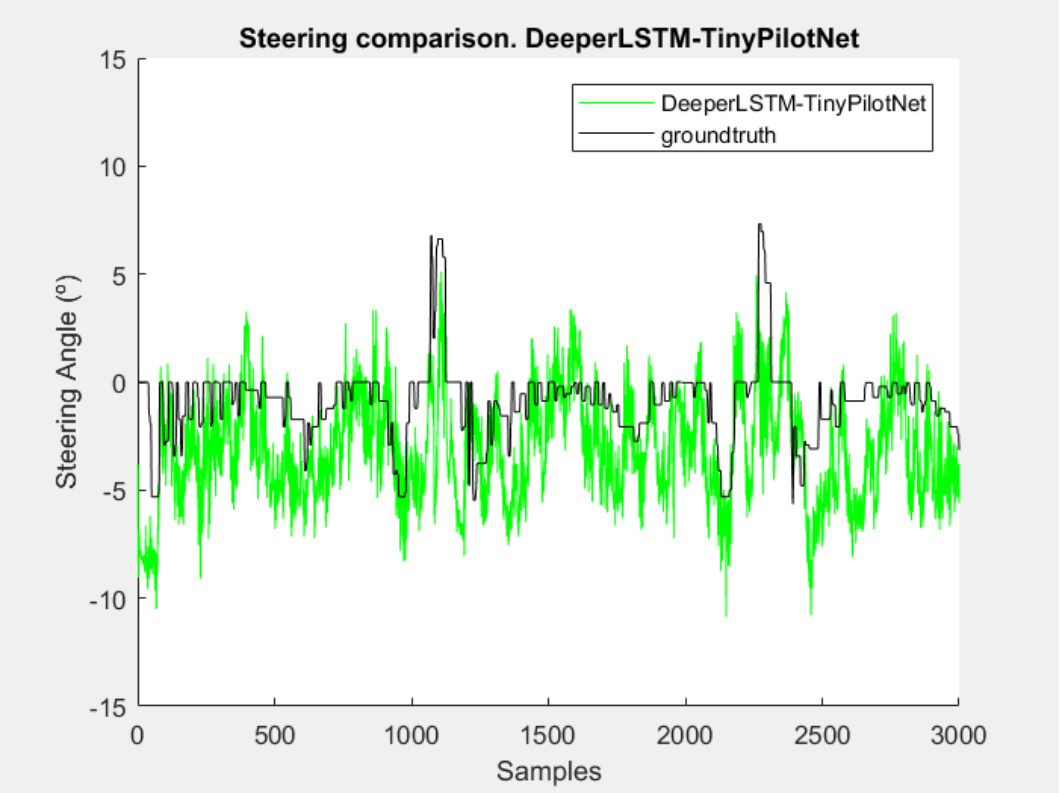

Ilustración 76 Comparativa frame-to-frame de ángulo de volante DeeperLSTM-TinyPilotNet acoplada

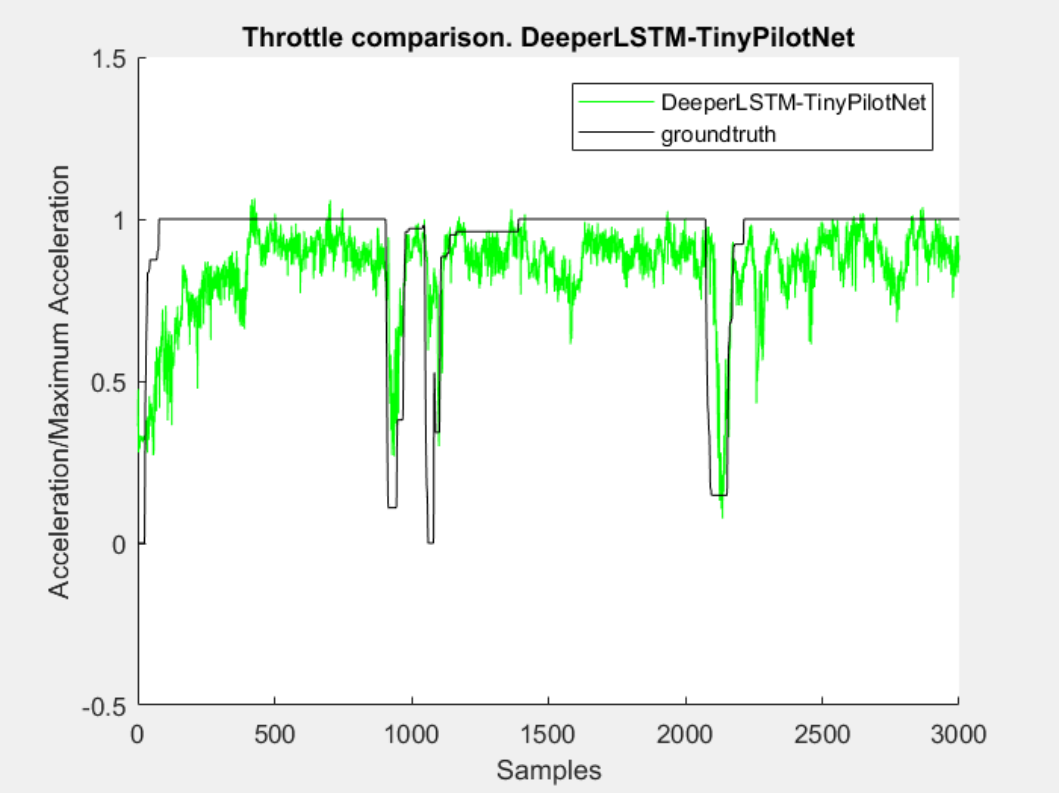

Ilustración 77 Comparativa frame-to-frame de aceleración DeeperLSTM-TinyPilotNet

6.3.4 Edge-DeeperLSTM-TinyPilotNet

Utilizando el método de detección de bordes Canny y la red DeeperLSTM-TinyPilotNet modificando su capa de salida obtenemos los siguientes resultados.

CNN	RMSE
Edge-DeeperLSTM-TinyPilotNet	0.114299

Tabla 41 RMSE de Edge-DeeperLSTM-TinyPilotNet controlando volante y acelerador acoplados

CNN	Error medio	Error máx.	Cabeceo medio	Cabeceo máx.
Edge-DeeperLSTM-TinyPilotNet	1.68	8.25	3.67	44.80

Tabla 42 Datos extraídos Edge-DeeperLSTM-TinyPilotNet controlando volante y acelerador acoplados

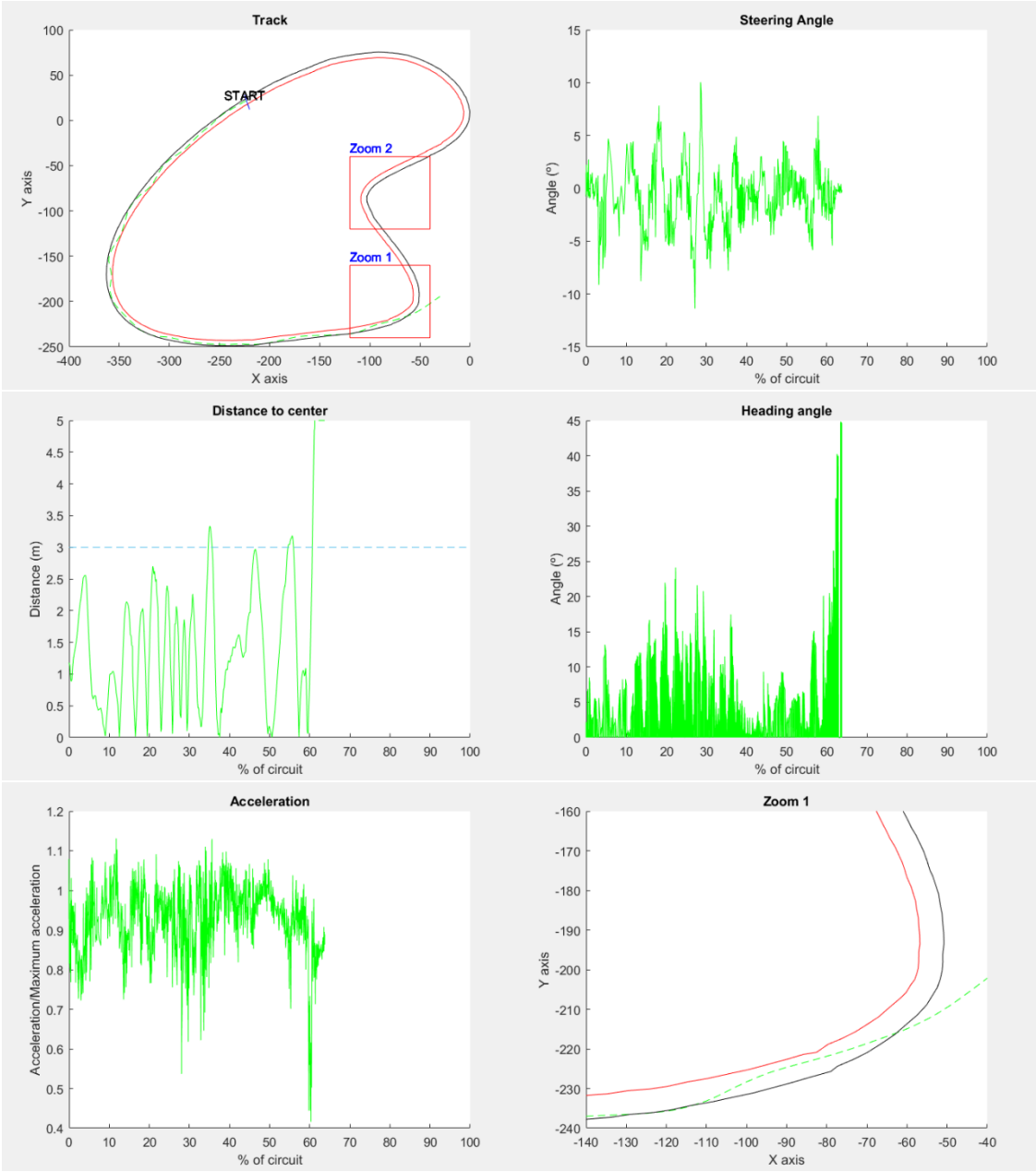

Ilustración 78 Gráfica de Edge-DeeperLSTM-TinyPilotNet controlando volante y acelerador acoplados

Como se observa en la gráfica anterior, el vehículo abandona el circuito en la curva Zoom1.

