

# Universidad Complutense de Madrid

## Facultad de Informática

Reconocimiento de vehículos y su movimiento mediante  
redes neuronales convolucionales en vídeos

Recognition of vehicles and their movement through  
convolutional neural networks in videos



TRABAJO DE FIN DE GRADO EN INGENIERÍA DEL SOFTWARE

Curso académico 2021 – 2022

Fernando Bellot Rodríguez

Esther Peñas Martínez

Alejandro Ruiz Valero

Director: Gonzalo Pajares Martinsanz

Colaboradora externa: Clara Isabel López González

# **Dedicatoria**

## **Fernando Bellot Rodríguez**

A todos mis compañeros y amigos de la carrera, en especial a Neku, Mario y Juan, con los que he reido y sufrido durante estos años, pero han sido lo mejor y lo más importante de toda mi carrera. A las asociaciones ASCII y LAG, que me han ayudado a crecer como persona, donde he participado en infinidad de actividades y donde he conocido a grandes personas y las mejores amistades. A todo el equipo de volleyball de la facultad, que este último año me ha acompañado y apoyado y donde he conocido gente maravillosa. A mis compañeros del TFG, que me han aguantado y ha sido un placer trabajar con ellos y son amigos con los que me quedo. A todo el personal de la facultad: profesores, personal de limpieza, de la biblioteca, de la cafetería, conserjes, etc. que siempre me han tratado bien y hacen posible que la facultad funcione. Y, por último, a mi familia que me ha acompañado y apoyado durante todo este camino.

## **Esther Peñas Martínez**

A mi familia, que siempre me ha apoyado en todo momento y están cuando más los necesito. A mis profesores y compañeros de carrera, en especial a Merche y a Jairo, que estuvieron a mi lado en infinidad de proyectos, me ayudaron, me enseñaron y se convirtieron en buenos amigos. A mis compañeros del TFG, con los que tuve la oportunidad de compartir este último trabajo antes de dar fin a esta etapa. A mis chiquis, quienes me han aguantado en mis momentos de bajón, me han dado la fuerza para seguir adelante y me hicieron confiar en mí misma. Y, por último, a mí porque estoy orgullosa de lo que he conseguido y logrado durante todos estos años de carrera.

## **Alejandro Ruiz Valero**

A mis padres, que han estado siempre en los buenos y malos momentos, y me han aportado la energía necesaria para seguir adelante. A mi hermana, mi ejemplo a seguir. A Javier Mendoza, quien más representa para mí la amistad. A mis compañeros de carrera, con especial mención a mis compañeros de TFG, con los cuales he pasado una bonita etapa de mi vida y que echaré de menos. Y, por último, y aunque ya no esté conmigo, a mi abuela Catalina, la persona que me dió las bases de lo que soy.

# **Agradecimientos**

Queremos dar un agradecimiento especial a nuestro profesor de IC y tutor del TFG, Gonzalo Pajares Martinsanz, por su excelente disposición al trabajo, su ayuda y su orientación constante. Realizar este trabajo con él ha sido un lujo y una experiencia bastante instructiva de cara a nuestro futuro como informáticos. La colaboración de Clara I. López merece igualmente una mención especial.

# Resumen

Este trabajo tiene como finalidad desarrollar una aplicación, que utiliza técnicas de Aprendizaje Profundo, o *Deep Learning* (DL), en Inteligencia Artificial para la detección de movimiento de vehículos en secuencias de imágenes, con fines de su identificación dentro del flujo de tráfico y como posible solución en las futuras ciudades inteligentes.

El objetivo es proporcionar una solución conceptual para el control y seguimiento de vehículos sobre la vía urbana. Dentro del AP se utilizan concretamente dos modelos de Redes Neuronales Convolucionales (RNC), exactamente *AlexNet* y *GoogleNet* para la identificación de los vehículos en movimiento, el cual se detecta mediante técnicas basadas en flujo óptico como técnica propia de Visión por Computador. Estos modelos requieren de un entrenamiento previo. Así se consigue extraer rasgos característicos de los vehículos, a partir de una serie de imágenes secuenciadas en vídeos, que pueden ser previamente seleccionados por el usuario.

La aplicación desarrollada en este proyecto permite al usuario modificar los parámetros necesarios para el entrenamiento de las RNC, con la finalidad de poder obtener un resultado más eficiente y óptimo. Tras realizar el entrenamiento, el usuario también puede seleccionar previamente un video para generar la clasificación de los vehículos e identificarlos dentro del tráfico de vehículos.

Complementariamente, los datos obtenidos tras la clasificación llevada a cabo se subirán a la nube mediante el uso de la plataforma *ThingSpeak*, como servicio proporcionado por Matlab dentro del paradigma de Internet de las Cosas (*IoT*, *Internet of Things*) que permite el almacenamiento y análisis de distintas categorías de datos. Esta plataforma permite mostrar los resultados y enviar mensajes a través de la red social *Twitter* para su posterior valoración e interpretación.

## Palabras clave

- Flujo óptico.
- Aprendizaje profundo.
- Redes neuronales convolucionales.
- *ThingSpeak*.
- Internet de las cosas.
- Ciudades inteligentes.

# **Abstract**

The purpose of this work is to develop an application that uses Deep Learning techniques in Artificial Intelligence for vehicle movement detection in image sequences, with the purpose of identifying them within the traffic flow and as a possible solution in future smart cities.

The objective is to provide a conceptual solution for the control and monitoring of vehicles on urban roads. Within the Deep Learning (DL), two models of Convolutional Neural Networks (CNN) are specifically used, AlexNet and GoogleNet, for the identification of moving vehicles which is detected using techniques based on optical flow, as a Computer Vision technique. These models require previous training. It is possible to extract typical features of the vehicles, from a series of images sequenced in videos, which can be previously selected by the user.

The application developed in this project allows the user to modify the necessary parameters for the CNN training, in order to obtain a more efficient and optimal result. After completing the training, the user can also previously select a video to generate the vehicle classification and identify them within the vehicle traffic.

In addition, the data obtained after the classification carried out will be uploaded to the cloud by using the ThingSpeak platform, as a service provided by Matlab within the Internet of Things paradigm that allows storage and analysis of different categories of data. This platform allows the results to be displayed and sent via the social network Twitter for subsequent assessment and interpretation.

# **Keywords**

- Optical flow.
- Deep learning.
- Convolutional neural networks.
- ThingSpeak.
- Internet of things.
- Smart cities.

# Índice

<b>Dedicatoria</b>	<b>1</b>
<b>Agradecimientos</b>	<b>1</b>
<b>Resumen</b>	<b>2</b>
<b>Palabras clave</b>	<b>2</b>
<b>Abstract</b>	<b>3</b>
<b>Keywords</b>	<b>3</b>
<b>1. Introducción</b>	<b>7</b>
1.1 Antecedentes	7
1.2 Objetivos	9
1.3 Motivación	10
1.4 Plan de trabajo	10
1.5 Contribuciones personales	13
1.5.1 Fernando Bellot Rodríguez	13
1.5.2 Esther Peñas Martínez	15
1.5.3 Alejandro Ruiz Valero	18
1.6 Estructura de la memoria	21
1.7 Introduction	22
1.7.1 Preliminary	22
1.7.2 Objectives	24
1.7.3 Work plan	25
<b>2. Métodos conceptuales y técnicas aplicadas</b>	<b>27</b>
2.1 Cálculo del flujo óptico	27
2.2 Detección de regiones en movimiento	29
2.3 Redes Neuronales Convolucionales	30
2.3.1 AlexNet	34
2.3.2 GoogleNet	36
<b>3. Diseño y análisis desarrollado</b>	<b>37</b>
3.1 Arquitectura	38
3.1.1 Lado del cliente	40
3.1.2 Lado del servidor	44
3.2 Metodología y gestión de proyecto	49
3.3 Herramientas empleadas	50
<b>4. Resultados obtenidos</b>	<b>54</b>
4.1 Interfaz de la aplicación	54
4.1.1 Training	55
4.1.2 Classification	57
4.1.3 ThingSpeak	58
4.2 Entrenamiento de las redes	59
4.3 Detección de vehículos	64
4.4 Resultados de clasificación	65
4.5 Algoritmo de conteo	67
4.6 Errores más frecuentes	69

4.7 Integración de los datos y resultados en ThingSpeak	72
<b>5. Conclusiones y trabajo futuro</b>	<b>73</b>
5.1 Conclusiones	73
5.2 Trabajo futuro	75
<b>6. Conclusions and future work</b>	<b>76</b>
6.1 Conclusions	76
6.2 Future work	78
<b>7. Bibliografía</b>	<b>79</b>
<b>8. Anexos</b>	<b>82</b>
Anexo 1: Diagramas del entrenamiento	82
Anexo 2: Licencias de los iconos de la aplicación	85
Anexo 3: Licencias del esquema de arquitectura	87
Anexo 4: Manual de usuario y acceso al código	88

## Índice de Figuras

Figura 2.1 Ejemplo de convolución 2-D	31
Figura 2.2 Representación de la función ReLU	32
Figura 2.3 Max pooling	33
Figura 2.4 Modelo de red AlexNet	36
Figura 2.5 Módulo inception con incorporación de unidades ReLU	37
Figura 2.6 Modelo completo GoogleNet (inception V1)	37
Figura 3.1 Esquema de la arquitectura del sistema	38
Figura 3.2 Diagrama de casos de uso	39
Figura 3.3 Main Menu	41
Figura 3.4 Training	41
Figura 3.5 Redimensión de imágenes	43
Figura 3.6 Classification	44
Figura 3.7 Configuración del canal y visualización del Channel ID	45
Figura 3.8 Interfaz de ThingSpeak en la aplicación	47
Figura 3.9 Gráficas de los resultados en ThingSpeak	47
Figura 3.10 Configuración de ThingTweet	48
Figura 3.11 Ejemplo Tweet Clasificación	48
Figura 3.12 Ejemplo Tweet Check Traffic	49
Figura 4.1 Pantalla Inicial	54
Figura 4.2 Selección de Modelo de Red	55

Figura 4.3 Training AlexNet	55
Figura 4.4 Training GoogleNet	56
Figura 4.5 Training Seleccionado	56
Figura 4.6 Classification Seleccionado	57
Figura 4.7 ThingSpeak Seleccionado	58
Figura 4.8 Training AlexNet, valores por defecto	59
Figura 4.9 Training GoogleNet, valores por defecto	60
Figura 4.10 Valores priorizando la velocidad de entrenamiento	60
Figura 4.11 Training AlexNet, valores priorizando la velocidad de entrenamiento	61
Figura 4.12 Training GoogleNet, valores priorizando la velocidad de entrenamiento	61
Figura 4.13 Valores priorizando mejores resultados	62
Figura 4.14 Training AlexNet, valores priorizando mejores resultados	62
Figura 4.15 Training GoogleNet, valores priorizando mejores resultados	63
Figura 4.16 Clasificación correcta de GoogleNet	64
Figura 4.17 Imágenes de dos frames consecutivos donde se demuestra el error del conteo	67
Figura 4.18 Variables de conteo	68
Figura 4.19 Cálculo de porcentaje	68
Figura 4.20 Error de una sola unidad	69
Figura 4.21 Error en la captura de elementos	70
Figura 4.22 Error en la identificación de las partes del coche	71
Figura 4.23 Error al reconocer motos	71
Figura 4.24 Error de reconocimiento de vehículos en los bordes	72
Figura 4.25 Tweet al obtener una afluencia de coches mayor al 70%	72
Diagrama 1. Network Analyzer	82
Diagrama 2. Training Images	83
Diagrama 3. Layers Analyzer	83
Diagrama 4. Validation Images	84
Diagrama 5. Weights	84

## **Índice de Tablas**

Tabla 2.1 - Parámetros aprendidos en el modelo AlexNet	35
Tabla 4.1 - Datos del conteo de la cámara situada en Moncloa a las 13:00	65

Tabla 4.2 - Datos de los porcentajes de la cámara situada en Moncloa a las 13:00	65
Tabla 4.3 - Datos del conteo de la cámara situada en Moncloa a las 17:00	65
Tabla 4.4 - Datos de los porcentajes de la cámara situada en Moncloa a las 17:00	66
Tabla 4.5 - Datos del conteo de la cámara situada en Villaverde a las 13:00	66
Tabla 4.6 - Datos de los porcentajes de la cámara situada en Villaverde a las 13:00	66

# 1. Introducción

## 1.1 Antecedentes

Los últimos paradigmas que se van generando a nuestro alrededor por el avance tecnológico y científico han contribuido a una evolución sobre la sociedad nunca antes vista. Todo ello nos ha encaminado hacia una automatización de la mayoría de los sistemas que nos rodean, desde la cotidianidad de cada persona hasta el proceso más complejo.

Esta evolución frenética conlleva a reflexionar hacia dónde nos dirigimos con estos nuevos escenarios tecnológicos que se presentan, y qué impacto debe tener en nuestra vida. El uso de la tecnología está determinado en gran parte por la óptica del sujeto que la utiliza y valora, y en segunda instancia en el contexto y el tiempo en el que nos encontramos. Refiriéndonos a esto último, tenemos como ejemplo la antigua idea de la civilización griega de la tecnología: un ejercicio intelectual de organización sobre un campo técnico. En un contexto más contemporáneo, estos campos pueden dividirse en dos grandes aspectos: primeramente, aquellas leyes ligadas directamente al conocimiento químico-físico y objetivo del entorno, lo comúnmente conocido como ciencia. En segundo lugar, las ligadas a la construcción de instrumentos, además del desarrollo de las necesidades de la sociedad, que serían las técnicas.

Generalmente, podemos considerar que un instrumento tecnológico es beneficioso cuando tiene efectos positivos que mejoran las condiciones de la población. Y actualmente existe una proliferación tecnológica exponencial, que cambia de manera acelerada conforme a las necesidades y a las circunstancias históricas. Como bien citó el escritor y artista canadiense Douglas Coupland, “incluso cuando te tomas unas vacaciones de la tecnología, la tecnología no se toma un descanso de ti”. Esta actualización constante de la tecnología ha afectado en muchos aspectos al ser humano. Uno de estos aspectos ha sido la urbanización y peregrinación de la población a las grandes ciudades, las cuales proporcionan y ofrecen, en términos generales, un mayor desarrollo tecnológico. Las ciudades deben innovar constantemente ante este avance, y para adaptarse deben aprender a gestionar y mantener un orden en red. Trabajar en red supone aceptar que se

debe avanzar vinculado a una estructura hilada y ordenada perfectamente y de manera responsable.

Todo este desarrollo de logística urbana nos lleva a un término actualmente muy utilizado, el de las ciudades inteligentes, ampliamente conocidas como *smart cities*. Este nuevo concepto de ciudad utiliza sabiamente las funcionalidades que ofrece la tecnología para implementar y promover un avance significativo, tanto en el desarrollo urbano, como en servicios públicos. Un ejemplo es el caso de la ciudad de Singapur (Levy y Gabriel, 2021), que utiliza una serie de sensores y un sencillo sistema de georreferenciación, en la que la red de autobuses y paradas de Singapur reporta de forma permanente su ubicación, el número de pasajeros en los vehículos y en las estaciones, al igual que la velocidad promedio, el tráfico en la vía, entre otras muchas variables, lo que posibilita mediante un sistema basado en Inteligencia Artificial, decidir de forma dinámica la frecuencia de operación, para garantizar una óptima prestación en los servicios. Esto permite afrontar el problema de la densidad y la movilidad a lo largo de su desarrollo. También se están integrando vehículos autónomos en el transporte público y el transporte mercante para hacer frente a las limitaciones de tierra y mano de obra. En este sentido evolucionan otros desarrollos inteligentes a nivel mundial (Low, 2021). De hecho, en esta referencia aparecen las diez ciudades con los mayores niveles de inteligencia, donde la ciudad española de Bilbao aparece en dicho rango.

Indagando en distintas optimizaciones que nos ofrecen las ciudades inteligentes en el ámbito de los servicios públicos, en el presente proyecto pondremos la vista en el sector del transporte, uno de los pilares de la evolución como sociedad. Hay que entender que la tecnología basada en el transporte es un sistema complejo que involucra la medición y análisis de una gran cantidad de datos, además de la investigación de los diferentes patrones que puedan surgir a raíz de estos. A partir de aquí entran en juego las diferentes herramientas que nos proporciona la tecnología, como por ejemplo *Big Data*, Internet de las Cosas (IoT, *Internet of Things*), Visión por Computador (VC). Y por supuesto, claro está, el Aprendizaje Profundo (AP), o *Deep Learning* (DL). Las técnicas, procesos y metodologías que ofrecen estas herramientas son clave para obtener una mejor respuesta a las necesidades de la población.

Sin embargo, hay que tener presente que al abordar esta problemática no todos los efectos son mensurables con el mismo patrón o tendencia, y también hay que tener en cuenta el dinamismo que conlleva este sector debido a multitud de factores. La clave está en minimizar estos riesgos utilizando las herramientas que obtenemos con el uso de las tecnologías avanzadas, mencionadas previamente.

Es por ello por lo que la idea sobre la que pivota este proyecto es ofrecer una solución conceptual mediante el uso de métodos y tecnologías basados en VC, AP e IoT. Más concretamente en el adecuado procesamiento de los datos, para su posterior uso, obtenidos a partir de imágenes procedentes de vídeos conteniendo diferentes tipos de vehículos en movimiento, que será de gran utilidad para una

mayor eficiencia del tráfico, mediante la identificación de dichos vehículos y el cómputo del tránsito de los mismos en un hipotético entorno de ciudad inteligente. Estos datos también nos proporcionan una predicción muy acertada sobre los diferentes flujos de tráfico que podríamos tener en diferentes sectores de una ciudad. En definitiva, el estudio de los resultados obtenidos en este proyecto genera una ventaja avanzada inteligente y tecnológica para la gestión eficiente del tráfico rodado en un núcleo urbano inteligente, contribuyendo a un cambio y una mejora en el modelo conceptual de la red de transporte.

## 1.2 Objetivos

El proyecto tiene como objetivo principal el desarrollo de una aplicación para detección e identificación de vehículos en movimiento mediante técnicas de VC, AP e IoT, cuya finalidad es proporcionar una solución de concepto para la mejora y desarrollo del flujo de vehículos en diferentes sectores de un ámbito urbano, controlando y automatizando los datos obtenidos. Va dirigido principalmente al concepto de las ciudades inteligentes para su contribución al desarrollo futuro.

Consecuentemente, la mejora del flujo de vehículos tiene un impacto positivo en varios aspectos dignos de destacar en entornos urbanos, con mayor énfasis en el marco de las futuras ciudades inteligentes. Uno de ellos es en la disminución de posibles atascos y fluidez del tráfico, con el resultado de que haya también menores accidentes de tráfico. También hay que tener en cuenta el aspecto ambiental de la ciudad, de forma que se obtendrán menores emisiones de elementos contaminantes como consecuencia de la mejora del flujo de vehículos. Además, la utilización pública de estos datos para uso informativo hacia la población generará un impacto positivo para los ciudadanos, mediante el uso de herramientas sociales. En el caso del presente proyecto, la red social *Twitter* será una buena aliada para proporcionar toda esa información a los usuarios.

Como consecuencia, considerando todos estos aspectos, se generan los siguientes objetivos específicos:

- Aplicar y utilizar técnicas basadas en VC para la detección del movimiento en secuencias de imágenes, exactamente métodos de flujo óptico.
- Utilizar y aplicar modelos de AP, en nuestro caso Redes Neuronales Convolucionales (RNC), concretamente *AlexNet* y *GoogleNet*.
- Integrar los procedimientos anteriores para su aplicación en cada una de las imágenes de la secuencia.
- Entrenar los modelos de redes bajo distintos parámetros de aprendizaje.
- Etiquetar los diferentes vehículos en movimiento obtenidos en cada imagen y cómputo del flujo de vehículos en la secuencia de imágenes procedentes del vídeo.

- Aplicar técnicas de comunicación y almacenamiento de datos en la nube, en este caso concreto mediante la utilización de la herramienta *ThingSpeak*.
- Desarrollar una interfaz adecuada y estructurada, que sea útil para su utilización por parte del usuario de la aplicación.
- Publicación de los datos obtenidos mediante la utilización de herramientas sociales como *Twitter* para el uso público.

## 1.3 Motivación

Con lo indicado previamente, podemos afirmar que la motivación principal para la realización de este proyecto es la investigación y el desarrollo de tecnologías vanguardistas relacionadas con la inteligencia artificial, como es el caso de la aplicación de técnicas de AP, VC o IoT.

Por otra parte, y ascendiendo un nivel más, cabe mencionar que el mundo tecnológico evoluciona y crea la necesidad de avanzar con él en paralelo. Un ejemplo son los datos. Cada vez se almacenan más datos de todo tipo (imágenes, vídeos, etc.), y es necesario crear tecnologías que sean capaces de procesar todos estos datos.

Aquí es donde entra el AP, como elemento sustancial y diferenciador de la propuesta formulada en este proyecto, para aprovechar toda la información subyacente en los datos a partir de las tecnologías que proporciona, para aplicarlas a casos y usos reales que puedan ser de utilidad.

Y es por eso por lo que este proyecto puede ser de gran importancia y tener el potencial suficiente para acercarnos un paso más hacia el concepto de las ciudades inteligentes. Aplicado a gran escala, y como se ha mencionado anteriormente, el presente proyecto podría servir, al menos conceptualmente, para analizar de forma precisa los flujos de tráfico para, por ejemplo, prever atascos y facilitar a los ciudadanos información relevante.

## 1.4 Plan de trabajo

Para la realización de este trabajo se han llevado a cabo las siguientes tareas de manera secuencial según este orden:

- **Identificación de las redes neuronales convolucionales**

Para tratar el aprendizaje a partir de los datos proporcionados tuvimos que buscar entre los muchos modelos de RNC existentes y así determinar cuál sería el más adecuado y eficiente para poder resolver los problemas planteados en este proyecto.

Al final nos decantamos por dos modelos pre-entrenados, *AlexNet* y *GoogleNet*. Estos dos tipos de redes neuronales convolucionales trabajan con millones de imágenes y su aprendizaje previo era bastante útil a la hora de realizar el proyecto, puesto que pudimos aprovechar el modelo original con sus pesos aprendidos que nos proporcionaban y así adaptarlo a nuestro trabajo de identificar y contar vehículos. Es lo que se conoce como transferencia de aprendizaje.

- **Análisis del flujo óptico**

Para poder analizar los vehículos en movimiento, a partir de secuencias de imágenes, era necesario aprender, comprender e implementar los métodos y funcionalidades adquiridas.

- **Elaboración de la interfaz y desarrollo de la arquitectura**

Previamente al diseño de la interfaz se analizó y estructuró la aplicación, contemplando los posibles casos de uso y diagramas necesarios para cubrir todas las funcionalidades exigidas por el usuario.

Después, a través de la herramienta de *Matlab App Designer*, se procedió a desarrollar una interfaz gráfica intuitiva, que consta de diferentes vistas y desde la que el usuario podrá manejar de manera sencilla la aplicación. Desde ella será posible elegir si se quiere entrenar o clasificar la red, seleccionar qué red neuronal se va a utilizar, redimensionar nuevas imágenes para el entrenamiento, controlar los videos elegidos, obtener los datos almacenados en la nube a través de *Thingspeak* y poder obtener la información previamente seleccionada mediante *tweets* de *Twitter*.

Una vez terminada la parte del diseño, se procedió a la codificación y a la fase de pruebas (*testing*).

- **Procesamiento de imágenes y redimensionamiento para el entrenamiento**

Teniendo en cuenta que los modelos de redes *AlexNet* y *GoogleNet*, que utilizamos, necesitan unas imágenes con unas características determinadas, era necesario poder redimensionar las imágenes utilizadas en el entrenamiento para poder evitar errores y así garantizar el buen funcionamiento del entrenamiento y la validación.

- **Configuración del proceso de entrenamiento de AlexNet y GoogleNet**

La configuración y el control de los parámetros de los diferentes modelos de redes es una tarea necesaria y bastante importante para llevar a cabo la detección de los vehículos.

Establecer los parámetros de aprendizaje y poder modificarlos es crucial y determina la precisión con la que se ejecuta el entrenamiento. Y la finalidad es conseguir la mayor precisión posible durante el proceso.

- **Detección y etiquetado de imágenes en movimiento para la clasificación**

Durante el proceso de clasificación, el modelo de red trabaja a partir de un video, que se reproduce *frame a frame*, con el objetivo de procesar las imágenes e identificar, en este caso, los vehículos en el flujo de tráfico.

A lo largo de esta sucesión de imágenes se detectan las diferentes regiones en movimiento, que se segmentan y extraen para la posterior clasificación y localización de los vehículos en plena circulación.

- **Sistema de conteo de los vehículos**

Se lleva a cabo un sistema de conteo a través de porcentajes con el objetivo de determinar qué tipos de vehículos tienen más peso en el tráfico, es decir, circulan con mayor frecuencia. De esta forma, se calcula cuántos vehículos en total pasaron por las regiones de la imagen, además de contabilizar también los que son de un tipo determinado.

Después se procede a cuantificar el tanto por ciento de cada tipo, evitando así el error de las repeticiones en el conteo de los vehículos.

- **Transferencia de datos a ThingSpeak**

Tras obtener los datos referidos al porcentaje y los tipos de vehículos que se han detectado en el tráfico, éstos se envían al canal correspondiente creado en *ThingSpeak*. Esta plataforma en la nube permite almacenar toda la información que queramos sobre los vehículos, manteniendo los datos disponibles para consultarlos o analizarlos de manera pública.

- **Comunicación cliente-servidor**

Engloba todas las operaciones de lectura y escritura realizadas en el proyecto.

