

Universidad Complutense de Madrid
Facultad de Informática

**Smart cities:
Aprendizaje profundo con imágenes**

**Smart cities:
Deep learning with images**

Ana Laura Corral Descargue
Guillermo Delgado Yepes
Víctor Goicoechea Enrique
Manuel Guerrero Moñús



Trabajo de Fin de Grado en Ingeniería del Software

Curso: 2019 - 2020

Director: Gonzalo Pajares Martinsanz

Dedicatoria

Ana Laura Corral Descargue

A toda mi familia, que siempre ha estado ahí dando tanto apoyo emocional como económico, a todas las personas que he conocido a lo largo de la carrera que me han aportado cosas positivas y negativas y aprender de todo ello, e ir madurando.

Guillermo Delgado Yepes

A mi familia, que me ha acompañado en todo momento, en lo bueno y en lo malo, y me ha brindado su apoyo. A mis amigos y compañeros de la UCM, por todas las risas y buenos momentos que han facilitado el paso por la universidad. A mis profesores que han aportado conocimientos y momentos míticos que nunca olvidaremos. A mis compañeros de trabajo, por acogerme tan fácilmente y ayudarme con su propia experiencia.

Víctor Goicoechea Enrique

A mi familia, que siempre ha estado ahí para apoyarme en los momentos más difíciles, a los profesores y compañeros, que han hecho más llevaderos estos años de universidad.

Manuel Guerrero Moñús

A toda mi familia, que siempre ha creído en que puedo alcanzar cualquier reto que me proponga. A mi grupo de amigas del instituto, que tanto me aprecian. A mis amigos y excompañeros de formación profesional de Joyfe, así como a mis amigos y compañeros de la facultad de informática de la UCM, que siempre me animaron a ir a más. A Lu, quien tanto me ha enseñado.

Agradecimientos

Queremos agradecer a nuestro tutor del TFG y profesor de la asignatura Ingeniería del Conocimiento, Gonzalo Pajares Martinsanz, la oportunidad de haber realizado bajo su orientación, un trabajo que ha resultado tan interesante como útil de cara al futuro de la Ingeniería Informática.

Resumen

Con el paso del tiempo los sistemas de la información y la comunicación se han integrado en las sociedades humanas de tal forma que pasan desapercibidos en el entorno, para proporcionar, cada vez más y de forma más rápida, una gran cantidad de información que, al ser tratada, puede hacer que la vida de las personas sea más cómoda y sencilla.

Gracias a esta computación ubicua que monitoriza constantemente los datos de las actividades humanas han surgido paradigmas como Aprendizaje Profundo (*Deep Learning*) en Inteligencia Artificial o IoT (*Internet of Things*), que presta una especial atención a problemas del mundo real resolubles y automatizables, mediante el empleo de sistemas hardware y software que aplican los mejores modelos de procesamiento y análisis de datos para lograr una mejor toma de decisiones.

En particular, en el presente trabajo se pone el foco en el problema del análisis del flujo de vehículos existente en las ciudades modernas, que evolucionan hacia el concepto de ciudades inteligentes. Se trata de un problema relevante de cara a la sostenibilidad, un problema de la historia reciente que trata algunas cuestiones, como por ejemplo la optimización de las rutas de transporte, la movilidad eficiente, la determinación de los flujos de transporte en función de los itinerarios de las ciudades, la gestión eficiente de las plazas de aparcamiento, entre otros aspectos relacionados.

Para abordar esta problemática se ha desarrollado un sistema software pensado para su implantación y uso en el mencionado contexto de las ciudades inteligentes.

Dicho sistema contiene las siguientes funcionalidades:

- Técnicas de análisis del movimiento en secuencias de imágenes mediante el cálculo del flujo óptico.
- Determinación de las regiones en movimiento mediante la aplicación técnicas de segmentación basadas en umbralización y etiquetado.
- Identificación de los vehículos en movimiento mediante técnicas basadas en Aprendizaje Profundo con Redes Neuronales Convolucionales.
- Desarrollo de un módulo de predicción del flujo de tráfico de vehículos con fines de prevención y planificación.
- Desarrollo de una arquitectura cliente-servidor para la implementación de métodos basados en el paradigma IoT, que permitan publicar y visualizar los datos en crudo y los resultantes del análisis del flujo de vehículos.

