

# Trabajo de Investigación

**Conducción autónoma basada en visión por ordenador.**

## **MEMORIA**

Autor: Bernat Bautista González

Tutor: Joan Pardó

Fecha: Octubre 16, 2023

Lugar: Llinars del Vallés





# Agradecimientos:

Quiero expresar mi más profundo agradecimiento a las siguientes personas e instituciones que hicieron posible la realización de este trabajo de investigación:

- **Universidad LaSalle:** Agradezco a la Universidad LaSalle por brindarme la oportunidad de utilizar su centro de investigación para llevar a cabo este proyecto. Además, estoy agradecido por haber sido nominado al premio del TDR, es decir, tres tutorías privadas con un profesor del centro, lo cual representa un honor y un reconocimiento invaluable.
- **Carlos Guerrero Mosquera:** Deseo agradecer de manera especial al profesor Carlos Guerrero Mosquera de LaSalle por su inestimable apoyo a lo largo de este proceso. Su orientación en el campo de la inteligencia artificial, su ayuda en la formulación del proyecto y la generosidad al proporcionar recursos.
- **Joan Pardó:** Agradezco a Joan Pardó, mi asesor del TDR, por su dedicación, sabiduría y guía durante todo el proceso de investigación. Su experiencia y consejos fueron fundamentales para la calidad y la dirección de este trabajo.

Estoy agradecido también a todos aquellos que, de una manera u otra, contribuyeron a este proyecto y creyeron en mí. Su apoyo ha sido esencial. Gracias nuevamente a todos por su valioso respaldo.

# Resumen:

Este trabajo de investigación aborda la aplicación de algoritmos de aprendizaje automático en el contexto de la conducción autónoma. Los objetivos incluyen la comprensión de dichos niveles de conducción autónoma, sus limitaciones, la arquitectura de los sistemas, los desafíos técnicos y finalmente el desarrollo de una IA.

La metodología se basa en la explicación de conceptos clave como redes neuronales convolucionales, redes neuronales recurrentes, aprendizaje por refuerzo, *deep learning*, las cuales serán usadas posteriormente.

Se explora la selección de un simulador para desarrollar un proyecto práctico relacionado con la conducción autónoma. El proyecto implica diseñar un sistema que abarca el entrenamiento y la explotación de modelos de aprendizaje automático, aparte de diseñar API's.

Los resultados revelan que el aprendizaje automático desempeña un papel fundamental en la conducción autónoma, pero también enfrenta desafíos en la percepción del entorno y la toma de decisiones seguras.

En conclusión, este estudio proporciona una visión integral de cómo la tecnología y los algoritmos de aprendizaje automático están impulsando la conducción autónoma, pero destaca la necesidad de abordar cuestiones críticas para lograr una implementación segura y eficiente.

This research work addresses the application of machine learning algorithms in the context of autonomous driving. The objectives include understanding various levels of autonomous driving, their limitations, system architecture, technical challenges, and ultimately, the development of an AI. The methodology is based on explaining key concepts such as convolutional neural networks, recurrent neural networks, reinforcement learning, deep learning, which will be used later on.

It explores the selection of a simulator for developing a practical project related to autonomous driving. The project involves designing a system that encompasses the training and deployment of machine learning models, in addition to designing APIs.

The results reveal that machine learning plays a fundamental role in autonomous driving but also faces challenges in environment perception and safe decision-making.

In conclusion, this study provides a comprehensive view of how technology and machine learning algorithms are driving autonomous driving, but it emphasizes the need to address critical issues to achieve safe and efficient implementation.

# Glosario

<b>ANN</b>	<i>Artificial Neural Network</i>
<b>CNN</b>	<i>Convolutional Neural Network</i>
<b>RNN</b>	<i>Recurrent Neural Network</i>
<b>IA</b>	<i>Inteligencia Artificial</i>
<b>ML</b>	<i>Machine Learning</i>
<b>DL</b>	<i>Deep Learning</i>
<b>RT</b>	<i>Ray Tracing</i>
<b>RL</b>	<i>Reinforcement Learning</i>
<b>HMI</b>	<i>Human Machine Interface</i>
<b>NHTSA</b>	<i>National Highway Traffic Safety Administration</i>
<b>SAE</b>	<i>Society of Automotive Engineers</i>
<b>FSD</b>	<i>Full Self Driving</i>
<b>SD</b>	<i>Self Driving</i>
<b>AA</b>	<i>Aprendizaje automático</i>
<b>ASIC</b>	<i>Application-Specific Integrated Circuit</i>
<b>GPU</b>	<i>Graphics Processing Unit</i>
<b>SoC</b>	<i>System-on-a-Chip</i>
<b>BCI</b>	<i>Brain Computer Interface</i>
<b>DoG</b>	<i>Diferencia de gaussiano</i>
<b>LoG</b>	<i>Laplaciano del Gaussiano</i>

# Contenido:

Glosario.....	5
Introducción.....	8
Motivación/Objetivos.....	9
Métodos:.....	10
3.1 SIFT.....	10
3.1.1 Scale-space Extrema Detection.....	10
3.1.2 Keypoint Localization.....	11
3.1.3 Orientation Assignment.....	11
3.1.4 Keypoint Descriptor & Matching.....	11
3.2 AdaBoost.....	12
3.3 TextonBoost.....	12
3.4 Computer Vision.....	12
3.4.1 ¿Cómo funciona la visión por ordenador?.....	12
Herramientas.....	13
4.1 OpenCV.....	13
4.2 UE5.....	13
4.3 Plataformas de desarrollo de inteligencia artificial.....	14
4.3.1 Tensorflow.....	14
4.3.2 PyTorch.....	15
4.3.3 Keras.....	15
4.3.4 Caffe.....	15
4.4 Procesadores y sistemas embebidos.....	15
4.4.1 ASIC.....	15
4.4.2 GPU.....	16
4.4.3 SoC.....	16
4.5 HMI.....	16
4.5.1 Visualización de información.....	16
4.5.2 Retroalimentación haptica y acústica.....	16
4.5.3 Comandos y control.....	16
4.5.4 Comunicación y estado del sistema.....	16
Explicación de la conducción autónoma.....	17
5.1 Niveles de conducción autónoma.....	17
5.1.1 Nivel 1: Asistentes de conducción.....	17
5.1.2 Nivel 2: Semi-autonomía.....	17
5.1.3 Nivel 3: Autónomo Controlado.....	17
5.1.4 Nivel 4: Alta automatización.....	17
5.1.5 Nivel 5: Automatización total.....	18
5.2 Limitaciones de la conducción autónoma.....	18
5.2.1 Fiabilidad.....	18
5.2.2 Tiempo.....	18
5.2.3 Cambios imprevistos.....	18
5.2.4 Costo.....	18
5.3 Arquitectura del sistema de conducción autónoma.....	18
5.3.1 Sensores.....	18
5.3.2 Percepción.....	19
5.3.3 Toma de decisiones.....	19
5.3.4 Control y actuación.....	19