En el caso de lectura, se utilizará para consultar y leer los campos proporcionados por *ThingSpeak* y reunirlos en una tabla para mostrarla en la aplicación.

En el caso de escritura, guardará los datos generados del porcentaje del tipo de vehículo y el conteo llevado a cabo.

- **Función del lado del servidor**

*ThingSpeak* ofrece varias Apps y proporciona diferentes funcionalidades interesantes. En este caso, utilizamos *ThingTweet* para poder enlazar una cuenta de *Twitter* y *React* y así lanzar varios avisos en forma de *tweet* según lo que el usuario quiera saber sobre el flujo de tráfico o lo que haya contabilizado la clasificación de vehículos.

## 1.5 Contribuciones personales

### 1.5.1 Fernando Bellot Rodríguez

Labores de investigación:

- Investigación y familiarización con la aplicación de Matlab y su lenguaje, mediante vídeos y guías en internet.
- Familiarización y comprensión del código con los ejemplos tutoriales para abordar este proyecto.
- Comprensión del funcionamiento de las redes neuronales convolucionales *AlexNet* y *GoogleNet*.
- Investigación de la herramienta de Matlab *App Designer*, centrada en el desarrollo de las interfaces del proyecto.
- Elección del modelo de gestión de proyecto a seguir y el tipo de metodología más adecuada para aplicar a nuestro proyecto, en compañía con el resto de los compañeros del equipo.
- Comprensión e investigación de la plataforma *ThingSpeak* y sus canales para almacenar datos e información.
- Investigación del funcionamiento de *Twitter* y de *ThingTweet*, una herramienta de *ThingSpeak* para la publicación de *tweets* de la aplicación. *ThingSpeak* es específica para *Internet of Things*.

- Investigación de la aplicación de Matlab para el dispositivo móvil, con el objetivo de visualizar los datos a través del teléfono.

#### Labores de desarrollo:

- Modificación en la estructura de los elementos de diseño aplicados en las vistas de la aplicación en conjunto con el resto de los miembros del equipo.
- Diseño inicial de las interfaces de la aplicación, conjuntamente con el resto de los compañeros del equipo.
- Implementación de los límites y los parámetros predeterminados del entrenamiento de las redes neuronales.
- Integración del código fuente para la clasificación de vehículos en colaboración con mi compañera Esther.
- Implementación del algoritmo de conteo en colaboración con el resto del equipo.
- Creación y configuración de la cuenta de *Twitter* del proyecto y de la herramienta *ThingTweet*.
- Testeo de las funcionalidades de la aplicación y el correcto funcionamiento de los botones.
- Implementación del código para la publicación automática de *tweets* después de cada clasificación.
- Grabación de nuevos vídeos para su posterior clasificación.
- Realización de labores de entrenamiento y clasificación para la realización de pruebas.

#### Labores de gestión:

- Organización de las imágenes utilizadas en el entrenamiento y de los videos disponibles y grabados en Google Drive.
- Monitorización del sistema de testeo.
- Revisión de errores en las diferentes versiones de la memoria.

- Elección y gestión de las localizaciones donde grabar los vídeos.

Labores de documentación de la memoria:

- Realización de la estructura del índice en colaboración con el resto del equipo y la supervisión del tutor.
- Realización del apartado 1.3, dónde se exponen las motivaciones del proyecto.
- Realización del apartado 1.7, donde se traducen al inglés los apartados 1.1, 1.2 y 1.4.
- Realización del apartado 3.1, donde se expone una visión general de la arquitectura de la aplicación y las capas, en colaboración con el resto del equipo.
- Realización del apartado 4.3, dónde se explica el funcionamiento del sistema de detección de vehículos utilizado.
- Realización del apartado 4.4, dónde se exponen los resultados de clasificación obtenidos después de varias pruebas y su comparación con los datos reales.
- Realización del apartado 5 junto al resto del equipo, donde se exponen las conclusiones y una serie de trabajos futuros que se podrían realizar para la mejora de la aplicación.
- Realización del apartado 6, donde se traduce al inglés el apartado 5.
- Realización del resumen general de la memoria en colaboración con el resto del equipo.
- Desarrollo de los Anexos 1 (en colaboración con el resto del equipo) y 3.
- Corrección general de cada uno de los apartados de la memoria en colaboración con el resto del equipo.

### **1.5.2 Esther Peñas Martínez**

Labores de investigación:

- Estudio para la comprensión de las redes neuronales convolucionales *AlexNet* y *GoogleNet*, que son los modelos utilizados en este trabajo.
- Familiarización y estudio del código guía, que sirve de ejemplo.
- Investigación sobre el lenguaje Matlab y su aplicación, por tratarse de un nuevo lenguaje no estudiado previamente.
- Búsqueda de información sobre el funcionamiento de la herramienta de Matlab, *App Designer*, para el desarrollo de las interfaces.
- Selección de videos a través de la plataforma de *Youtube* para gestionar los elementos del diseño utilizados en la aplicación.
- Elección del modelo de gestión de proyecto a seguir y el tipo de metodología más adecuada para aplicar a nuestro proyecto, en compañía con el resto de los compañeros del equipo.
- Búsqueda de una plataforma para la obtención de los iconos amigables y vistosos para usarlos en la interfaz de la aplicación. En este caso, *Flaticon*.
- Búsqueda de una aplicación para realizar el seguimiento del trabajo, tratando de obtener una organización adecuada para la repartición y el desarrollo de las tareas. En este caso, *Trello*.
- Investigación del funcionamiento de *ThingSpeak* y la creación de canales en la plataforma, especialmente diseñada para *Internet of Things*.

Labores de desarrollo:

- Implementación y diseño de la interfaz del menú principal utilizando *App Designer*.
- Implementación y diseño inicial (prototipo) de las diferentes ventanas de la aplicación utilizando la herramienta *App Designer*.
- Modificación de la estructura de los elementos de diseño aplicados en las vistas de la aplicación, conjuntamente con el resto del equipo.
- Integración del código fuente para el entrenamiento de *AlexNet* en colaboración con mi compañero Alejandro.
- Integración del código para la redimensión de imágenes, tanto para *AlexNet* como para *GoogleNet* con mi compañero Alejandro.

- Integración del código fuente para la clasificación de vehículos en colaboración con mi compañero Fernando.
- Implementación del vídeo de clasificación en colaboración con mi compañero Alejandro.
- Manejo de los botones y su funcionalidad.
- Implementación del algoritmo de conteo en colaboración con el resto del equipo.
- Realización de labores de clasificación para el análisis y la captura de errores.
- Implementación del código de consultas, su enlace con *ThingSpeak* y *ThingTweet* en colaboración con mi compañero Alejandro.
- Testeo de la aplicación para la búsqueda de errores.

Labores de gestión:

- Repartición de las diversas tareas a realizar en el proyecto.
- Creación de un repositorio en Github y organización de carpetas del proyecto.
- Creación de un tablero Kanban mediante la herramienta Trello.
- Creación de la unidad compartida de Google Drive para el almacenamiento de la información y de las versiones de la memoria y del proyecto.
- Gestión de Trello tras cada reunión para que el tablero Kanban se mantuviera actualizado.
- Organización de las reuniones del proyecto y revisiones.
- Establecer fechas de entrega para las tareas.
- Apuntar los fallos para consultarlos en las reuniones establecidas con nuestro tutor.
- Revisión de errores en las diferentes versiones de la memoria antes de realizar su envío al tutor en compañía con el resto del equipo.

Labores de documentación en la memoria:

- Realización de la estructura del índice en colaboración con el resto del equipo y la supervisión del tutor.
- Realización del apartado 1.4, dónde se expone el plan de trabajo y las tareas realizadas a lo largo del proyecto.
- Realización del apartado 1.6, con la estructura de la memoria.
- Realización del apartado 3.1, que contiene una visión general de la arquitectura de la aplicación y las capas, en colaboración con el resto del equipo.
- Realización del apartado 3.2, dónde se expone la metodología y gestión utilizadas en el proyecto.
- Realización del apartado 4.1 completo, dónde se detallan las funcionalidades de la interfaz de la aplicación.
- Realización del apartado 4.5, dónde se explica el algoritmo de conteo utilizado.
- Realización del apartado 4.6, dónde se exponen los errores más frecuentes de la clasificación de los vehículos.
- Realización del apartado 5 junto al resto del equipo, donde se explican las conclusiones derivadas del proyecto y se exponen una serie de trabajos futuros como acciones de mejora de la aplicación.
- Realización del resumen general de la memoria en colaboración con el resto del equipo.
- Desarrollo de los Anexos 1 (en colaboración con el resto del equipo), 2 y 4.
- Corrección general de cada uno de los apartados de la memoria en colaboración con el resto del equipo.

### **1.5.3 Alejandro Ruiz Valero**

Labores de investigación:

- Familiarización con los ejemplos de código que sirven de inicio para el desarrollo del proyecto, comparándolo con otros modelos y ejemplos para comprender su funcionamiento.
- Investigación y comprensión de cómo aplicar el lenguaje Matlab, mediante el uso de vídeos y la realización de un curso sobre ello, con la consiguiente obtención de un título oficial sobre el tema.
- Investigar la implementación de la interfaz, usando *App Designer*, que MATLAB pone a su disposición, y que se encarga del diseño y desarrollo de la interfaz.
- Elección del modelo de gestión de proyecto a seguir y el tipo de metodología más adecuada para aplicar al proyecto, en compañía con el resto de los compañeros del equipo.
- Trabajo de investigación sobre la plataforma *ThingSpeak* y sus componentes, comprendiendo cómo anclar los datos obtenidos en el proyecto con la plataforma, teniendo en cuenta los parámetros del paradigma *Internet of Things* que es el uso final de esta plataforma.
- Búsqueda de una plataforma para realizar el almacenamiento del historial de versiones del proyecto, con el objetivo de organizar la evolución y desarrollo tanto del proyecto como de la memoria. En este caso, *Git* y *Google Drive* respectivamente.

#### Labores de desarrollo:

- Modificación en la estructura de los elementos de diseño aplicados en las vistas de la aplicación en conjunto con el resto del equipo.
- Integración del código fuente para el entrenamiento del modelo *GoogleNet* en colaboración con mi compañera Esther.
- Desarrollo, implementación y solución de errores del código necesario para la redimensión de imágenes con mi compañera Esther.
- Implementación del vídeo de clasificación en colaboración con mi compañera Esther.
- Implementación de la funcionalidad y la lógica de los cambios de vistas mediante la creación de botones que dirijan al usuario a la pantalla deseada de la aplicación.

- Implementación y solución de errores en los diferentes botones utilizados para configurar la detención del video durante la clasificación de las redes.
- Configuración de la organización de los resultados obtenidos en las clasificaciones, ordenándose cronológicamente.
- Implementación del algoritmo de conteo en colaboración con el resto del equipo.
- Implementación del código de consultas, su enlace con *ThingSpeak* y *ThingTweet* en colaboración con mi compañera Esther.
- Solución de posibles errores a la hora de enviar los resultados de la clasificación a la plataforma de *ThingSpeak*.
- Realización de labores de entrenamiento y capturas para el análisis del entrenamiento de las redes.

Labores de gestión:

- Comunicar al tutor los avances que progresivamente se iban consiguiendo en el proyecto por medio de correo electrónico, identificando las tareas resueltas, y la gestión de la respuesta por parte del tutor tras su revisión y conformidad.
- Gestión e identificación de las diferentes tareas en Trello para la realización del proyecto.
- Identificar las dudas para consultarlas en las reuniones establecidas con el tutor.
- Revisión de errores en las diferentes versiones de la memoria antes de realizar su envío al tutor en compañía con el resto del equipo.

Labores de documentación en la memoria:

- Realización de la estructura del índice en colaboración con el resto del equipo y la supervisión del tutor.
- Realización del apartado 1.1, dónde se exponen los antecedentes del proyecto.
- Realización del apartado 1.2, en el que se mencionan los objetivos principales.

- Realización del apartado 3.1, donde se expone una visión general de la arquitectura de la aplicación y las capas, en colaboración con el resto del equipo.
- Realización del apartado 3.3, donde se mencionan las herramientas empleadas.
- Realización del apartado 4.2 completo, donde se plasma el entrenamiento de las redes.
- Realización del apartado 4.7, donde se expone la integración de los datos y resultados en *ThingSpeak*.
- Realización del apartado 5 junto al resto del equipo, donde se exponen las conclusiones y una serie de trabajos futuros que se podrían incluir en la aplicación para su mejora.
- Realización del resumen general de la memoria en colaboración con el resto del equipo.
- Desarrollo del Anexo 1 en colaboración con el resto del equipo.
- Corrección general de cada uno de los apartados de la memoria en colaboración con el resto del equipo.

## 1.6 Estructura de la memoria

La estructura de la memoria consta de la siguiente disposición de secciones:

- **Introducción**

En este punto se especifican los antecedentes y algunos ejemplos sobre las posibles situaciones del panorama actual que puedan asimilarse a los resultados obtenidos con el trabajo. Asimismo, se enumeran los objetivos del proyecto, las motivaciones para realizarlo, el plan seguido para elaborarlo y las contribuciones personales de cada miembro que conforma el equipo.

- **Métodos conceptuales y técnicos aplicados**

En esta sección se engloban todos los aspectos teóricos y las técnicas utilizadas para realizar el cálculo del flujo óptico y la detención. Además de la explicación de las redes neuronales convolucionales empleadas para procesar la información del tráfico de los vehículos.

- **Diseño y análisis desarrollado**

Descripción detallada de aquellos aspectos relacionados con la Ingeniería del Software, tales como la arquitectura de la aplicación, la gestión del proyecto, la metodología de trabajo utilizada y las herramientas empleadas para el desarrollo de la aplicación y la organización del trabajo.

- **Resultados obtenidos**

Tras realizar las pruebas, en este punto se exponen los resultados. Analizando y explicando la precisión de los datos obtenidos tras el entrenamiento de las redes, su tiempo de ejecución, la clasificación llevada a cabo y la subida de datos del *ThingSpeak*.

- **Conclusiones y trabajo futuro**

Se expone un resumen de las conclusiones obtenidas a lo largo de la ejecución del trabajo y se plantean mejoras, que nos servirán para perfeccionar los resultados no satisfactorios o para elaborar opciones nuevas de cara al futuro.

- **Anexos**

Se incluyen varios anexos donde se exponen:

1. Los diagramas del entrenamiento de las redes.
2. Las licencias de las imágenes y los iconos que han sido utilizados en el desarrollo de la aplicación y en la memoria.
3. Un manual de usuario para la instalación y la ejecución de la aplicación. Además del método de visualización del código.

## 1.7 Introduction

### 1.7.1 Preliminary

The latest paradigms that are being generated around us due to technological and scientific progress have contributed to an evolution of society never seen before. All this has led us towards an automation of most of the systems that surround us, from the daily life of each person to the most complex process.

This frenetic evolution has led us to think about where we are headed with these new technological scenarios that are presented to us, and what impact it should

have on our lives. The use of technology is largely determined by the point of view of the subject who uses and values it, and secondly by the context and time in which we are in. Referring to this, we have as an example the Greek civilization's ancient idea of technology: an intellectual exercise of organization in a technical field. In a more contemporary context, these fields can be divided into two main aspects: first, those techniques directly linked to physico-chemical and objective knowledge of the environment, what is commonly known as science. Secondly, those linked to the construction of instruments, in addition to the development of society's needs, technologies.

Generally, we can consider that a technological instrument is beneficial when it has positive effects that improve the conditions of the population. And today we have an exponential technology, which changes rapidly according to the needs and historical circumstances. As Canadian writer and artist Douglas Coupland rightly quoted, "even when you take a holiday from technology, technology doesn't take a break from you". The constantly updating technology has affected human beings in many aspects. One of these aspects has been the urbanization and pilgrimage of the population to the big cities, which provide and offer, in general terms, greater technological development. Cities must constantly innovate in the face of this progress, and to adapt they must learn to manage and maintain network order. Working in a network means accepting that progress must be made linked to a perfectly responsible well-organized structure.

All this urban logistics development leads us to a term that is currently widely used: smart cities. This new concept of city wisely uses the functionalities offered by technology to implement and promote significant progress, both in urban development and in public services. An example is the case of the city of Singapore (Levy and Gabriel, 2021), which uses a series of sensors and a simple georeferencing system, in which the network of buses and bus stops in Singapore permanently reports their location, the number of passengers in the vehicles and in the stations, as well as the average speed, the traffic on the road, among many variables, which is enabled through a system based on artificial intelligence to dynamically decide the frequency of operation, to guarantee optimal performance in services. This allows dealing with the problem of density and mobility throughout its development. Autonomous vehicles are also being integrated into public transport and merchant transport to deal with land and labor constraints. In this way, other intelligent developments are evolving worldwide (Low, 2021). In fact, in this reference appear the ten cities with the highest levels of intelligence, where the Spanish city of Bilbao appears in that range.

From all the optimizations that smart cities offer us in the field of public services, in this project we will look at the transport sector, one of the pillars of evolution as a society. We must understand that transport-based technology is a complex system that involves the measurement and analysis of a large amount of data, in addition to the analysis of the different patterns that may arise as a result of these. From here,

the different tools provided by technology come into play, such as Big Data, Internet of Things (IoT), Computer Vision (CV) and of course, Deep Learning. The techniques, processes and methodologies offered by these tools are key to obtaining a better response to the needs of the population.

However, we must keep in mind that when approaching this problem, not all the effects are measurable with the same pattern or trend, and the dynamism that this sector entails due to a multitude of factors must also be considered. The key is to minimize these risks using the tools that we obtain with the use of the advanced technologies previously mentioned.

That is why the idea on which this project revolves is to offer a conceptual solution using methods and technologies based on CV, DL and IoT. More specifically in the proper processing of data for later use obtained from images from videos containing different types of vehicles in motion, which will be very useful for greater traffic efficiency, through the identification of said vehicles and their transit calculation in a hypothetical smart city environment. These data also provide us with a very accurate prediction about the different traffic flows that we could have in different sectors of a city. In conclusion, the study of the results obtained in this project generates an advanced intelligent and technological advantage for the efficient management of road traffic in an intelligent urban area, contributing to a change and an improvement in the conceptual model of the transport network.

### **1.7.2 Objectives**

The project's main objective is the development of an application for the detection and identification of moving vehicles using CV, DL and IoT techniques, whose purpose is to provide a concept solution for the improvement and development of the vehicle flow in different sectors of an urban area, controlling and automating the data obtained. It is mainly aimed at the concept of smart cities for their contribution to future development.

Consequently, the improvement in the vehicle flow has a positive impact on several aspects worth highlighting in urban environments, with greater emphasis in the framework of future smart cities. One of them is in the reduction of possible traffic jams, leading to fewer traffic accidents too. The environmental aspect of the city must also be considered, so that lower emissions of polluting elements will be obtained as a result of the improvement in the flow of vehicles. In addition, the public use of these data towards the population for informative purposes will generate a positive impact for citizens, by using social tools. In the case of this project, the social network Twitter will be a good ally to provide all this information to users.

Consequently, considering all these aspects, the following specific objectives are generated:

- Apply and use techniques based on Computer Vision for motion detection in image sequences, exactly optical flow methods.
- Use and apply DL models, in our case CNN, specifically AlexNet and GoogleNet.
- Integrate the previous procedures for their application in each one of the image sequences.
- Train the network models under different learning parameters.
- Label the different vehicles in motion obtained in each image and calculate the vehicle flow in the image sequences from the video.
- Apply communication and data storage techniques in the cloud, in this specific case by using the ThingSpeak tool.
- Development of an appropriate and structured interface that is useful for the application's user.
- Publication of the data obtained through the use of social tools such as Twitter for public use.

### **1.7.3 Work plan**

To carry out this work, the following tasks have been done sequentially in this order:

- **Identification of convolutional neural networks:**

To deal with learning from the data provided, we had to search among the many neural network models that can be found and determine which one would be the most appropriate and efficient to solve the problems raised in this project.

In the end we opted for two pre-trained models, AlexNet and GoogleNet. These two types of convolutional neural networks work with millions of images and their previous learning was quite useful when carrying out the project, since we were able to take advantage of the original model that they provided us and adapt it to our work of identifying and counting vehicles. This is known as transfer learning.

- **Optical flow analysis:**

In order to analyze moving vehicles from image sequences, it was necessary to learn, understand and implement the methods and functionalities acquired.

- **Interface and architecture development:**

Prior to the design of the interface, the application was analyzed and structured, considering the possible use cases and diagrams necessary to cover all the functionalities required by the user.

Then, through the Matlab App Designer tool, an intuitive graphical interface was developed, which consists of different views and from which the user can easily manage the application. It will be possible to choose whether to train or classify the network, select which neural network is going to be used, resize new images for training, control the chosen videos, obtain the data stored in the cloud through ThingSpeak and be able to obtain the information previously selected from a tweet on Twitter.

Once the design part was finished, the coding and testing phase started.

- **Image processing and resizing for training:**

Bearing in mind that the AlexNet and GoogleNet network models that we use require images with certain characteristics, it was necessary to be able to resize the images used in the training in order to avoid errors and guarantee the proper functioning of the training and validation.

- **AlexNet and GoogleNet training process configuration:**

Configuring and controlling the parameters of the different network models is a necessary and quite important task to carry out the vehicle detection. Setting the learning parameters and being able to modify them determines how accurately the training runs. And the purpose is to achieve the greatest possible precision during the process.

- **Detection and labeling of moving images for classification:**

During the classification process, the network model works from a video, which is played frame by frame, with the aim of processing the images and identifying, in this case, the traffic vehicles.

Throughout this succession, the different moving regions are detected, which are segmented and extracted for the subsequent classification and location of vehicles driving around.

- **Vehicle counting system:**

A counting system through percentages is carried out in order to know which types of vehicles have more weight in traffic.

It calculates how many vehicles in total passed through the regions of the image, in addition to also counting those of a certain type.

Then the percentage of each type is quantified, thus avoiding the error of the repetitions in the counting of the vehicles.

- **Data transfer to ThingSpeak:**

After obtaining the percentages and types of vehicles that have been detected in the traffic, these are sent to the corresponding channel created in ThingSpeak. This cloud platform allows us to store all the information we want about the vehicles, keeping the data available for public consultation or analysis.

- **Client-server communication:**

Encapsulates all read and write operations performed on the project. In the case of reading, it will be used to query and read the fields provided by ThingSpeak and collect them in a table to display in the application. In the case of writing, it will save the generated data of the percentage of the types of vehicles and the count carried out.

- **Server side feature:**

ThingSpeak offers numerous apps and provides different interesting features. In this case, we use ThingTweet to be able to link Twitter and React accounts and launch notifications in the form of a tweet depending on what the user wants to know about the traffic flow or the vehicle classification count.

## 2. Métodos conceptuales y técnicas aplicadas

La metodología aplicada consiste en la detección de objetos en movimiento a través de las distintas escenas procedentes de los vídeos analizados. Para ello se aplican técnicas de VC para detección del movimiento, concretamente basadas en el flujo óptico. Tras lo cual se aíslan las regiones en movimiento, también aplicando métodos de VC. Estas regiones son las que se procesan mediante las RNC para identificar la naturaleza de los vehículos que han originado el movimiento. Por ello, en este capítulo se analizan ambas técnicas, en primer lugar las concernientes al flujo óptico y en segundo lugar y en extenso, las basadas en las RNC, que constituyen el núcleo central de la aplicación desarrollada.

### 2.1 Cálculo del flujo óptico

El movimiento en secuencias de imágenes y por tanto con intervención de la variable temporal se puede detectar con técnicas avanzadas de visión por computador, más concretamente mediante lo que se conoce como flujo óptico. Se basa en el cómputo de las variaciones de intensidad que se producen entre las distintas partes u objetos presentes en sendas imágenes de una determinada

secuencia temporal. Dichas imágenes de la secuencia se conocen como *frames* y el conjunto de todas ellas constituye el vídeo completo. Dos *frames* consecutivos están separados por un tiempo que se identifica como  $dt$  (Pajares y Cruz, 2007; Farneback, 2003). El cómputo del flujo óptico proporciona la posición, dirección y velocidad de un objeto en movimiento por comparación entre los dos *frames* analizados. El procedimiento que se describe a continuación calcula exactamente el flujo a nivel de píxel, esto es, para todos y cada uno de los píxeles involucrados.

La obtención del gradiente es un método aproximado que se utiliza para calcular el flujo óptico. Se basa en identificar los cambios del nivel de intensidad entre dos imágenes consecutivas. Así pues, la obtención de la intensidad de un píxel en la posición  $(x,y)$  en un instante de tiempo  $(t)$  se define a través de la función  $f(x,y,t)$ . Debido al dinamismo del movimiento en el tiempo, un píxel dado en una imagen se posicionará en una ubicación próxima en la siguiente secuencia, dado que  $dt$  es pequeña también lo es el desplazamiento  $(dx,dy)$ . Estas ideas son las que se reflejan en la ecuación (2.1).

$$f(x + dx, y + dy, t + dt) = f(x, y, t) \quad (2.1)$$

Expandiendo la expresión (2.1) mediante un polinomio de Taylor de primer orden, se obtiene el resultado mostrado en la ecuación (2.2).

$$\begin{aligned} f(x + dx, y + dy, t + dt) &= f(x, y, t) + \frac{\delta f}{\delta x} dx + \frac{\delta f}{\delta y} dy + \frac{\delta f}{\delta t} dt + O(\delta^2) \\ &= f(x, y, t) + f_x dx + f_y dy + f_t dt + O(\delta^2) \end{aligned} \quad (2.2)$$

Conviene señalar que en la anterior aproximación las derivadas pueden interpretarse como las diferencias existentes entre las intensidades de los píxeles adyacentes.