Palabras clave

- Flujo óptico.
- Aprendizaje automático.
- Aprendizaje profundo.
- Redes neuronales convolucionales.
- Análisis predictivo.
- Visión por computador.
- Internet de las cosas.
- Ciudades inteligentes.

Abstract

Over time, information and communication systems have been introduced into human societies in such a way that they are unnoticed in the environment to provide even more and more quickly, a lot of information that, when it's processed, can make people's lives more simple and comfortable.

Thanks to this ubiquitous computing that is permanently monitoring the data of human activities, paradigms such as Deep Learning in artificial intelligence or IoT have emerged, they pay special attention to real world problems that can be solved and automated by using hardware and software systems by applying the best data processing and analysis models to achieve a better decision making.

Particularly, in the present work, the focus is put on the problem of vehicle flow analysis in modern cities, which are evolving to the concept of smart cities. This is a relevant problem in terms of sustainability, a problem of recent history that think about some questions, such as the optimization of transport routes, efficient mobility, determination of transport flows according to the timetables of the cities, efficient management of parking spaces, etc.

To solve this problem we have developed a software system thought for its implantation and use in the context of smart cities.

This system contains the following functionalities:

- Determination of regions in movement applying segmentation techniques based on umbralization and labeling.
- Movement analysis techniques applied to image sequences through the computation of optical flow.
- Vehicles identification in movement through techniques based on Deep Learning with Convolutional Neural Networks.
- Development of a prediction module for vehicle traffic flow with the aim of prevention and planification.
- Development of a client-server architecture for the implementation of methods based on the IoT paradigm, which allows to publish and to visualize raw data and the resulting data of vehicle flow analysis.

Keywords

- Optical flow.
- Machine learning.
- Deep Learning.
- Convolutional Neural Networks.
- Predictive analysis.
- Computer Vision.
- Internet of Things.
- Smart cities.

Índice

Dedicatoria	1
Agradecimientos.....	1
Resumen	2
Palabras clave.....	3
Abstract.....	3
Keywords	4
1. Introducción.....	7
1.1 Antecedentes	7
1.2 Motivación.....	9
1.3 Objetivos	9
1.4 Plan de trabajo.....	10
1.5 Estructura de la memoria	12
1.6 Contribuciones personales.....	13
1.6.1 Ana Laura Corral Descargue	13
1.6.2 Guillermo Delgado Yepes.....	15
1.6.3 Víctor Goicoechea Enrique.....	17
1.6.4 Manuel Guerrero Moñús	20
1.7 Introduction	22
1.7.1 Preliminary.....	22
1.7.2 Objectives	24
1.7.3 Work plan.....	25
2. Métodos conceptuales aplicados	26
2.1 Visión por Computador	26
2.1.1 Cálculo del flujo óptico	27
2.1.2 Detección de regiones en movimiento.....	28
2.2 Redes Neuronales Convolucionales	31
2.2.1 Capas del modelo AlexNet	33
2.3 Series temporales: Autorregresión.....	41
3. Diseño y análisis de la plataforma	42
3.1 Análisis arquitectónico	45
3.1.1 Lado del cliente.....	45
3.1.2 Lado del servidor	50
3.1.2.1 Canal de detección de vehículos.....	52
3.1.2.2 Canales de predicción	54
3.2 Metodología de desarrollo empleada.....	57
3.3 Herramientas y recursos software	58

4. Resultados	60
4.1 Interfaz de la aplicación	61
4.1.1 Learning.....	62
4.1.1.1 Retraining	63
4.1.2 Car Detection.....	63
4.1.3 ThingSpeak.....	64
4.1.3.1 Queries de ThingSpeak.....	65
4.1.4 Forecast.....	66
4.2 Visión por computador	67
4.2.1 Detección de movimiento.....	67
4.2.2 Proceso de binarización	68
4.2.3 Proceso de segmentación	69
4.2.4 Clasificación del flujo de tráfico.....	69
4.2.5 Cómputo del flujo del tráfico	71
4.3 Aprendizaje profundo	75
4.3.1 Ajustes del entrenamiento e interpretación de los resultados.....	75
4.3.2 Clasificación post-entrenamiento.....	79
4.4 Procesamiento y predicción en ThingSpeak	80
4.4.1 Resultados obtenidos	81
4.5 Resultados del contador de vehículos de ThingSpeak	83
5. Conclusiones y trabajo futuro	85
6. Conclusions and future Work	87
7. Bibliografía	91
8. Anexos	94
Anexo I : Diagrama de clases de la capa de presentación.....	94
Anexo II : Diagrama de clase de la capa de negocio.	95
Anexo III: Resultados ThingSpeak Predicción.....	96
Anexo IV: Resultados ThingSpeak CarDetection.	99
Anexo V : Manual de usuario.....	99