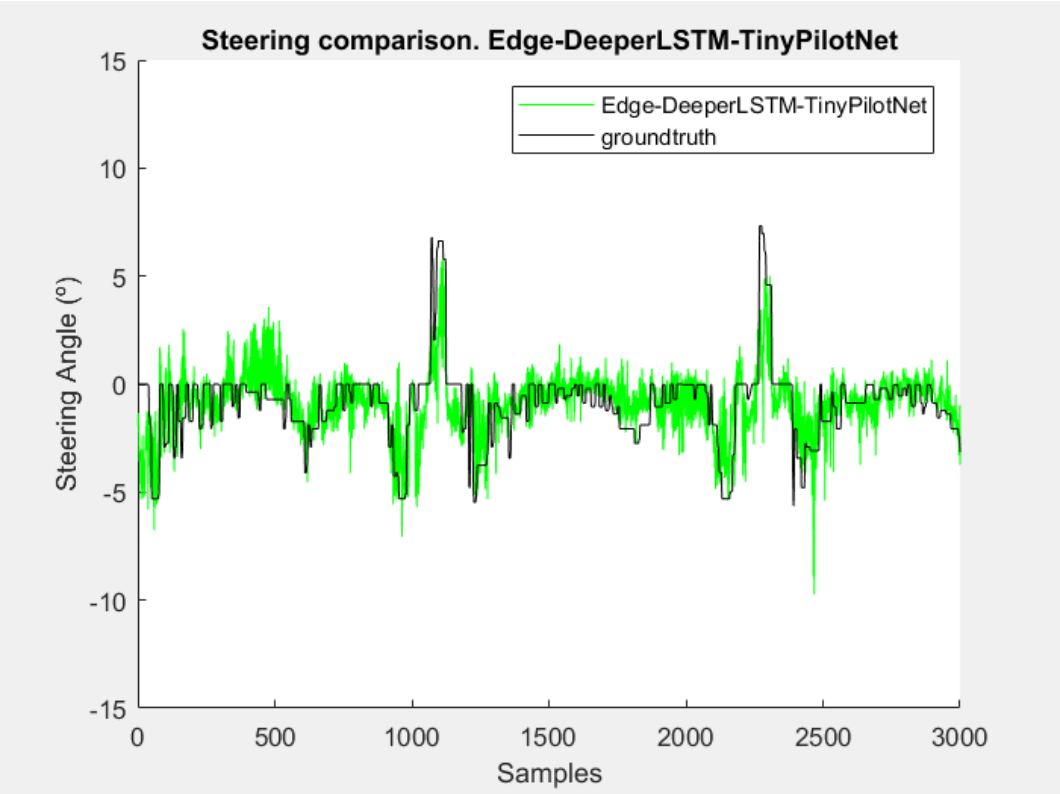

Ilustración 79 Comparativa frame-to-frame de volante Edge-DeeperLSTM-TinyPilotNet acoplada

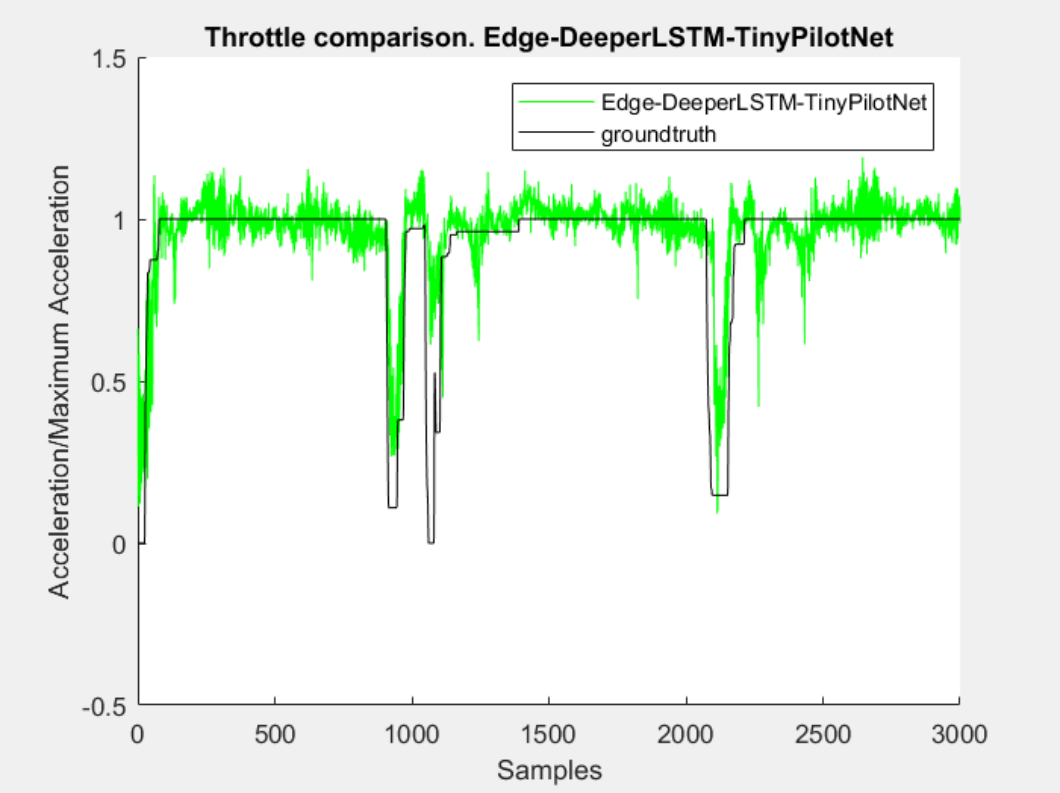

Ilustración 80 Comparativa frame-to-frame de aceleración Edge-DeeperLSTM-TinyPilotNet

6.3.5 DeepestLSTM-TinyPilotNet

Aplicando arquitectura de esta red, descrita en el apartado 4.1.2.2, sobre el simulador de conducción, obtenemos los resultados que se muestran a continuación.

CNN	RMSE
DeepestLSTM-TinyPilotNet	0.116321

Ilustración 81 RMSE de DeepestLSTM-TinyPilotNet controlando volante y acelerador acoplados

CNN	Error medio	Error máx.	Cabeceo medio	Cabeceo máx.
DeepestLSTM-TinyPilotNet	0.72	5.00	2.15	26.36

Tabla 43 Datos extraídos de CNN mayor, con LSTM y detección de bordes controlando volante y acelerador acoplados

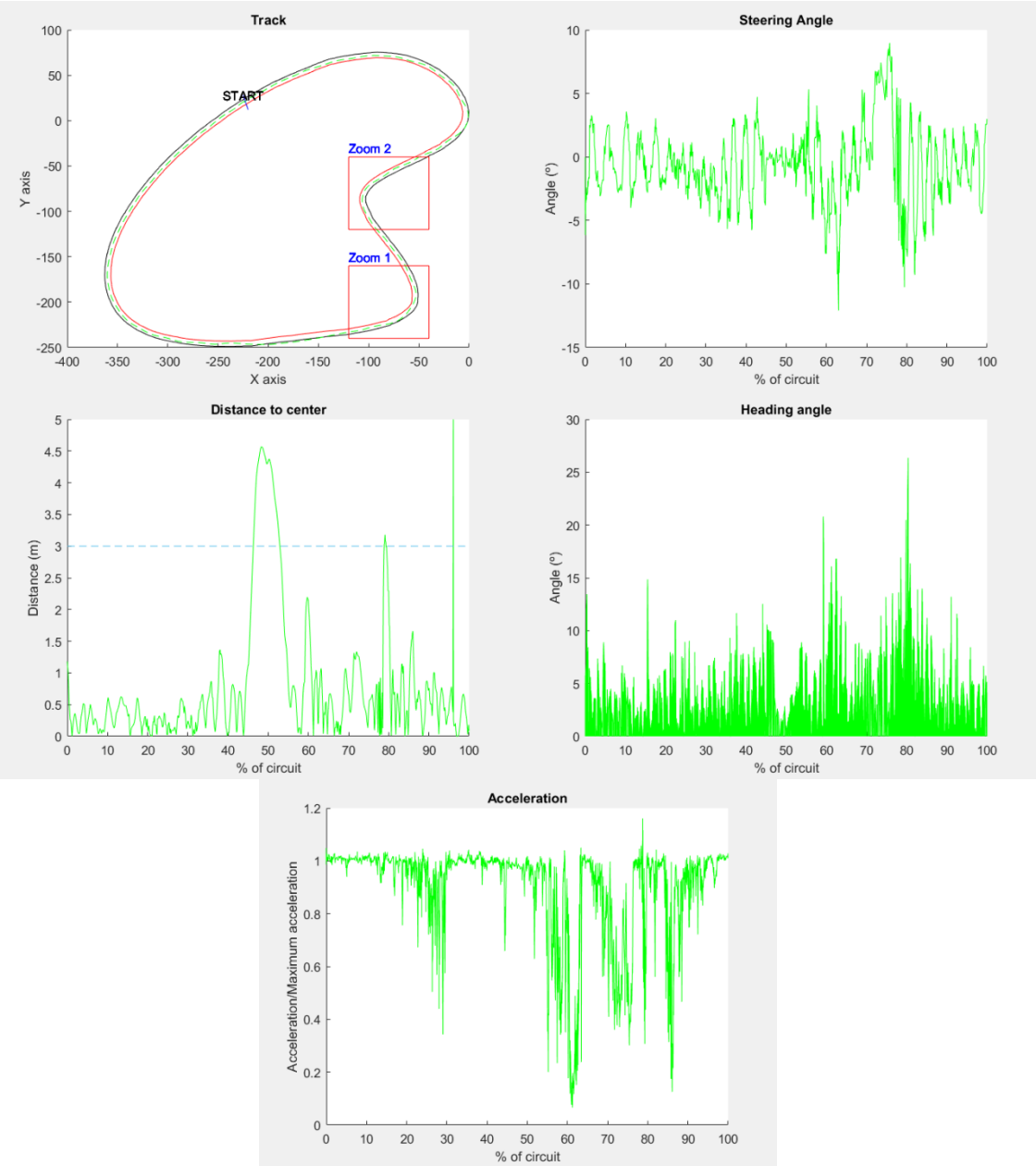

Ilustración 82 Gráfica de DeepestLSTM-TinyPilotNet controlando volante y acelerador acoplados