5.4 Desafíos técnicos y de implementación.....	19
5.4.1 Percepción y comprensión del entorno.....	19
5.4.2 Toma de decisiones segura.....	19
5.5 Casos de uso y aplicaciones.....	19
Algoritmos de aprendizaje automático.....	21
6.1 CNN.....	21
6.1.1 Capa de convolución.....	21
6.1.2 Capa de activación.....	21
6.1.3 Capa de agrupación.....	22
6.1.4 Capa completamente conectada.....	22
6.2 RNN.....	22
6.3 RL.....	23
6.3.1 Agente.....	23
6.3.2 Entorno.....	23
6.3.3 Recompensa.....	24
6.3.4 Exploración.....	24
6.3.5 Explotación.....	24
6.3.6 Q-Learning.....	24
6.3.7 SARSA.....	24
6.3.8 Policy Gradient.....	24
6.3.9 DQN.....	24
6.3.10 Q Function.....	24
6.4 DL.....	25
6.4.1 Inicialización.....	25
6.4.2 Forward Pass.....	25
6.4.3 Loss.....	25
6.4.4 Backpropagation.....	25
6.4.5 Iteración.....	26
6.4.6 Clasificación de Imágenes.....	26
6.4.7 NLP.....	26
6.4.8 Juegos.....	26
6.4.9 Conducción Autónoma.....	26
6.4.10 Medicina.....	26
6.5 ReLU.....	26
6.5.1 El problema de ReLU.....	27
Simulación.....	28
7.1 Selección del simulador.....	28
Proyecto Práctico.....	29
8.1 Descripción del Proyecto.....	29
8.2 Diseño del sistema.....	29
8.2.1 Entrenamiento:.....	29
8.2.2 Explotación:.....	30
Webgrafía.....	31

# 1

## Introducción

La conducción autónoma se ha convertido en un asunto muy interesante en el ámbito de la inteligencia artificial. La idea de que los vehículos puedan conducirse de forma autónoma y segura es revolucionaria, ya que tiene el potencial de transformar la manera en la que nos desplazamos. Sin embargo, todavía quedan muchos desafíos técnicos y regulativos por superar antes de que se puedan implementar vehículos completamente autónomos en la carretera.

En mi trabajo de investigación, me centraré en simular un vehículo de conducción autónoma en un ordenador. Mi objetivo es abarcar diferentes aspectos de la inteligencia artificial, incluyendo *computer vision, training, predicción y detección de objetos*.

En este trabajo, presentaré una detallada descripción de mi enfoque y los métodos que utilizaré para simular el vehículo de conducción autónoma. También discutiré los resultados y las conclusiones obtenidas a partir de mi investigación y cómo pueden ser útiles para la futura investigación en este campo.

# 2

## Motivación/Objetivos

Mi motivación para trabajar en la conducción autónoma en inteligencia artificial proviene de mi interés en la programación desde temprana edad. A los 6 años, empecé a aprender por mi cuenta y he ido avanzando en mi conocimiento a lo largo de los años. Desde que descubrí el potencial de la inteligencia artificial, me cautivó el rápido avance de este campo y las posibilidades que ofrece. Por esta razón, decidí enfocar mis esfuerzos en simular un vehículo de conducción autónoma en ordenador para abarcar diferentes aspectos de la IA y el aprendizaje automático, incluyendo visión por ordenador, entrenamiento de ML, predicción y detección de objetos. Mi objetivo es contribuir a la investigación en este campo y aprender al máximo.

# 3

## Métodos:

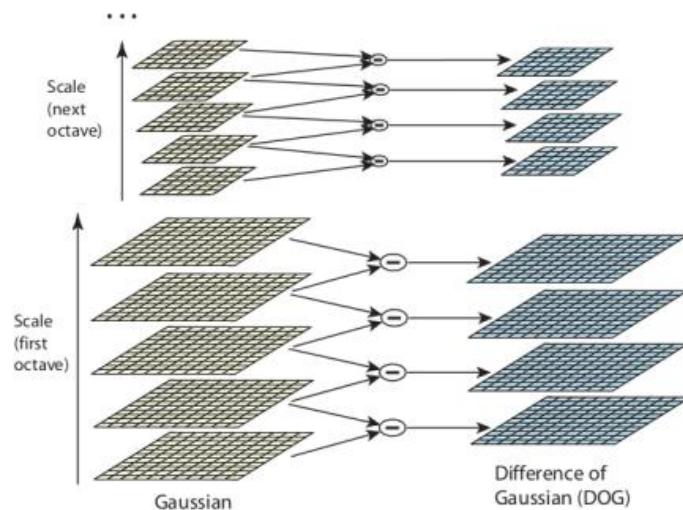
### 3.1 SIFT

Es un algoritmo de detección y descripción de características utilizado en “computer vision” y aprendizaje automático. Se emplea para encontrar puntos únicos y descripciones detalladas de una imagen que pueden ser comparados y emparejados con otras imágenes para lograr tareas como la superposición de imágenes y la reconstrucción 3D. SIFT, es una metodología que le es invariantes la escala, lo que significa que las características detectadas son las mismas independientemente de la escala de la imagen. Este algoritmo es robusto en temas de cambios de iluminación y rotación.

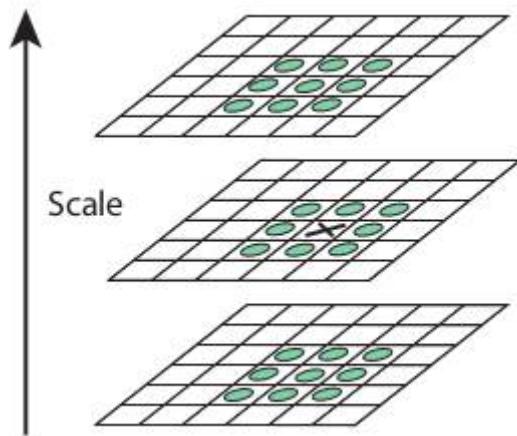
Tal como escribió D.Lowe, de la Columbia Británica con sede en Vancouver en 2004, en el paper “Distinctive Image Features from Scale-Invariant Keypoints” (considerado el mejor material sobre SIFT, además de ser sencillo de leer), nos comenta que hay cuatro pasos principales en el proceso del algoritmo de SIFT.

#### 3.1.1 Scale-space Extrema Detection

En primer lugar, utilizaremos un filtro espacio-escala, que consiste en el laplaciano de Gauss con varios valores de  $\sigma$ . El LoG actúa como un detector de puntos clave que puede detectar estos puntos en diferentes tamaños debido a su variación en  $\sigma$  (que actúa como parámetro de escala). Podemos encontrar máximos locales en la escala y el espacio, lo que nos proporciona una lista de valores ( $x, y, \sigma$ ), lo que indica la presencia de un posible punto clave en  $(x, y)$  en la escala  $\sigma$ . Lamentablemente, el cálculo del LoG es costoso, por lo que el algoritmo SIFT usa la aproximación de la diferencia de gaussianos, que es una aproximación del LoG. La diferencia de gaussiano se obtiene restando dos imágenes suavizadas con diferentes valores de  $\sigma$ , donde uno es  $\sigma$  y el otro es  $k\sigma$ . Este proceso se realiza en diferentes octavas de la imagen en la Pirámide Gaussiana.