1. Introducción

1.1 Antecedentes

Con el paso del tiempo el ser humano ha ido orientando su desarrollo tecnológico hacia la sostenibilidad, cambiando su modo de vida de forma intencional para apostar cada vez más por el ahorro de los recursos energéticos, las energías renovables, el consumo responsable y el respeto por el medio ambiente.

Bajo este planteamiento las ciudades inteligentes constituyen un elemento clave en este complejo engranaje. Dentro de ellas, la gestión de la movilidad se establece como uno de los objetivos prioritarios, para lo cual es necesario el manejo del tráfico de vehículos de forma eficiente y controlada. Por otra parte, el desarrollo tecnológico está posibilitando la aparición de soluciones tecnológicas eficientes para abordar la problemática derivada de estos planteamientos. A ello contribuyen, la mejora de los sensores, la capacidad de procesamiento de los sistemas disponibles y el desarrollo de paradigmas tales como internet de las cosas (*Internet of Things*, IoT). Es aquí donde se centra el proyecto que se presenta en este trabajo, orientado a contribuir a una disminución de las emisiones contaminantes, el ruido y los tiempos de viaje hacia la sostenibilidad.

Actualmente, existen soluciones enmarcadas dentro de lo que se conocen como Advanced Traffic Management Systems (ATMS) y Advanced Traveler Information Systems (ATIS) para un control y manejo eficiente de los flujos de tráfico, con especial énfasis en los contextos urbanos²⁵. Son diversos los sistemas ATMS/ATIS diseñados con tal propósito, destacando por ejemplo el uso de redes celulares de sensores que envían información a las estaciones de procesamiento o sistemas de localización espacial como Waze²⁶, basados en el movimiento de los usuarios.

Bajo esta perspectiva, aunque desde una posición diferente, en el presente trabajo se plantea una solución de concepto, de forma que mediante la captura de imágenes a color a través de una cámara, es posible aplicar técnicas de Visión por Computador para determinar el movimiento de vehículos, y realizar su identificación mediante la aplicación de técnicas de Aprendizaje Profundo, y el manejo de esta información procediendo a su publicación con posibilidades de procesamiento bajo el mencionado paradigma IoT, que facilita el acceso a la misma por parte de usuarios, particulares o institucionales, para el conocimiento y la gestión eficiente del tráfico rodado o la predicción del mismo. Es evidente que este planteamiento se diferencia de los anteriores en el sentido de que se trata de un único sensor (cámara) cuyos datos son imágenes que se procesan en unidades con capacidad suficiente y cuyos resultados se ponen a disposición de usuarios con fines de visualización y predicción.

En cualquier caso, tanto en los sistemas anteriores como en la propuesta del presente trabajo, el objetivo final se encamina al procesamiento y análisis inteligente de los datos disponibles para la extracción de información, y el desarrollo de modelos para realizar una mejor toma de decisiones en cuanto a eficiencia y sostenibilidad, que en el caso del presente proyecto se orienta hacia una mejor distribución de los flujos de tráfico de vehículos en las ciudades.

Dentro de esta gestión eficiente del tráfico en las ciudades inteligentes, se contemplan las estaciones de recarga de vehículos eléctricos, tanto de carácter público como privadas en centros de trabajo, de forma que se garantice un flujo apropiado de cara a sus posibilidades, evitando así esperas y pérdidas de tiempo innecesarias. Tal es el caso de la empresa Elmec Informática en Varese (Italia), que ha automatizado al completo su sede central en asociación con Everynet mediante tecnologías IoT¹.

En el parking del edificio la empresa ha situado siete nodos Libelium Smart Parking²² para gestionar de forma eficiente los puntos de carga de los vehículos eléctricos. Estos dispositivos informan del grado de ocupación del parking, de forma que en soluciones integradas con la propuesta que se formula en este proyecto, esta información ha de resultar de gran utilidad para resolver el problema de la movilidad, que es en definitiva el objetivo final.