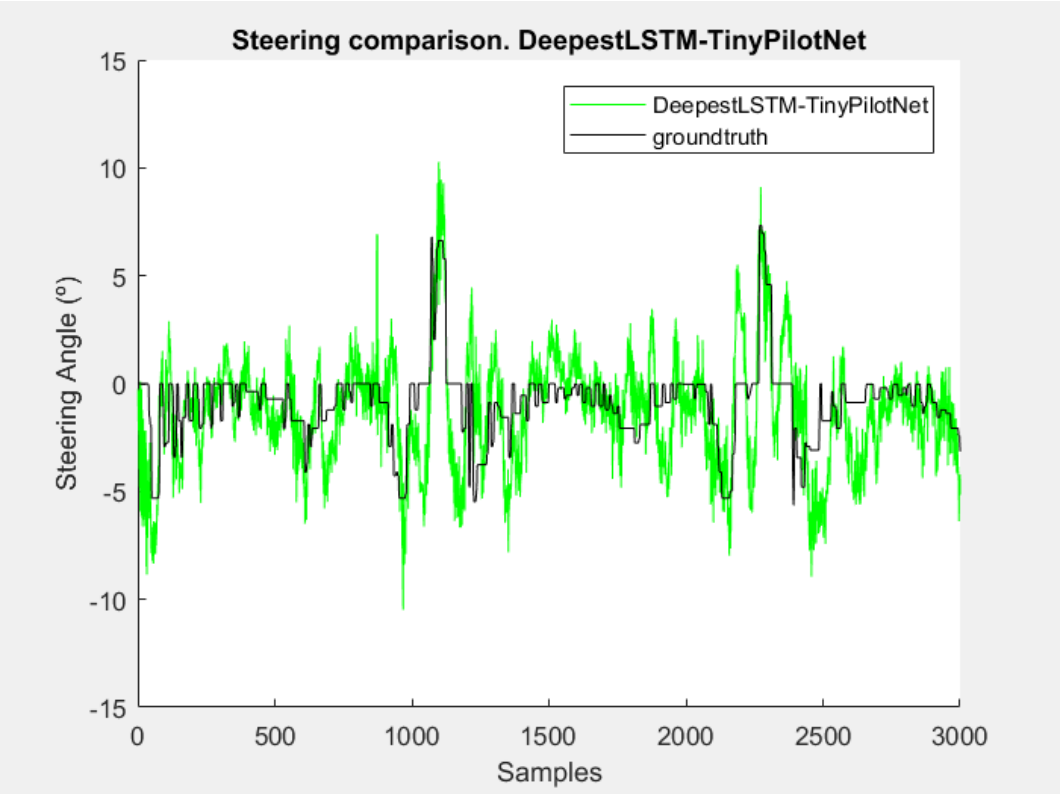

Ilustración 83 Comparativa frame-to-frame de ángulo de volante DeepestLSTM-TinyPilotNet acoplada

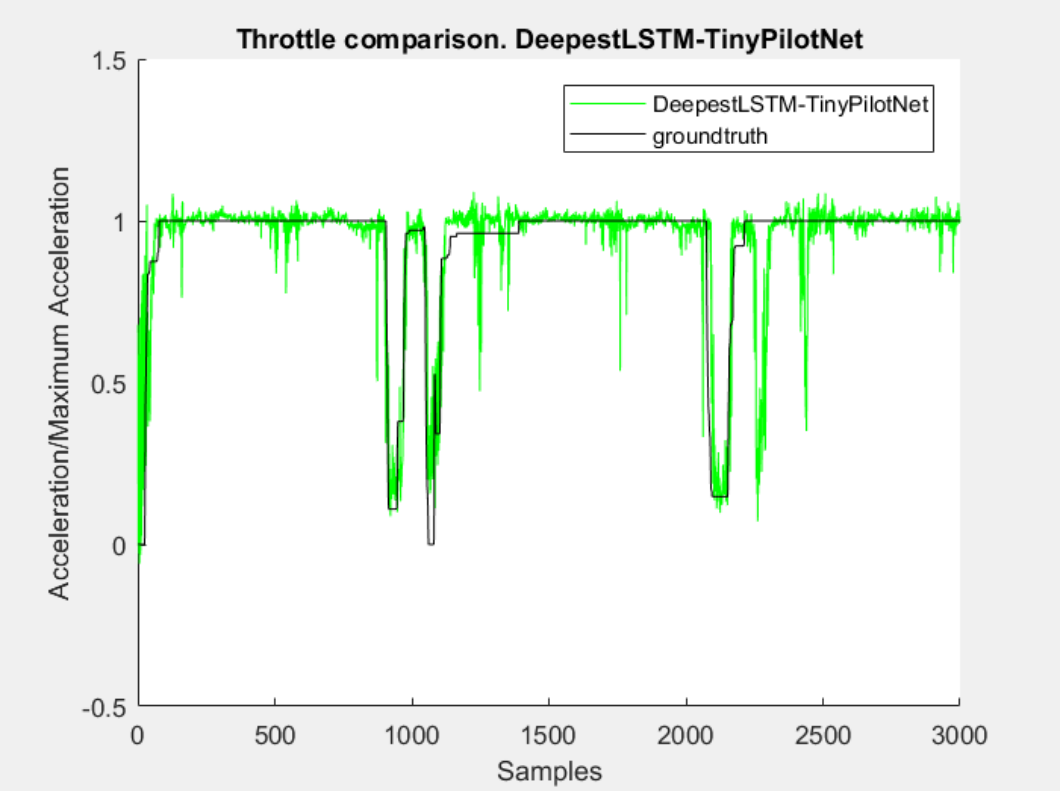

Ilustración 84 Comparativa frame-to-frame de aceleración DeepestLSTM-TinyPilotNet acoplada

6.3.6 Edge-DeepestLSTM-TinyPilotNet

Introduciendo el filtro detector de bordes sobre la arquitectura DeepestLSTM-TinyPilotNet obtenemos los siguientes resultados:

CNN	RMSE
Edge-DeepestLSTM-TinyPilotNet	0.1008005

Tabla 44 RMSE de Edge-DeepestLSTM-TinyPilotNet controlando volante y acelerador acoplados

CNN	Error medio	Error máx.	Cabeceo medio	Cabeceo máx.
Edge-DeepestLSTM-TinyPilotNet	0.91	9.46	2.49	39.53

Tabla 45 Datos extraídos de Edge-DeepestLSTM-TinyPilotNet controlando volante y acelerador acoplados

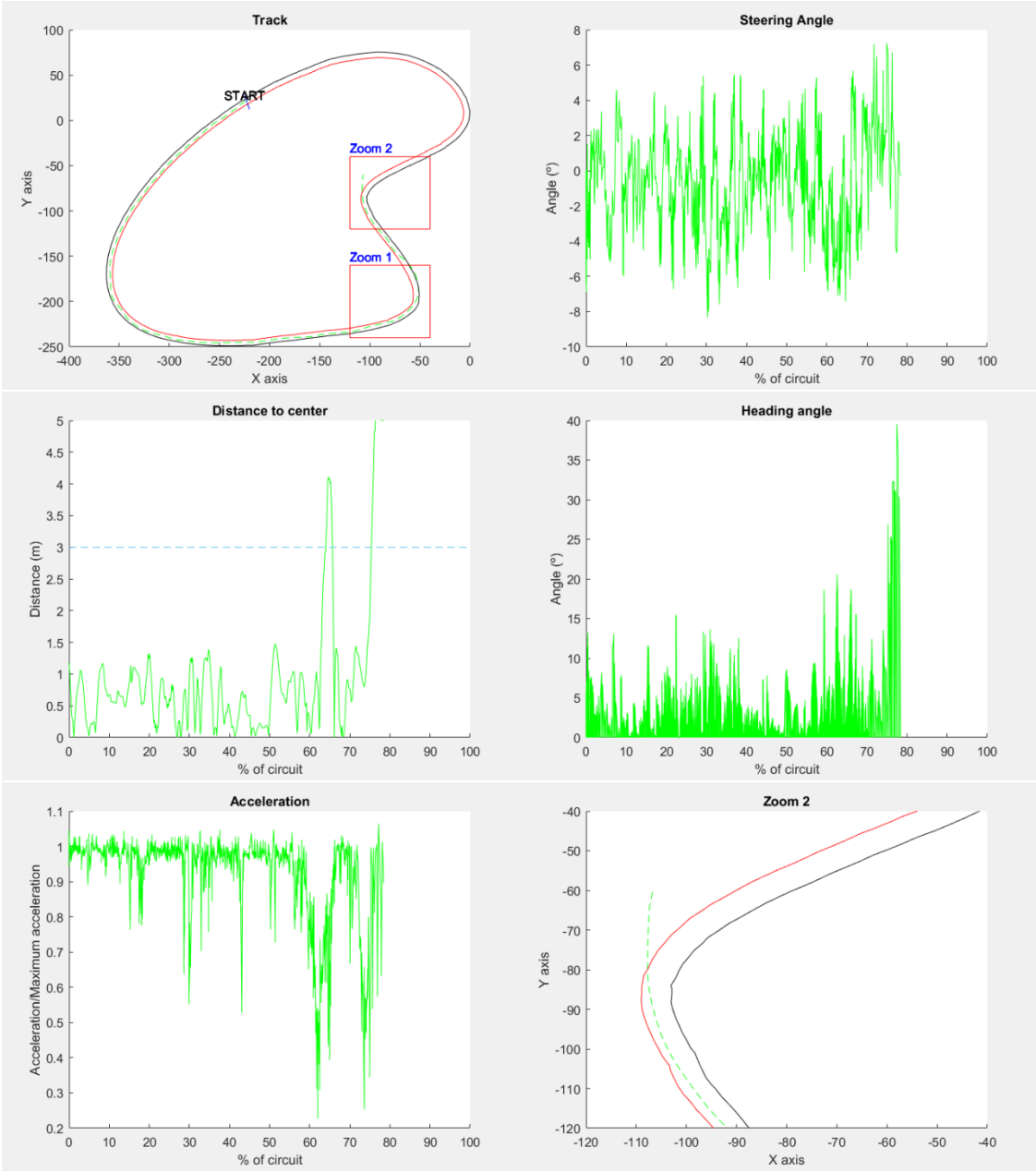

Ilustración 85 Gráfica de Edge-DeepestLSTM-TinyPilotNet controlando volante y acelerador acoplados

Como se observa en la gráfica, el vehículo abandona el circuito en la curva Zoom2.