Despreciando el término de orden superior, se obtiene una igualdad aproximada, que da lugar al resultado mostrado en la ecuación (2.3).

$$-f_t = f_x \frac{dx}{dt} + f_y \frac{dy}{dt} \quad (2.3)$$

Así se llega a la ecuación que expresa cómo la diferencia de intensidad  $f$ , en una misma posición e instante de tiempo se debe a la diferencia de intensidad espacial, lo cual se debe a la existencia y magnitud del movimiento.

Considerando que la intensidad de las imágenes de la secuencia se mantiene constante debido al pequeño tiempo transcurrido ( $\delta f / \delta t = 0$ ) y que el vector gradiente en la ecuación (2.3) es constante, se llega a la ecuación (2.4).

$$\left[ \frac{\delta^2 f}{\delta x^2}, \frac{\delta^2 f}{\delta x \delta y}; \frac{\delta^2 f}{\delta x \delta y}, \frac{\delta^2 f}{\delta y^2} \right] [u \ v] = -\frac{\delta(\nabla f)}{\delta t} \quad (2.4)$$

Por otra parte, a la hora de llevar a cabo este cómputo se considera un entorno de vecindad  $\Omega$  para todos los píxeles alrededor de uno dado, y que van a intervenir en el cómputo, llegando así a la ecuación (2.5).

$$\left[ \sum_{\Omega} f_x^2 \sum_{\Omega} f_x f_y \sum_{\Omega} f_x f_y \sum_{\Omega} f_y^2 \right] [u \ v] = - \left[ \sum_{\Omega} f_t f_x \sum_{\Omega} f_t f_y \right] = A V = b \quad (2.5)$$

Mediante la técnica de mínimos cuadrados aplicada a la ecuación (2.5) se resuelve el sistema de ecuaciones involucrado, llegando a la solución dada por la ecuación (2.6).

$$V = (A^T A)^{-1} (A^T b) \quad (2.6)$$

Esta ecuación determina ya el flujo óptico, que es un vector expresado como  $V(u,v)$  para todo píxel de una imagen, cuyas componentes horizontal y vertical son  $u$  y  $v$ , respectivamente.

## 2.2 Detección de regiones en movimiento

Una vez obtenido el flujo óptico  $V(u,v)$  en todos los puntos de una imagen, la agrupación de los correspondientes vectores en una zona de la imagen, esto es del *frame*, permite determinar una región sobre la que se ha producido un movimiento. En el caso de este trabajo, vehículos en movimiento.

Se ha mencionado previamente, que el flujo óptico proporciona un vector, y por tanto con su correspondiente módulo o magnitud y también dirección. La magnitud del flujo óptico se utiliza para calcular el valor medio de la magnitud del movimiento en una región dada y por ende la desviación estándar en relación con dicha magnitud para la misma región. El valor promediado y la desviación estándar sobre una determinada región se utiliza para obtener un valor de forma que, si el mismo supera un determinado umbral, fijado de antemano, la región correspondiente se identifica como de movimiento, procediendo en tal caso a generar una imagen binaria de forma que las regiones en movimiento se expresan en blanco y el resto en negro (Pajares y Cruz, 2007).

El siguiente proceso es lo que se conoce como etiquetado de componentes conexas, de forma que a cada región supuestamente de movimiento se le asigna una etiqueta. El proceso de etiquetado se lleva a cabo mediante el conocido como algoritmo de Haralick y Shapiro (1992), que consta de dos fases y utiliza una tabla de equivalencias. En la primera fase se recorre la imagen binaria, considerando en primer lugar todas las filas de arriba hacia abajo y luego recorriendo los píxeles de izquierda a derecha. Cuando un píxel no posee etiqueta, se evalúan los ocho píxeles adyacentes para determinar si alguno de ellos está ya etiquetado y puede heredar esa etiqueta.

Se distinguen varios casos a la hora de plantear el algoritmo:

- a) Si ningún píxel adyacente posee etiqueta, entonces se puede asignar a éste una etiqueta nueva, además, se inserta en la tabla de equivalencias un elemento tipo par, cuya clave y valor, es el valor de la etiqueta asignada.
- b) Si solamente uno de los píxeles adyacentes posee etiqueta, éste la propaga inmediatamente al píxel actual.
- c) Si más de un píxel adyacente puede propagar el valor de su etiqueta al píxel actual, entonces éste toma el valor de la etiqueta menor y se realiza un cambio en la tabla de equivalencias. Este proceso consiste en actualizar los pares que tienen como clave las etiquetas mayores que se intentaron propagar, los valores de estos pares se cambian por el valor de la etiqueta menor y, así, se indica que son equivalentes.

En la segunda fase se vuelve a realizar un recorrido de la misma forma que en la primera fase, si bien con el objetivo de cambiar el valor de cada píxel por el valor de su etiqueta equivalente, mediante la consulta de la tabla de equivalencias mencionada.

Una vez finaliza el algoritmo, y por tanto con todas las regiones donde se ha producido movimiento etiquetadas, se obtienen una serie de propiedades de cada región. Dentro de estas propiedades, la más relevante es la obtención de la ubicación y el tamaño de las áreas conexas, etiquetadas y delimitadas por un rectángulo que se conoce como *Bounding Box*. Estos rectángulos son los que permiten extraer la parte de la imagen contenida en ellos, realizando lo que se conoce como recorte. Pues bien, es precisamente este recorte el que se proporciona como entrada a la correspondiente RNC para identificar las regiones en movimiento y en el caso que nos ocupa los distintos tipos de vehículos.

Otra propiedad interesante asociada con cada región es la de su centroide de las áreas que las definen. El centroide, que viene a ser el centro geométrico de la región, permite identificar cada región como única, y por tanto cada vehículo de cara al cómputo final del tráfico existente en una determinada vía de circulación.

## 2.3 Redes Neuronales Convolucionales

Una RNC es un tipo específico de red neuronal, cuyo fundamento base son las conocidas capas de convolución, que dan pie a su nombre (Pajares y col, 2021). Cualquier modelo de este tipo consta de dos fases: aprendizaje y clasificación. Durante la fase de entrenamiento se ajustan, y en esto consiste el aprendizaje, los denominados pesos del modelo en lo que se conoce como núcleos de convolución. Los modelos así ajustados son los que se utilizan para la clasificación de las diferentes imágenes.

En este trabajo se han utilizado dos modelos de RNC, concretamente *AlexNet* y *GoogleNet*, que se describen a continuación. En cualquiera de estos modelos se realizan diferentes operaciones, con definición de una serie de parámetros, que constituyen la clave del proceso. Son las que se describen a continuación.

### Operaciones aplicadas en las RNC

#### a) Convolución

La convolución es una operación que se define como sigue para el procesamiento de imágenes, que como se sabe son estructuras bidimensionales 2-D,

$$C(i, j) = (K * I)(i, j) = \sum_m \sum_n I(i + m, j + n)K(m, n) \quad (2.7)$$

El resultado de la convolución es  $C(i, j)$  para un determinado píxel  $(i, j)$  cuando la entrada es una imagen  $I$  o bien un elemento de un tensor cuando se trata de capas más internas. En la ecuación anterior  $K$  hace referencia a lo que se denomina núcleo de convolución. La aplicación de esta expresión al contexto de las imágenes resulta en el esquema gráfico mostrado en la figura 2.1. El núcleo de convolución  $K$  con sus correspondientes valores, se posiciona sobre la cuadrícula superior izquierda para obtener el resultado  $P$  que se posiciona en la cuadrícula superior izquierda. A continuación, se desplaza el núcleo una posición hacia la derecha para obtener de forma similar el resultado  $Q$  y así sucesivamente se desplaza el núcleo de izquierda a derecha y de arriba hacia abajo hasta completar los valores de las celdas en la matriz resultante.

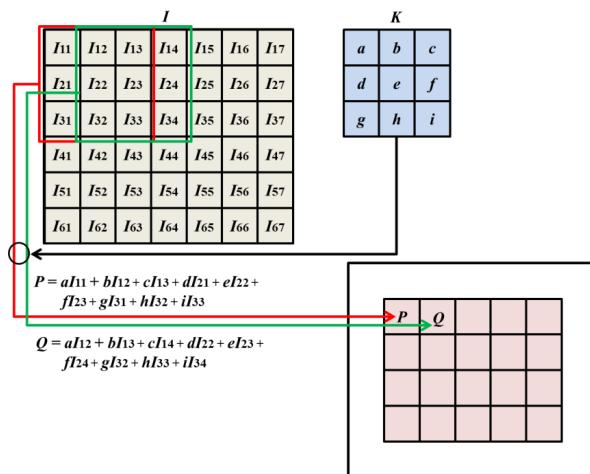


Figura 2.1 Ejemplo de convolución 2-D

No obstante, si este núcleo en lugar de desplazarse una única celda se desplaza más de una, es necesario establecer lo que se conoce como *stride(s)* que puede tomar valores mayores que la unidad. Además, como el núcleo puede desbordar la imagen o tensor de entrada cuando la celda correspondiente al valor  $e$  del núcleo  $K$  se sitúa en las filas o columnas más exteriores, existen valores del núcleo que no se

utilizan, en cuyo caso estas filas o columnas quedan sin procesar. Por esta razón, si se desea que el resultado tenga las mismas dimensiones de la imagen, es necesario añadir filas y columnas llenas de ceros, a esta operación se le conoce como *padding*( $p$ ). En función de los valores  $s$  y  $p$  para una imagen de entrada con dimensión  $i$  y para valores dados de  $p$  y  $s$  se obtiene la correspondiente dimensión de salida  $o$  según las siguientes relaciones. En estas expresiones  $k$  es la dimensión del núcleo de convolución, que por ejemplo en el caso del ejemplo anterior,  $k = 3$ .

$$\text{Relación 5: } o = \left\lceil \frac{i - k}{s} \right\rceil + 1 \quad \text{Relación 6: } o = \left\lceil \frac{i + 2p - k}{s} \right\rceil + 1 \quad (2.8)$$

### b) ReLU

Esta función conocida como *Unidad Lineal Rectificada* (ReLU, *Rectified Linear Unit*) se define como  $f(x) = \max(0, x)$ , cuya representación es la que se muestra en la figura 2.2, cuyas características y su utilidad se centran en lo siguiente:

- a) Evitar la saturación del gradiente durante el proceso de optimización por el método del gradiente descendente.
- b) Disminuir la complejidad computacional por el hecho de reducir valores negativos a cero.

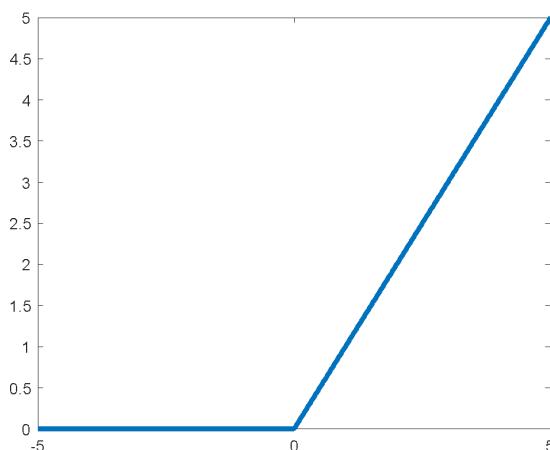


Figura 2.2 Representación de la función ReLU

### c) Normalización

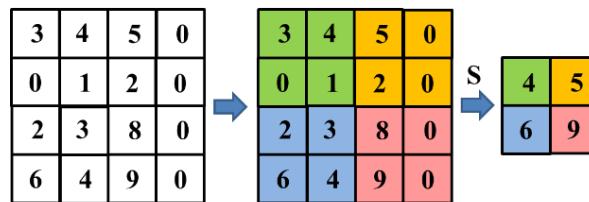
La operación de normalización se aplica con el fin de acelerar la convergencia durante el proceso de aprendizaje. En la expresión siguiente se define una función de normalización por lotes,

$$b_{x,y}^i = a_{x,y}^i \left/ \left( k + \alpha \sum_{j=\max(0,i-n/2)}^{\min(N-1,i+n/2)} (a_{x,y}^i)^2 \right)^\beta \right.$$
(2.9)

donde N es el número de filtros de la capa dada,  $a_{x,y}^i$  es la actividad de la neurona y k,  $\alpha$ , n,  $\beta$  son hiperparametros. En Krizhevsky (2012) se proponen como más apropiados los siguientes valores para los parámetros  $k = 2$ ,  $n = 5$ ,  $\alpha = 10^{-4}$ ,  $\beta = 0.75$ .

#### d) Pooling

Se trata de una función cuyo objeto es agrupar valores en las capas de características, de tal forma que dada una pequeña traslación en la entrada no afecte a los valores de las salidas. En la figura 2.3 se muestra un ejemplo gráfico de agrupamiento por máximos, es decir seleccionando el valor máximo en cada una de las ventanas de agrupamiento, en este caso de dimensión 2x2.



**Figura 2.3 Max pooling**

#### e) Dropout

Es un mecanismo para evitar sobresaturaciones en la red neuronal. Consiste en anular determinado tipo de neuronas de forma aleatoria mediante un hiperparámetro que indique la probabilidad de supervivencia de cada neurona.

#### f) Softmax

Consistente en proyectar los valores de la salida en la capa final al rango [0,1], proporcionando así una especie de probabilidad de pertenencia a cada una de las clases para una imagen de entrada dada. Su función se define según la siguiente expresión.

$$\text{softmax}(\mathbf{x})_i = \frac{\exp(x_i)}{\sum_{j=1}^n \exp(x_j)}$$
(2.10)

### g) Gradiente descendente

Es una técnica de minimización utilizada para el ajuste de los pesos involucrados en los modelos de las redes durante el proceso de retro-propagación. La función para optimizar se conoce como función *objetivo* y cuando se está minimizando, se le denomina también *loss function* o *error function*. Se trata de minimizar una función objetivo que tiene la forma,

$$J(w) = \frac{1}{n} \sum_{i=1}^n J_i(w) \quad (2.11)$$

donde el parámetro  $w$  que minimiza  $J(w)$  debe estimarse, constituyendo el objetivo principal del aprendizaje.  $J_i$  se asocia con la  $i$ -ésima observación en el conjunto de datos utilizados para el entrenamiento (ajuste). El gradiente descendente se usa para minimizar esta función de forma iterativa, con iteraciones  $t$ ,

$$w(t+1) = w(t) - \varepsilon \frac{1}{n} \sum_{i=1}^n \nabla J_i(w) \quad (2.12)$$

De esta forma iterativa el método recorre el conjunto de entrenamiento y realiza la actualización anterior para cada muestra de entrenamiento. Se pueden realizar varios pasos sobre el conjunto de entrenamiento hasta que el algoritmo converja. Estas muestras así seleccionadas constituyen lo que se denomina **batch** como se describe más adelante a la hora de seleccionar los parámetros de entrenamiento. En este sentido, es habitual utilizar exactamente la técnica conocida como Gradiente Descendente Estocástico (SGD, *Stochastic Gradient Descent*) (Bishop, 2006; Ruder, 2017).

Las implementaciones típicas pueden usar una razón de aprendizaje adaptativa para que el algoritmo converja. En pseudocódigo, el método de gradiente descendente estocástico es como sigue:

1. Elegir un vector inicial de parámetros  $w$  (puede ser aleatoriamente) y razón de aprendizaje  $\varepsilon$ .
2. Repetir hasta que se consiga un mínimo aproximado.
  - 2.1. Seleccionar aleatoriamente ejemplos en el conjunto de entrenamiento
  - 2.2. Para  $i = 1, 2, \dots, n$ , hacer  $w(t+1) = w(t) - \varepsilon \nabla J_i(w)$

#### 2.3.1 AlexNet

La red *AlexNet* (ImageNet, 2020; Russakovsky y col., 2015; Krizhevsky y col., 2012; BVLC AlexNet Model, 2022) como se ha mencionado previamente, es una de las utilizadas en el ámbito de las RNC, ganadora de la competición ImageNet LSVRC (2012) para el reto del año 2012.

En la figura 2.4 se muestra la estructura de la red *AlexNet* con ilustración gráfica de las operaciones involucradas, identificando las mismas con indicación de las dimensiones de filtros en las capas convolucionales y totalmente conectadas, para las operaciones de Convolución (*conv*), ReLU (*relu*), Normalización (*norm*), Pooling (*pool*), dropout (*drop*) o softmax. Se incluyen las dimensiones de los filtros, el *stride* (*s*), el *padding* (*p*) y el número de filtros (*K*) en cada capa.

En la tercera columna de la tabla 2.1 se indican los parámetros (pesos) que se aprenden en este modelo, que son los pesos en las capas (primera columna) de convolución (*conv1* a *conv5*) y las totalmente conectadas (*Fully Connected*, *fc6*, *fc7* y *fc8*), independientemente de que se aplique *dropout* o no en ellas. Obsérvese que como pesos a aprender se incluyen en cada capa de convolución un valor de *bias* por filtro y que es un parámetro de ajuste adicional. Por otra parte, en la tabla 2.1 también se muestran en la segunda columna las dimensiones de salida correspondientes a la capa que se indica.

Nombre de capa	Dimensión de salida	Pesos
conv1	55×55×96	$W = 11 \times 11 \times 3 \times 96$ $b = 1 \times 1 \times 96$
conv2	27×27×256	$W = 5 \times 5 \times 48 \times 256$ $b = 1 \times 1 \times 256$
conv3	13×13×384	$W = 3 \times 3 \times 256 \times 384$ $b = 1 \times 1 \times 384$
conv4	13×13×384	$W = 3 \times 3 \times 192 \times 384$ $b = 1 \times 1 \times 384$
conv5	13×13×256	$W = 3 \times 3 \times 192 \times 256$ $b = 1 \times 1 \times 256$
fc6	1×1×4096	$W = 4096 \times 9216$ $b = 4096 \times 1$
fc7	1×1×4096	$W = 4096 \times 4096$ $b = 4096 \times 1$
fc8	1×1×4096	$W = 1000 \times 4096$ $b = 1000 \times 1$

**Tabla 2.1 Parámetros aprendidos en el modelo AlexNet**

Además, en la estructura de la figura 2.4 se indican los números de *Relación*, que establecen precisamente la relación entre las dimensiones de salida de cada capa (*o*) considerando los parámetros de entrada (*i*, *k*, *p*, *s*), según la ecuación (2.8), se definen las distintas relaciones posibles, junto con su número correspondiente asociado.

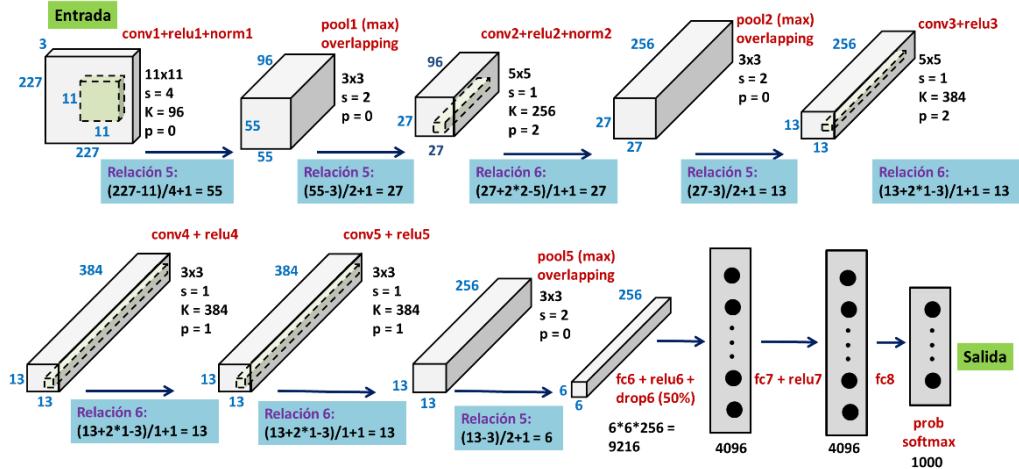


Figura 2.4 Modelo de red AlexNet

En algunas implementaciones, Matlab (2022), tras la función *relu7* se realiza una operación de *dropout* del 50% modificando las dimensiones de la salida de esta capa.

### 2.3.2 GoogleNet

La red ganadora del concurso ILSVRC 2014 fue *GoogleNet* (también conocida como Inception V1) de *Google* (BVLC GoogleNet Model, 2022), proviene de la publicación de Szegedy y col. (2014), habiendo logrado una tasa de error de 6.67%. Se trata de un resultado muy cercano al nivel humano, de modo que los organizadores del desafío necesitaron de la intervención experta. Resulta que, en realidad, la evaluación por parte del experto era bastante difícil de realizar, requiriendo algún entrenamiento adicional para determinar la precisión de *GoogleNet*. La red usada se inspiró en LeNet, añadiendo un elemento novedoso denominado módulo *inception*. Un módulo *inception* (Inc) consta de varias convoluciones relativamente pequeñas para reducir drásticamente el número de parámetros. La arquitectura del modelo *GoogleNet* consiste en una RNC de 22 capas de profundidad, que consigue una reducción en el número de parámetros de 60 millones (*AlexNet*) a 4 millones, de esas 22 capas, 9 son de tipo *inception* de distintas categorías. En total el número de capas es de 144, donde cada una de las 9 capas *inception* contiene la estructura mostrada en la figura 2.5, en la que se han incorporado las unidades ReLU presentes en el módulo. El modelo completo propuesto por Szegedy y col. (2014) es el mostrado en la figura 2.6, donde aparecen las distintas capas: convolución (Conv) indicando las dimensiones de los filtros; Pooling, bien average (AvgPool) o máximo (MaxPool) con indicación en este caso también del tamaño de la ventana; Normalización (LRN, Local Response Normalization); capas totalmente conectadas (FC) y por supuesto los módulos *Inception* (Inc), en este caso V1 que constituyen la parte nuclear de este tipo de redes. En las capas de convolución y *pooling* se indica también el desplazamiento (*stride*), en este caso se expresa entre paréntesis con el símbolo *s* seguido del correspondiente valor de desplazamiento en el caso '*same*' y *V* también con el correspondiente valor de desplazamiento, si bien en este caso de

tipo ‘*valid*’. Como puede comprobarse, este esquema presenta dos redes extra, que son exactamente sendos clasificadores auxiliares, y constan de un *pooling* promediado de dimensión  $5 \times 5$  y  $s = 3$  de tipo V que proporciona salidas de tamaños  $4 \times 4 \times 512$  y  $4 \times 4 \times 528$  respectivamente. Le sigue una capa de convolución  $1 \times 1$  con 128 filtros para reducción de la dimensionalidad y una unidad ReLU. Continúa una unidad *dropout* con capas totalmente conectadas finalizando con unidades *softmax* (Softmax0 y Softmax1). La salida final de la red también es una unidad *softmax* (Softmax2).

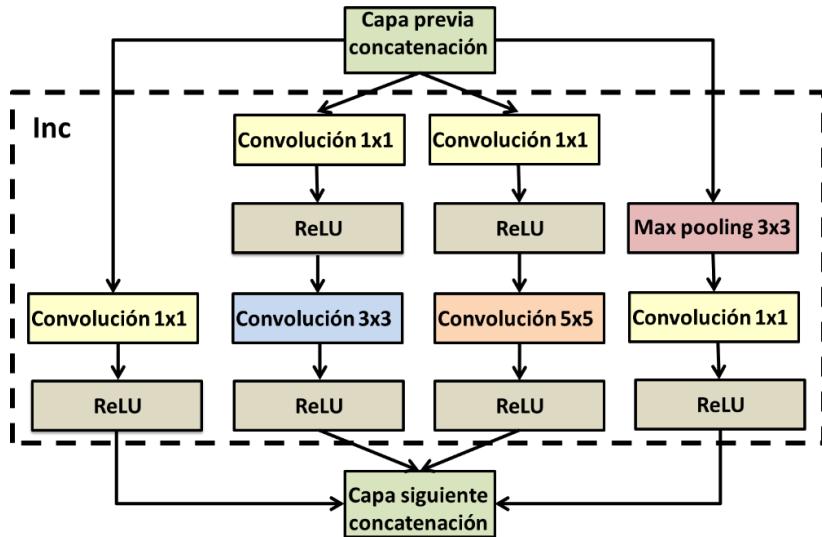


Figura 2.5 Módulo inception con incorporación de unidades ReLU

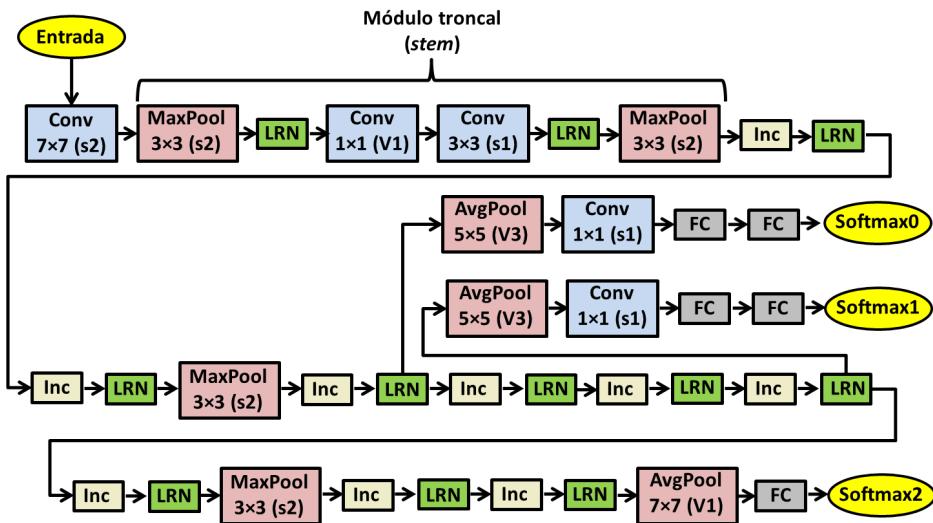


Figura 2.6 Modelo completo GoogleNet (inception V1)

### 3. Diseño y análisis desarrollado

Tras el estudio de los modelos conceptuales, a continuación se describe el diseño de la arquitectura para la integración de los mismos en la aplicación. En este sentido se plantea el diseño de la arquitectura propuesta, tanto desde el lado del cliente como del servidor. El siguiente aspecto abordado es el concerniente a la

metodología y gestión del proyecto, para finalizar con la descripción de las herramientas utilizadas en todo el desarrollo.