Una vez que se ha calculado esta diferencia de gaussiano (DoG), buscamos máximos locales en la escala y el espacio.



### 3.1.2 Keypoint Localization

Una vez que hemos identificado posibles ubicaciones de puntos clave, es necesario refinar estas ubicaciones para obtener resultados más precisos. El DoG tiende a responder más a los bordes, por lo que es necesario eliminar los puntos clave ubicados en bordes. Para lograr esto, se utiliza una matriz de Hesse ( $H$ ) de  $2 \times 2$  para calcular la curvatura principal. En el artículo original se emplea una función simple que establece que si esta relación es mayor que un umbral, se descarta ese punto clave, como se menciona en el artículo.

### 3.1.3 Orientation Assignment

A continuación, asignamos una orientación a cada punto clave para lograr invariancia a la rotación de la imagen. Tomamos una vecindad alrededor de la ubicación del punto clave de acuerdo con su escala y calculamos la magnitud y dirección del gradiente en esa región. Luego, creamos un histograma de orientación con 36 contenedores que cubren 360 grados. Se selecciona el pico más alto del histograma y cualquier pico que supere el 80% de ese valor también se considera para calcular la orientación. Esto suscita puntos clave con la misma ubicación y escala pero en diferentes direcciones.

### 3.1.4 Keypoint Descriptor & Matching

Luego, originamos el descriptor del punto clave. Tomamos los valores de los píxeles en un área de  $16 \times 16$  alrededor del punto clave y lo dividimos en 16 sub bloques de  $4 \times 4$ . Para cada sub bloques, creamos un histograma de orientación con 8 contenedores. Esto resulta en un vector de 128 valores de bins. Este vector se representa de manera que se toman medidas para lograr robustez frente a cambios.

Finalmente, relacionamos los puntos clave entre dos imágenes buscando sus vecinos más cercanos. En algunos casos, la coincidencia puede estar muy cerca de la primera debido al ruido. En tales casos, calculamos la relación entre la distancia más cercana y la segunda distancia más cercana. Si esta relación es mayor que 0.8, se rechaza la coincidencia, lo que elimina alrededor del 90% de las coincidencias falsas y solo descarta el 5% de las coincidencias correctas.

## **3.2 AdaBoost**

Es un algoritmo de aprendizaje para mejorar la precisión de otros algoritmos de aprendizaje supervisado. Es un acrónimo de Adaptive Boosting y funciona aumentando el peso de las muestras de entrenamiento mal clasificadas en cada iteración y, así, brindar más importancia a aquellas muestras que son difíciles de clasificar correctamente. El resultado final es una combinación de varios modelos débiles que se combinan para formar un modelo más fuerte.

## **3.3 TextonBoost**

Es un algoritmo de aprendizaje automático desarrollado por J. Shotton y J. Winn en 2006, es usado en “computer vision” para la detección de objetos y la clasificación de imágenes. Se basa en la técnica de aprendizaje por refuerzo, AdaBoost. Este, utiliza características textuales en lugar de características puntuales para clasificar las imágenes, lo que permite una mayor robustez ante las variaciones en la escala y la orientación de los objetos.

## **3.4 Computer Vision**

La visión por ordenador (computer vision) es una de las ramas de la inteligencia artificial que permite a los ordenadores poder ver lo que ocurre en pantalla, poder realizar análisis de imágenes, o incluso poder ver a través de una cámara, es decir, cualquier estímulo visual, que pueda ser enviado por algún método a un ordenador. Esta visión funciona como los ojos humanos, recibimos unos inputs, que estos serían las emisiones de luz capturadas por nuestros ojos, en el caso de un ordenador, sería el color, de cada pixel en pantalla, o incluso, para facilitar el trabajo al ordenador, se suele convertir la imagen a blanco y negro, para que el ordenador solo tenga que analizar mediante un 0 si el color del pixel es negro y un 1 si el color del pixel es blanco.

Seguidamente de recibir el input, este mismo, es mandado al cerebro, para que pueda interpretar de lo que se trata el objeto que estamos percibiendo, en caso de no tener conocimiento de lo que es el mismo, poder analizarlo y la próxima vez que lo veamos, saber lo que es.

La visión por ordenador, tiene la ventaja, a diferencia de la visión humana, de que puede analizar miles de imágenes por minuto, y detectar la más mínima protuberancia, así, superando las capacidades humanas.

### **3.4.1 ¿Cómo funciona la visión por ordenador?**

La visión por ordenador funciona mediante “data”, es decir, toda la información que ha recibido mediante la etapa de entrenamiento, en esta etapa de entrenamiento, es necesario tener una cantidad masiva de imágenes, para que esta misma pueda ser entrenada.

Para que esta tecnología pueda ser ejecutada, hay dos tipos de “machine learning” que son esenciales, el DL (Deep Learning) y los CNN (Convolutional Neural Network).

El Machine Learning se basa en usar modelos algorítmicos que le permiten aprender sobre el contexto de unos datos visuales, al darle muchos datos, esta va a ser capaz de diferenciar entre objetos, al igual que hace la visión humana.

# 4

## Herramientas

### 4.1 OpenCV

Es una biblioteca de “computer vision” de código abierto, que proporciona funciones para el procesamiento de imágenes y videos. Se utiliza en una amplia variedad de aplicaciones de inteligencia artificial, incluyendo la detección de objetos, el seguimiento de objetos, la reconstrucción 3D, el reconocimiento de patrones y la segmentación de imágenes. Este fue desarrollado originalmente por Intel y se ha convertido en uno de los principales proyectos de “computer vision”, con una amplia comunidad de desarrolladores y usuarios.

### 4.2 UE5

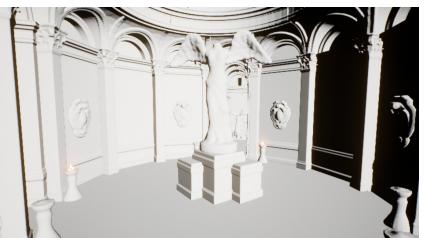
Unreal Engine 5, es un motor de videojuegos, el cual te proporciona una amplia variedad de funciones y recursos. La marketplace de Unreal Engine 5, nos proporciona 5 assets de pago, de forma totalmente gratuita cada mes. Luego tenemos Lumen y Nanite, unas técnicas de RT y de virtualización de la geometría. Las cuales hacen que el “realismo” aumente, consumiendo consigo menos recursos.



Scene View



Global Distance Field



Mesh Distance Fields

Aparte de todo esto, UE5, nos proporciona un proyecto llamado CitySample, implementando todas estas tecnologías, aparte de IA para pederastians (peatones), vehículos y su sistema de Chaos, para conducción y colisiones.