Por otro lado, en algunas ciudades como Dordrecht (Holanda) se ha desarrollado un proyecto de investigación² en el que se muestra la utilidad de aplicar el paradigma IoT a algunos de los problemas anteriormente mencionados. Para ello se instalaron en los cruces de varias de sus calles ocho equipos Meshlium IoT Gateways de Libelium²², éstos sirven para conseguir que los datos recibidos por los sensores lleguen hasta una plataforma que permita la gestión de la información. Los dispositivos Meshlium están sincronizados por un reloj digital externo y a través de un escáner wifi detectaban la dirección física (*mac address*) de los smartphones y de los dispositivos manos libres de los vehículos. La recogida de datos se realiza accediendo a la base de datos local de cada sensor para luego ser almacenados en una base de datos PostgreSQL, una vez comprobada la persistencia de los datos, éstos son analizados mediante scripts de Python y visualizados en un mapa geográfico gracias a la herramienta QGIS. Con los datos recogidos y conociendo la relación de distancia entre dos sensores, se puede obtener la cantidad de movimiento de cada dispositivo detectado. Teniendo en cuenta lo anterior, junto con los criterios de uso de cada calle de la ciudad sobre la que se estaba realizando la monitorización, se pueden reconocer tres tipos de usuarios de estos dispositivos: peatones, ciclistas y vehículos.

Esto ha servido para estudiar la relación entre estas tres categorías a lo largo del día, lo que ha sido útil tanto para identificar las calles preferidas por cada tipo de usuario, como para conocer los hábitos de los usuarios a lo largo del tiempo.

Queda patente, que toda la información disponible y en especial la relativa a la densidad de tráfico, como es el caso del presente proyecto, permite el trazado de rutas más eficaces en los desplazamientos dentro y fuera de las ciudades, lo que justifica la propuesta que se formula en el presente proyecto.

1.2 Motivación

Según lo expuesto previamente, los motivos que originan el trabajo se centran en el interés por el desarrollo de una solución inteligente con vistas a un futuro, no muy lejano, en el que se integren métodos y tecnologías de vanguardia, actualmente en pleno auge, lo cual da pie a sumergirnos en dos campos que se engloban en el ámbito de la inteligencia artificial, éstos son: Visión por Computador (*Computer Vision*)⁴ y Aprendizaje Profundo (*Deep Learning*)²⁷.

Estos campos de estudio se combinan y se integran en el emergente paradigma IoT, basado en la idea de un mundo futuro en el que diferentes dispositivos sensores se conectan entre sí para compartir y recopilar diferentes datos procedentes de actividades humanas. Aunque lo normal es el empleo de dichos dispositivos para el suministro de grandes cantidades de datos, en el presente proyecto los datos son extraídos a partir de vídeos conteniendo vehículos en movimiento. Dichos datos se procesan posteriormente mediante la aplicación de técnicas inteligentes que extraen patrones o conclusiones que pueden ser, y de hecho son, de gran utilidad de cara a la sostenibilidad de las ciudades inteligentes del futuro.

Siguiendo este planteamiento, y tras estudiar los ejemplos expuestos anteriormente junto con algunos otros en pleno uso, como los que se han implantado en Montpellier (Francia)²³ para el control inteligente de los parkings y los accesos a éstos desde las vías urbanas, se considera que el desarrollo de las mencionadas tecnologías inteligentes, así como la integración de estas en el IoT, proporcionan los ingredientes motivacionales necesarios para la elaboración del trabajo que aquí se presenta.

1.3 Objetivos

El objetivo general de este proyecto es el desarrollo de una aplicación conceptual que pueda implantarse en el ámbito de las ciudades inteligentes en un futuro no muy lejano, para la detección del flujo de vehículos en las vías de acceso a éstas, lo cual se realiza mediante el análisis de las imágenes contenidas en vídeos. Gran parte del procesamiento se lleva a cabo en el lado del cliente de esta aplicación, mediante el empleo de máquinas que cuenten con los procesadores y las unidades de procesamiento gráfico necesarias para soportar el reentrenamiento de una Red Neuronal Convolutacional y la detección de imágenes a la mayor velocidad posible, que son las tecnologías a aplicar.