### 3.1 Arquitectura

Para este proyecto se han desarrollado e implementado tres módulos principales que permiten la utilización de las diferentes herramientas mencionadas anteriormente. En la figura 3.1 podemos observar el esquema con los diferentes módulos que representan la arquitectura de la aplicación y que se describen a continuación.

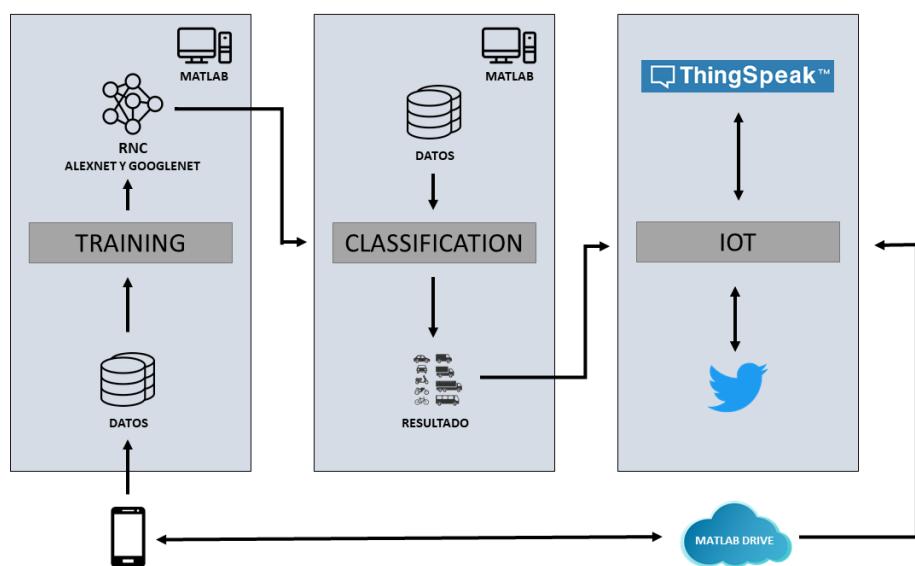


Figura 3.1 Esquema de la arquitectura del sistema

- **Módulo Training**

Se dispone de un conjunto suficiente de imágenes de vehículos capturadas con un teléfono móvil que se encuentran etiquetadas según la categoría a la que pertenecen. Se han integrado dos modelos de RNC (*AlexNet* y *GoogleNet*) configurados y listos para entrenar. Una vez preparado todo, se procede a entrenar en Matlab (2022) ambos modelos de RNC de forma separada para su posterior clasificación.

- **Módulo Classification**

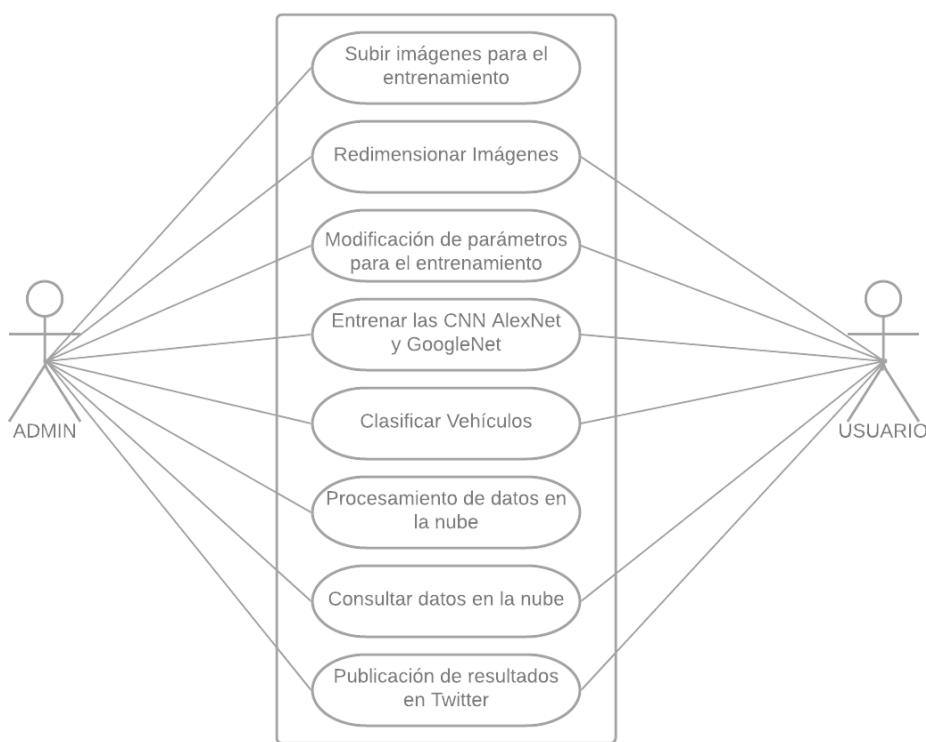
Previamente, se utilizará el dispositivo móvil para capturar los vídeos que se almacenan en el ordenador para su posterior clasificación. Mediante las funcionalidades que proporciona Matlab (2022), en este módulo se procede a la clasificación de los vehículos según el modelo de RNC seleccionado (*AlexNet* o *GoogleNet*). El resultado obtenido se envía automáticamente al servidor en la nube *ThingSpeak* (2022) para su almacenamiento y análisis en

el siguiente módulo (IoT). En lo que sigue, a veces se hace referencia a la nube como sinónimo de este servidor. Conviene recordar que *ThingSpeak* es la plataforma que ofrece Matlab en el contexto de IoT.

- **Módulo IoT**

Los resultados obtenidos en el módulo *Classification* se almacenan automáticamente en la base de datos de *ThingSpeak* y se envían en un *tweet*. También es posible buscar y filtrar los datos de clasificaciones anteriores almacenados en *ThingSpeak*.

Seguidamente, se explica el diagrama de casos de uso que se muestra en la Figura 3.2, que reúne todas las operaciones que contiene la aplicación. Dependiendo del rol que tenga una persona podrá realizar determinadas funciones en la aplicación. En nuestro caso, distinguimos dos tipos de roles: **usuario** y **administrador**.



**Figura 3.2 Diagrama de casos de uso**

El administrador es el encargado de manejar la aplicación desde una perspectiva de gestión y, por lo tanto, realiza las siguientes operaciones: subida de nuevas imágenes para el módulo *Training*, redimensión de imágenes, modificación de parámetros para el desarrollo eficiente del entrenamiento de la red, entrenamiento de las redes *AlexNet* y *GoogleNet*, clasificación de vehículos, procesamientos y consulta de datos en la nube *ThingSpeak* y publicación de resultados en *Twitter*.

En cuanto al usuario corresponde a cualquier persona que pueda utilizar la aplicación. Puede realizar las mismas operaciones anteriormente descritas, exceptuando la subida de imágenes y el procesamiento de datos en *ThingSpeak*.

La arquitectura en la que se basa el proyecto es un modelo multicapa, por lo que el sistema software está organizado en varias capas. Cada una de ellas con unas características propias, tal y como se especifica a continuación.

- **Capa de presentación**

Donde se desarrolla una interfaz para lograr la interacción entre el usuario y la aplicación.

- **Capa de negocio**

En esta capa se encuentra la lógica y todas las operaciones necesarias para realizar las tareas de la aplicación, incluyendo todo lo relacionado con las redes neuronales.

- **Capa de integración**

En el caso del diseño que se propone, la capa de integración no es necesaria, ya que el almacenamiento y el acceso de los datos no se realiza a través de una base de datos relacional como tal, sino que se utiliza *ThingSpeak* como recurso de control de datos en la nube. Además, *ThingSpeak* es una plataforma remota con acceso integrado en MATLAB y con acceso directo, lo que lo convierte en un recurso bastante cómodo y sencillo para el control de datos sin tener que requerir de otro servicio.

### 3.1.1 Lado del cliente

En el lado del cliente se desarrollan todas las operaciones de la aplicación mostradas en la figura 3.2, exceptuando aquellas partes que se relacionan con *ThingSpeak* y *Twitter*. Por lo que las funciones que el cliente puede aplicar son las que se indican a continuación.

- **Entrenamiento de las redes**

En el menú principal de la aplicación (*MAIN MENU*) se encuentra la opción de *Training*, que permitirá al usuario entrenar o reentrenar la red que seleccione posteriormente, ya sea *AlexNet* o *GoogleNet*, Figura 3.3

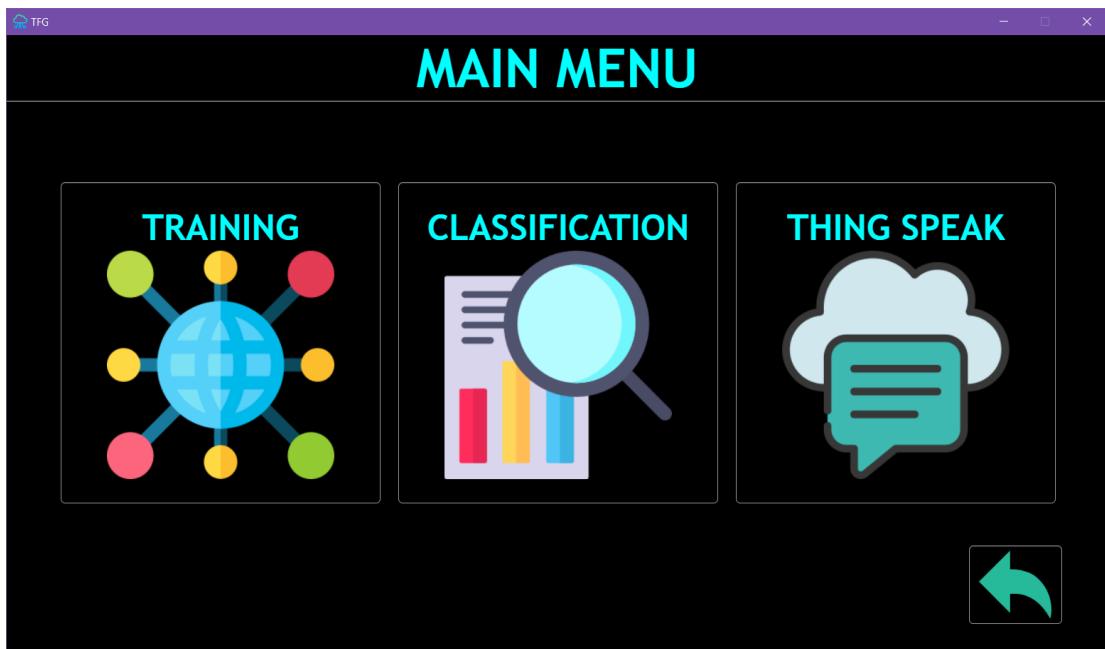


Figura 3.3 Main Menu

Una vez elegido el tipo de red, la vista perteneciente a la opción de *Training* de la red seleccionada aparecerá en pantalla, permitiendo al usuario acceder a la redimensión de las imágenes y al ajuste de parámetros que se van a utilizar para el entrenamiento, tal y como muestra la figura 3.4.

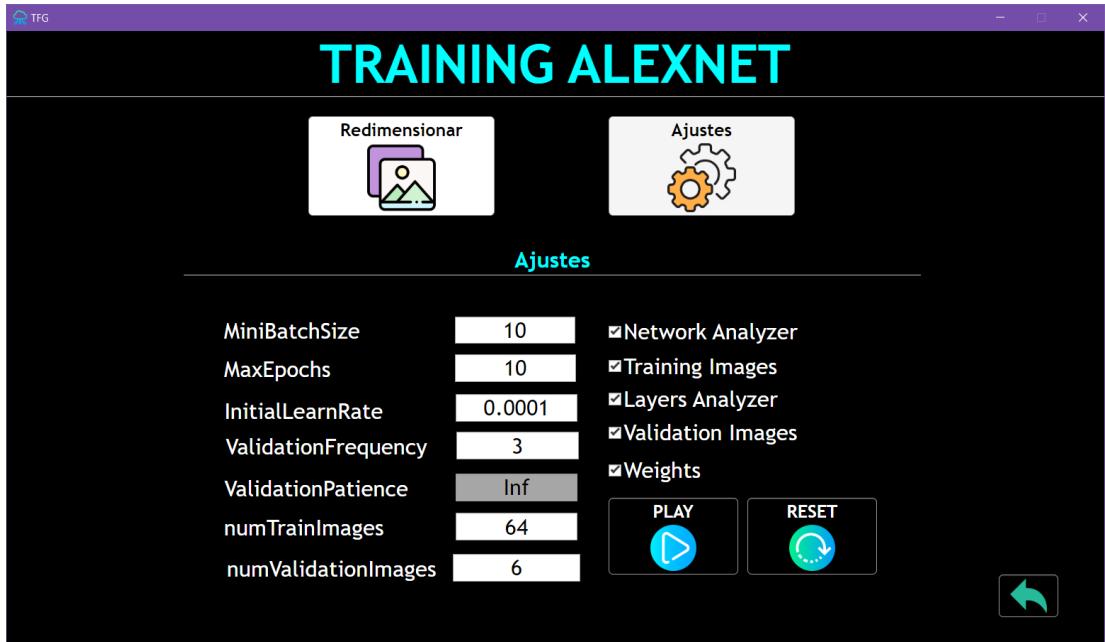


Figura 3.4 Training

Concretamente, los parámetros y configuraciones que se pueden establecer son los que se describen a continuación:

- Dimensión por lotes (mini-batch size).** Define la cantidad de imágenes que se utilizan en cada iteración.

- b. **Epochs e iteraciones.** Representa el paso de todas las imágenes disponibles para el entrenamiento. Se ha establecido un máximo posible.
- c. **Razón de aprendizaje (learning rate).** Establece la rapidez con la que la red aprende según el método de actualización de los pesos de cada modelo y que constituyen el objetivo del aprendizaje.
- d. **Frecuencia de validación (validation frequency).** Que representa el número de iteraciones entre las evaluaciones que establecen las métricas de validación.
- e. **Validation Patience.** Indica la detención del proceso de entrenamiento tras el número de *epochs*, fijado por dicho parámetro, para el que no se consiguen mejoras relevantes en la precisión.
- f. **Número de imágenes de entrenamiento (numTrainImages).** Número de imágenes utilizadas para el entrenamiento.
- g. **Número de imágenes de validación (numValidationImages).** Número de imágenes utilizadas para la validación de los resultados.

Una vez seleccionados los valores de los parámetros indicados, el usuario puede comenzar a entrenar la red elegida tras pulsar *Play*, lo que provoca que los datos de los parámetros vuelvan al *Controller* de la aplicación y de ahí se lance el proceso, tras lo cual se guardan los resultados de aprendizaje, concretamente el modelo de red con sus correspondientes pesos ajustados en las correspondientes capas.

- **Modificación de parámetros para el entrenamiento**

Como se puede observar en la Figura 3.4, la aplicación ejecuta el entrenamiento con los parámetros predefinidos y mostrando las vistas seleccionadas. En cualquier momento, el usuario puede modificar los parámetros y, si desea revertir los cambios, puede volver a los ajustes predeterminados pulsando el botón *Reset*.

Los valores predeterminados del entrenamiento son cargados desde los archivos *ajustesTrainingAlex.mat* y *ajustesTrainingGoogle.mat*, dependiendo de la red que vaya a entrenar, que se establece de antemano.

Estos archivos con la extensión *mat* son contenedores de datos binarios que utiliza el programa MATLAB y que permiten clasificar los datos iniciales de las variables a utilizar para el entrenamiento por defecto.

- **Redimensionar imágenes**

Si se pulsa sobre el botón Redimensionar, aparece una vista como se muestra en la figura 3.5. La funcionalidad de redimensionar tiene como objetivo convertir las imágenes al formato correcto para su posterior

entrenamiento. En el caso de *AlexNet*, 227 x 227 x 3 píxeles y para *GoogleNet*, 224 x 224 x 3 píxeles.

Mediante el botón Cargar Imagen, el usuario escoge la imagen que desea redimensionar ubicada en algún contenedor de su ordenador.



Figura 3.5 Redimensión de imágenes

Una vez cargada la imagen, el usuario pulsa el botón *Start*. Aparece un mensaje de éxito y la imagen queda redimensionada correctamente para su posterior entrenamiento.

- **Clasificación de vehículos**

Cuando el usuario selecciona la opción de *Classification* en el menú principal mostrado en la figura 3.3, se despliega la ventana que aparece en la figura 3.6.

La funcionalidad de clasificar es, junto con la de entrenamiento, una aplicación clave, ya que es la que realiza la detección de vehículos, utilizando para ello la red que se indique. En nuestro caso el usuario elige entre las dos redes neuronales dispuestas y entrenadas con tal finalidad (*AlexNet* y *GoogleNet*).

Una vez seleccionada la red, el usuario podrá escoger un video desde una ubicación dada en su ordenador para cargarlo en la ventana de la aplicación y proceder a controlarlo a través de los botones *play* y *stop*.

Seguidamente, cuando el usuario pulse *play*, el video se reproducirá *frame a frame* y a la derecha (*Searcher*) se mostrarán las imágenes encontradas en la búsqueda para la detección de vehículos.

Cuando el video finaliza o el usuario pulsa *stop*, los datos recopilados se envían a través del *Controller* a otra ventana, dónde se guardarán y se procesarán para su posterior envío a la nube y publicación.



Figura 3.6 Classification

### 3.1.2 Lado del servidor

En el lado del servidor se alojan y tratan las operaciones relacionadas con la extracción y manejo de datos en la aplicación representadas en la figura 3.2. Por lo tanto, las funcionalidades que trata el servidor son:

- **Procesamiento de datos en la nube**

Cuando hablamos de la nube, nos referimos a los servidores a los que accedemos a través de Internet, además del software y base de datos que se ejecutan en ellos. En nuestro caso, procesamos y enviamos los datos obtenidos a partir de la clasificación de vehículos realizada con anterioridad. Para ello, la plataforma encargada de realizar este procesamiento es *ThingSpeak*, que es la específica de Matlab para IoT.

La figura 3.7 muestra, en esta plataforma, la configuración de un canal, con su nombre (Name), identificador (Channel ID), junto con los campos y sus nombres asociados para alojar los datos procesados, como se explica a continuación.

Inicialmente, el registro de los datos obtenidos es llevado a cabo a través de la creación de canales. Como se ha indicado previamente, estos nos permiten almacenar los resultados recopilados en nuestra aplicación. En consecuencia, para poder enviar, procesar y recuperar datos, debemos configurar primeramente el canal de almacenamiento de los datos.

### Channel Settings

Percentage complete	30%
Channel ID	1564684
Name	<input type="text" value="Detección Vehículos"/>
Description	<input type="text"/>
Field 1	<input type="text"/> <input type="checkbox"/>
Field 2	<input type="text" value="Coches"/> <input checked="" type="checkbox"/>
Field 3	<input type="text" value="Buses"/> <input checked="" type="checkbox"/>
Field 4	<input type="text" value="Motos"/> <input checked="" type="checkbox"/>
Field 5	<input type="text" value="Camiones/Furgonetas"/> <input checked="" type="checkbox"/>
Field 6	<input type="text"/> <input type="checkbox"/>
Field 7	<input type="text"/> <input type="checkbox"/>
Field 8	<input type="text"/> <input type="checkbox"/>
Metadata	<input type="text"/>
Tags	<input type="text"/> <small>(Tags are comma separated)</small>

**Figura 3.7 Configuración del canal y visualización del Channel ID**

Cada canal se compone de ocho campos generales para el almacenamiento de cualquier tipo de dato, tres campos para datos de ubicación, y un campo de estado. En nuestro caso, se han utilizado cuatro campos generales, donde se almacena el porcentaje de vehículos (coches, autobuses, motos, y camiones o furgonetas) con respecto al total de los clasificados. Los vehículos indicados representan las categorías objeto del aprendizaje y clasificación. Otros aspectos a destacar, que, aunque no han sido utilizados consideramos que hay que mencionar, son la posibilidad de poder almacenar el código de *ThingSpeak* en un repositorio de Github, o la opción de introducir etiquetas clave que identifiquen el canal.

- **Consulta de datos en la nube**

Una vez configurado un canal determinado se procede a su creación. Esta acción genera automáticamente dos API keys, las cuales se componen de una serie de caracteres, formada por letras y números. Una de ellas genera la posibilidad de escritura de datos en el canal, y otra se utiliza para poder leer los datos incluidos del canal. Estas claves son añadidas a la aplicación de Matlab, y permiten, llamando a una clave u otra, acceder al modo de escritura o de lectura de *ThingSpeak*. Existe, además, la posibilidad de generar más API keys o de eliminarlas por si se produce alguna vulnerabilidad en el canal.

En la pestaña de configuración del canal, cabe resaltar la visualización del *Channel ID*, como se ha mencionado previamente en la figura 3.7, un número identificativo clave que nos permite conectar nuestro código a *ThingSpeak*. Sin este número no es posible transferir ni analizar los datos obtenidos a partir de nuestra clasificación de vehículos, por lo que es necesario incluirlo en la aplicación.

Conocidas las claves de acceso, cabe la posibilidad de que diferentes usuarios accedan al canal para la escritura concurrente de datos. Para evitar fallos de sincronización o de multiacceso simultáneo, *ThingSpeak* obliga a establecer un intervalo de 15 segundos entre cada acceso de escritura, devolviendo un mensaje de error en caso de que haya más de un acceso de escritura a la vez y aceptando únicamente el primero que llegue.

Una vez estructurada toda la configuración de *ThingSpeak*, el usuario de la aplicación posee en el menú principal la opción de visualizar los datos en la nube, además de poder generar nuevos datos. Para ello, el usuario tiene a su disposición la ventana que aparece en la figura 3.8.

A la derecha de la pantalla se muestra una tabla con los resultados obtenidos a partir de cada clasificación realizada, por defecto organizada cronológicamente. A partir de la segunda columna, podemos comprobar el porcentaje de cada tipo de vehículo que se ha tenido como resultado de la clasificación. A la izquierda de la pantalla aparece un panel de selección de vehículos, con el cual podemos filtrar en la tabla los datos, dependiendo de si queremos visualizar solo los porcentajes de un determinado tipo de vehículo, y un botón llamado *accept* para confirmar el cambio. La aplicación dispone, además, de un botón para poder visualizar todos los tipos de vehículos, por si anteriormente hemos filtrado los datos y queremos de nuevo obtener una visualización completa.

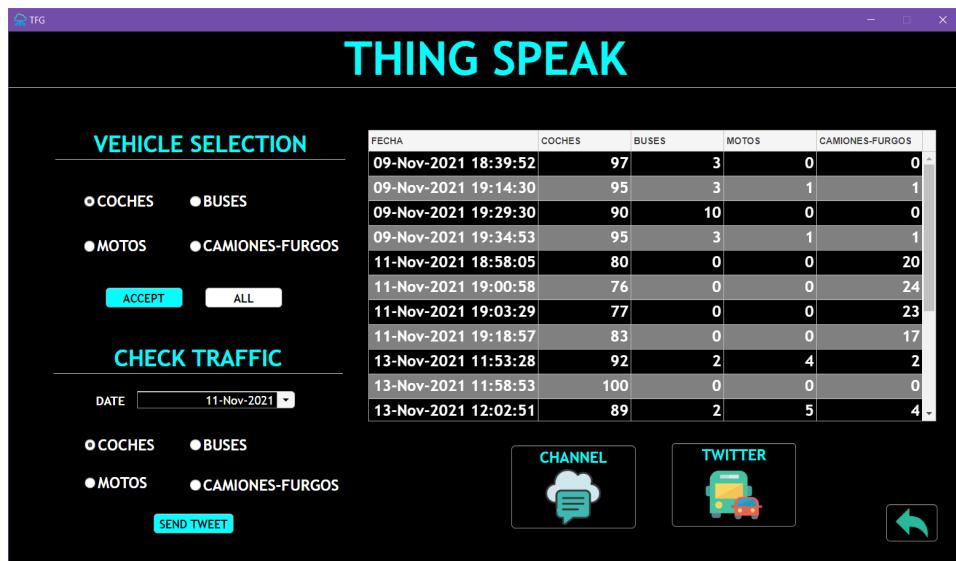


Figura 3.8 Interfaz de ThingSpeak en la aplicación

Hay que tener en cuenta el botón situado debajo de la tabla de consultas, denominado *channel*, el cual nos redirige a la visualización de la vista del canal. Cuando se presiona este botón, se pueden visualizar cuatro gráficas, una por cada tipo de vehículo, las cuales en los ejes de ordenadas expresan el porcentaje de cada dato obtenido, y en los ejes de abscisas se representa la fecha en la que se ha obtenido dicho resultado. Estas gráficas ayudan a visualizar de manera sencilla los resultados obtenidos en cada clasificación.

Como ejemplo ilustrativo de los resultados disponibles en un momento determinado, se pueden distinguir las cuatro gráficas que se muestran en la figura 3.9.