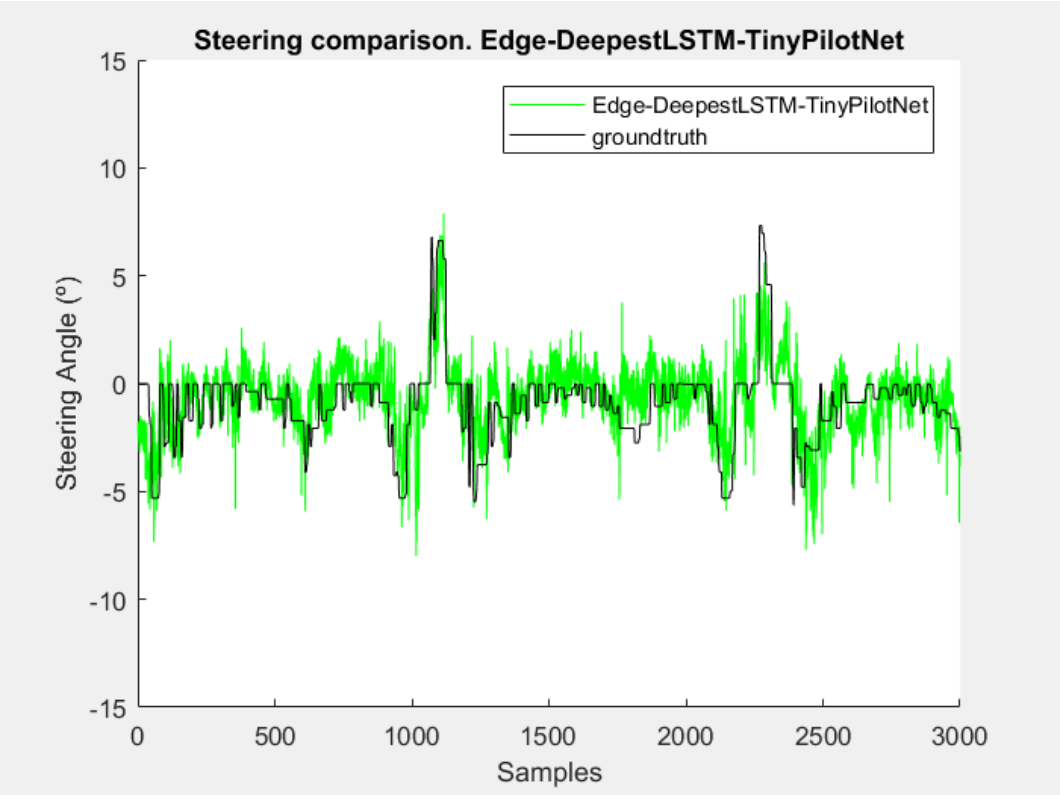

Ilustración 86 Comparativa frame-to-frame de ángulo de volante Edge-DeepestLSTM-TinyPilotNet desacoplada

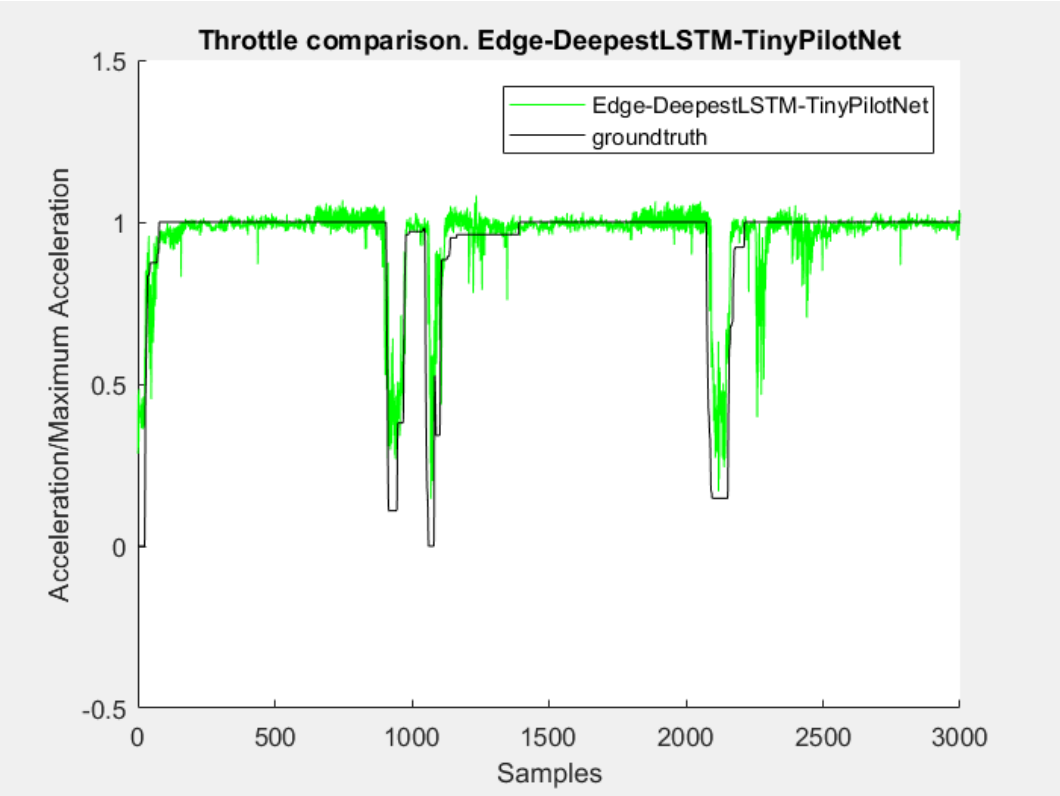

Ilustración 87 Comparativa frame-to-frame de aceleración Edge-DeepestLSTM-TinyPilotNet desacoplada

6.3.7 Conclusiones sobre el uso de una misma CNN para volante y acelerador

A continuación, se establecerá una comparación entre las redes neuronales convolucionales puestas a prueba en el apartado anterior respecto a la red TinyPilotNet, concluyendo qué elementos mejoran el comportamiento del vehículo y cuáles lo empeoran a la hora de controlar el ángulo del volante y la aceleración del vehículo empleando una misma CNN.

6.3.7.1 Root-Mean Square Error

La siguiente tabla muestra los valores de error obtenidos para una comparativa fotograma a fotograma realizada para cada una de las CNN con una base de datos del circuito.

CNN	RMSE	Mejora
TinyPilotNet volante	0.0912475	0%
TinyPilotNet de mayor resolución volante	0.083217	**9%**
DeeperLSTM-TinyPilotNet volante	0.16482	-81%
Edge-DeeperLSTM-TinyPilotNet volante	0.114299	-25%
DeepestLSTM-TinyPilotNet volante	0.116321	-27%
Edge-DeepestLSTM-TinyPilotNet volante	0.1008005	-10%

Tabla 46 Comparativa de RMSE para diferentes CNNs controlando volante y acelerador desacoplados

Como ocurría en las comparaciones anteriores, las redes con detección de bordes resultan más eficaces a la hora de realizar predicciones conociendo tan solo un único fotograma de la situación del vehículo.

Sin embargo, la inclusión de capas LSTM y el disponer de la información previamente aportada produce una conducción más suave como se indica en las siguientes métricas.

6.3.7.2 Desviación respecto al centro del carril

En la siguiente tabla se pueden ver los valores medios y máximos para el parámetro de calidad de desviación respecto al centro del carril para cada CNN.

CNN	Error medio	Mejora media	Error máx.	Mejora máx.
TinyPilotNet	1.07	0%	7.17	0%
TinyPilotNet de mayor resolución	0.90	16%	3.62	**50%**
DeeperLSTM-TinyPilotNet	1.43	-34%	9.71	-35%
Edge-DeeperLSTM-TinyPilotNet	1.68	-57%	8.25	-15%
DeepestLSTM-TinyPilotNet	0.72	**33 %**	5.00	30%
Edge-DeepestLSTM-TinyPilotNet	0.91	15%	9.46	-32%

Tabla 47 Comparación de desviación al centro del carril para distintas CNN controlando volante y acelerador conjuntamente

Las redes TinyPilotNet, DeeperLSTM-TinyPilotNet y las entrenadas con detección de bordes no son capaces de mantener el vehículo en la carretera.

El valor de error máximo de DeepestLSTM-TinyPilotNet es superior debido a que en el tramo de circuito del puente el vehículo circula pegado al muro, que impide la salida del circuito.

Siguiendo el criterio de la mejora del error medio respecto al centro del carril para el resto de las redes, el orden de desempeño de mejor a peor es el siguiente:

1. DeepestLSTM-TinyPilotNet
2. TinyPilotNet de mayor resolución

Al aumentar el número de capas, obteniendo mayor posibilidad de configuración de distintos parámetros de la red, observamos una mejora importante, siendo incluso superior en el caso de la red sin detección de bordes.

Por lo tanto, las redes necesitan una mayor cantidad de parámetros a entrenar para ser capaces de mantener el vehículo dentro del circuito.

6.3.7.2 Ángulo de cabeceo

En la siguiente tabla se pueden ver los valores medios y máximos para el parámetro de calidad de ángulo de cabeceo para cada CNN.