Los datos resultantes del procesamiento de la información en el lado del cliente son transferidos a un servidor remoto ThingSpeak²⁴, allí éstos quedan almacenados para su visualización, consulta y otros procesamientos, tales como predicción.

Como se ha mencionado previamente, se proporciona, de este modo, una solución conceptual inteligente surgida a partir de la aplicación de técnicas basadas en Visión por Computador⁴ y Aprendizaje Profundo²⁷, e integradas en el contexto de IoT, lo que constituye en sí el objetivo general bajo el que se desarrolla el presente proyecto.

El planteamiento anterior da lugar a los siguientes objetivos específicos:

- Desarrollar técnicas basadas en el análisis del flujo óptico en imágenes para la detección de vehículos en movimiento.
- Desarrollar técnicas de segmentación de imágenes para extraer las regiones en movimiento mediante umbralización y el etiquetado de regiones conexas.
- Desarrollar un módulo de transferencia de conocimiento mediante el uso de Redes Neuronales Convolucionales pre-entrenadas y re-entrenadas mediante una fase de aprendizaje, que sirve para la detección de vehículos en una segunda fase de clasificación.
- Desarrollar un método de cómputo de vehículos en la secuencia de imágenes para proceder a la carga de datos en la nube.
- Desarrollar técnicas de acceso a la nube y procesamiento de datos en ésta.
- Desarrollar técnicas de análisis predictivo en la nube con el fin de determinar el flujo de vehículos en el futuro
- Desarrollar interfaces amigables para la captura, la visualización de los datos y los resultados extraídos de estos.
- Integrar las técnicas y módulos anteriores bajo una arquitectura de software.

1.4 Plan de trabajo

Tareas específicas para llevar a cabo los objetivos anteriores:

- **Análisis del flujo óptico:** Consiste en el estudio e implementación de técnicas para el análisis de vehículos en movimiento a partir de los datos de entrada. Para ello se hace uso de los métodos y funcionalidades proporcionadas por el entorno de desarrollo.

- **Umbralización, etiquetado y extracción de las regiones en movimiento:** Determinación exacta de las regiones con gran cantidad de movimiento, ya que éstas resultan ser las zonas más prometedoras de localización de vehículos en circulación. Supone el estudio e implementación de métodos basados en técnicas de segmentación de imágenes para la extracción de regiones.
- **Identificación del modelo de Red Neuronal Convolucional y aplicación de la técnica de transferencia de aprendizaje:** Existen muchos y variados modelos de redes neuronales, lo que implica tener que decantarse por uno de ellos que sea lo suficientemente accesible y eficiente como para poder resolver parte de la problemática de este proyecto. Hemos considerado varios modelos ya pre-entrenados con miles e incluso millones de imágenes con el objetivo de re-aprovechar su aprendizaje para adaptarlo al problema de identificación y conteo de vehículos. Esto puede realizarse, con relativa facilidad, mediante la sustitución de algunas capas finales del modelo original y se conoce como “transferencia de aprendizaje”¹³, ya que se aprovecha lo que el modelo original tiene aprendido.
- **Preprocesamiento de imágenes y aumento del número de éstas para el entrenamiento y la validación:** Habiendo optado por el uso de la Red Neuronal Convolutiva AlexNet⁹, cuyos detalles se definen más adelante, esta tarea consiste tanto en evitar la introducción errónea de imágenes con las que este modelo no puede trabajar, como en la formación de un set de imágenes de tamaño superior al inicial para enriquecer los procesos de entrenamiento y validación.
- **Configuración del proceso de entrenamiento del modelo AlexNet:** Se centra en el estudio de los parámetros del modelo AlexNet que guardan relación con el proceso de entrenamiento. Algunos de tales parámetros son: número de ciclos de entrenamiento (*epochs*), número de imágenes a pasar por la red por cada iteración (*batch size*) o factor de aprendizaje (*learn rate factor*).
- **Generación de informes del entrenamiento y clasificación:** El objetivo de esta tarea consiste en proporcionar información adicional al usuario con respecto a la bondad del proceso de entrenamiento y validación.
- **Desarrollo de un sistema de conteo de vehículos:** Consiste en el estudio y la aplicación de ciertas propiedades de las regiones de una imagen, extraídas mediante las técnicas citadas con anterioridad y con el objetivo de contabilizar los vehículos que pasen por determinada zona de la imagen sobre la que se ha realizado la extracción.