*Nanite y Lumen están siendo usados.*

## 4.3 Plataformas de desarrollo de inteligencia artificial

Para trabajar con inteligencia artificial, hay unas plataformas, llamadas plataformas de desarrollo de inteligencia artificial, las cuales nos facilitan las tareas del desarrollo de esta misma, estas normalmente son las que se encargan de poder almacenar los datos de entrenamiento, y de usar sus algoritmos específicos para que la IA, pueda ver de una forma más veloz en esos datos de entrenamiento y consumiendo menos recursos, unas de ellas podrían ser:

### 4.3.1 Tensorflow

Es una biblioteca OpenSource sobre el aprendizaje automático a través de un rango de tareas, esta fue desarrollada principalmente por Google, ya que estos requerían de sistemas capaces de construir y entrenar redes neuronales para detectar y descifrar patrones y correlaciones, según Tensorflow, sus beneficios son la creación de modelos de una forma sencilla mediante el uso de la API de alto nivel de Keras, a parte de eso te permite realizar iteración instantánea y depuración intuitiva, para tareas grandes de AA, esta API hace uso de estrategias de distribución de entrenamiento distribuido en diferentes configuraciones del hardware sin cambiar la definición del modelo. Aparte de eso, tensorflow ofrece un camino directo hacia producción.

### **4.3.2 PyTorch**

PyTorch es una librería para Python y C++ la cual soporta un end-to-end machine learning, es decir, un aprendizaje automático que abarca todo, desde la adquisición de datos hasta la entrega de los mismos. Pytorch está listo para producción y permite el uso de distribuir el entrenamiento, actualmente tiene un ecosistema muy robusto y están implementando una función beta, la cual es, que los usuarios de la comunidad de PyTorch, puedan subir sus propios datos de entrenamientos a la página para que cualquiera los pueda usar. PyTorch, fue desarrollada por FaceBook y al igual que tensorflow es de código abierto. Una de sus características mas destacadas es el enfoque que tiene en la computación de gráficos, que permite a sus usuarios definir modelos de ML como grafos cílicos dirigidos (DAG) de operaciones matemáticas, esto hace que PyTorch sea muy efectivo en la creación de modelos de redes neuronales profundas, también añade soporte de compatibilidad con otras librerías como Numpy, SciPy y Pandas.

### **4.3.3 Keras**

Keras, es un sistema montado encima de Tensorflow, que te permite montar clusters grandes de GPU's o un TPU pod entero (también puede ser usado en CPU). Es una biblioteca de redes neuronales escrita en Python y sirve para crear y entrenar modelos de IA usando una interfaz de alto nivel. Keras se centra en la facilidad de uso, para que no tenga complejión para nadie, la modularidad y la extensibilidad, aparte de ser usada en Tensorflow, también te permite usarla en Theano, Microsoft Cognitive Toolkit entre otras.

### **4.3.4 Caffe**

Caffe es un framework de DL OpenSource desarrollado por BAIR, Berkely AI Research en la Universidad de California, Berkely. Se usa para *computer vision*, procesamiento del lenguaje natural y reconocimiento de voz. Éste ha sido diseñado para ser rápido y eficaz utilizando el lenguaje C++ para su desarrollo, contiene una interfaz en Python y MATLAB para la construcción y entrenamiento de modelos. Incluye una amplia gama de capas de CNN, compatibilidad con múltiples GPUs para el proceso de entrenamiento y predicción y muchas herramientas de visualización y análisis de los modelos.

## **4.4 Procesadores y sistemas embebidos**

Los procesadores y sistemas embebidos proporcionan la capacidad de realizar cálculos complejos en tiempo real.

Los procesadores que son utilizados en los sistemas embebidos deben ser capaces de manejar múltiples tareas de forma simultánea, como la adquisición de datos de sensores, el procesamiento de imágenes, el reconocimiento de objetos y la toma de decisiones en tiempo real.

### **4.4.1 ASIC**

Los ASIC son chips diseñados para realizar una tarea específica de manera eficiente. En la conducción autónoma, se han desarrollado ASIC's especializados en el procesamiento de imágenes y el reconocimiento de objetos. Estos pueden realizar operaciones de CV de manera mucho más rápida y eficiente que los procesadores convencionales.

## **4.4.2 GPU**

Las GPU son procesadores altamente paralelos que se usan comúnmente en aplicaciones gráficas, aunque también son usados en CV. Estas pueden realizar operaciones en paralelo en grandes conjuntos de datos, lo que las hace ideales para acelerar algoritmos de ML y procesamiento de imágenes.

## **4.2.3 SoC**

Los SoC son sistemas que integran todo en un único chip, como procesadores, GPU, memoria y otros periféricos. Estos sistemas son ampliamente utilizados en dispositivos móviles y también se están usando cada vez más en aplicaciones de conducción autónoma. Los que son diseñados exclusivamente para la conducción autónoma suelen incluir capacidades de procesamiento de imágenes y aprendizaje automático,

## **4.5 HMI**

La HMI, es un componente esencial en el campo de la conducción autónoma. Es la forma de comunicación entre los sistemas, y los humanos.

Esta desempeña un papel crucial en la experiencia del usuario. Una interfaz intuitiva y bien diseñada puede mejorar la confianza del conductor y garantizar una comunicación efectiva entre el vehículo autónomo y los pasajeros o usuarios.

Aspectos importantes:

### **4.5.1 Visualización de información**

Una presentación clara, es crucial para el humano que está dentro del vehículo, esta debe contener datos relevantes y fundamentales, tales como visualización de sensores, mapas, rutas planificadas, estados del sistema, advertencias...

### **4.5.2 Retroalimentación háptica y acústica**

Además de la información visual, es crucial que el conductor también reciba “inputs” mediante otras vías, como puede ser la acústica y háptica, que el vehículo mande una vibración o un sonido de alerta, puede hacer que el conductor, se dé cuenta mucho antes, que mirando una pantalla, de que algo está fallando, o que ocurre algo.

### **4.5.3 Comandos y control**

La HMI, debe permitir al usuario controlar el vehículo de forma cómoda y eficaz, esto puede incluir el hecho de controlarlo mediante controles de voz, táctiles, gestos o incluso BCIs.

### **4.5.4 Comunicación y estado del sistema**

Este debe contener avisos tanto para los usuarios externos al vehículo como para los pasajeros del mismo, tanto podrían ser alertas de sonido, usar los intermitentes vehiculares...

# 5

## Explicación de la conducción autónoma.

### 5.1 Niveles de conducción autónoma

Actualmente hay 5 niveles en la conducción autónoma, los cuales han sido estipulados por NHTSA y la SAE.

#### 5.1.1 Nivel 1: Asistentes de conducción

Este nivel, es el que engloba la amplia mayoría de vehículos, estos, son los que tienen un control de velocidad en crucero o incluso un asistente para cambio de carril, los vehículos que están en este nivel, no pueden llegar a hacer esas dos cosas a la vez, sino que solo pueden hacer una. El conductor es siempre quien toma la acción de cualquier movimiento o acción que haga el vehículo, este requiere una supervisión total y constante.