CNN	Cabeceo medio	Mejora media	Cabeceo máx.	Mejora máx.
TinyPilotNet	3.38	0%	44.91	0%
TinyPilotNet de mayor resolución	3.46	-2%	31.24	30%
DeeperLSTM-TinyPilotNet	4.01	-19%	35.92	20%
Edge-DeeperLSTM-TinyPilotNet	3.67	-9%	44.80	0%
DeepestLSTM-TinyPilotNet	2.15	**36 %**	26.35	**41 %**
Edge-DeepestLSTM-TinyPilotNet	2.48	27%	39.53	12%

Tabla 48 Comparación de ángulo de cabeceo para distintas CNNs controlando volante y acelerador conjuntamente

Las únicas redes que mejoran el parámetro de cabeceo medio de la red TinyPilotNet son las que incluyen un mayor número de capas, como ocurre con relación al parámetro de desviación respecto al centro del carril.

En concreto, la red DeepestLSTM-TinyPilotNet mejora hasta un 36 % el cabeceo, produciendo una circulación más suave y natural sin aplicar detección de bordes a la información de entrada que, aunque mejor a TinyPilotNet, empeora lo conseguido sin aplicarlo.

Por tanto, el uso de una red con capas LSTM y un mayor número de parámetros configurables produce una conducción más natural, con menor cabeceo y menor desviación.

Sin embargo, el hecho de controlar volante y acelerador hacen que los resultados sean ligeramente peores que los obtenidos en el caso del control de volante en exclusiva, por lo que la información de aceleración no aporta información al volante.

6.3.7.3 Valores de aceleración

Para comparar el desempeño en la aceleración de las diferentes CNN se muestran a continuación los valores de salida aportados por la neurona de la última capa encargada de controlar este parámetro.

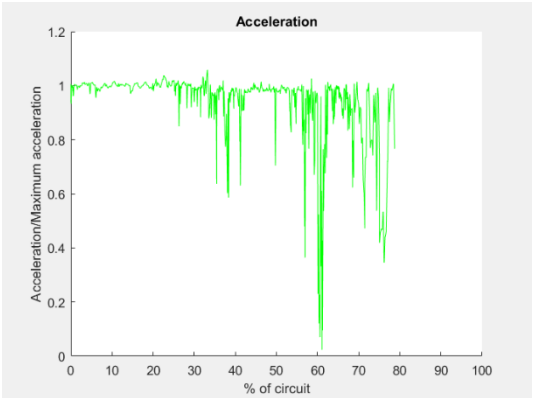

Ilustración 88 Gráfica de aceleración de TinyPilotNet

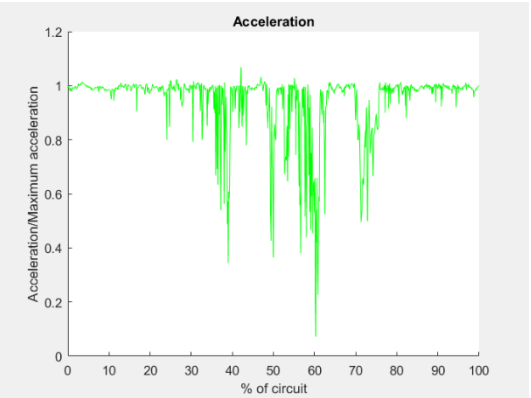

Ilustración 89 Gráfica de aceleración de TinyPilotNet de mayor resolución

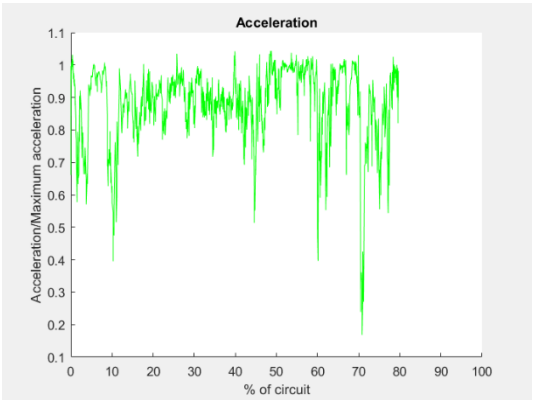

Ilustración 90 Gráfica de aceleración de DeeperLSTM-TinyPilotNet

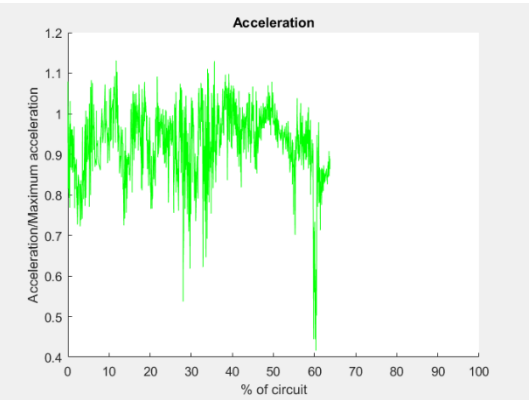

Ilustración 91 Gráfica de aceleración de Edge-DeeperLSTM-TinyPilotNet

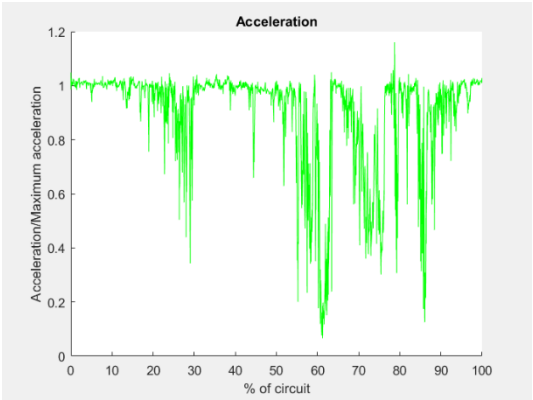

Ilustración 92 Gráfica de aceleración de DeepestLSTM-TinyPilotNet

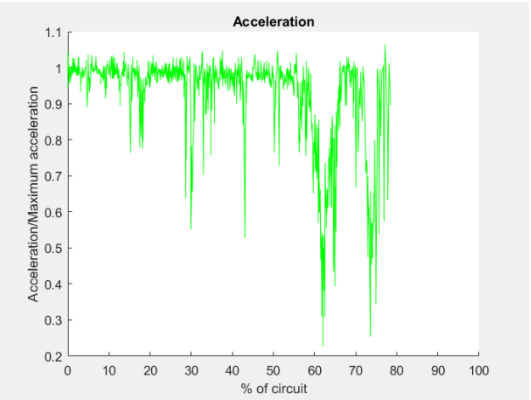

Ilustración 93 Gráfica de aceleración de Edge-DeepestLSTM-TinyPilotNet

Al igual que ocurría cuando la aceleración era controlada con la misma CNN que el volante, se produce una disminución de la aceleración al llegar al 60% del circuito en la mayoría de los casos, para hacer frente a la curva Zoom 1.

Sin embargo, al mantener en todos los casos una aceleración superior a 0, el vehículo circula siempre a la máxima velocidad del simulador, ya que en ningún caso deja de acelerar ni frena.

6.4 Comparativa general

A continuación, se establece una comparativa entre los resultados obtenidos con las diferentes arquitecturas según el método de control del vehículo, véase solo volante para los casos en los que la CNN solo produce los valores de ángulo de volante, desacopladas para los casos en que una CNN controla el volante y otra controla el acelerador, y acoplada para los casos en que una CNN produce los valores de volante y acelerador.

CNN	Conducción	Control	RMSE
TinyPilotNet	Solo Volante	Volante	0.083478
	Desacopladas	Volante	0.082959
		Acelerador	0.099855
	Acoplada	Volante y acelerador	0.0912475
TinyPilotNet de mayor resolución	Solo Volante	Volante	0.081172
	Desacopladas	Volante	**0.072945**
		Acelerador	**0.072789**
	Acoplada	Volante y acelerador	0.083217
LSTM-TinyPilotNet	Solo Volante	Volante	0.08725
	Desacopladas	Volante	-
		Acelerador	-
	Acoplada	Volante y acelerador	-
DeeperLSTM-TinyPilotNet	Solo Volante	Volante	0.095224
	Desacopladas	Volante	0.097846
		Acelerador	0.14497
	Acoplada	Volante y acelerador	0.16482
DeepestLSTM-TinyPilotNet	Solo Volante	Volante	-
	Desacopladas	Volante	-
		Acelerador	-
	Acoplada	Volante y acelerador	0.116321

Tabla 49 Comparativa general de CNNs según RMSE

Se observa que los mejores valores de RMSE son producidos por la red TinyPilotNet de mayor resolución cuando controla volante y acelerador de forma desacoplada, mejorando un 12% la precisión en valores de ángulo de volante y hasta un 27% los valores de aceleración a la red TinyPilotNet desacoplada.

CNN	Conducción	Error medio	Error máx.	Cabeceo medio	Cabeceo máx.
TinyPilotNet	Solo Volante	0.54	2.41	2.44	18.91
	Desacopladas	0.85	3.35	3.08	26.25
	Acoplada	1.07	7.17	3.38	44.91
TinyPilotNet de mayor resolución	Solo Volante	0.70	3.39	**1.71**	23.06
	Desacopladas	0.85	3.39	3.61	29.30
	Acoplada	0.90	3.62	3.46	31.24
LSTM-TinyPilotNet	Solo Volante	0.58	2.35	2.10	27.03
	Desacopladas	-	-	-	-
	Acoplada	-	-	-	-
DeeperLSTM-TinyPilotNet	Solo Volante	**0.453**	**1.98**	1.98	**18.37**
	Desacopladas	0.95	9.54	3.32	37.23
	Acoplada	1.43	9.71	4.01	35.92
DeepestLSTM-TinyPilotNet	Solo Volante	-	-	-	-
	Desacopladas	-	-	-	-
	Acoplada	0.72	5.00	2.15	26.36

Tabla 50 Comparativa general de CNNs según las nuevas métricas de evaluación

Se puede observar como para una misma red neuronal convolucional, por lo general, el desempeño de ambos factores de métrica de evaluación empeora al añadir el cálculo de los valores de aceleración, tanto de forma desacoplada como acoplada.