- **Diseño y desarrollo de una arquitectura de software multicapa:** Se analizará la posibilidad de estructurar la aplicación siguiendo un modelo multicapa, investigando las diferencias existentes entre el lenguaje utilizado en la aplicación y los utilizados en los estudios de grado y de esta manera aplicar lo aprendido a nuestra aplicación. Posteriormente se realizará una fase de modelado de software que terminará con la realización de diagramas de clase y casos de uso. A continuación se procederá a la codificación y fase de testing.
- **Diseño y desarrollo de las interfaces de usuario:** Se estudiará el uso de los elementos y funcionalidades que proporciona el entorno de desarrollo utilizado en el proyecto (MATLAB App Designer). Se desarrollarán interfaces intuitivas y amigables con el usuario, facilitando la curva de aprendizaje en el manejo de la aplicación.
- **Desarrollo de la comunicación cliente-servidor:** Se centra en la realización de operaciones de lectura y escritura. En el caso de escritura, guardará en el servidor los datos recopilados referentes al conteo de los tipos de vehículos. En el caso de lectura, se usará para leer determinados campos del canal de ThingSpeak²⁴ para mostrarlos en la aplicación.
- **Programación del lado del servidor:** Se utilizarán determinadas apps y funcionalidades que proporciona ThingSpeak, concretamente ThingTweet, para enlazar una cuenta de Twitter y React para lanzar avisos en la cuenta de Twitter, cuando se escriban nuevos datos en el canal asignado.
- **Desarrollo de un algoritmo en la nube para la predicción del flujo del tráfico:** Se utilizará MATLAB Analysis de ThingSpeak para la realización de un modelo AR para hacer predicciones del flujo de tráfico que circula por un punto concreto de la M30 en Madrid.

1.5 Estructura de la memoria

El documento tiene la siguiente disposición de apartados:

- 1. Introducción:** Aquí se exponen algunos ejemplos reales de carácter similar a lo que se ha hecho en este trabajo, así como aquellos factores que nos han motivado como para querer realizarlo. También se mencionan las tareas que son necesarias para completar el trabajo y las áreas de estudio que se han abordado para aprender conceptos con los que resolver dichas tareas.
- 2. Métodos conceptuales aplicados:** En esta sección se profundiza en los temas de Visión por Computador⁴, Redes Neuronales Convolucionales y por último Series Temporales con Autoregresión¹⁶.

- 3. Análisis y diseño:** En este apartado se describen aquellos aspectos propios de la Ingeniería del Software, tales como la arquitectura de la aplicación, la metodología de trabajo empleada para la realización de este proyecto y los recursos hardware y software que han sido utilizados, tanto para el diseño y el desarrollo del sistema inteligente, como para la organización del trabajo.
- 4. Análisis de resultados:** Una vez completadas las distintas partes de este trabajo se examinan los resultados obtenidos, exponiendo las apreciaciones tanto positivas como negativas.
- 5. Conclusiones y trabajo futuro:** Se expondrán las diferentes conclusiones obtenidas a lo largo de la realización del trabajo, utilizando éstas para la concepción de futuras mejoras que nos servirán para tratar de mejorar los resultados obtenidos que no han sido satisfactorios.

1.6 Contribuciones personales

1.6.1 Ana Laura Corral Descargue

El desarrollo de una aplicación, como la que se presenta en este trabajo, conlleva una serie de tareas que se estructuran según las secciones que se indican a continuación.

Labores de investigación:

- Investigación sobre la posibilidad de disponer de vídeos en directo para su análisis on-line, si bien por motivos de privacidad de datos no se ha llegado a una solución viable.
- Búsqueda de un DataSet de flujos de tráfico, con el fin de aproximarse lo más posible a datos reales. En este sentido, indicar que se encontró un repositorio oficial mantenido por el Ayuntamiento de Madrid¹⁵, que es el utilizado para hacer predicciones.
- Investigación para determinar el formato de subida de datos al servidor IoT remoto de ThingSpeak²⁴.
- Investigación del funcionamiento de MATLAB Analysis para la unión de distintos canales en ThingSpeak.
- Análisis del problema de la ubicación de un sistema de conteo de vehículos basado en el uso de las propiedades básicas de las regiones conexas

extraídas de una imagen. Esta tarea se ha realizado en colaboración con los otros tres miembros del grupo.