#### 5.1.2 Nivel 2: Semi-autonomía

Este nivel, es la evolución de los asistentes de conducción, son vehículos capaces de combinar el control de velocidad y el asistente de cambio de carril, también puede incluir detectores de ángulos muertos o sistemas de frenadas de emergencia. Este nivel permite al conductor estar más despreocupado, incluso llegando momentos donde no tiene que tocar ni pedales ni volante, aunque sigue requiriendo una atención total.

#### 5.1.3 Nivel 3: Autónomo Controlado

En este nivel, el vehículo ya puede monitorear su entorno y determinar cualquier imprevisto y actuar a su favor. El vehículo, de forma autónoma, puede cambiar de carril, realizar una frenada de emergencia e incluso frenar en caso de posible colisión cuando un vehículo se nos cruce, eso sí, manteniéndose dentro de las líneas de carril. El conductor sigue siendo un elemento necesario, ya que el sistema requiere de la actuación cuando el sistema falle, en este nivel se encuentra el “Tesla autopilot”, o el sistema de conducción autónoma del Audi A8.

#### 5.1.4 Nivel 4: Alta automatización

En este nivel, el humano empieza a dejar de ser relevante, ya que el sistema, aunque se le produzca un fallo, puede regresar a una zona de confort, como podría ser aparcar a un lado de la calzada, es decir, conducir hasta una situación de riesgo mínimo.

## **5.1.5 Nivel 5: Automatización total**

En este nivel, el conductor deja de ser requerido, el vehículo puede conducir de forma totalmente autónoma y ocuparse de todo por cuenta propia, está preparado para responder ante cualquier eventualidad, en los prototipos futuros que hemos logrado ver, los vehículos destacan en no tener volantes ni pedales, como puede ser el vehículo que presento Tesla con su FSD (el Tesla Model 2).

## **5.2 Limitaciones de la conducción autónoma**

Por muy prometedora que la conducción autónoma sea, aun existen limitaciones, las cuales, pueden ser solucionadas con mayor entrenamiento y mejoras en los vehículos.

### **5.2.1 Fiabilidad**

Por muy fiable que sea la tecnología, aún, al igual que un humano, no es 100% segura y se le pueden producir fallos en la misma, aún faltan años de entrenamiento, para que la IA, pueda abordar todos los problemas que puedan aparecer en el proceso de conducción.

### **5.2.2 Tiempo**

La mayoría de estos sistemas, se basan en el uso de cámaras y sensores, los cuales con el mal tiempo se pueden ver afectados, muchas empresas están adoptando cartas en el asunto, como Tesla, tal como se vio en el video [<https://www.youtube.com/watch?v=fKXztwtXaGo>] se puede ver como el sistema de IA analiza si la carretera está sucia, mojada, rota...

### **5.2.3 Cambios imprevistos**

Estos sistemas, aún no están del todo preparados para lidiar con cambios imprevistos que podrían suceder en el día a día, como el de que un semáforo no funcione o un policía este dando señales, un accidente automovilístico...

### **5.2.4 Costo**

La implementación de estos sistemas, aún puede llegar a ser muy costosa, ya que para que la misma funcione, se requiere de un alto coste de recursos.

## **5.3 Arquitectura del sistema de conducción autónoma**

La conducción autónoma requiere de una arquitectura de sistema compleja para lograr su funcionamiento de manera efectiva. Ésta arquitectura está basada en una serie de componentes que interactúan entre sí para permitir la percepción del entorno, la toma de decisiones y la ejecución de estas mismas. Esta arquitectura es variable dependiendo del nivel de autonomía vehicular, explicados en el punto 5.1.

### **5.3.1 Sensores**

Los sensores son los encargados de capturar la información del entorno para tener una percepción del mismo y saber como actuar en cada situación, los más comunes son las cámaras, lidars y los sensores de ultrasonido.

### **5.3.2 Percepción**

Cuando los datos han sido capturados por los sensores, son procesados para hacer una percepción del entorno y poder comprender-lo, así mismo poder detectar objetos relevantes como vehículos, peatones, señales de tráfico y obstáculos.

### **5.3.3 Toma de decisiones**

Una vez que ya todos los datos han sido procesados y el vehículo ya conoce todo su entorno, este mismo procede a tomar las decisiones, esto implica evaluar la situación actual, planificar rutas, maniobras, y determinar las acciones adecuadas para cada momento.

### **5.3.4 Control y actuación**

Las decisiones tomadas en el punto 5.3.3 se traducen en comandos para el vehículo, para que este mismo pueda, girar el volante, encender luces, frenar, pisar el acelerador...

## **5.4 Desafíos técnicos y de implementación**

La conducción autónoma plantea una serie de desafíos técnicos y de implementación que deben abordarse para lograr un funcionamiento seguro y confiable.

### **5.4.1 Percepción y comprensión del entorno**

Es fundamental que el vehículo sea capaz de detectar y comprender de manera precisa toda la información de su alrededor, como objetos, peatones, vehículos, señales de tráfico, y otros elementos relevantes en el entorno de conducción. Esto implica desafíos como la variabilidad de iluminación en escena, la oclusión de estos objetos, la detección y seguimiento de los mismos de una forma veloz...

### **5.4.2 Toma de decisiones segura**

Deben ser capaces de tomar decisiones seguras de forma rápida, esto implica evaluar constantemente el entorno que les rodea, anticiparse ante posibles escenarios que puedan ocurrir y tomar las decisiones en base a la normativa de tráfico. Ésta debe considerar la velocidad, la posición de otros vehículos, las señales de tráfico, las condiciones de la carretera y las preferencias del conductor.

## **5.5 Casos de uso y aplicaciones**

La conducción autónoma puede tener distintos casos de uso en diferentes áreas, unos ejemplos podrían ser:

- Transporte de pasajeros: Empresas de tecnología y fabricantes de automóviles están desarrollando vehículos autónomos de transporte compartido, como podría ser Tesla o Waymo, estos servirían de taxis autónomos y shuttles
- Logística y entrega: Tal como ideó Tesla, con su camión, el Tesla Semi, un sistema de camiones autónomos en forma de Convoy, es decir, un conductor humano en el primer vehículo, y todos los

demás, compartiendo sus sensores entre vehículos, yendo detrás del primero, incluso, el primero siendo manejado de forma autónoma.

- Seguridad y vigilancia: Se están planteando la idea de desarrollar vehículos autónomos para el cuerpo policial, que patrullen las zonas urbanas, para detectar infracciones de tránsito y actividades sospechosas por parte de los civiles. Estos vehículos pueden proporcionar una vigilancia constante y reducir el riesgo para el personal de seguridad.