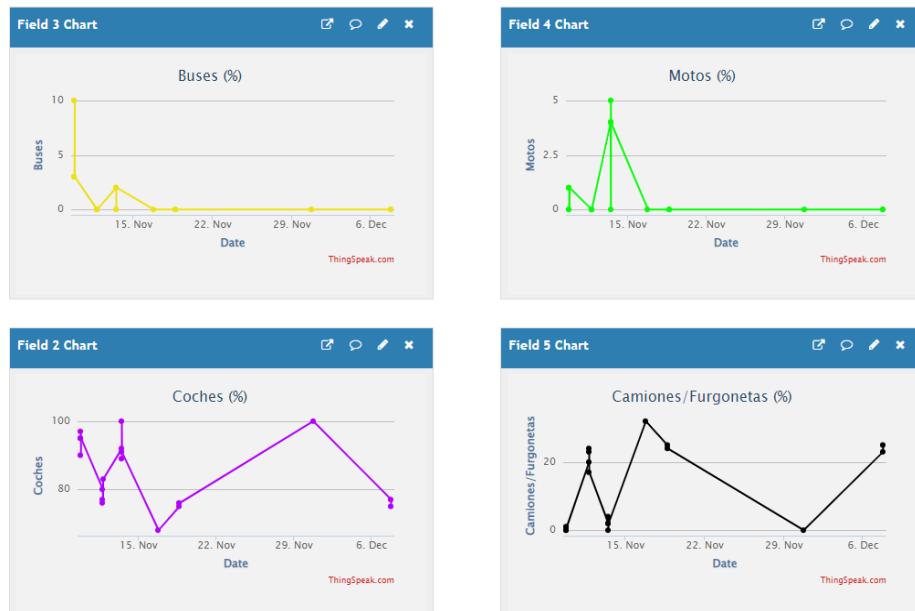


Figura 3.9 Gráficas de los resultados en ThingSpeak

Obsérvese cómo dichas gráficas muestran las mencionadas clasificaciones según las categorías de vehículos que participan en la clasificación.

- **Publicación de resultados en Twitter**

Por último, hay que indicar que *ThingSpeak* cuenta con una funcionalidad dentro de su entorno de aplicaciones llamada *ThingTweet*. Gracias a *ThingTweet*, es posible conectar una cuenta de *ThingSpeak* a una cuenta de *Twitter* generando una *API Key* tal y como se muestra en la figura 3.10.

Twitter Account	API Key	Action
RecVehiculosTFG	NHBVCHXRUW6P8IU3	<button>Regenerate API Key</button> <button>Unlink Account</button>

Figura 3.10 Configuración de *ThingTweet*

Esto es de gran utilidad a la hora de crear alertas y enviar *tweets* automáticamente cuando se deseé según la programación establecida en cada momento. En nuestro caso, la aplicación envía un *tweet* a nuestra propia [cuenta de Twitter](#) cada vez que se realiza una clasificación mostrando los datos obtenidos. Un ejemplo de esta operatividad se observa en la figura 3.11.



Figura 3.11 Ejemplo Tweet Clasificación

Volviendo a la ventana de *ThingSpeak*, que se muestra en la figura 3.8, podemos observar que el usuario dispone de un apartado llamado *Check Traffic*. De esta forma el propio usuario puede seleccionar una fecha y el tipo de vehículo según su interés, y al pulsar sobre el botón *Send Tweet*, la

aplicación envía un *tweet* con esa información a la cuenta de *Twitter*. En la figura 3.12 se muestra un ejemplo de aplicación de esta funcionalidad.



Figura 3.12 Ejemplo Tweet Check Traffic

## 3.2 Metodología y gestión de proyecto

Uno de los primeros temas a resolver al comienzo del proyecto fue el tipo de metodología a utilizar. Al tratarse de un equipo pequeño y un proceso flexible, se descartaron las metodologías tradicionales por completo. Por tanto, pasamos directamente a analizar las metodologías ágiles, cuyas características se adaptan mejor al tipo de proyecto a realizar.

Finalmente, se decidió utilizar la metodología *Scrumban*, que aprovecha la naturaleza prescriptiva de *Scrum* para ser ágil y la mejora de procesos de *Kanban* para permitir que el equipo mejore continuamente su proceso, otorgando también una cómoda visualización del flujo de trabajo y la autoorganización del equipo.

Una vez elegida la metodología, procedimos a gestionar el proyecto adecuando las características propias de *Scrumban* a nuestro trabajo.

Primeramente, organizamos las tareas a realizar. Al no existir roles, todos los miembros del equipo se dedicaron a desarrollar la aplicación de manera equilibrada y a un nivel semejante, tanto en la fase de codificación como en la redacción de la memoria. Por lo que la figura del “líder” no estaba definida en una sola persona, sino que iba rotando según el área del proceso en la que nos encontráramos, estableciendo así una dinámica sincronizada de equipo, que centraba todos los esfuerzos en alcanzar un objetivo común.

A continuación, establecimos la forma más adecuada para llevar un seguimiento del proceso. En nuestro caso, descartamos el uso de *sprints* y de estimaciones, determinando en su lugar reuniones y puntos de control establecidos por el equipo de trabajo y el profesor director. Este último era el responsable de guiar y decidir si el trabajo realizado hasta la fecha cumplía los requisitos y las especificaciones marcadas durante las reuniones.

Las reuniones realizadas para el desarrollo del proyecto son las siguientes.

- **Reunión de planificación**

Aproximadamente cada mes el profesor nos presentaba las funcionalidades a implementar durante una reunión. Acto seguido, el equipo de desarrollo, que está formado por todos los integrantes del proyecto, preguntamos cualquier tipo de duda para comprender los requisitos de la aplicación con detalle. Después, se decidía cuantas funcionalidades se aplicarían finalmente.

- **Dailys**

Semanalmente se realizaba una reunión los martes por la tarde a través de *Discord*, debido a la disponibilidad completa de todo el equipo. En esta reunión se hablaba de todos los cambios y avances realizados por cada miembro del equipo y, posteriormente, se relataban los problemas encontrados, se solucionaban los errores y se realizaban determinadas tareas en conjunto. Además, utilizando *Trello*, se revisaba el tablero Kanban para actualizarlo. Teniendo así una visión más amplia y completa de todo el proyecto, del trabajo que había sido completado y del que quedaba por hacer.

- **Reunión de revisión**

Aproximadamente cada veinte días, de manera telemática a través de *Google Meet*, se realizaba una reunión de revisión. En ella nos centramos, principalmente, en la aplicación final y presentamos al profesor todas las funcionalidades que se han realizado en la aplicación. De esta manera, el profesor podía inspeccionar el producto y comprobar el correcto funcionamiento de la aplicación para darnos un *feedback*.

- **Reunión de retrospectiva**

Esta reunión se realizaba con cierta frecuencia, pero sin un rango específico de tiempo y, básicamente, en ella nos dedicamos a determinar el grado de avance de los trabajos. Después, se identificaban los elementos más importantes con resultados positivos, a la vez que se comentaban posibles mejoras, para finalmente elaborar un plan sobre estas de cara al siguiente paso.

### **3.3 Herramientas empleadas**

Las herramientas empleadas en el proyecto, tanto desde el punto de vista del desarrollo e implementación como de la gestión, son las siguientes.

- **Matlab R2022a (2022)**

Es un lenguaje de cálculo técnico de alto nivel y un entorno interactivo que proporciona un lenguaje basado en matrices y permite la expresión más natural de las matemáticas computacionales. Facilita el análisis de datos con diferentes algoritmos y el estudio de los cambios en el comportamiento y decisiones tomadas por los propios algoritmos. También proporciona flexibilidad para el diseño de interfaces según las necesidades. En nuestro caso, se ha hecho uso de algunas de las diferentes herramientas (*toolboxes*) que proporciona, como son: *Computer Vision*, *Image Processing* y *Deep Learning*, para procesar de manera correcta los modelos de redes utilizados y su representación. Además, se ha utilizado la extensión que proporciona Matlab de *Design App*, para crear la interfaz gráfica de la aplicación.

- **ThingSpeak (2022)**

Es una plataforma especializada en el análisis de datos IoT, respaldada por Matlab. Al igual que otros servicios en la nube, permite crear, visualizar, eliminar y enviar datos desde cualquier dispositivo. Asimismo, estos datos se pueden almacenar en canales privados, o hacerlos públicos si así se requiere. Estos datos pueden ser también analizados y relacionados mediante una gran variedad de herramientas que proporciona la propia plataforma, con interfaz directa con Matlab.

- **Simulink**

Es un entorno de diagramas de bloque integrado en Matlab que permite simular modelos de sistemas físicos y de control mediante diagramas. El comportamiento de estos sistemas se define mediante funciones y operaciones matemáticas. Ofrece un editor gráfico para poder simular estos sistemas.

- **ThingTweet**

Aplicación integrada en *ThingSpeak* que permite enlazar la plataforma con los datos disponibles y enviar información y resultados de procesamiento a una cuenta de *Twitter* para distribuirlos de manera pública.

- **React**

Permite realizar acciones y reaccionar convenientemente a la llegada de valores de los canales que estén integrados en *ThingSpeak* para realizar una

acción programada de antemano. Estas acciones pueden ser realizadas cuando los datos del canal cumplan una determinada condición.

- **Microsoft Office Word**

Entorno de trabajo destinado a la creación, edición, almacenamiento y transmisión de documentos online. Concretamente se ha utilizado para la realización de la memoria en su versión en la nube.

- **Google Drive**

Servicio de almacenamiento en la nube de cualquier tipo de archivo. Utilizado para guardar el progreso de la memoria de forma online, desde la que cualquier miembro del equipo tiene acceso a la manipulación y edición de los archivos que se encuentran en ella.

- **Git**

Herramienta de control de versiones para el proyecto orientada al trabajo en equipo. Permite gestionar de manera óptima y ordenada el progreso del proyecto, ya que permite guardar un historial de versiones. De tal manera que se puede mantener un control sobre el flujo de trabajo y realizar cambios con facilidad.

- **Gitkraken**

Interfaz gráfica de *git* que sirve como complemento para la gestión del proyecto, pudiendo gestionar de forma intuitiva y sencilla el seguimiento del repositorio. Cada actualización del proyecto se visualiza mediante una serie de ramas, cada una de ellas pertenecientes a un miembro del equipo. Esto proporciona la ventaja de poder trabajar en distintos aspectos de la aplicación de manera simultánea y pudiendo, posteriormente, mezclar estas actualizaciones a la rama principal del proyecto.

- **Whatsapp**

Aplicación móvil de mensajería instantánea, cuyo uso principal es la comunicación entre los diferentes miembros del equipo. También permite el envío de imágenes y documentos de manera rápida. Ha sido usada principalmente para pequeñas consultas y para la organización de las reuniones de equipo.

- **Gmail**

Servicio de correo electrónico, que combina las funciones del correo tradicional con la inmediatez online. Utilizada principalmente como medio de comunicación con el profesor para la organización de las reuniones y la rápida resolución de dudas puntuales.

- **Discord**

Herramienta de comunicación mediante voz y texto pensado para tener conversaciones entre los miembros del equipo. Permite la creación de un grupo privado con el que los miembros del equipo pueden comunicarse a la vez de forma telemática. Además, permite presentar y compartir la pantalla con los demás integrantes de la sala, lo que facilita la resolución de problemas y el desarrollo de las reuniones.

- **Google Meet**

Plataforma orientada a la realización de llamadas y videollamadas online. Permite realizar reuniones virtuales o presentaciones dentro del ámbito del teletrabajo. Principalmente usada como medio de comunicación para las reuniones periódicas con el profesor, exponiendo el progreso del trabajo mediante las presentaciones online a través de la plataforma.

- **Lucid.app**

Herramienta que permite trabajar en tiempo real, creando diagramas de flujo, organigramas, esquemas y diseños UML para el desarrollo basado en modelos del dominio. Es usado para la creación de diagramas de clases y casos de uso, los cuales permiten tener una comprensión clara de la construcción y desarrollo del proyecto.

- **Trello**

Aplicación destinada a la gestión y asignación de tareas del proyecto. Ayuda a mejorar las rutinas habituales del trabajo, asignando a cada miembro del equipo, mediante tableros, las diferentes tareas a realizar. Utilizada para la organización del equipo, así como la elección y la finalización de tareas.

- **Flaticon**

Es una base de datos preparada para la descarga de iconos gratuitos y personalizables. Los iconos utilizados para el diseño de la interfaz de la aplicación propuesta han sido seleccionados a través de esta plataforma.

## 4. Resultados obtenidos

A continuación, se describen los resultados obtenidos como consecuencia de la ejecución de la aplicación en las distintas partes de la arquitectura y secciones descritas previamente, siguiendo la misma distribución de contenidos.

### 4.1 Interfaz de la aplicación

Una vez se ejecuta la aplicación, aparece una pantalla inicial como la que se muestra en la Figura 4.1.



Figura 4.1 Pantalla Inicial

Tras pulsar el botón *START*, comienza el reconocimiento de vehículos propiamente dicho, apareciendo el *Main Menu*, anteriormente mostrado en la Figura 3.3, con las siguientes opciones:

- **Training:** Módulo encargado de ejecutar el entrenamiento de la red previamente elegida.
- **Classification:** Módulo encargado de ejecutar la detección y la clasificación de los tipos de vehículos contenidos en los videos a procesar.
- **ThingSpeak:** Módulo encargado de la consulta, almacenamiento y tratamiento de datos y estadísticas del canal de *ThingSpeak* y de la cuenta de *Twitter*.

#### 4.1.1 Training

Al pulsar en el botón *TRAINING*, aparecerá una ventana como la mostrada en la Figura 4.2 para seleccionar el modelo de red que se quiere entrenar.



Figura 4.2 Selección de Modelo de Red

En este caso, se permite elegir entre los dos modelos de redes implementados, tal como se ha mencionado previamente:

- **ALEXNET:** Si se selecciona la opción de la izquierda, la pantalla muestra la ventana de *TRAINING ALEXNET*, tal y como se muestra en la Figura 4.3.

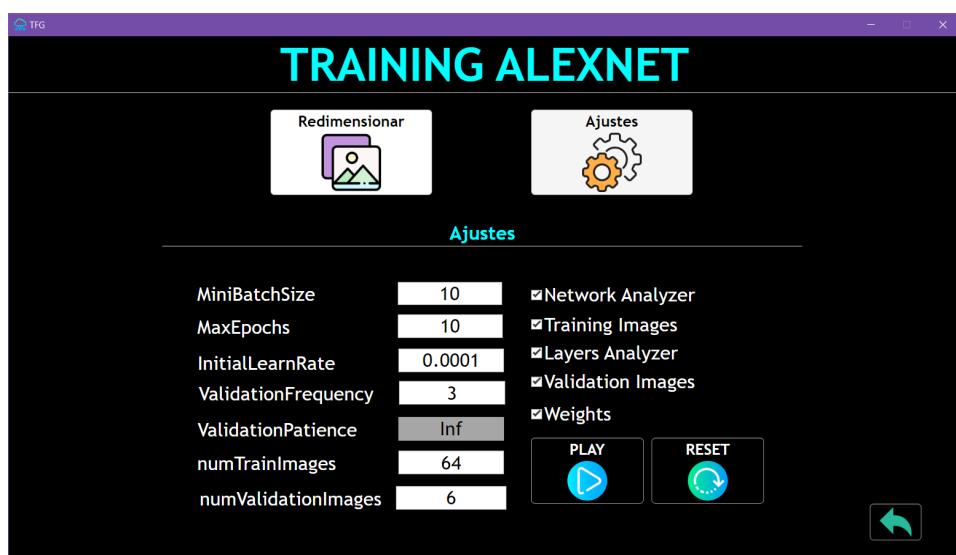


Figura 4.3 Training AlexNet

- **GOOGLENET:** Si se selecciona la opción de la derecha, en la pantalla aparece la ventana de *TRAINING GOOGLENET*, según se muestra en la Figura 4.4.

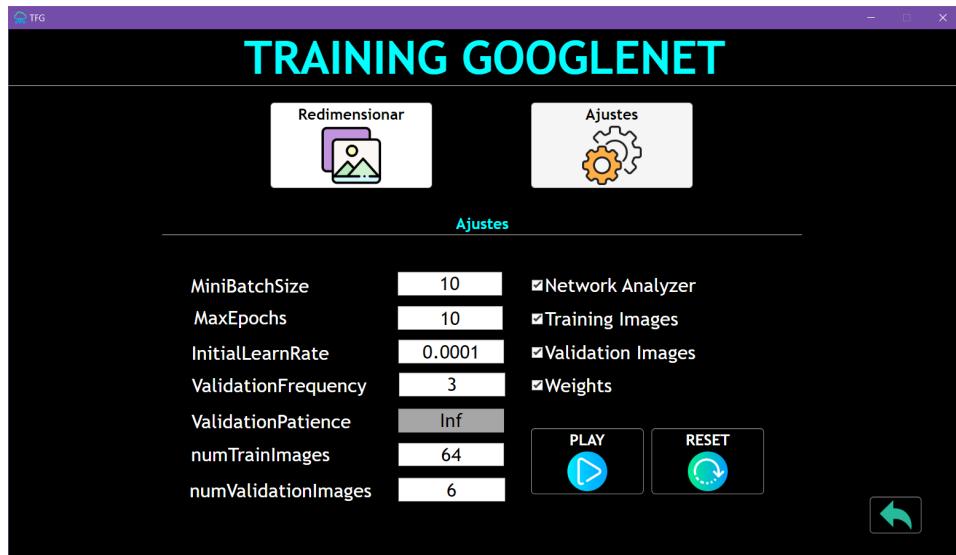


Figura 4.4 Training GoogleNet

Ambas pantallas de entrenamiento constan de un botón para la redimensión de imágenes (recuadro rojo mostrado en la Figura 4.5), un botón para modificar los ajustes de los parámetros (recuadro azul mostrado en la Figura 4.5), un botón *PLAY* (recuadro verde mostrado en la Figura 4.5) para iniciar el entrenamiento y un botón *RESET* (recuadro amarillo mostrado en la Figura 4.5) para inicializar los parámetros a su valor por defecto.

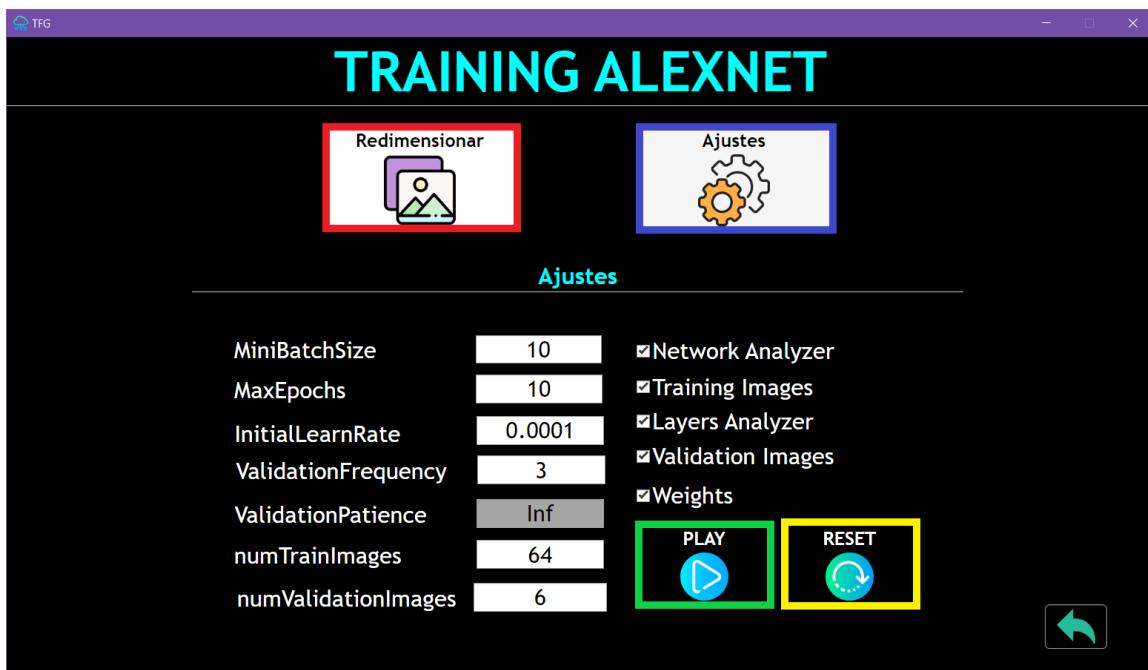


Figura 4.5 Training Seleccionado

Las pantallas correspondientes a Redimensionar y Ajustes son las mostradas previamente en las Figuras 3.4 y 3.5 respectivamente.

#### 4.1.2 Classification

Al pulsar en el botón *CLASSIFICATION*, aparecerá una ventana como la mostrada en la Figura 4.6, dónde se muestran las diferentes secciones que se pueden encontrar.

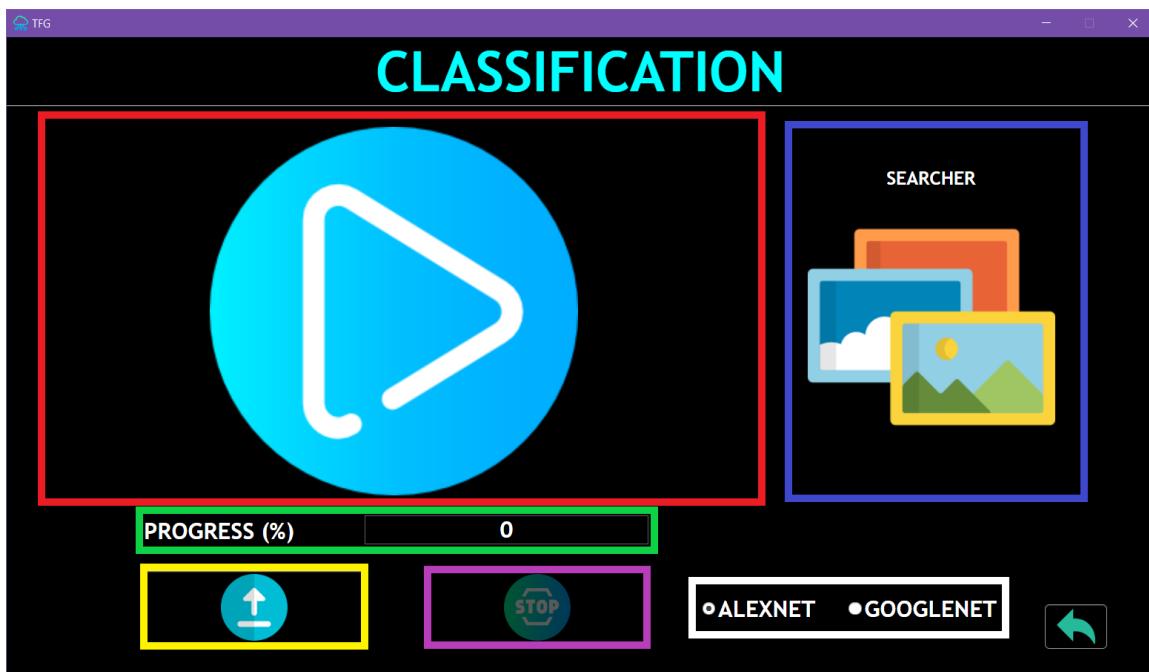


Figura 4.6 Classification Seleccionado

- **Sección roja:** Muestra el video *frame* a *frame* con las etiquetas de identificación de los vehículos.
- **Sección azul:** Muestra las imágenes en las que se ha detectado movimiento.
- **Sección verde:** Muestra porcentualmente el progreso del vídeo.
- **Sección amarilla:** Botón para seleccionar el video que se quiera clasificar, almacenado en el ordenador.
- **Sección morada:** Botón para parar la ejecución de la clasificación y el video en cualquier punto del proceso.
- **Sección blanca:** Selecciona el tipo de red que se quiere clasificar.

#### 4.1.3 ThingSpeak

Al pulsar en el botón *THING SPEAK*, aparecerá una ventana como la mostrada en la Figura 4.7, dónde se señalan las diferentes secciones que se pueden encontrar.