Además, se aprecia que para conseguir unos valores de correcto funcionamiento para una red con los datos acoplados próximos a los mejores obtenidos con el control de volante únicamente, es necesario crear redes neuronales de mayor profundidad.

Capítulo 7.
Conclusiones generales y trabajos futuros.

A continuación, se desarrollarán las conclusiones extraídas del presente Trabajo Fin de Grado relacionadas con el objetivo de obtener una conducción humana en un vehículo controlado por una Convolutional Neural Network en un simulador.

- Un ligero aumento en la resolución de la imagen de entrada produce mejoras notables en ambos factores de calidad sin suponer un aumento significativo de la dimensión de la red ni del tiempo de procesamiento.

- La inclusión de capas Long Short-Term Memory (LSTM) en la salida de una red neuronal convolucional aporta una influencia de los valores previamente aportados por la misma, lo que conlleva una conducción más suave.

- El uso de imagen de entrada a color RGB en lugar de usar exclusivamente la saturación del espacio de color HSV produce una menor comprensión del entorno por parte de la CNN, lo que conlleva en una mala conducción. Al usar el canal de saturación la carretera queda destacada en negro, mientras que el exterior de esta obtiene colores más claros, produciendo una sencilla distinción del recorrido del circuito.

- La información acerca de aceleración no produce un mejor control del ángulo de volante, tanto en una misma red como en redes desacopladas.

- Para obtener valores de métrica de evaluación similares a los obtenidos por una CNN que solo controla el volante en una CNN que controle volante y acelerador es necesario aumentar la profundidad de la red.

- La mayor definición de bordes en la imagen de entrada mediante el filtro Canny no produce mejoría significativa en ninguno de los dos parámetros de calidad establecidos.

- Realizar un recorte en la imagen de entrada para extraer únicamente la información de la carretera, eliminando la información del entorno, no mejora la conducción.

Este trabajo tiene una gran aplicación en la actualidad, por lo que podría continuar desarrollándose como una herramienta puntera próxima al estado del arte, introduciendo nuevas técnicas como el entrenamiento y la realización de pruebas sobre un vehículo real o la inclusión de capas LSTM en las capas iniciales de las CNN tal y como propone [18], desarrollando una mayor comprensión espacio-temporal por parte de la red.

También, siguiendo el planteamiento propuesto en [18], existe la posibilidad de introducir una serie de capas que produzca una imagen de los elementos más significativos para la CNN, aportando información sobre cómo se focaliza la red sobre los límites de la carretera para mejorar su funcionamiento.

Anexo I. Pliego de condiciones

A continuación, se describe el material necesario para el desarrollo del presente Trabajo Fin de Grado, tanto hardware como software.

Hardware empleado

En este caso se ha utilizado un único ordenador portátil para ejecutar todo el desarrollo, que tiene los siguientes componentes:

- Intel Core i7 4770-HQ
- NVIDIA 960M
- 8GB RAM
- 1TB HDD

Además, para una mejor adquisición de datos de entrenamiento se ha empleado un volante Logitech Driving Force Pro, conectado por USB al ordenador mencionado.

Software empleado

En el caso del software ha sido necesario trabajar en dos sistemas operativos en función de la tarea realizada

- Ubuntu 16.04

  Utilizado para el desarrollo de CNNs y simulación del comportamiento. El software instalado en este SO es el siguiente:

  - Python 3
  - Distintas librerías: numpy, flask-socketio, eventlet, pillow, h5py, Keras, TensorFlow

- Windows 10

  Utilizado para realizar modificaciones sobre el simulador, extracción de conclusiones y para la escritura del trabajo. El software incluido es el siguiente:

  - Matlab R2017a
  - Microsoft Word 2016
  - Unity 5.5.1f1

Anexo II. Manual de usuario

En este anexo se detallarán los pasos seguidos para la extracción de datos de entrenamiento del simulador, el desarrollo y entrenamiento de arquitecturas de CNN, la prueba de CNN en el simulador y la obtención de los datos de métrica de evaluación mediante MATLAB.

1. Extracción de datos de entrenamiento del simulador

La obtención de estos datos es crucial para el correcto entrenamiento posterior de las distintas CNN. Es por ello por lo que la conducción debe ser lo más perfecta posible para evitar que la red aprenda de posibles fallos humanos.

Para una mejor calidad de datos obtenidos es conveniente emplear un volante con pedales que disponga de conexión USB con el ordenador. En el desarrollo de este Trabajo Fin de Grado se emplea un volante Logitech Driving Force Pro.

Una vez descargado el simulador [8], se ejecuta en SO Windows preferiblemente, ya que los drivers del volante están creados para esta plataforma.

Se selecciona el primero de los circuitos y la opción “Training Mode”.

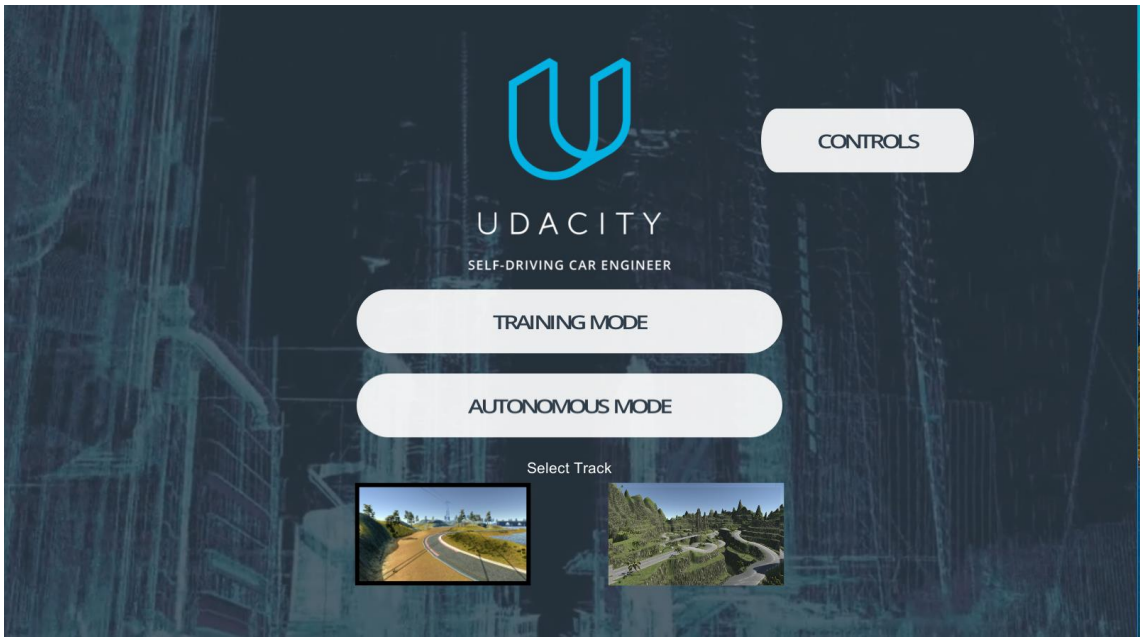

Ilustración 94 Menú principal de simulador Udacity

Una vez en el circuito, se pulsa en el icono RECORD, ubicado en la esquina superior derecha de la pantalla.

Ilustración 95 Simulador en modo entrenamiento

A continuación, deberemos indicar dónde se almacenarán los datos de entrenamiento.

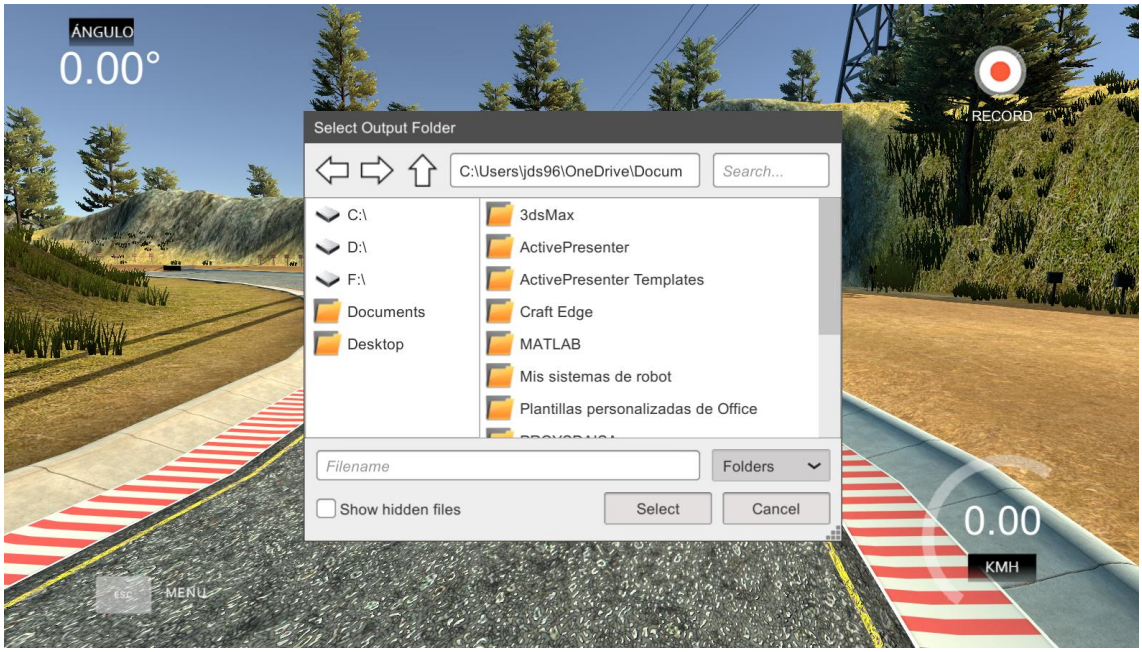

Ilustración 96 Simulador en modo entrenamiento. Indicar lugar para almacenamiento de datos

Una vez realizados estos pasos, aparecerá el mensaje “Recording” próximo al botón de inicio de grabación y los datos de la conducción que realicemos quedarán capturados por el simulador.