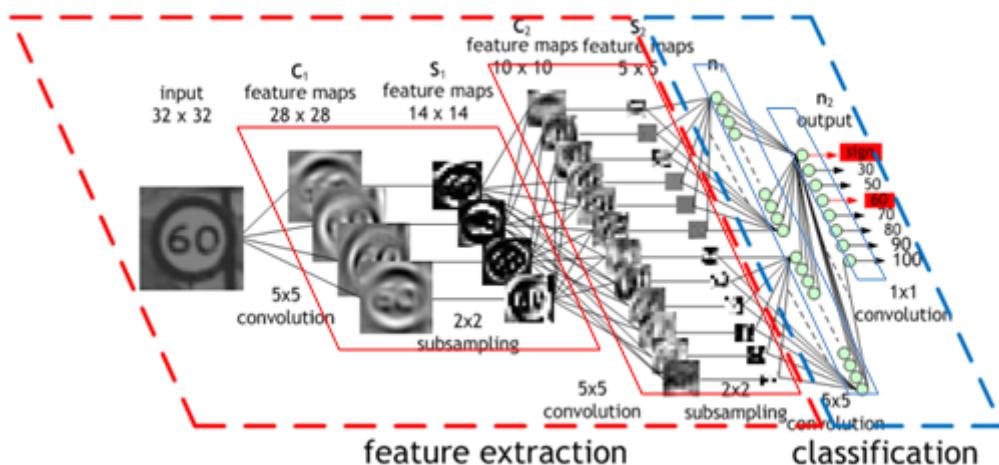
- Servicios de emergencia: La conducción autónoma también se está explorando en el ámbito de emergencia como bomberos y ambulancias, donde estos puedan llegar rápidamente a la escena de un accidente o emergencia proporcionando una asistencia rápida y reduciendo el tiempo de respuesta,

# 6

## Algoritmos de aprendizaje automático

### 6.1 CNN

Las CNN son un tipo de arquitectura de DL que se suele usar en el campo de procesamiento de imágenes, reconocimiento de patrones y visión por ordenador. Estas mismas fueron diseñadas para emular el funcionamiento de la visión humana, proporcionando que estas mismas puedan clasificar objetos, detectar objetos...



El desarrollo de las CNN fue inspirado en la organización del sistema visual del cerebro humano, las neuronas en las primeras capas responden a características locales como bordes y texturas, mientras que las capas superiores se activan delante de los patrones que son de nivel mas complejo y abstracto. Las CNN se aplican mediante varias capas:

#### 6.1.1 Capa de convolución

En esta capa la CNN aplica filtros en la imagen para que ésta pueda extraer características relevantes, estos filtros, son los nombrados filtros kernels. Los filtros son pequeñas matrices que se desplazan a través de la imagen realizando operaciones de convolución.

#### 6.1.2 Capa de activación

Se aplica una función de activación como ReLU (explicado en el punto 6.2) para introducir no linealidades en la red y permitir que ésta pueda aprender tareas más avanzadas.

### **6.1.3 Capa de agrupación**

Reduce las dimensiones del mapa de características de la convolución al tomar el valor máximo o promedio de una agrupación de píxeles. Ésto ayuda a reducir la cantidad de parámetros que la CNN tiene que tener en cuenta, además de obtener una representación más compacta y significativa.

### **6.1.4 Capa completamente conectada**

Los datos son introducidos en un vector y alimentan a una o mas capas de neuronas completamente conectadas para realizar tareas de clasificación, detección u otras tareas.

## **6.2 RNN**

Las RNN son un tipo de arquitectura diseñada para procesar secuencias de datos, donde la información del pasado influye en la del presente y, a su vez, afecta a la futura. Estas, son especialmente adecuadas para tareas que involucran, de alguna forma, secuencias temporales, como el procesar texto, la traducción, el reconocimiento de voz...

A diferencia de las redes neuronales convencionales, las RNN tienen conexiones recurrentes que forman bucles en su arquitectura. Esto le da acceso a que esta información se pueda “recordar” o se pueda tener en cuenta en un futuro.

Su formulación matemática en un paso de tiempo ( $t$ ), se puede expresar de la siguiente forma:

$$h_t = f(W_{hh} * h_{(t-1)} + W_{xh} * x_t + b_h)$$

En esta formula tenemos que:

$h_t$  es el estado oculto en el paso de tiempo  $t$

$x_t$  es la entrada del paso de tiempo  $t$

$W_{hh}$  es la matriz de pesos que conecta el estado oculto anterior  $h_{t-1}$  con el estado oculto actual  $h_t$

$W_{xh}$  es la matriz de pesos que conecta la entrada  $x_t$  con el estado oculto actual  $h_t$

$b_h$  es el sesgo de la capa oculta

El estado oculto en cada paso contiene información sobre las observaciones de la secuencia anterior y se usa para la entrada para el siguiente paso de tiempo.

## 6.3 RL

El Reinforcement Learning, es una rama del aprendizaje automático que se enfoca en enseñar a un agente a tomar decisiones óptimas en un entorno incierto para maximizar la recompensa a lo largo del tiempo. Este paradigma está inspirado en el proceso de aprendizaje en los seres vivos.

RL se basa en un modelo de interacción tripartita, que incluye:

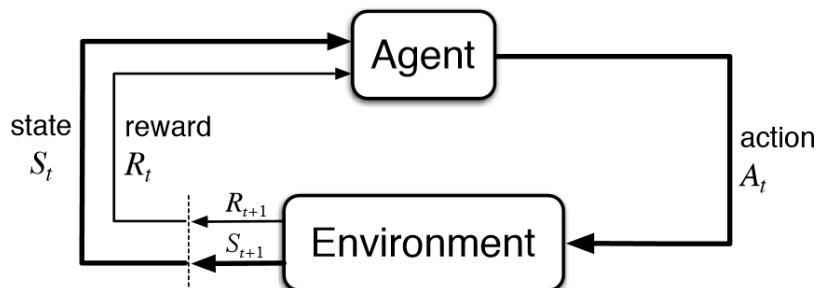
1. Agente
2. Entorno
3. Recompensa

El objetivo del agente en RL es aprender una política, que es una función que mapea los estados a acciones. Esta política le permite al agente decidir qué acción tomar en cada estado para maximizar la recompensa total que acumula a lo largo del tiempo.

El proceso de entrenamiento implica:

1. Exploración
2. Explotación

El enfoque fundamental en RL es el equilibrio entre exploración y explotación, este agente requiere explotar lo suficiente para descubrir acciones y estados desconocidos, pero también debe explotar el conocimiento adquirido para maximizar las recompensas en estados conocidos.



Algunos de los algoritmos más populares utilizados en RL son:

- Q-learning
- SARSA
- Policy Gradient
- Deep Q-Networks (DQN)

### 6.3.1 Agente

Es el sistema que realiza las acciones en el entorno y aprende a través de la experiencia.

### 6.3.2 Entorno

Es el mundo con el que el agente interactúa y donde se desarrollan todas las acciones.