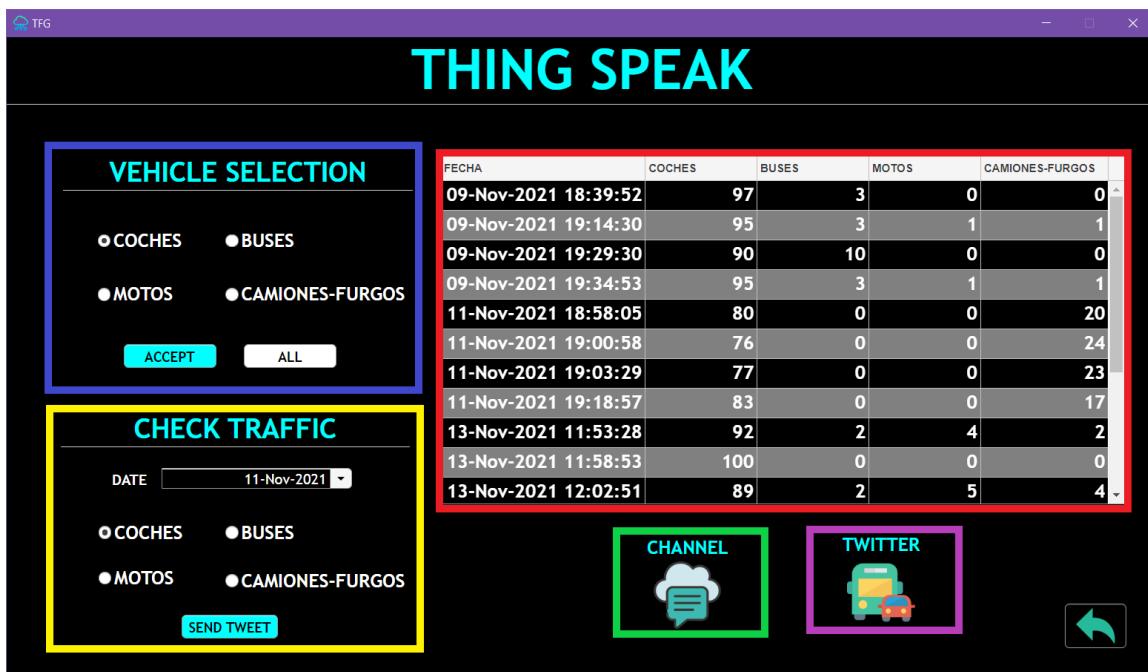


Figura 4.7 ThingSpeak Seleccionado

- **Sección roja:** Tabla con la fecha y la hora en la que se realizó la clasificación de los vehículos y el porcentaje correspondiente al flujo de tráfico de cada uno de ellos en ese momento.
- **Sección azul:** Selección de vehículos para filtrar los datos que se muestran en la tabla.
- **Sección amarilla:** Zona de consulta del tráfico por medio de *Twitter*. A través de un *tweet* informa del porcentaje de tráfico que tuvo el vehículo elegido el día seleccionado en el calendario. En el caso de que no se haya realizado ninguna clasificación el día propuesto, se lanzará un *tweet* con el siguiente mensaje: “No hay flujo de tráfico en el día <fecha seleccionada>”. En caso contrario el mensaje será: “El flujo de tráfico de <vehículo seleccionado> en el día <fecha seleccionada> es del <porcentaje>”.
- **Sección verde:** Botón que accede al canal de *ThingSpeak* y, por el cual, se pueden observar las gráficas de progreso anteriormente descritas en el punto 3.1.2.
- **Sección morada:** Botón para acceder a la cuenta de *Twitter* que está asociada a *ThingSpeak*.

## 4.2 Entrenamiento de las redes

Para el aprendizaje de las redes, se han llevado a cabo dos tipos de entrenamiento, procedentes de dos videos distintos. Primeramente, se ha utilizado el modelo de red *AlexNet*, y posteriormente *GoogleNet*.

En la interfaz de entrenamiento aparecen una serie de parámetros. Como se puede observar en la figura 4.5, se aprecia la interfaz de entrenamiento en el caso de *AlexNet*. Por defecto, se proporcionan unos valores iniciales, los cuales pueden ser modificados, a excepción de *validationPatience*, el cual siempre va a tener un valor fijo igual a infinito. Además, los parámetros de la sección de la derecha permiten mostrar los diferentes diagramas analíticos utilizados en el proceso.

El número de entrenamientos realizados se divide dependiendo de los parámetros utilizados. En primer lugar, se realiza un entrenamiento basado en los valores por defecto, tal y como se puede apreciar en la figura 4.8. En el caso de *AlexNet*, se observa un alto porcentaje de precisión, en este caso de un 96,97%. El tiempo de realización de todo el entrenamiento ha sido de 19 minutos y 23 segundos. El número de iteraciones totales han sido 300, dado que cada *epoch* ha realizado 30 iteraciones. Cabe destacar una pequeña anomalía en la función de pérdida (*loss*) al comienzo del tercer *epoch*, dado que aumenta casi a un valor de 1. En la parte superior de esta figura se muestra la evolución de la precisión a lo largo del entrenamiento, observándose cómo en la primera *epoch* se alcanzan niveles altos de precisión, oscilando en torno al 95% y con un alto nivel de estabilidad. Estos porcentajes son deseables, ya que el ideal sería 100%. En la parte inferior se muestra la evolución de la función de pérdida o *loss* que decrece con cierta rapidez a partir también de la primera *epoch* tendiendo hacia el valor cero, que es lo ideal en los entrenamientos, ya que tal y como se ha indicado previamente, el objetivo es minimizar la función de pérdida.

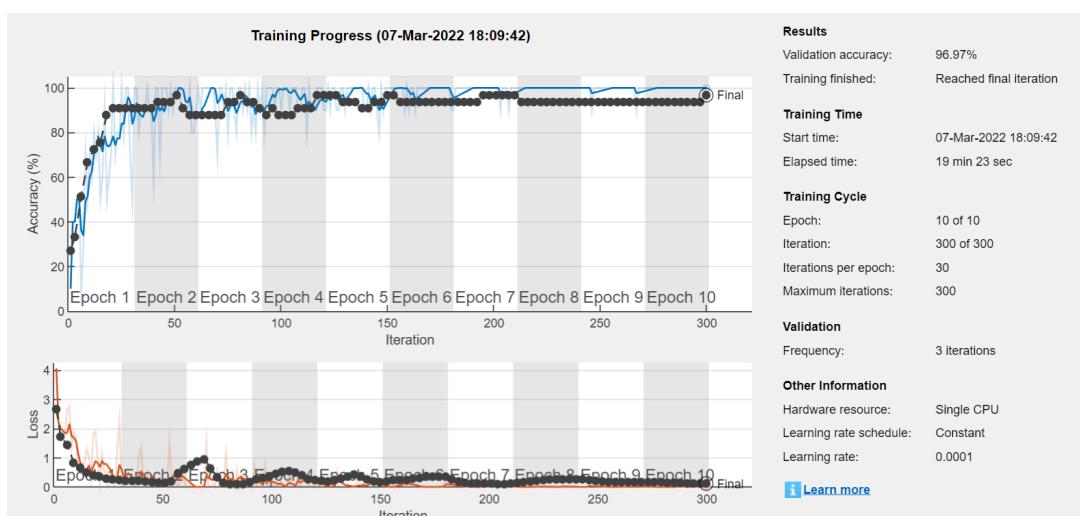
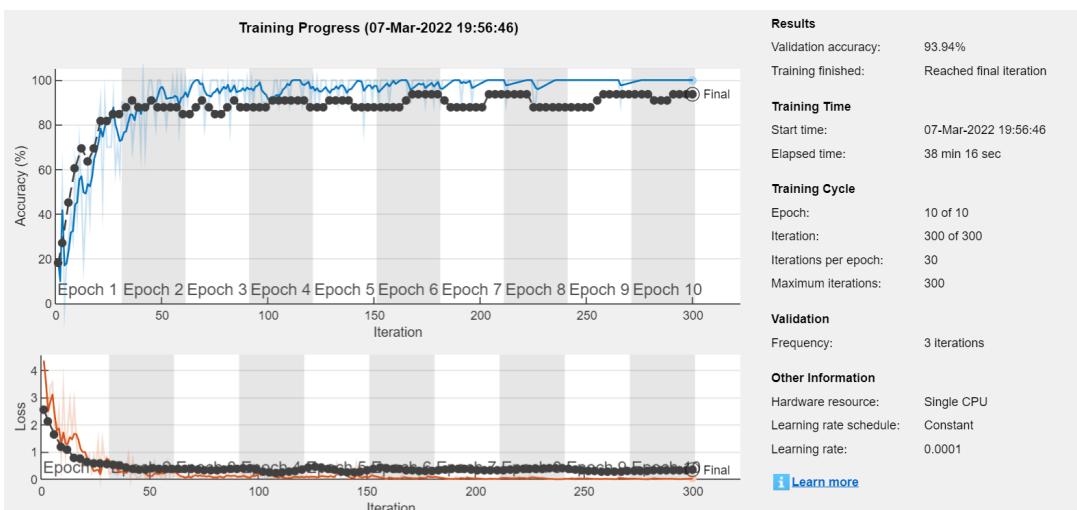


Figura 4.8 Training AlexNet, valores por defecto

Sin embargo, y como se puede observar en la figura 4.9, el entrenamiento basado en *GoogleNet* no presenta anomalías en la función de pérdida. El número de iteraciones es el mismo que en *AlexNet*, pero el tiempo empleado fue prácticamente el doble, con un total de 38 minutos y 16 segundos. Incluso habiendo resultado un tiempo mayor, hemos obtenido un porcentaje de precisión en la validación menor, concretamente un 93,94%. Estos datos plasman una conclusión rotunda, *AlexNet* proporciona, en un menor tiempo, una mayor precisión. Tanto la precisión como la función de pérdida evolucionan a valores aceptables a partir de la primera epoch con algunas oscilaciones a lo largo del proceso.



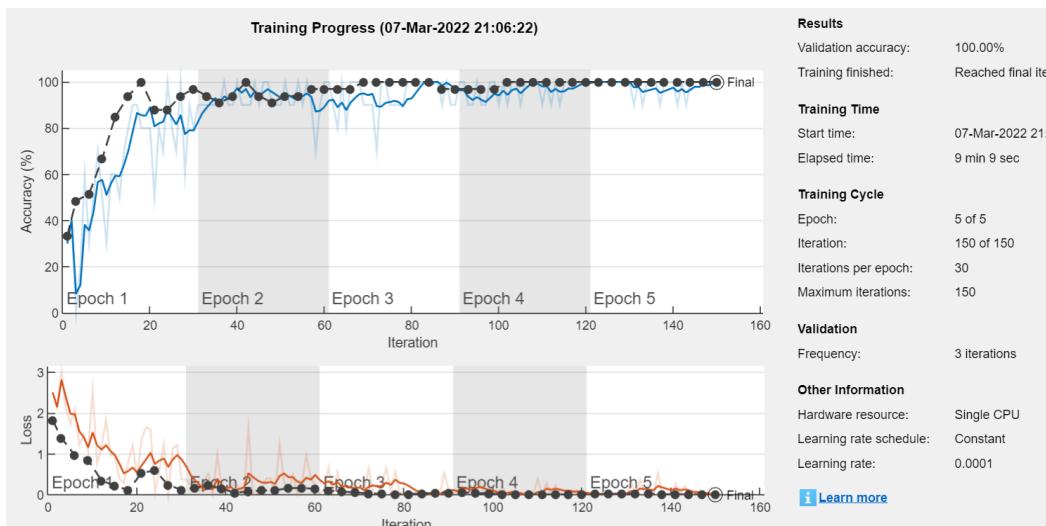
**Figura 4.9 Training GoogleNet, valores por defecto**

Posteriormente a los resultados obtenidos con los valores de los parámetros y opciones por defecto, se procede a modificar dichos valores, de modo que se prioriza la velocidad por delante del alto porcentaje de aciertos. Se puede observar en la figura 4.10 este cambio de parámetros, que concretamente realiza la mitad de iteraciones, en total 150, y valida menos imágenes, en este caso 4 en vez de 6 como se tenía por defecto. El número de imágenes de entrenamiento por defecto era de 64, y se ha disminuido notablemente, hasta 10 imágenes.

MiniBatchSize	10
MaxEpochs	5
InitialLearnRate	0.0001
ValidationFrequency	3
ValidationPatience	Inf
numTrainImages	10
numValidationImages	4

**Figura 4.10 Valores priorizando la velocidad de entrenamiento**

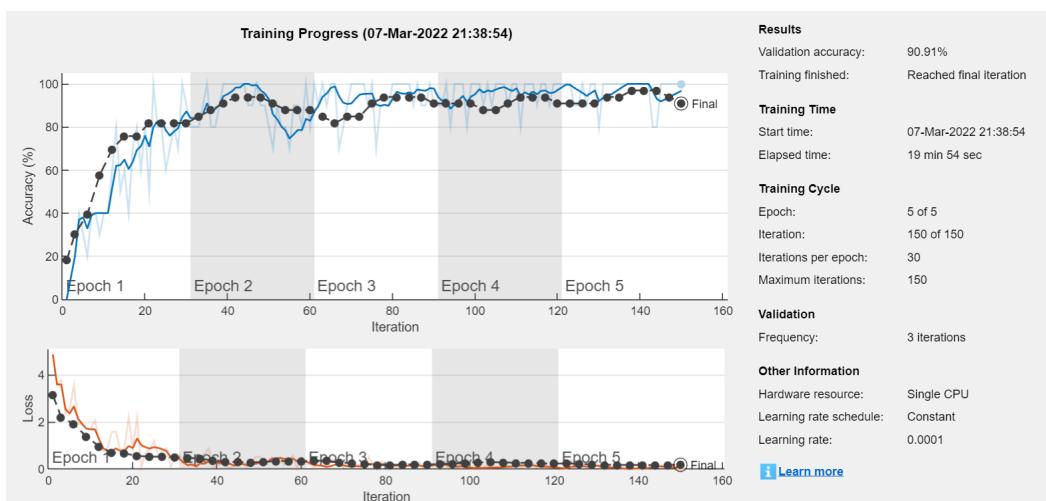
Con los resultados obtenidos en *AlexNet* (Figura 4.11), se puede resaltar la disminución de tiempo en la que se ha realizado con estos valores, de los 19 a los 9 minutos y 9 segundos exactamente.



**Figura 4.11 Training AlexNet, valores priorizando la velocidad de entrenamiento**

Sin embargo, el dato más curioso es el porcentaje de precisión. Se obtiene un 100%, y esto no se debe a la disminución del tiempo, sino a la del número de imágenes de entrenamiento. Este parámetro afecta al tiempo total, pero aún más, al porcentaje de validación de imágenes. En estos casos, la evolución de la precisión y la función de pérdida son apropiados.

En el caso de *GoogleNet* (Figura 4.12), se confirma la tendencia de que, comparada con *AlexNet*, el proceso de entrenamiento es más lento. El porcentaje de precisión, en este caso, se observa con una tendencia más a la baja, concretamente logrando un 90,91% y mayores oscilaciones durante su evolución en el proceso de entrenamiento.



**Figura 4.12 Training GoogleNet, valores priorizando la velocidad de entrenamiento**

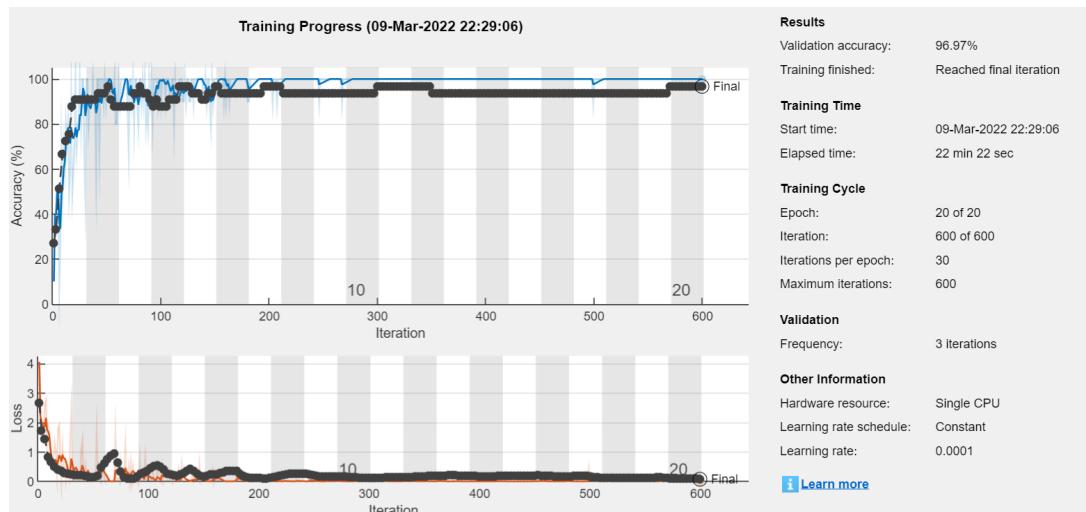
También cabe apreciar, en este caso, cómo la función de pérdida se encuentra prácticamente próxima a cero al finalizar el proceso. En conclusión, en *GoogleNet* obtenemos unos datos mínimamente diferenciados con respecto a los valores por defecto.

Por último, tratando de buscar un proceso de entrenamiento con unos resultados más precisos, se propone modificar los valores de los parámetros correspondientes, como se puede apreciar en la figura 4.13.

MiniBatchSize	10
MaxEpochs	20
InitialLearnRate	0.0001
ValidationFrequency	3
ValidationPatience	Inf
numTrainImages	64
numValidationImages	10

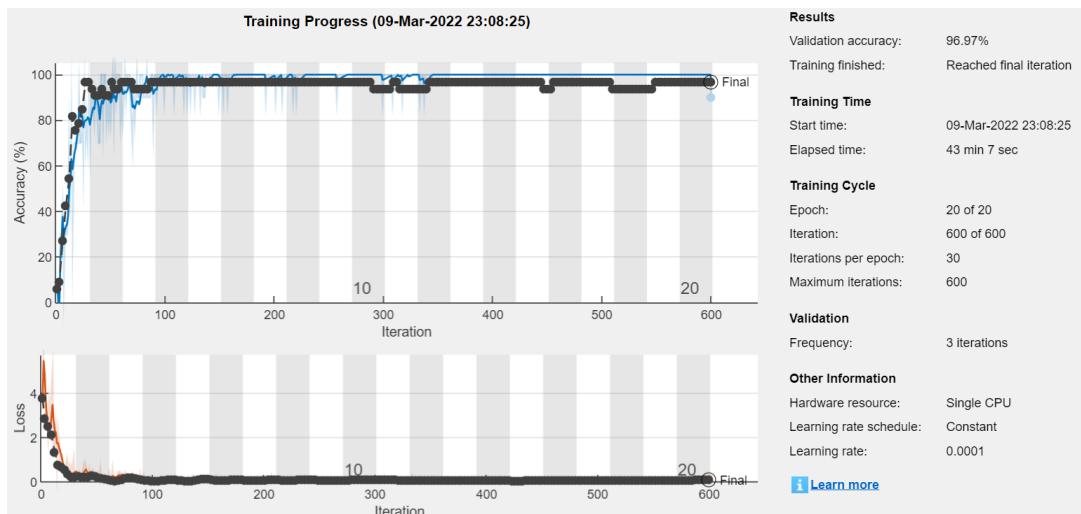
**Figura 4.13 Valores priorizando mejores resultados**

Se espera que al cambiar los mencionados valores se obtengan resultados más precisos. Sin embargo, al realizar el procedimiento de entrenamiento y como se puede comprobar en la figura 4.14, se obtiene un porcentaje de precisión del 96,97%. Comparado con los valores por defecto, permite concluir que el cambio de parámetros no aporta un beneficio lo suficientemente destacado como para obtener un resultado más óptimo y preciso, a pesar de realizar el doble de iteraciones, lo cual dictamina que llega un punto en el que un mayor número de iteraciones no beneficia el rendimiento obtenido en relación a la precisión de la solución.



**Figura 4.14 Training AlexNet, valores priorizando mejores resultados**

Por otro lado, conviene destacar el tiempo de entrenamiento con *GoogleNet* reflejado en la figura 4.15, el cual es exactamente de 43 minutos y 7 segundos. Se consigue una validez ligeramente superior a la que se obtuvo anteriormente. Aunque, dado que el porcentaje de precisión no ha sido especialmente significativo, queda como conclusión que, tanto en *AlexNet* como en *GoogleNet*, no se consigue la efectividad esperada, y por tanto los valores por defecto resultan ser los más óptimos.



**Figura 4.15 Training GoogleNet, valores priorizando mejores resultados**

Por último, cabe destacar también algunos de los diagramas que se obtienen durante el entrenamiento y que se pueden visualizar en el Anexo 1.

- **Network Analyzer:** proporciona la arquitectura de un modelo de red, comprueba que se ha definido la arquitectura correctamente, y detecta problemas antes de iniciarse el entrenamiento.
- **Validation images:** proporciona de forma visual los diferentes tipos de imágenes, que se han obtenido a partir del número de imágenes entrenadas. El número de diferentes tipos de imágenes se ajusta a partir del parámetro *numValidationImages*.
- **Weights:** Son parámetros de especial relevancia, ya que dentro de la red neuronal representa los valores aprendidos tras el entrenamiento. Estos pesos, en realidad son tensores de distintas dimensiones según cada modelo de red y capa, tal y como se muestra en la tabla 2.1. Transforman los datos de entrada a cada capa oculta de la red. En el diagrama se muestran los diferentes pesos obtenidos durante la realización del entrenamiento en la primera capa de aprendizaje de la red *Alexnet*. En caso de que un valor de peso sea nulo, se visualiza con una tonalidad en negro, por el contrario tonalidades más claras representan valores de pesos más altos.

## 4.3 Detección de vehículos

La detección de vehículos utilizando las redes *AlexNet* y *GoogleNet* depende de varios factores, entre los que se incluye el nivel de entrenamiento previo y los algoritmos utilizados. En general, la detección de vehículos y su clasificación durante todo el proceso se realiza de forma aceptable, como se analiza seguidamente.

Debido al sistema de detección utilizado, que se detalla posteriormente, el conteo de vehículos que se realiza no es totalmente exacto, pero sí realiza una estimación aproximada.

En la imagen mostrada en la figura 4.16 se puede observar la interfaz con un ejemplo ilustrativo del proceso de clasificación, donde se localizan los distintos tipos de vehículos utilizando recuadros de colores:

- **Recuadro negro** para los camiones o las furgonetas.
- **Recuadro azul** para la parte frontal de los coches.
- **Recuadro rojo** para la parte trasera de los coches.
- **Recuadro amarillo** para los autobuses.
- **Recuadro verde** para las motos.



Figura 4.16 Clasificación correcta de GoogleNet

En este caso, se puede ver que la aplicación ha detectado correctamente los tres vehículos que aparecen (un autobús y dos coches). También se puede observar el vehículo que se encuentra analizando en ese momento, la red seleccionada que está siendo utilizada (*GoogleNet*) y el progreso de la clasificación del vídeo.

## 4.4 Resultados de clasificación

Para comprobar el funcionamiento de la clasificación y el conteo, se procedió a analizar tres vídeos distintos grabados en dos zonas de Madrid (Moncloa y Villaverde). Los tres vídeos fueron clasificados con *AlexNet* y *GoogleNet*, entrenadas con los valores predeterminados.

El resultado de estas clasificaciones se puede observar en las siguientes tablas. En la primera se muestra el conteo de los vehículos y en la segunda los porcentajes. En ambos casos se tiene en cuenta el resultado real y el obtenido al clasificar con ambas redes.

- Moncloa (13:00)

CONTEOS								
Tipo	Coches		Buses		Motos		Camiones-furgo	
	Real	Detectado	Real	Detectado	Real	Detectado	Real	Detectado
AlexNet	28	1482	3	112	2	184	2	195
GoogleNet		1285		140		118		311

**Tabla 4.1 - Datos del conteo de la cámara situada en Moncloa a las 13:00**

PORCENTAJES								
Tipo	Coches		Buses		Motos		Camiones-furgo	
	Real	Detectado	Real	Detectado	Real	Detectado	Real	Detectado
AlexNet	80%	75%	9%	6%	6%	9%	6%	10%
GoogleNet		69%		8%		6%		17%

**Tabla 4.2 - Datos de los porcentajes de la cámara situada en Moncloa a las 13:00**

- Moncloa (17:00)

CONTEOS								
Tipo	Coches		Buses		Motos		Camiones-furgo	
	Real	Detectado	Real	Detectado	Real	Detectado	Real	Detectado
AlexNet	21	1133	1	82	0	15	0	133
GoogleNet		1014		47		72		156

**Tabla 4.3 - Datos del conteo de la cámara situada en Moncloa a las 17:00**

PORCENTAJES								
Tipo	Coches		Buses		Motos		Camiones-furgo	
	Real	Detectado	Real	Detectado	Real	Detectado	Real	Detectado
AlexNet	95%	83%	5%	6%	0%	1%	0%	10%
GoogleNet		79%		4%		6%		12%

**Tabla 4.4 - Datos de los porcentajes de la cámara situada en Moncloa a las 17:00**

- Villaverde (13:00)

CONTEOS								
Tipo	Coches		Buses		Motos		Camiones-furgo	
	Real	Detectado	Real	Detectado	Real	Detectado	Real	Detectado
AlexNet	59	1477	0	12	0	19	4	159
GoogleNet		1374		37		2		238

**Tabla 4.5 - Datos del conteo de la cámara situada en Villaverde a las 13:00**

PORCENTAJES								
Tipo	Coches		Buses		Motos		Camiones-furgo	
	Real	Detectado	Real	Detectado	Real	Detectado	Real	Detectado
AlexNet	94%	89%	0%	1%	0%	1%	6%	10%
GoogleNet		83%		2%		1%		14%

**Tabla 4.6 - Datos de los porcentajes de la cámara situada en Villaverde a las 13:00**

En todos los casos el conteo de la aplicación es notablemente superior al conteo real de vehículos. Esto se debe al diseño del algoritmo de conteo que será explicado en el siguiente punto. Sin embargo, los datos de los porcentajes son más fieles a la realidad.

Cabe mencionar que las redes de *AlexNet* y *GoogleNet* han sido entrenadas con imágenes de entrenamiento similares, y por tanto los resultados de ambas son parecidos entre sí.

En ocasiones, el algoritmo detecta un tipo de vehículo cuando no aparece en el vídeo. Esto se debe a que, durante el transcurso de la clasificación, el algoritmo capta el mismo vehículo y lo clasifica de diferentes maneras a lo largo de su movimiento y, aunque luego lo vuelva a clasificar de forma correcta, el conteo erróneo se mantiene en los datos.

También mencionar el hecho de que en las zonas más alejadas, los vehículos son menos visibles y su imagen posee peor calidad, por lo que es más difícil realizar un análisis correcto. Además, en los bordes del vídeo, los vehículos quedan cortados y es muy probable que la información requerida por la red no sea suficiente y los clasifique de forma incorrecta al mostrarse parcialmente y no estar visibles en su totalidad.