Gestión del proyecto software:

- Identificación de las tareas del módulo de predicción, módulo del que soy la principal responsable.
- Gestión de las tareas y del Work in Progress del módulo de predicción mediante el uso de un tablero Kanban a través de la herramienta Trello, que permite la gestión eficiente.
- Realización de reuniones con los miembros del equipo y tutor para resolver dudas, comunicar nuestro progreso con la tarea asignada y valoración de esta misma.

Diseño y desarrollo de software:

- Implementación de un sistema de conteo de vehículos mediante el uso de las propiedades que fueron extraídas de aquellas regiones conexas en las que se ha detectado una cantidad de movimiento superior al valor umbral, tarea realizada en colaboración con el resto de miembros del grupo con distribución específica de tareas.
- Creación de un canal en ThingSpeak^{19 24} para la subida de los datos extraídos del DataSet del Ayuntamiento de Madrid¹⁵.
- Desarrollo e implementación de código para la extracción de los datos procedentes del DataSet oficial del Ayuntamiento de Madrid¹⁵ para realizar las predicciones.
- Creación de un canal en ThingSpeak para almacenar las peticiones de datos de la parte de presentación relativa a la vista de predicción, concretamente día de la semana durante el que se quiere realizar la consulta y número de días a consultar.
- Creación de un MATLAB Analysis para la unión de los canales mencionados anteriormente.
- Creación de un canal específico para el almacenamiento de los datos resultantes del MATLAB Analysis, con el fin de mostrarlo posteriormente en la interfaz de predicción en forma de gráfico.

- Depuración del código fuente de la aplicación para el subsanamiento de los errores hallados, realización de pruebas y detección de incompatibilidades de la aplicación al ser usada en sistemas de tipo GNU/Linux. Tarea realizada en colaboración con el resto de compañeros del equipo.

Parte de memoria:

- Revisión de la memoria.
- Desarrollo de los puntos 1.5, 2.3 de la memoria.
- Apartados redactados en colaboración con mis compañeros: 4.4, 4.4.1 y 7.

1.6.2 Guillermo Delgado Yepes

Como en el caso anterior, las tareas se estructuran según las siguientes secciones, teniendo en cuenta que algunas de ellas se solapan, dado el planteamiento de trabajo en grupo.

Labores de investigación:

- En colaboración con el miembro del equipo Víctor Goicoechea, realicé un prototipo de interfaz de prueba para determinar cómo se realiza la lectura y escritura entre ThingSpeak²⁴ y MATLAB.
- Análisis del problema de la ubicación de un sistema de conteo de vehículos, basado en el uso de las propiedades básicas de las regiones conexas extraídas de una imagen. En colaboración con los otros tres miembros del equipo.
- Investigar la implementación de interfaces usando la herramienta que MATLAB pone a disposición de los usuarios llamada “App Designer”, que hace posible el diseño y desarrollo de interfaces.
- Investigar cómo unir los distintos elementos gráficos resultantes de la detección de vehículos en una única interfaz con los componentes que nos ofrece MATLAB.
- Investigar la posible arquitectura de la aplicación mediante los patrones establecidos y siguiendo las metodologías de Ingeniería de Software y Modelado de Software.

- Investigar los Toolbox de MATLAB que es necesario instalar para añadir las funcionalidades necesarias para la detección de vehículos.
- Investigar metodologías de Programación Orientada a Objetos y el uso de clases en Matlab, los modificadores de acceso a variables y funciones de la clase, así como el uso de la instanciaión única.

Gestión del proyecto software:

- Identificación de las tareas del módulo de Arquitectura del Software (SA), Interfaces de Usuario (UI) y Experiencia de Usuario (UX), módulo del cual soy el principal responsable.
- Gestión de las tareas y del Work in Progress del módulo de Arquitectura del Software (SA), Interfaces de Usuario (UI) y Experiencia de Usuario (UX), mediante el uso de un tablero Kanban.