### **6.3.3 Recompensa**

Es la señal de retroalimentación que recibe el agente después de realizar una acción. Indica cuán buen o mala fue esa acción con respecto al objetivo del agente.

### **6.3.4 Exploración**

El agente explora el entorno realizando acciones y recopilando datos para aprender sobre las recompensas y la dinámica del entorno.

### **6.3.5 Explotación**

El agente utiliza la información de la exploración para tomar decisiones más óptimas y maximizar la recompensa esperada.

### **6.3.6 Q-Learning**

Es un algoritmo de aprendizaje por valor (value-based) que aprende la función Q, que estima el valor esperado de tomar una acción en un estado específico.

### **6.3.7 SARSA**

Es un algoritmo de aprendizaje por valor que actualiza la función Q en función de las experiencias de un par de estados y acciones consecutivas.

### **6.3.8 Policy Gradient**

Su enfoque se centra directamente en la política, ajustando los pesos de una red neuronal para mejorar su rendimiento.

### **6.3.9 DQN**

Es una combinación entre Q-Learning y redes neuronales profundas, lo que le permite aprender funciones Q más complejas y aplicar RL a problemas de mayor dimensionalidad.

### **6.3.10 Q Function**

La función Q se define de la siguiente manera:

$$Q(s, a)$$

Donde:

Q es la función Q, que toma como entrada un estado s y una acción a.

s representa el estado actual en el entorno.

a representa una acción que el agente puede tomar en el estado s.

La función Q puede ser interpretada como la “calidad” o “valor” que se obtiene al realizar la acción (a) en el estado (s)

## 6.4 DL

El DL es una rama de ML que utiliza algoritmos de redes neuronales artificiales para modelar y resolver problemas complejos. El DL se caracteriza por el uso de redes neuronales profundas, que son modelos con múltiples capas ocultas, lo que permite aprender representaciones jerárquicas y abstractas de los datos.

El proceso de entrenamiento de DL implica:

1. Inicialización
2. Propagación hacia adelante (Forward Pass)
3. Cálculo de pérdida (Loss)
4. Retropropagación (Backpropagation)
5. Iteración

El DL ha resultado ser altamente eficiente en la resolución de problemas en una variedad de dominios, como CV, procesamiento de lenguaje natural, reconocimiento de voz, juegos, robótica, entre otros. Entre las aplicaciones destacadas se encuentran:

- Clasificación de Imágenes
- Procesamiento del Lenguaje Natural (NLP)
- Juegos
- Conducción Autónoma
- Medicina

### 6.4.1 Inicialización

Los pesos de la red se inicializan aleatoriamente o mediante algún esquema predefinido.

### 6.4.2 Forward Pass

Los datos se introducen en la red y se propagan a través de las capas para obtener predicciones.

### 6.4.3 Loss

Se evalúa qué tan buenas son las predicciones en comparación con las etiquetas reales mediante una función perdida.

### 6.4.4 Backpropagation

Se calculan los gradientes de la función de pérdida con respecto a los pesos de la red y se utilizan para actualizar los pesos de la red y se utilizan para actualizar los pesos mediante algún algoritmo de optimización.

#### **6.4.5 Iteración**

Se repiten los pasos 2 a 4 varias veces (épocas) hasta que la red converja una solución o se alcance un criterio de parada.

#### **6.4.6 Clasificación de Imágenes**

Reconocimiento de objetos, detección de rostros, clasificación de escenas, etc.

#### **6.4.7 NLP**

Traducción automática, generación de texto, análisis de sentimientos, etc.

#### **6.4.8 Juegos**

Aprendizaje de máquina mediante el juego, como en los algoritmos AlphaGo y AlphaZero.

#### **6.4.9 Conducción Autónoma**

Percepción del entorno, toma de decisiones y control en vehículos autónomos.

#### **6.4.10 Medicina**

Diagnóstico de enfermedades a partir de imágenes médicas y análisis de datos pacientes.

### **6.5 ReLU**

ReLU significa Unidad Lineal Rectificada, es una función de activación que se puede usar en las CNN, esta función introduce no linealidad en la red y se define de la siguiente manera:

$$f(x) = \max(0, x)$$

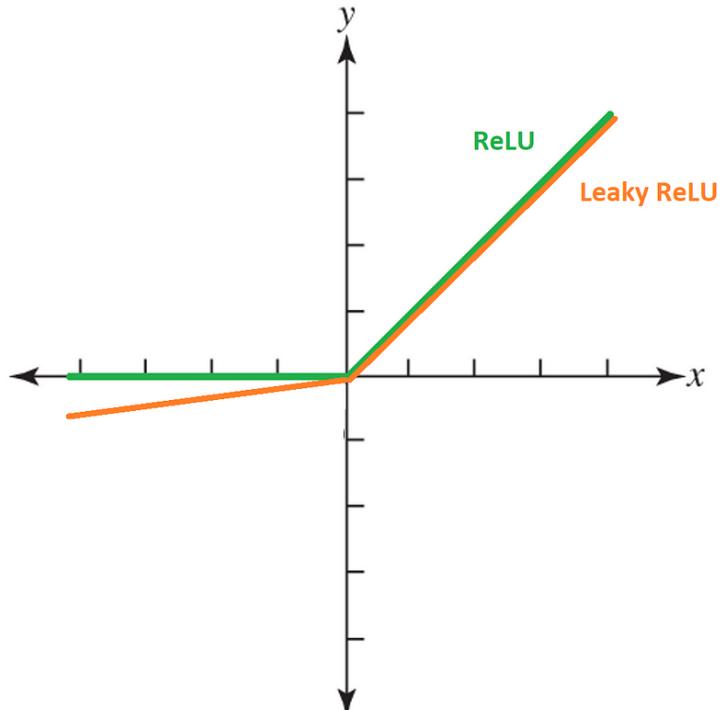
La función ReLU toma su primer valor (x) y devuelve el máximo entre ese valor y 0.

Si el valor de entrada es positivo, ReLU devuelve el mismo valor, en el caso de ser negativo, devuelve 0.

La principal ventaja de ReLU es que es computacionalmente eficiente y facilita el entrenamiento de redes profundas, ya que evita el problema de desvanecimiento del gradiente que se solía presentar en otras funciones como la sigmoide o la tangente hiperbólica.

### 6.5.1 El problema de ReLU

ReLU puede presentarnos un problema llamado “Dying ReLU”, en el cual algunas de sus neuronas quedan completamente inactivas y no se actualizan en peso durante su entrenamiento, lo que puede dar a una degradación del rendimiento del modelo. Para solucionar este problema, se han desarrollado forks de ReLU, como Leaky ReLU, Parametric ReLU... que permiten un pequeño flujo de información para valores negativos, evitando así la inactividad de las neuronas.