Ilustración 97 Simulador en modo entrenamiento. Grabación de datos

Una vez realizado el circuito recolectando datos de la conducción, se debe pulsar en el icono de la esquina superior derecha para finalizar la captura de datos. En ese momento aparecerá el mensaje “Capturing Data” acompañado de un valor porcentual que indicará el total de datos almacenado de todos los realizados. Entonces se mostrará por pantalla el recorrido realizado a la vez que el valor porcentual crece hasta alcanzar el 100%.

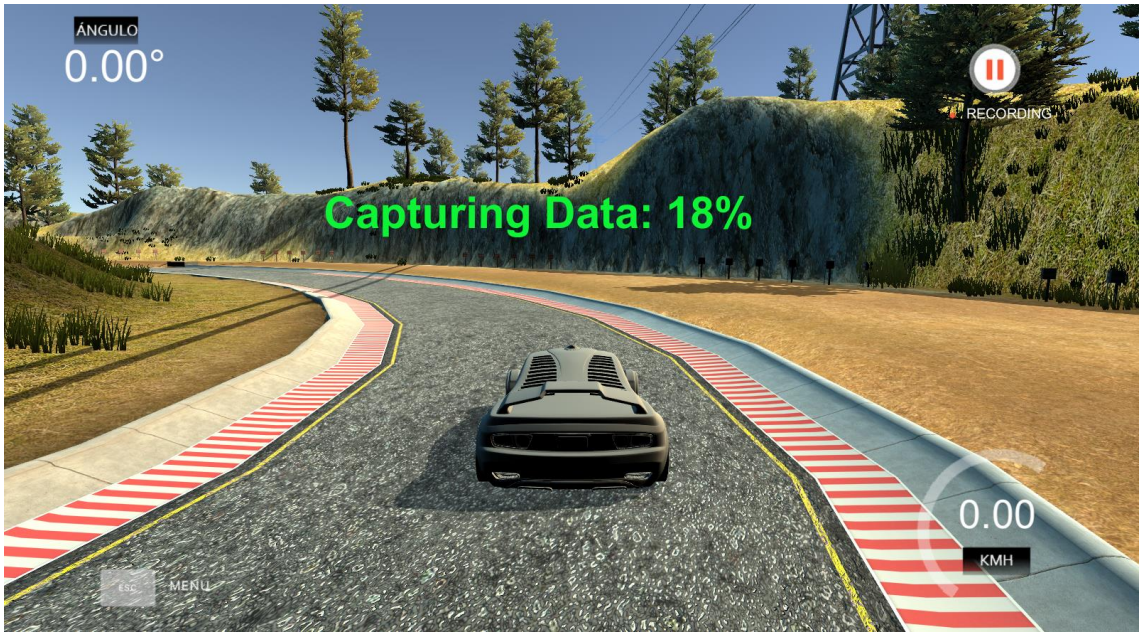

Ilustración 98 Simulador en modo entrenamiento. Almacenamiento de los datos capturados

2. Desarrollo y entrenamiento de arquitecturas de CNN

2.1 Arquitectura

Como se ha mencionado a lo largo del Trabajo Fin de Grado, las arquitecturas de red están elaboradas en lenguaje de alto nivel Keras [28], que incluye las siguientes funciones de arquitectura.

```
def steering_model():
    model = Sequential()
    model.add(Lambda(lambda x: x/127.5 - 1.,input_shape=(rows,cols,1),batch_input_shape=(1,rows,cols,1)))

    model.add(Convolution2D(8, 3, 3, init='normal',border_mode='valid'))
    model.add(Activation('relu'))
    model.add(MaxPooling2D((2,2),border_mode='valid'))

    model.add(Convolution2D(8, 3, 3,init='normal',border_mode='valid'))
    model.add(Activation('relu'))
    model.add(MaxPooling2D((2,2),border_mode='valid'))

    model.add(Dropout(0.2))
    model.add(Reshape(target_shape=conv_to_LSTM_dims, name='Reshape_Conv2LSTM'))

    model.add(ConvLSTM2D(filters=8, kernel_size=(3, 3), input_shape=conv_to_LSTM_dims, padding='same', return_sequences=True,  stateful = True))
    model.add(ConvLSTM2D(filters=8, kernel_size=(3, 3), input_shape=conv_to_LSTM_dims, padding='same', return_sequences=True,  stateful = True))
    model.add(ConvLSTM2D(filters=8, kernel_size=(3, 3), input_shape=conv_to_LSTM_dims, padding='same', return_sequences=True,  stateful = True))
    model.add(ConvLSTM2D(filters=8, kernel_size=(3, 3), input_shape=conv_to_LSTM_dims, padding='same', return_sequences=True,  stateful = True))

    model.add(Flatten())
    model.add(Dense(1))
    model.summary()
    return model
```

Ilustración 99 Código de arquitectura de DeeperLSTM-TinyPilotNet

Para conformar la arquitectura de la red DeeperLSTM-TinyPilotNet se enumeran en orden las capas que la conforman.

Las capas convolucionales son instanciadas mediante la función Convolution2D, indicando a continuación el número de filtros (8), y el tamaño del kernel (3x3).

Las capas de activación se incluyen mediante la función Activation, aportando como argumento el tipo de activación deseado (ReLU).

Las capas de submuestreo de tipo maxpooling se incluyen mediante la función MaxPooling2D, detallando además el tamaño del kernel (2x2).

La capa dropout se incluye mediante la función Dropout, y se introduce como argumento la probabilidad de que la neurona aparezca durante el entrenamiento (20%).

La capa reshape para adaptar las capas Convolutional+LSTM al resto de la arquitectura se introduce mediante el comando Reshape, acompañada de la dimensión de salida deseada.

Las capas Convolutional+LSTM se instancian mediante la función ConvLSTM2D, indicando el tamaño del filtro (8) y del kernel (3x3), además de la dimensión de salida de la capa.

El efecto flatten para aplanar el tensor de salida de las capas anteriores se introduce mediante el comando Flatten. No es necesario especificar ningún argumento.

Finalmente, la neurona de salida se crea mediante una capa fully connected, denominada Dense en Keras, indicando en el argumento el número de neuronas que componen la capa.

2.2 Entrenamiento

El lenguaje Keras también dispone de funciones que facilitan el entrenamiento de las CNN, como se verán a continuación.

```
model = steering_model()
adam = Adam(lr=0.001, beta_1=0.9, beta_2=0.999, epsilon=1e-08, decay=0.0)
model.compile(loss='mean_squared_error',optimizer='adam')
history = model.fit(train_features, train_labels,batch_size=1, epochs=6,verbose=1, validation_data=(test_features, test_labels),shuffle=False)
```

Ilustración 100 Código de entrenamiento de DeepestLSTM-TinyPilotNet

El optimizador Adam recoge las variables deseadas para ajustar el entrenamiento.

La función compile, aplicada a la red "model", recibe como argumento qué función de pérdidas aplicar y las variables del optimizador Adam.

El entrenamiento se realiza mediante la función fit, indicando los datos de entrada "train_features" y salida "train_labels", así como el tamaño del batch empleado y el número de epochs o vueltas realizadas sobre el conjunto de datos. También se indica un conjunto de datos de entrada y salida "test_features y test_labels" que sirven para identificar cómo de correcto es el funcionamiento de la CNN.

Para realizar el entrenamiento de la red se debe llamar al archivo creado en Python3 que contiene los comandos mencionados en los puntos 2.1 y 2.2 de este Anexo II como se indica a continuación:

```
$ sudo python3 DeeperLSTM-TinyPilotNet.py
```

3. Prueba de CNN en simulador y obtención de datos

Una vez entrenada la CNN es necesario poner a prueba su correcto funcionamiento. Para ello deberá controlar un vehículo en el simulador empleado durante el entrenamiento.

Para comprobar la validez de la conducción es necesario extraer datos de la conducción realizada en el simulador. Estos datos se almacenarán en un fichero model.csv, que posteriormente se analizará en MATLAB.

El simulador provee un archivo denominado drive.py que recibe imágenes del simulador y las envía a la red previamente entrenada, obteniendo los datos de salida de la red, que devuelve al simulador. Para poner en ejecución el simulador con una CNN determinada debemos introducir como argumento el fichero .json generado en el entrenamiento de la red como aparece a continuación:

```
$ sudo python3 drive.py DeeperLSTM-TinyPilotNet.json
```

Simultáneamente se debe abrir el simulador clickando en "Autonomous Mode" en el menú inicial que aparece en la ilustración 63.

4. Análisis de datos en MATLAB

Para comprobar los valores de las métricas de evaluación establecidos en el capítulo 5 de este Trabajo Fin de Grado se elaboran diversas funciones en código MATLAB.

Estas funciones extraerán la información de los ficheros model.csv obtenidos en el apartado 3 de este Anexo II, creando gráficas del recorrido por el circuito, así como de ángulos de volante y acelerador obtenidos y desviación respecto al centro del carril y ángulo de cabeceo resultante, lo que permitirá establecer comparativas entre las diferentes CNN evaluadas.