En los siguientes puntos se explica el algoritmo de conteo y se detallan mejor los errores más frecuentes que se han encontrado.

## 4.5 Algoritmo de conteo

A la hora de aplicar el algoritmo desarrollado para la detección de vehículos y su diferenciación, se produce un fallo en la lógica del diseño. Y es que, durante la progresión de las imágenes en el video, el algoritmo cuenta y suma los vehículos que localiza *frame a frame* según su tipo. Por lo que, si el mismo vehículo vuelve a aparecer en el siguiente *frame*, el procedimiento lo vuelve a contabilizar de nuevo y realiza la suma de un mismo vehículo varias veces consecutivas. Esto provoca que se obtengan datos incorrectos e imprecisos a la hora de realizar la clasificación.

Un ejemplo de esta falla se puede observar en la figura 4.17 dónde se muestran dos *frames* sucesivos y la identificación de los mismos vehículos para contabilizar.

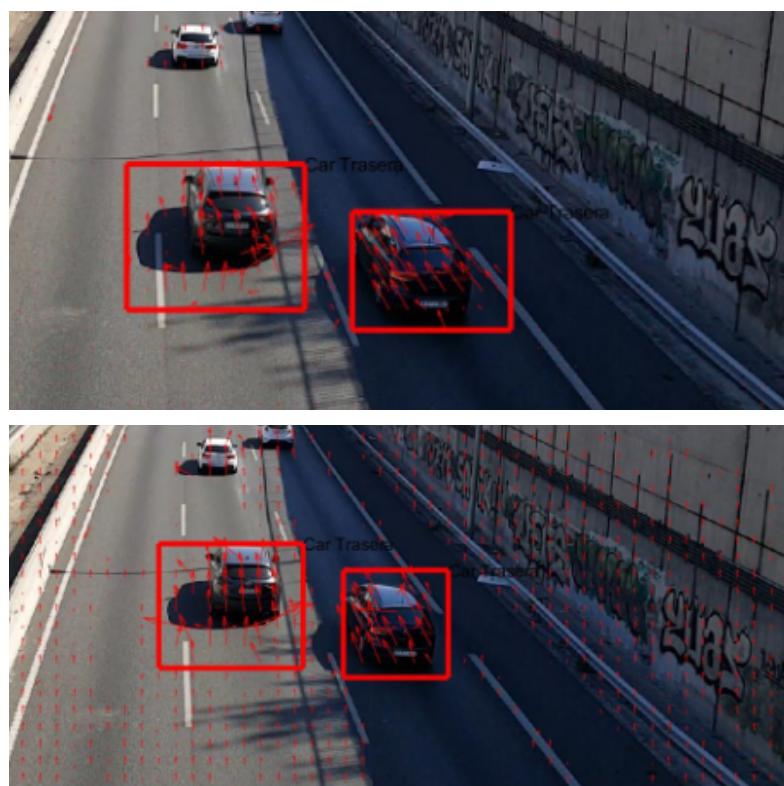


Figura 4.17 Imágenes de dos frames consecutivos donde se demuestra el error del conteo

Como solución a este problema, se desarrolló un nuevo procedimiento para contabilizar los vehículos a través de porcentajes. De esta manera, aunque el conteo se repitiera, el porcentaje seguiría siendo fiel al total de vehículos contabilizados y se obtendría un resultado más cercano al real.

En el desarrollo del procedimiento, se tienen en cuenta varias variables de conteo:

- vehículos: contabiliza el total de vehículos que aparecen en el video, sin tener en cuenta su tipo.
- coches: contabiliza el total de los coches, sumando las partes delanteras y traseras.
- buses: contabiliza el total de autobuses.
- motos: contabiliza el total de motos.
- camionesFurgos: contabiliza el total de furgonetas y camiones en conjunto.

Todas ellas inicializadas por defecto a cero como se muestra en la figura 4.18.

```
vehiculos = 0; coches = 0; buses = 0; motos = 0; camionesFurgos = 0;
```

**Figura 4.18 Variables de conteo**

Después de realizar la clasificación, se calcula el porcentaje total de cada tipo de vehículo considerando las sumas obtenidas en cada variable de conteo. Tal y como se muestra en la figura 4.19.

```
coches = round(coches / vehiculos * 100, 0);
buses = round(buses / vehiculos * 100, 0);
motos = round(motos / vehiculos * 100, 0);
camionesFurgos = round(camionesFurgos / vehiculos * 100, 0);
```

**Figura 4.19 Cálculo de porcentaje**

A pesar de que las variables de conteo alcanzan unos valores elevados debido a la suma de vehículos repetidos, el porcentaje consigue arreglar el desajuste y logra más precisión a la hora de cumplir el objetivo de controlar el flujo de los vehículos en el tráfico.

Sin embargo, esta manera de detección de vehículos generó otro problema. En el caso de que la red no se entrene perfectamente, es decir, no alcance el 100% de precisión, en algún *frame* del video se puede detectar de manera errónea el tipo de vehículo, provocando un error en el conteo. Por lo que, si este error se produce varias veces, se generarán pequeños desajustes en el porcentaje.

Esta anomalía es algo difícil de corregir teniendo en cuenta que en el procedimiento no se conoce internamente si se está detectando el vehículo incorrectamente y causa un cambio insignificante, considerando que el vehículo predominante siempre acabará teniendo el porcentaje mayor por mucho que haya algún error en la diferenciación.

## 4.6 Errores más frecuentes

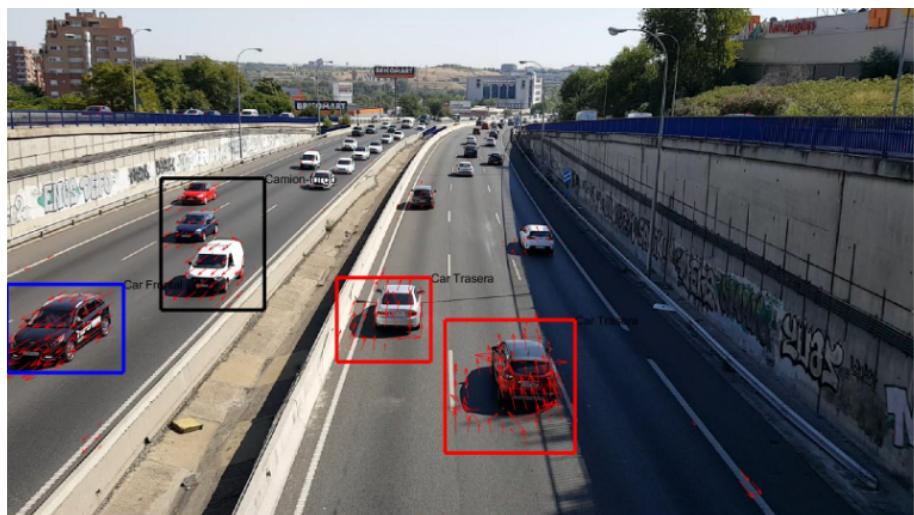
Los errores más frecuentes detectados en los videos durante la clasificación de los vehículos son:

- **Duplicación de vehículos por el algoritmo de conteo.**

Como se explicó en el punto anterior y se mostró en la figura 4.17, el método de conteo es incapaz de detectar si ya ha contabilizado un recuadro, por lo que se provoca una duplicación de vehículos en diferentes puntos del vídeo y las variables de conteo alcanzan valores demasiado elevados con respecto al número de vehículos reales.

- **Captura de dos o más vehículos como una sola unidad.**

En ocasiones, en la clasificación se seleccionan varios vehículos en un *frame* determinado y los clasifica en un mismo recuadro como pertenecientes al mismo tipo de vehículo. Por lo que un conjunto de vehículos puede ser catalogado como uno solo de mayor tamaño. Este error se puede observar en la figura 4.20 y se debe a un fallo de entrenamiento en la red.



**Figura 4.20 Error de una sola unidad**

Además, también afecta al problema de la duplicidad anteriormente mencionado, pues se realizará una detección de tipo de vehículo errónea y el conteo se realizará con imprecisiones.

- **Detección errónea de varios elementos del video.**

Durante el procesamiento del video se detectan elementos que no se asemejan con lo que son en realidad. Por ejemplo, podemos ver cómo un coche puede clasificarse como un camión durante varios *frames* en la figura 4.21.

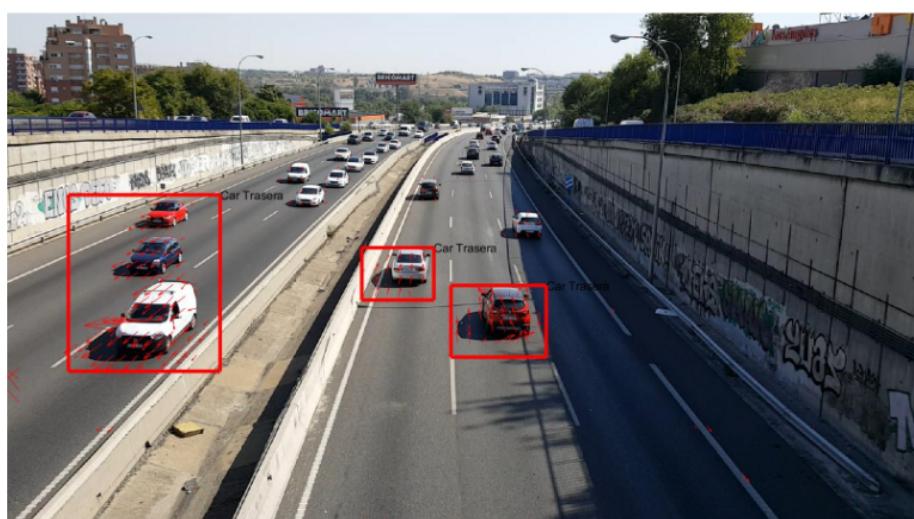
Este problema se genera tras realizar entrenamientos con baja precisión y también potencia el error de la duplicidad.



*Figura 4.21 Error en la captura de elementos*

- **Confusión entre la parte trasera y delantera en la identificación de un coche.**

Esta anomalía se debe principalmente a confundir la detección de vehículos de la parte trasera con la de la parte delantera y viceversa. Sin embargo, al tratar las dos partes como una sola, no supone un gran problema a la hora de contabilizar los coches. En la figura 4.22 vemos dos ejemplos de este caso.



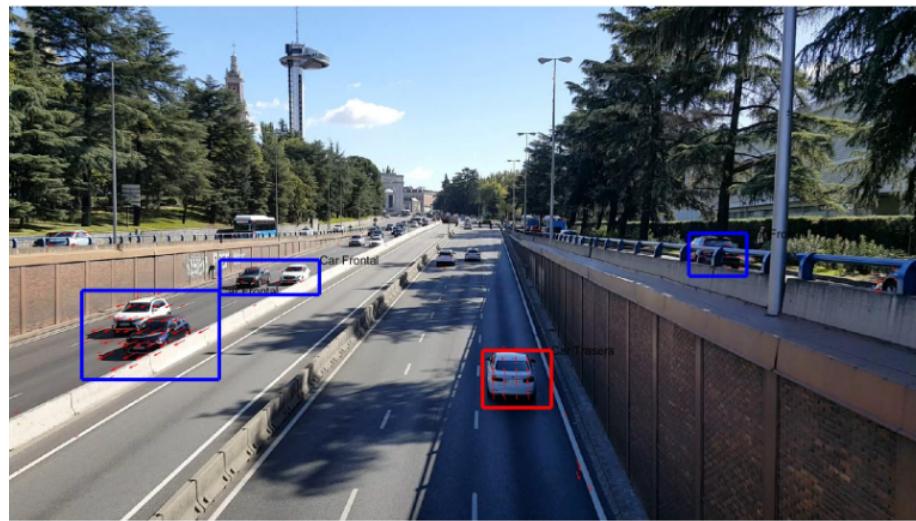


Figura 4.22 Error en la identificación de las partes del coche

- **Confusión entre una sombra y un vehículo.**

La sombra de varios vehículos puede alterar al algoritmo y provocar confusiones al clasificar. Esto se debe a que la sombra también se encuentra en movimiento asociada a un vehículo y moviéndose a la misma velocidad, por lo que puede ser clasificada como cualquier vehículo o cambiar el tipo de vehículo detectado por uno más grande. Este último caso se puede apreciar en la figura 4.21, anteriormente mostrada en la detección errónea de elementos.

- **Reconocimiento de motos como otro elemento del entorno.**

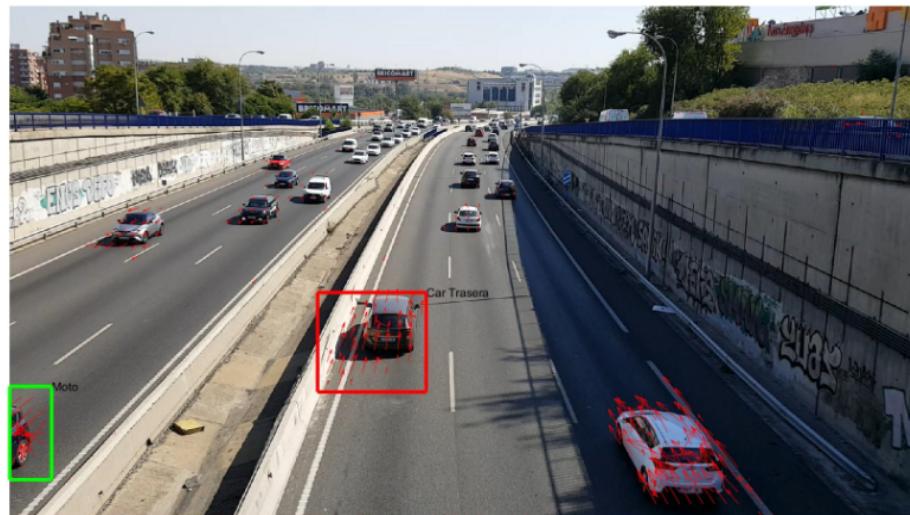
Al ser vehículos relativamente pequeños, comparados con los otros, en movimiento, el algoritmo tiende a relacionar las motos con objetos o líneas de menor tamaño a la hora de clasificarlas, como pasa en la figura 4.23.



Figura 4.23 Error al reconocer motos

- Reconocimiento del tipo de vehículo en los bordes de cada frame.

Puntualmente, cuando el algoritmo reconoce un vehículo cortado por los bordes de la imagen, tiende a relacionarlo con otro tipo diferente al que es en realidad. Normalmente, suele clasificarlo como una moto, como se muestra en la figura 4.24, pues es el vehículo que menor tamaño posee.



*Figura 4.24 Error de reconocimiento de vehículos en los bordes*

## 4.7 Integración de los datos y resultados en ThingSpeak

Al finalizar la detección de vehículos, la aplicación dispone de la interfaz *ThingSpeak*, la cual muestra las diferentes opciones en las que se pueden filtrar los resultados obtenidos. Se puede observar, además, el registro histórico de los resultados que se han ido obteniendo en cada fase de clasificación de vehículos. La consulta de este registro se puede realizar también en *ThingSpeak*, el cual muestra, de forma gráfica, el porcentaje de cada vehículo y la relación histórica con los demás resultados. Estas gráficas están directamente relacionadas con los datos de la tabla de la interfaz, y disponer de esta clasificación de los datos posibilita un análisis visual beneficioso y eficaz.

Mediante el botón *SEND TWEET*, se envían los datos determinados, y se genera la publicación de los resultados en la red social (*Twitter*). Todo ello se realiza en remoto dentro de la plataforma *ThingSpeak*, la cual recoge estos datos y los publica en la cuenta asociada.

Name	Message	Last Sent	Twitter Account
Aviso afluencia de coches	Gran afluencia de coches	2022-03-17 17:40	RecVehiculosTFG

*Figura 4.25 Tweet al obtener una afluencia de coches mayor al 70%*

Se ha modificado además el tipo de mensajes que se pueden difundir a través de *Twitter* y, para ello, se ha utilizado una app que proporciona *ThingSpeak*. Esta app, llamada *React*, genera una condición, que si se cumple, la cuenta asociada a *Twitter* publicará automáticamente un mensaje. Como se puede observar en la figura 4.25, la interfaz de *ThingTweet* muestra el mensaje que se ha publicado tras haber obtenido un porcentaje de coches mayor a un 70%.

## 5. Conclusiones y trabajo futuro

### 5.1 Conclusiones

En el presente trabajo se ha desarrollado una aplicación para la identificación y detección de diferentes tipos de vehículos en movimiento en tránsito en vías urbanas a partir de imágenes en videos previamente grabados. Para ello, se han utilizado diversas tecnologías en el ámbito de la Inteligencia Artificial y más concretamente relacionadas con la Visión por Computador (VC). Para ello se han aplicado estrategias específicas basadas en técnicas de Aprendizaje Profundo (AP) para entrenar Redes Neuronales Convolucionales (RNC) y ejecutarlas utilizando Matlab como herramienta base para ello. Además, se ha extendido su uso para incluir funcionalidades específicas de IoT.

El objetivo principal ha sido el desarrollo de una interfaz gráfica, intuitiva y vistosa, con la que cualquier usuario pueda interactuar de manera sencilla, la cual integra y contempla el entrenamiento de las RNC, la realización de clasificaciones de vehículos según su tipo y el almacenamiento de los datos obtenidos a través de la plataforma *ThingSpeak*, específica de IoT.

Como ya se ha mencionado previamente, Matlab, a través de sus correspondientes *toolboxes* proporciona las funcionalidades necesarias para implementar los conceptos de VC y AP y utiliza, en este caso, los modelos de redes *AlexNet* y *GoogleNet*. Las dos están pre-entrenadas y se re-entrenan a través de un conjunto de imágenes y una selección de parámetros, que se ajustan de la forma más precisa posible, con el objetivo de alcanzar la mayor tasa de acierto.

El entrenamiento consiste principalmente en el aprendizaje de los pesos involucrados en ambas redes para la identificación de varios vehículos seleccionados: camiones, buses, motos, coches por su parte delantera y coches por su parte trasera. Cabe mencionar que los datos recopilados sobre los coches se almacenan en *ThingSpeak* de manera única, sumando ambas partes.

Además, el usuario tiene acceso a las carpetas que contienen el conjunto de imágenes utilizadas para el entrenamiento, pudiendo añadir más, si lo desea, y redimensionarlas para su correcto funcionamiento.

La aplicación extrae la información necesaria a partir de los datos (imágenes) mediante el entrenamiento realizado, utilizando las mencionadas técnicas de AP. Para ello, se basa en una serie de criterios previamente establecidos por el usuario y obtiene como resultado una mayor o menor tasa de acierto. Tras los diversos experimentos realizados en el presente trabajo, tomando en conjunto los datos adquiridos, y reflejados en la memoria se concluye que la red *AlexNet* proporciona una tasa de acierto mejor que la de *GoogleNet* en relación al tiempo de entrenamiento.

En cuanto a la estrategia de detección de vehículos y su clasificación, se utiliza una combinación de técnicas de VC basadas en la detección de flujo óptico y la segmentación de regiones, que permiten captar las imágenes en movimiento (*frame* a *frame*) dentro de una secuencia de vídeo. Tras la selección de vehículos, mediante el uso de un método de diferenciación de cada vehículo tras su paso por una línea concreta de cada *frame*, se ha diseñado un algoritmo de conteo para identificar la cantidad porcentual de vehículos de cada tipo que transcurren por el tráfico y que conduce a una solución bastante aceptable con respecto a la realidad de monitorización del flujo de vehículos en vías de circulación.

Para el control de la clasificación, se han realizado diferentes pruebas, capturando vídeos en distintos puntos de vías urbanas de Madrid (Moncloa y Villaverde). De esta manera, se puede comprobar el grado de acierto que posee la clasificación de vehículos y el algoritmo de conteo aplicado. Observando que, normalmente, la detección de vehículos es bastante aceptable en ambas redes, a pesar de que su identificación y su conteo generaron ciertos problemas, que se deberán tener en cuenta para futuras mejoras de la aplicación, como se indica posteriormente.

Todos los resultados de los análisis se guardan cronológicamente en la plataforma pública *ThingSpeak*, utilizando la idea de almacenamiento en la nube y quedando a disposición de cualquiera usuario que quiera realizar una consulta sobre ellos. Además, a través de esta aplicación, el usuario podrá realizar diferentes consultas, teniendo la opción de elegir el día y el tipo de vehículo del que quiere saber el porcentaje de afluencia en el tráfico. Asimismo, todos los resultados obtenidos son expuestos y comunicados a través de la red social *Twitter*, que tiene conexión con *ThingSpeak* y, por tanto, como una utilidad adicional en IoT.

A raíz de las pruebas realizadas, se concluye con carácter general, que la aplicación se encuentra integrada y funciona de forma apropiada según el objetivo planteado inicialmente.

## 5.2 Trabajo futuro

Como continuación al desarrollo de este proyecto, han surgido algunas ideas venideras que pueden ser proyectadas en un futuro. Entre las posibles, se destacan:

- Tener en cuenta a los vehículos contrarios al sentido de la marcha que no sean automóviles, puesto que, al realizar el porcentaje de vehículos totales, se contabiliza como coches las detecciones tanto de la parte delantera como la trasera, y por parte de camiones, motos y autobuses no se aplica este criterio.
- Incorporación de videos en directo, pudiendo realizar cualquier tipo de estudio en situaciones reales y que sean más útiles para su aplicación en entornos reales y por ende a la sociedad, dentro del desarrollo de ciudades sostenibles.
- Mejorar la interfaz incorporando más modelos de RNC, como pueden ser entre otros, algunos de los existentes con carácter general y los descritos en Pajares y col. (2021), tales como ResNet, VGG19, MobileNet, para dar al usuario otras posibilidades con las que trabajar que no sean sólo *AlexNet* y *GoogleNet*.
- Llevar un detalle del monitoreo de las visitas y razones por las cuales los usuarios consultaron la aplicación como procedimiento de retroalimentación para enriquecer la información y la funcionalidad.
- Mejora de la velocidad de respuesta a la hora de realizar el entrenamiento de imágenes y la clasificación del video seleccionado, explorando alternativas para aumentar su rapidez, entre otras su implementación en sistemas de altas prestaciones o al menos con uso de GPUs.
- Desarrollar otro tipo de interacciones para el usuario con respecto a las publicaciones en redes sociales, aplicando más información en ellas y utilizando otras plataformas para su notificación, como puede ser el correo electrónico.
- Mejorar los entrenamientos de las redes neuronales aportando más imágenes y ampliando el número de entrenamientos realizados para encontrar el mejor equilibrio entre el grado de precisión, eficacia y tiempo.
- Incorporar más elementos a la base de datos, para detectar otros vehículos de asistencia tales como ambulancias o coches de policía, aumentando las predicciones y análisis de los datos obtenidos.

- Incluir otros elementos del espacio donde se mueven los vehículos, como señales o semáforos, bastante útiles para controlar el tráfico y la detección de posibles atascos. En este sentido, se podrían incorporar detectores de objetos en el ámbito de las RNC, tales como R-CNN, Mask R-CNN y otros (Pajares y col., 2021).
- Corregir el problema de las “sombra”, para evitar que su movimiento sea detectado como un vehículo o un vehículo de distinta naturaleza por la asociación entre vehículo y sombra.
- Tener en cuenta los diferentes estados meteorológicos que se pueden encontrar en los videos, haciendo que la clasificación no se vea afectada a pesar de la presencia de fenómenos meteorológicos lluvia, nieve o derivados de la propia evolución del día como sol intenso o situaciones de baja iluminación por la noche.
- Ajustar el algoritmo de conteo, estudiando una forma más eficaz de definir la franja de detección de vehículos en la imagen para evitar la duplicidad de los vehículos.
- Aumentar el número de vídeos grabados, aportando más localizaciones para obtener una base de datos más extensa y afinar el grado de acierto en la clasificación de vehículos.
- Implementar un método de predicción ya que, con una buena base de datos consolidada, sería posible predecir el flujo de tráfico aproximado de una localización concreta. En este sentido se podría proponer la utilización de modelos tales como Long Short-Term Memories (Pajares y col., 2021).
- Consulta de datos y clasificación mediante un dispositivo móvil.

## 6. Conclusions and future work

### 6.1 Conclusions

In the present work, an application has been developed for the identification and detection of different types of vehicles transiting on urban roads from previously recorded video images. To do this, various technologies have been used in the field of Artificial Intelligence and more specifically related to Computer Vision (CV). Specific strategies based on Deep Learning (DL) techniques have been applied to train Convolutional Neural Networks (CNN) and execute them using Matlab as a base tool for it. In addition, its use has been extended to include specific IoT functionalities.

The main objective has been the development of a graphic, intuitive and showy interface, with which any user can interact in a simple way, which integrates and contemplates the RNC training, the vehicle classification according to their type and the data storage through the ThingSpeak platform, specific to IoT.

As previously mentioned, Matlab, through its corresponding toolboxes, provides the necessary functionalities to implement the CV and DL concepts and uses, in this case, the AlexNet and GoogleNet network models. Both are trained through a set of images and a selection of parameters, which are adjusted as precisely as possible, with the aim of achieving the highest success rate.

The training consists mainly of learning the weights involved in both networks to identify several selected vehicles: trucks, buses, motorcycles, cars from the front and cars from the rear. It is worth mentioning that the data collected about the cars is stored in ThingSpeak in a unique way, adding both parts.

In addition, the user has access to the folders that contain the set of images used for training, being able to add more, if desired, and resize them for proper functioning.