# 7

# Simulación

## 7.1 Selección del simulador

Después de una exhaustiva búsqueda y evaluación de distintos simuladores disponibles, se procedió a la selección del simulador más adecuado para este proyecto. A continuación, se presenta un resumen de los simuladores considerados y las razones clave que llevaron a la elección de GTA V con ScriptHookV:

- AirSim, Apollo y CARLA: Si bien son simuladores potentes, su adaptación resultaba compleja para este proyecto, lo que podría retrasar significativamente el desarrollo.
- DeepTraffic: Aunque es amigable para usuarios promedio, su limitación para modificar parámetros de algoritmo de RL no lo hacía ideal para implementar un enfoque personalizado.
- Robot-benchmark: Aunque potente y relativamente fácil de adaptar, carecía de la capacidad de reiniciar el entorno de forma automática, un requisito esencial para el entrenamiento de RL.

Finalmente, se optó por GTA V con ScriptHookV debido a su idoneidad para este proyecto con una adaptación relativamente sencilla. Además, brinda la flexibilidad necesaria para modificar cualquier parámetro del entorno de manera eficiente, lo que lo convierte en la elección más adecuada para la investigación y desarrollo de algoritmos de RL en este contexto.

# 8

# Proyecto Práctico

## 8.1 Descripción del Proyecto

El proyecto práctico se trata de una iniciativa diseñada con simplicidad y facilidad de implementación en mente. Su propósito principal es ofrecer una plataforma de simulación autónoma con una API intuitiva, pensada para futuros desarrolladores interesados en continuar o contribuir al proyecto. En el marco de este proyecto, se ha desarrollado un programa capaz de detectar y clasificar todos los objetos detectables por un vehículo a través de una única cámara (para el uso de múltiples cámaras, se requerirán varias instancias del proyecto). Los resultados de estas detecciones pueden exportarse de manera sencilla a través de la API integrada, facilitando así la recopilación de datos y la interacción con el entorno simulado.

Todo el proyecto, se puede encontrar en su repositorio público de GitHub:  
<https://github.com/MayiVT/FSD>

## 8.2 Diseño del sistema

El proyecto se ha estructurado en dos partes fundamentales: el proceso de "Entrenamiento" y el de "Explotación".

### 8.2.1 Entrenamiento:

#### Entrenamiento de IA:

Para la fase de entrenamiento, se requiere un vídeo o un conjunto de vídeos unidos de vehículos como entrada. Se recomienda contar con más de 20 horas de material de video para lograr resultados óptimos. El proceso comienza con la importación del video a la aplicación diseñada para este propósito. La aplicación se encargará de procesar el video y exportará los resultados en una carpeta denominada “results”. Cada fotograma del video generará un archivo en formato `api-exported-data-<fotograma>.json`. Se debe tener en cuenta que este proceso puede ocupar una cantidad considerable de espacio en disco.

#### Generación de Datos de Entrenamiento:

Luego, se deben importar estos archivos generados previamente en la carpeta “data” del programa de entrenamiento. Posteriormente, se ejecutará el programa de entrenamiento. En esta etapa, el programa presentará cada fotograma uno por uno y solicitará instrucciones sobre la acción que el vehículo debe realizar bajo las condiciones detectadas en ese momento. Aunque este proceso puede ser tedioso y llevar tiempo, es esencial para entrenar la inteligencia artificial de manera efectiva.

### **Resultado del Entrenamiento:**

Una vez completado el entrenamiento, el programa proporcionará un archivo que contiene todo el modelo de IA entrenado, listo para ser utilizado en la fase de explotación.

### **8.2.2 Explotación:**

#### **Implementación de la IA entrenada:**

En la fase de explotación, la IA previamente entrenada se importará al programa de servidor. Esto permitirá la utilización del modelo para tomar decisiones en tiempo real durante la simulación de conducción autónoma.

#### **Detección y Ejecución de Acciones:**

Para iniciar la instancia de detección y operación, se seleccionará la cámara deseada y se iniciará la detección junto con la API. El programa del servidor supervisará las condiciones actuales y determinará la acción que el vehículo debe llevar a cabo. Esta acción se ejecutará en el cliente del simulador, permitiendo así una conducción autónoma basada en las decisiones de la IA previamente entrenada.

En mi caso, como faltaba potencia de computación, lo que se realizó fue:

1. El simulador (GTA 5) queda congelado.
2. El programa detecta el fotograma y hace la predicción.
3. Envía la data mediante la API.
4. El programa hace la predicción y dice cual es la mejor acción que tomar en ese momentos, a su misma vez, este programa está apoyado por la IA interna del propio juego para mejorar su predicción, y a su vez, miles de “Michales”, uno de los tres personajes principales del juego, está marcando la ruta que el vehículo debe seguir, al igual que hay otros anclados al personaje. Con esto se logra que el vehículo pueda detectar distancias de manera mas sencilla, y tener una ruta ya prescrita.
5. Ejecuta la acción en el ordenador principal y descongela el siguiente fotograma del simulador.

# Webgrafía

- IBM. "Computer Vision". <https://www.ibm.com/topics/computer-vision>.
- IBM. "Machine Learning". <https://www.ibm.com/topics/machine-learning>.
- Lowe, David G. "Distinctive Image Features from Scale-Invariant Keypoints". International Journal of Computer Vision, 2, 2004. <https://www.cs.ubc.ca/~lowe/papers/ijcv04.pdf>.
- Schapire, Robert E. "Explaining AdaBoost". <http://rob.schapire.net/papers/explaining-adaboost.pdf>.
- Rodríguez Fernández, Jose Marcos. "TextonBoost for Image Classification". Universitat Politècnica de Catalunya, 2014. <https://upcommons.upc.edu/bitstream/handle/2099.1/21343/95066.pdf>.
- OpenCV. "OpenCV". <https://opencv.org/>.
- Epic Games. "Nanite: Virtualized Geometry in Unreal Engine". <https://docs.unrealengine.com/5.0/en-US/nanite-virtualized-geometry-in-unreal-engine/>.
- Epic Games. "Lumen Technical Details in Unreal Engine". <https://docs.unrealengine.com/5.0/en-US/lumen-technical-details-in-unreal-engine/>.
- Epic Games. "Unreal Engine FAQ". <https://www.unrealengine.com/es-ES/faq>.
- Society of Automotive Engineers Levels of Driving Automation - Standard for Self-Driving Vehicles. <https://www.sae.org/news/press-room/2018/12/sae-international-releases-updated-visual-chart-for-its-%E2%80%9Clevels-of-driving-automation%E2%80%9D-standard-for-self-driving-vehicles>
- Massachusetts Institute of Technology MIT 6.S091: Introduction to Deep Reinforcement Learning. <https://deeplearning.mit.edu>
- Bing Xu, Naiyan Wang, Tianqi Chen and Mu Li Empirical Evaluation of Rectified Activations in Convolution Network. arXiv:1505.00853v2 [cs.LG] 27 Nov 2015
- Kyushik Min and Hayoung Kim DRL Based Self Driving Car Control. [https://github.com/MLJejuCamp2017/DRL\\_based\\_SelfDrivingCarControl](https://github.com/MLJejuCamp2017/DRL_based_SelfDrivingCarControl)