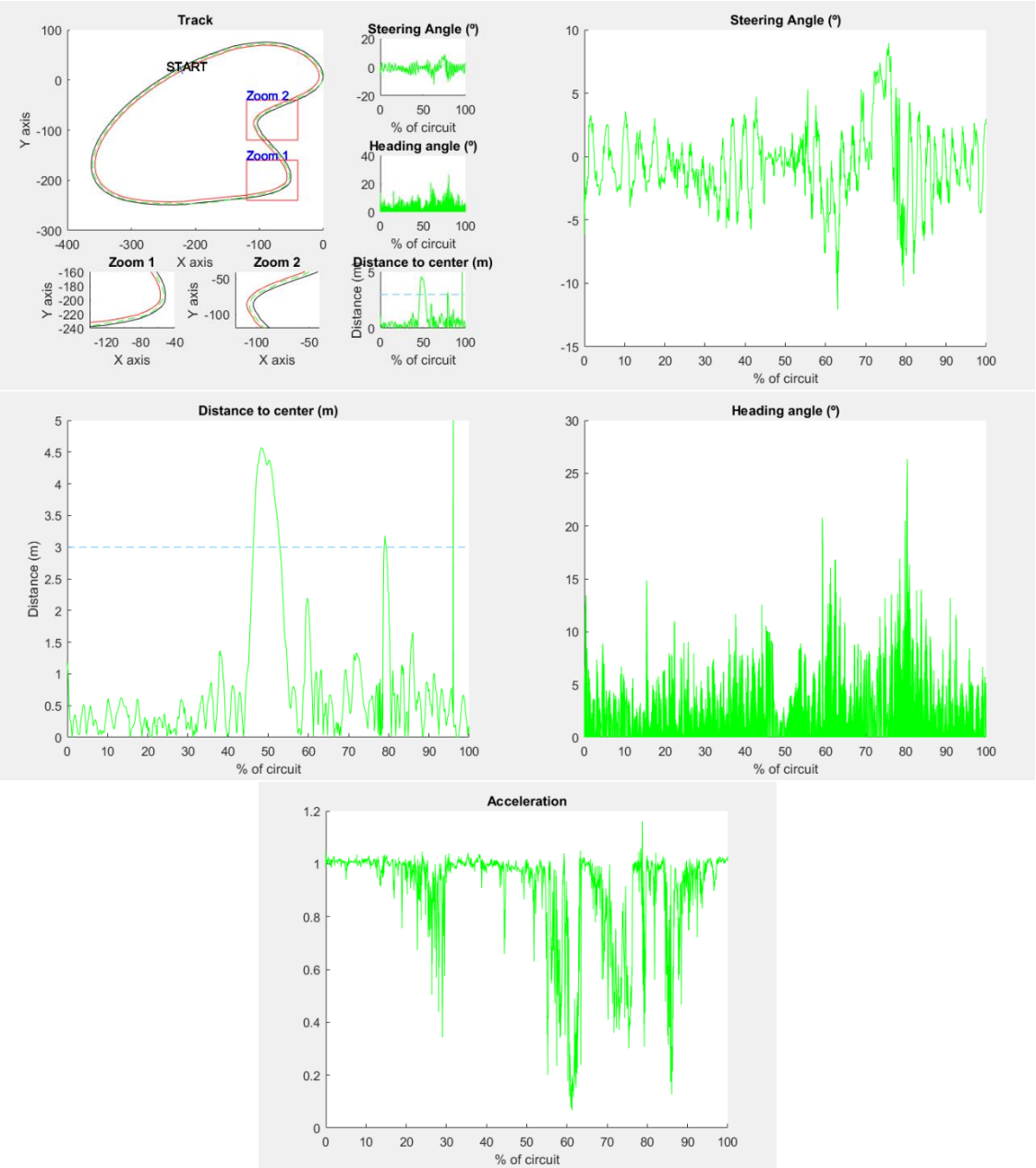

Ilustración 101 Gráficas extraídas en MATLAB a partir de la conducción de DeepestLSTM-TinyPilotNet

CNN	Error medio	Error máx.	Cabeceo medio	Cabeceo máx.
DeepestLSTM-TinyPilotNet	0.72	5.00	2.15	26.36

Tabla 51 Tabla de datos extraída de MATLAB a partir de la conducción de DeepestLSTM-TinyPilotNet

Anexo III. Presupuesto

A continuación, se describen los costes estimados para el desarrollo del proyecto en los aspectos de mano de obra y hardware y software necesario.

Coste Hardware

Material	Precio unitario	Unidades	Precio total
Ordenador Lenovo Y50-70	850 €	1	850 €
Logitech Driving Force Pro	150 €	1	150 €

Tabla 52 Costes de material hardware

Costes Software

Para la realización del proyecto se ha empleado diverso material software descrito en el Anexo I. A continuación, se especifican los costes del material empleado.

Material	Precio
Matlab R2017a	0 € (aportado por UAH)
Ubuntu 16.04	0 €
Python 3 y librerías	0 €
Microsoft Word 2016	0 € (aportado por UAH)
Unity 5.5.1f1	0 €

Tabla 53 Costes de material software

Costes de personal

Para la estimación del coste que supondría la realización de este Trabajo Fin de Grado se supone un sueldo medio de 15 euros por hora de un Ingeniero Industrial.

Material	Precio/hora	Horas	Subtotal
Análisis de funcionamiento de CNNs y adaptación para su uso en este proyecto	15 €/hora	60 horas	900 €
Manejo de simulador open-source y recolección de datos para entrenamiento	15 €/hora	50 horas	750 €
Entrenamiento de CNNs	15 €/hora	250 horas	3.750 €
Obtención y análisis de resultados de conducción autónoma	15 €/hora	50 horas	750 €
Documentación y escritura del Trabajo Fin de Grado	15 €/hora	40 horas	600 €

Tabla 54 Costes de personal

Costes de ejecución totales

Concepto	Subtotal
Costes hardware	1000 €
Costes software	0 €
Costes personal	6.750 €
Subtotal final	**7.750 €**

Tabla 55 Costes de ejecución totales

Gastos generales y beneficio industrial

Siguiendo la tipificación del PEM 2018, los gastos generales se estiman en un 13% de los costes de ejecución totales, mientras que el beneficio industrial se sitúa en torno a un 6% de los costes de ejecución totales.

Concepto	**Subtotal**
Costes de ejecución material	7.750 €
Gastos generales (13 %)	1.007,5 €
Beneficio industrial (6%)	465 €
Subtotal final	1.472,5 €

Tabla 56 Gastos generales y beneficio industrial

Presupuesto de ejecución por contrata

Este presupuesto se calcula a partir de los costes de ejecución de material y los gastos generales y beneficio industrial.

Concepto	**Subtotal**
Costes de ejecución material	7.750 €
Gastos generales y beneficio industrial	1.475,5 €
Subtotal final	9.225,5 €

Tabla 57 Presupuesto de ejecución de contrata

Importe total del presupuesto

El presupuesto final de este Trabajo Fin de Grado finaliza con la aplicación del IVA, estipulado en 21%, a partir del presupuesto de ejecución por contrata.

Concepto	**Subtotal**
Presupuesto de ejecución por contrata	9.225,5 €
Porcentaje de IVA	21 %
Subtotal final	1.937,35€
Importe final	11.162,85 €

Tabla 58 Importe total del presupuesto

El presupuesto final estimado para la realización de este Trabajo Fin de Grado asciende a un total de 10.943,24 € (diez mil novecientos cuarenta y tres con veinticuatro euros).

Bibliografía

[1] Mariusz Bojarsk *et al*. (2016)
"End to End Learning for Self-Driving Cars. NVIDIA".
Available: https://arxiv.org/pdf/1604.07316v1.pdf

[2] Tesla Autopilot webpage. https://www.tesla.com/es_ES/autopilot

[3] A. Krizhevsky, I. Sutskever, and G. E. Hinton (2012), "Imagenet classification with deep convolutional neural networks," in Advances in neural information processing systems, pp. 1097–1105.

[4] O. Russakovsky et al. (Dec. 2015).
"ImageNet Large Scale Visual Recognition Challenge".
Available: http://dx.doi.org/10.1007/s11263-015-0816-y

[5] M. D. Zeiler and R. Fergus (Sept. 2014), "Visualizing and Understanding Convolutional Networks," in Computer Vision – ECCV 2014
Available: http://dx.doi.org/10.1007/978-3-319-10590-1_53

[6] K. Simonyan and A. Zisserman, "Very Deep Convolutional Networks for Large-Scale Image Recognition," arXiv:1409.1556 [cs], Sep. 2014, arXiv: 1409.1556.

[7] K. He, X. Zhang, S. Ren, and J. Sun (2016), "Deep residual learning for image recognition," in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 770–778.

[8] E. Shelhamer, J. Long, and T. Darrell (Apr. 2017), "Fully Convolutional Networks for Semantic Segmentation," IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 39, no. 4, pp. 640–651.

[9] T. Pohlen, A. Hermans, M. Mathias, and B. Leibe (Nov. 2016), "Full-Resolution Residual Networks for Semantic Segmentation in Street Scenes," arXiv:1611.08323 [cs], arXiv: 1611.08323.

[10] "TensorFlow documentation". Available: https://www.tensorflow.org/

[11] "MATLAB CNNs documentation".
Available: https://es.mathworks.com/help/nnet/convolutional-neural-networks.html

[12] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," in Proceedings of the 22nd ACM international conference on Multimedia. ACM, 2014, pp. 675–678.
Available: http://dl.acm.org/citation.cfm?id=2654889

[13] "PyTorch documentation — PyTorch master documentation."
Available: http://pytorch.org/docs/master/index.html

[14] “Keras documentation”. Available: https://keras.io/

[15] Yunming Shao (2017). End-to-End Learning for Self-Driving Cars.
Available: https://github.com/ymshao/End-to-End-Learning-for-Self-Driving-Cars

[16] Y. LeCun, U. Muller, J. Ben, E. Cosatto and B. Flepp (2015).
“Off-road obstacle avoidance through end-to-end learning” in NIPS.

[17] M. Bojarski and U. Muller (2017).
“Explaining how a deep neural network trained with end-to-end learning steers a car”.

[18] Lu Chi and Yadong Mu, (2017). “Deep Steering: Learning End-to-End Driving Model from Spatial and Temporal Visual Cues”.
Available: https://arxiv.org/pdf/1708.03798.pdf

[19] Adam Gibson, Chris Nicholson, Josh Patterson.
A Beginner’s Guide to Recurrent Networks and LSTMs.
Available: https://deeplearning4j.org/lstm.html

[20] Fei-Fei Li, Justin Johnson, Serena Yeung.
CS231n Convolutional Neural Networks for Visual Recognition.
Available: http://cs231n.stanford.edu/

[21] Udacity Self-Driving Car Simulator project
Available: https://github.com/udacity/self-driving-car-sim

[22] M. A. Nielsen (2015), “Neural Networks and Deep Learning,”.
Available: http://neuralnetworksanddeeplearning.com

[23] C. Szegedy, W. Liu et al. (2015), “Going deeper with convolutions,” in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 1–9.
Available:
http://www.cvfoundation.org/openaccess/content_cvpr_2015/html/Szegedy_Going_Deeper_With_2015_CVPR_paper.html

Universidad de Alcalá

Escuela Politécnica Superior

ESCUELA POLITECNICA SUPERIOR