The application extracts the necessary information from the data (images) through the training carried out, using the mentioned DL techniques. To do this, it is based on a series of criteria previously established by the user and results in a higher or lower success rate. After the various experiments carried out in the present work, analyzing the data acquired and reflecting it in the memory, it is concluded that the AlexNet network provides a better success rate than GoogleNet in relation to the training time.

Regarding the vehicle detection strategy and its classification, a combination of CV techniques based on optical flow detection and region segmentation are used, which allow moving images to be captured (frame by frame) within a video sequence. After the vehicle selection, through the use of a differentiation method of each vehicle after passing through a specific line of each frame, a counting algorithm has been designed to identify the percentage amount of vehicles of each type that transit through the traffic and that leads to a quite acceptable solution with respect to the reality of monitoring the vehicle flow on roads.

To control the classification, different tests have been carried out, capturing videos at different points on urban roads in Madrid (Moncloa and Villaverde). This way, the success rate of the vehicle classification and the applied counting algorithm can be verified. Observing that, normally, the vehicle detection is quite acceptable in both networks, despite the fact that their identification and their counting generated certain problems, which should be taken into account for future improvements, as indicated later.

All the results are saved chronologically on the public platform ThingSpeak, using the idea of cloud storage and being available for any user who wants to consult them. In addition, through the application, the user will be able to make different queries, having the option to choose the day and the type of vehicle for which he wants to know the percentage of traffic flow. Likewise, all the results obtained are exposed and communicated through the social network Twitter, which is connected to ThingSpeak and, therefore, as an additional utility in IoT.

As a result of the tests, it is concluded generally speaking that the application is integrated and works properly according to the objective initially set.

## 6.2 Future work

Following the development of this project, some ideas have emerged that can be projected in the future. Among the possible ones, the following stand out:

- Take into account vehicles which go opposite to the direction of travel that are not automobiles, since, when calculating the percentage of total vehicles, cars are counted both the front and the rear but for trucks, motorcycles and buses this criterion does not apply.
- Incorporation of live videos, being able to perform any type of study in real situations and that are more useful for their application in real environments and therefore to society, within the development of sustainable cities.
- Improve the interface by incorporating more CNN models, such as, some of the existing ones in general and those described in Pajares y col. (2021), such as ResNet, VGG19, MobileNet, to give the user other possibilities to work with than just AlexNet and GoogleNet.
- Be elaborately monitoring the visits and reasons why users consulted the application as a feedback procedure to enrich the information and functionality.
- Response speed improvement when performing the image training and the classification of the selected video, exploring alternatives to increase it, for example its implementation in high-performance systems or at least with the use of GPUs.
- Develop other types of interactions for the user relating to social network publications, applying more information to them and using other platforms to notify, such as email.

- Improve neural network training by providing more images and expanding the number of trainings performed to find the best balance between the degree of accuracy, efficiency and time.
- Incorporate more elements to the database, to detect other assistance vehicles such as ambulances or police cars, increasing the predictions and analysis of the data obtained.
- Include other elements of the space, such as signs or traffic lights, which are quite useful for controlling traffic and detecting possible traffic jams. In this sense, object detectors could be incorporated in the field of CNNs, such as R-CNN, Mask R-CNN and others (Pajares y col., 2021).
- Fix “shadows” issue, to avoid its movement from being detected as a vehicle.
- Consider the different meteorological states that can be found in the videos, making sure that the classification is not affected despite the presence of weather events such as rain, snow or those derived from the evolution of the day itself, such as intense sun or low lighting situations at night.
- Adjust the counting algorithm, studying a more efficient way to define the vehicle detection zone in the image to avoid duplication of vehicles.
- Increase the number of recorded videos, providing more locations to obtain a more extensive database and improve the degree of accuracy in vehicle classification.
- Implement a prediction method since, with a good consolidated database, it would be possible to predict the approximate traffic flow of a specific location. In this sense, the use of models such as Long Short-Term Memories (Pajares y col., 2021) could be proposed.
- Data query and classification through a mobile device.

## 7. Bibliografía

1. Bishop, C.M. (2006). Pattern Recognition and Machine Learning, Springer, NY, USA.
2. BVLC GoogleNet Model (2022). Disponible on-line: [https://github.com/BVLC/caffe/tree/master/models/bvlc\\_googlenet](https://github.com/BVLC/caffe/tree/master/models/bvlc_googlenet) (Accedido Marzo 2022).

3. Farneback, G. (2003). Two-Frame Motion Estimation Based on Polynomial Expansion. In *Proceedings of the 13th Scandinavian Conference on Image Analysis*, 363 - 370. Halmstad, Sweden: SCIA.
4. Haralick, R.M., Shapiro, L.G. (1992). Computer and Robot Vision. Addison-Wesley, Boston.
5. Imagenet (2022). Disponible on-line: <http://www.image-net.org> (Accedido Diciembre 2021).
6. ImageNet LSVRC (2012). Large Scale Visual Recognition Challenge 2012 (ILSVRC2012). Disponible on-line: <http://www.image-net.org/challenges/LSVRC/2012/> (Accedido Marzo 2022).
7. Krizhevsky, A., Sutskever, I., Hinton, G.E. (2012). ImageNet Classification with Deep Convolutional Neural Networks. In Proc. 25th Int. Conf. on Neural Information Processing Systems (NIPS'12), vol. 1, pp. 1097-1105.
8. Levy, B., Gabriel, E. (2021). El liderazgo de Singapur como ciudad inteligente. Disponible on-line: <https://www.andinalinksmartcities.com/el-liderazgo-de-singapur-como-ciudad-inteligente/> (Accedido Enero 2022).
9. Low, D. (2021). The straits times. Singapore is world's smartest city for the third year: IMD Smart City Index. <https://www.straitstimes.com/tech/tech-news/singapore-is-worlds-smartest-city-for-the-third-year-imd-smart-city-index> (Accedido Marzo 2022).
10. Matlab (2022). MATLAB para inteligencia artificial. Disponible on-line: <https://es.mathworks.com/> (Accedido Enero 2022).
11. Pajares, G., Cruz, J.M. (2007). Visión por Computador: Imágenes digitales y aplicaciones. RA-MA, Madrid.
12. Pajares, G., Herrera, P.J, Besada, E. (2021). Aprendizaje Profundo. RC-Libros, Madrid.
13. Ruder, S. (2017). An overview of gradient descent optimization algorithms. arXiv:1609.04747v2 [cs.LG].

14. Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., Rabinovich, A. (2014). Going Deeper with Convolutions. Computing Research Repository. arXiv:1409.4842 [cs.CV].
15. ThingSpeak (2022). ThingSpeak for IoT Projects. Disponible on-line: <https://thingspeak.com/> (Accedido Enero 2022).
16. MathWorks, Analyzer Network, Analyze deep learning network architecture. Disponible on-line: <https://www.mathworks.com/help/deeplearning/ref/analyzenetwork.html> (Accedido Marzo 2022).
17. MathWorks, Redes Neuronales Convolucionales. Disponible on-line: <https://es.mathworks.com/discovery/convolutional-neural-network-matlab.html> (Accedido Marzo 2022).
18. Diseño de interfaz gráfica de Usuario con App Designer MATLAB. Disponible on-line: <https://youtu.be/ewbU48LB6QE>
19. Graficar en App Designer Matlab UIAxes. Disponible on-line: <https://youtu.be/ac1KGSE7D7U>
20. Check Box & Drop Down App Designer Matlab. Disponible on-line: [https://youtu.be/RH9mwD\\_tvZk](https://youtu.be/RH9mwD_tvZk)
21. Uso del Indicador Gauge App Designer Matlab. Disponible on-line: [https://youtu.be/6\\_UPbde454M](https://youtu.be/6_UPbde454M)
22. Cajas de texto en App Designer Matlab. Disponible on-line: <https://youtu.be/DXffSnaXJO8>
23. Metodología Scrumban. Disponible on-line: <https://openwebinars.net/blog/que-es-scrumban/>
24. Scrumban: la fusión de Kanban y Scrum. Disponible on-line: <https://blog.comparasoftware.com/scrumban-la-fusion-de-kanban-y-scrum/>
25. Las mejores metodologías ágiles. Disponible on-line: <https://www.inesdi.com/blog/que-son-las-metodologias-agiles-tipos-y-ejemplos/>
26. Tableros Kanban. Disponible on-line: <https://www.atlassian.com/es/agile/kanban/boards>

## 8. Anexos

En esta sección se incluyen una serie de anexos donde se describen los diferentes diagramas relacionados con el entrenamiento de las redes neuronales.

También se recopilan todas las licencias de las imágenes e iconos que se han utilizado para el diseño de la memoria y en el desarrollo de la aplicación.

Por último, se facilita un manual de usuario para la instalación y la ejecución de la aplicación.

### Anexo 1: Diagramas del entrenamiento

A continuación, se muestran los diferentes diagramas referidos al entrenamiento de las redes de *AlexNet* y *GoogleNet*.

- **Network Analyzer:** análisis de la arquitectura de un modelo de red para la detección de problemas antes del entrenamiento, Diagrama 1.

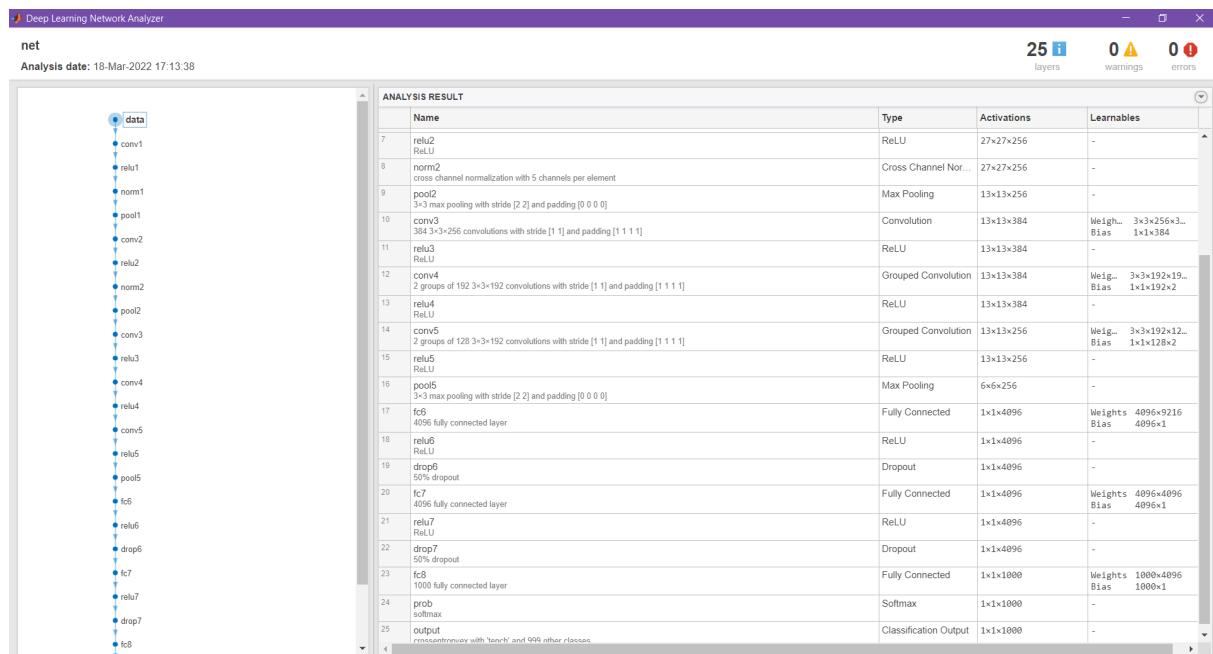
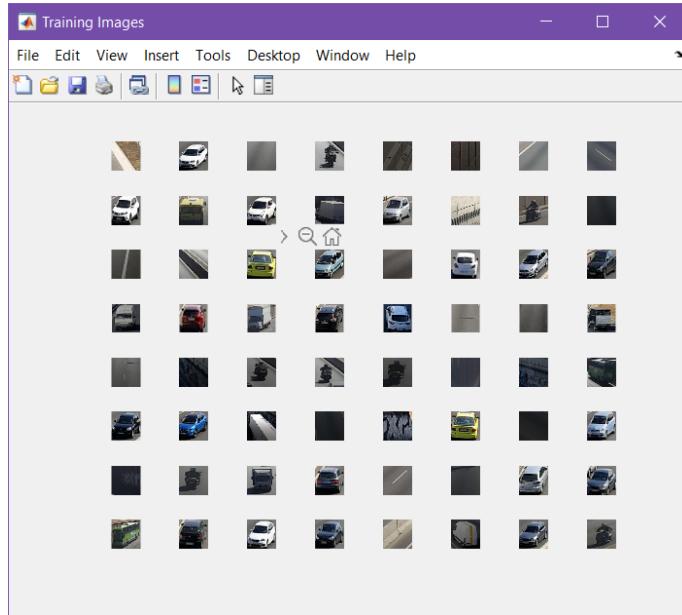


Diagrama 1. Network Analyzer

- **Training Images:** Conjunto de imágenes previamente seleccionadas para el entrenamiento, que se muestran como representantes del conjunto total, Diagrama 2.



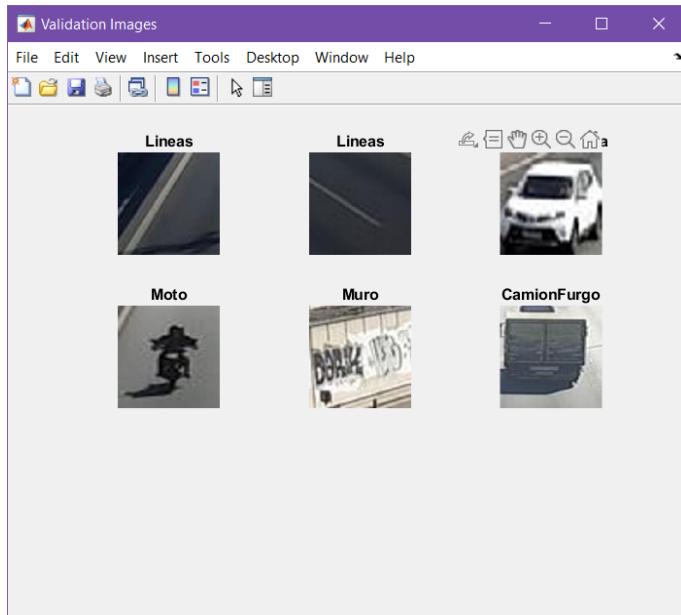
**Diagrama 2. Training Images**

- **Layers Analyzer:** análisis de las capas de un modelo de red para la detección de problemas antes del entrenamiento, a la vez que se muestra la estructura del modelo de red, Diagrama 3.

ANALYSIS RESULT				
	Name	Type	Activations	Learnables
1	data 227x227x3 images with 'zerocenter' normalization	Image Input	227x227x3	-
2	conv1 96 11x11x3 convolutions with stride [4 4] and padding [0 0 0]	Convolution	55x55x96	Weights: 11x11x3x96 Bias: 1x1x96
3	relu1 ReLU	ReLU	55x55x96	-
4	norm1 cross channel normalization with 5 channels per element	Cross Channel Nor.	55x55x96	-
5	pool1 3x3 max pooling with stride [2 2] and padding [0 0 0]	Max Pooling	27x27x96	-
6	conv2 2 groups of 128 5x5x48 convolutions with stride [1 1] and padding [2 2 2 2]	Grouped Convolution	27x27x256	Weights: 5x5x48x12... Bias: 1x1x128x2
7	relu2 ReLU	ReLU	27x27x256	-
8	norm2 cross channel normalization with 5 channels per element	Cross Channel Nor.	27x27x256	-
9	pool2 3x3 max pooling with stride [2 2] and padding [0 0 0]	Max Pooling	13x13x256	-
10	conv3 384 3x3x256 convolutions with stride [1 1] and padding [1 1 1 1]	Convolution	13x13x384	Weights: 3x3x256x3... Bias: 1x1x384
11	relu3 ReLU	ReLU	13x13x384	-
12	conv4 2 groups of 192 3x3x192 convolutions with stride [1 1] and padding [1 1 1 1]	Grouped Convolution	13x13x384	Weights: 3x3x192x19... Bias: 1x1x192x2
13	relu4 ReLU	ReLU	13x13x384	-
14	conv5 2 groups of 128 3x3x192 convolutions with stride [1 1] and padding [1 1 1 1]	Grouped Convolution	13x13x256	Weights: 3x3x192x12... Bias: 1x1x128x2
15	relu5 ReLU	ReLU	13x13x256	-
16	pool5 3x3 max pooling with stride [2 2] and padding [0 0 0]	Max Pooling	6x6x256	-
17	fc6 4096 fully connected layer	Fully Connected	1x1x4096	Weights: 4096x9216 Bias: 4096x1
18	relu6 ReLU	ReLU	1x1x4096	-
19	drop6 50% dropout	Dropout	1x1x4096	-

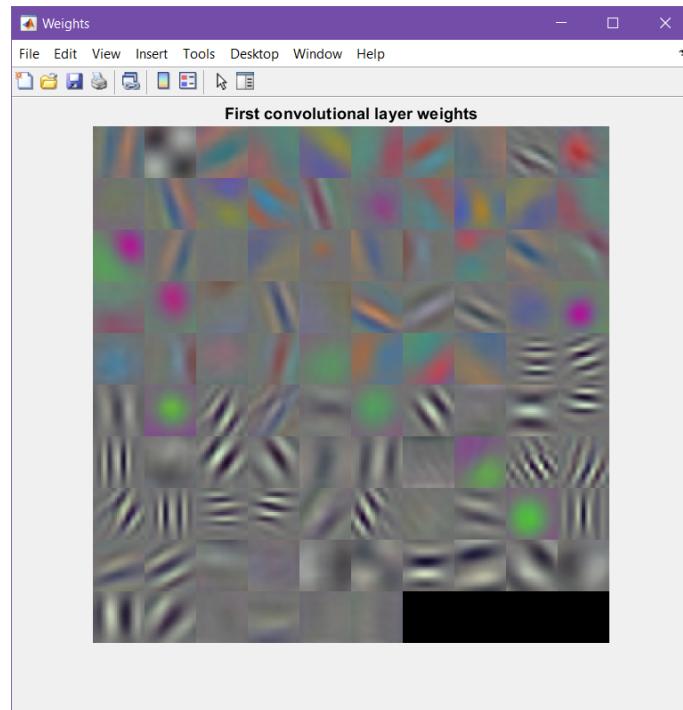
**Diagrama 3. Layers Analyzer**

- **Validation Images:** Conjunto de imágenes validadas y ordenadas según los tipos previamente definidos, Diagrama 4.



**Diagrama 4. Validation Images**

- **Weights:** Parámetros que representan dentro de la red neuronal los valores aprendidos tras el entrenamiento, Diagrama 5.



**Diagrama 5. Weights**

## Anexo 2: Licencias de los iconos de la aplicación

Los siguientes iconos, que se utilizan en la capa de presentación de la aplicación, se han diseñado con recursos de Flaticon.com.

- Imagen A.png: Disponible desde este [enlace](#).



- Imagen atras.png: Disponible desde este [enlace](#).



- Imagen boton-de-carga.png: Disponible desde este [enlace](#).



- Imagen clasification.png: Disponible desde este [enlace](#).



- Imagen conexion.png: Disponible desde este [enlace](#).



- Imagen configuraciones.png: Disponible desde este [enlace](#).



- Imagen G.png: Disponible desde este [enlace](#).



- Imagen imagenes.png: Disponible desde este [enlace](#).



- Imagen imagenes2.png: Disponible desde este [enlace](#).



- Imagen interrogacion.jpg: Disponible desde este [enlace](#).



- Imagen loading.png: Disponible desde este [enlace](#).



- Imagen pausa.png: Disponible desde este [enlace](#).



- Imagen play.png: Disponible desde este [enlace](#).



- Imagen senal-de-stop.png: Disponible desde este [enlace](#).



- Imagen thingSpeak.png: Disponible desde este [enlace](#).



- Imagen training.png: Disponible desde este [enlace](#).
- 
- Imagen vehiculos.png: Disponible desde este [enlace](#).



### Anexo 3: Licencias del esquema de arquitectura

Los siguientes iconos, que se utilizan en la imagen del esquema de la arquitectura de la aplicación, se han diseñado con recursos de Flaticon.com.

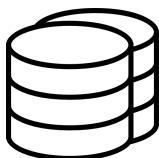
- Imagen 488.png: Disponible desde este [enlace](#).



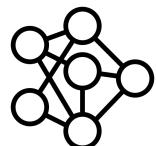
- Imagen 73443.png: Disponible desde este [enlace](#).



- Imagen 149208.png: Disponible desde este [enlace](#).



- Imagen 2103620.png: Disponible desde este [enlace](#).



- Imagen vehiculos-de-motor.png: Disponible desde este [enlace](#).



- Imagen Twitter-logo.svg.png: Disponible desde este [enlace](#).  

- Imagen kisspng-cloud-blue.png: Disponible desde este [enlace](#).  

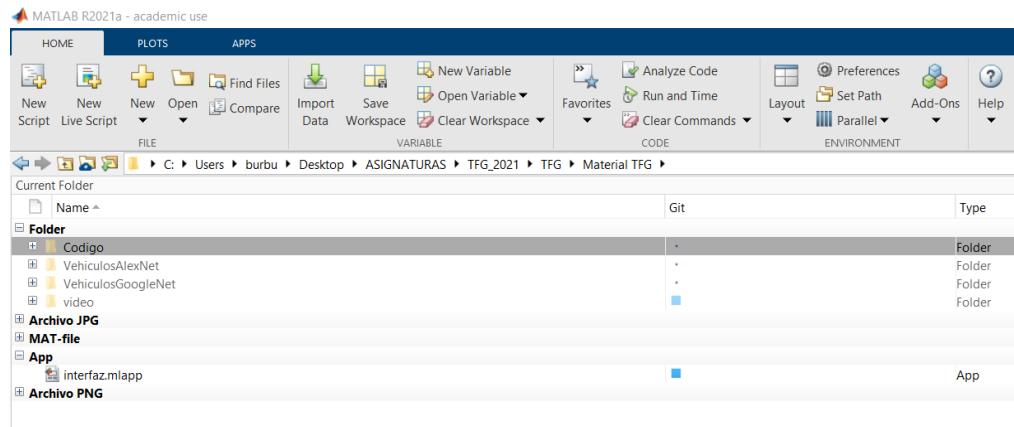

- Imagen 92153\_thingspeak\_logo.png: Disponible desde este [enlace](#).



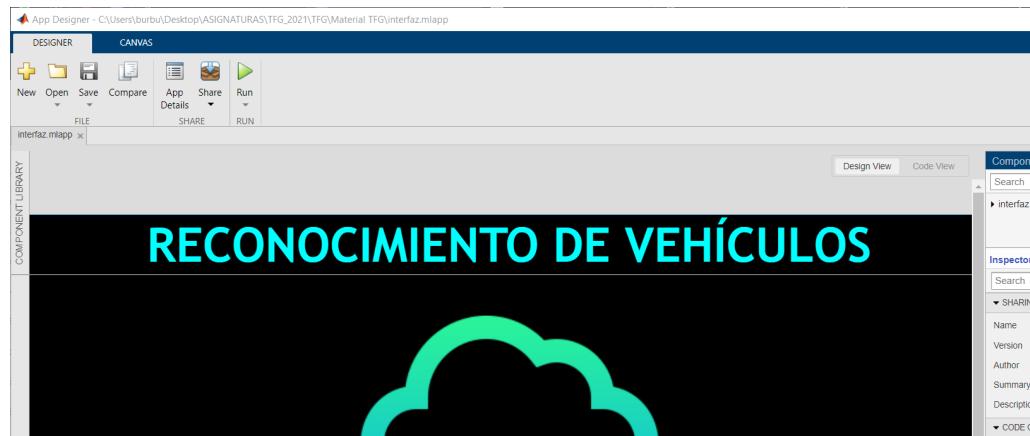
## Anexo 4: Manual de usuario y acceso al código

A continuación, se muestran los pasos a seguir para ejecutar la aplicación correctamente:

1. Descargar el código del repositorio de Github:  
<https://github.com/Estrella148/TFG>
2. Instalar MATLAB R2021a, así como las siguientes extensiones:
  - a. Computer Vision Toolbox
  - b. Deep Learning Toolbox
  - c. Deep Learning Toolbox Model for AlexNet Network
  - d. Deep Learning Toolbox Model for GoogleNet Network
  - e. Image Processing Toolbox
  - f. Parallel Computing Toolbox
3. A partir de este punto existen dos opciones para ejecutar la aplicación:
  - a. Abrir MATLAB y seleccionar la carpeta *Material TFG* que se encuentra dentro del repositorio de Github, previamente descargado. Después, aparecerá una estructura parecida a la mostrada a continuación.



Finalmente, se debe seleccionar el archivo *interfaz.mlapp* para iniciar la aplicación de *App Designer* y poder ejecutarla a través del botón *Run*.



- b. Acceder directamente y de manera local a la carpeta *Material TFG* que se encuentra dentro del repositorio de Github, previamente descargado, y seleccionar *interfaz.mlapp*.
4. Una vez iniciada la aplicación, es muy importante realizar primero el *TRAINING* de las dos redes y asegurarse de que los archivos *netTransferAlex.mat* y *netTransferGoogle.mat* se generan correctamente para realizar posteriormente la *CLASSIFICATION*.
5. Los videos disponibles para la sección *CLASSIFICATION* se encuentran en [este enlace](#) y deberán guardarse en una carpeta local para posteriormente acceder a ellos y ejecutarlos mediante la aplicación.

Para acceder al código, se debe realizar todo lo mencionado anteriormente hasta el punto 3a incluido y seleccionar la pestaña de *Code View* del *App Designer*.