Diseño y desarrollo de software:

- Realización de diagramas de clase como parte de la fase de diseño.
- Realización de varios mockups de ejemplo como parte del diseño de las interfaces de la aplicación, para visualizar resultados provisionales, cuyo objetivo es determinar el progreso de los desarrollos y validación en su caso.
- Colaboré con Víctor Goicoechea para comunicar las interfaces que hacen uso de ThingSpeak²⁴ para conseguir la funcionalidad adecuada.
- Implementación de las interfaces siguiendo los mockups realizados previamente y siguiendo los principios básicos de claridad, flexibilidad, unificación y coherencia para con los usuarios de la aplicación, de forma que su uso sea lo más sencillo y amigable posible.
- Colaboré con Manuel Guerrero para conectar el módulo de Visión por Computador (*Computer Vision*)⁴ y Aprendizaje Profundo (*Deep Learning*)²⁷ con sus respectivas interfaces de usuario, para que así éstas contasen con las funcionalidades necesarias para su utilización.
- Colaboré con Ana L. Corral para conectar la parte de predicción en el servidor con la interfaz que implementa dicha funcionalidad.
- Adaptar todo el código de la aplicación a la arquitectura diseñada previamente, siguiendo un modelo multicapa.

- Estructurar el código de la aplicación en los distintos paquetes y directorios de forma que los distintos módulos queden perfectamente definidos.
- Implementación de un sistema de conteo de vehículos mediante el uso de las propiedades que fueron extraídas de aquellas regiones conexas en las que se ha detectado una cantidad de movimiento superior al valor umbral. Tarea realizada en colaboración con los compañeros Ana L. Corral y Manuel Guerrero.
- Depuración del código fuente de la aplicación para el subsanamiento de los errores hallados, realización de pruebas y detección de incompatibilidades de la aplicación al ser usada en sistemas de tipo GNU/Linux. Tarea realizada en colaboración con el resto de miembros del equipo.

Memoria:

- Revisión técnica y de redacción.
- Redacté los apartados: 3, 3.1.1. y 4.1
- Apartados redactados en colaboración con mis compañeros: 3.1.2.1, 3.1.2.2, 4.4, 4.4.1 y 7.

1.6.3 Víctor Goicoechea Enrique

Como en los casos anteriores, las tareas realizadas se estructuran como sigue:

Labores de investigación:

- Investigué cómo realizar la comunicación entre el canal de ThingSpeak²⁴ y MATLAB, realizando un prototipo de interfaz de prueba para determinar cómo se realiza la lectura y escritura entre ThingSpeak y MATLAB realizada en colaboración con el miembro del equipo Guillermo Delgado.
- Estudié si la licencia de ThingSpeak era viable para poder ser utilizada en el proyecto, analizando las funcionalidades necesarias sin restricciones y determinado si éstas eran viables en relación a los objetivos planteados.
- Investigué las diferentes Apps que facilita ThingSpeak²⁴ para poder hacer que los datos fueran visualizados en diferentes dispositivos/plataformas, como puede ser el teléfono móvil o Twitter.

- Investigué diferentes datasets para alimentar con imágenes el entrenamiento de la CNN Alexnet⁹, comprobando que tuvieran la correspondiente licencia para su uso con fines académicos.

Gestión del proyecto software:

- Colaboré en la identificación de las diferentes tareas en Trello que se tenían que llevar a cabo para la realización del proyecto.
- Gestioné las diferentes tareas en Trello, moviéndolas entre las diferentes columnas para que el tablero Kanban estuviera constantemente actualizado.
- Valoración de las imágenes y vídeos a utilizar para el aprendizaje de la Red Neuronal Convolutiva (Convolutional Neural Network, CNN). Finalmente nos decantamos por la grabación de nuestros propios vídeos y su posterior tratamiento, extrayendo de los mismos las imágenes para poder enriquecer el entrenamiento de la CNN.

Diseño y desarrollo de software:

- Implementé la llamada de escritura en el canal de ThingSpeak que guarda el conteo de los diferentes tipos de vehículos (FrontCar, BackCar, FrontMotorbike, BackMotorbike, FrontTruckVan, BackTruckVan, FrontBus, BackBus), en la parte de la aplicación donde se realiza la detección de vehículos.
- Participé en el diseño de la interfaz de la aplicación referente a ThingSpeak, comentando las condiciones necesarias para poder realizar las consultas a ThingSpeak, y poder mostrar los datos en la interfaz.
- Implementé la llamada de lectura para el canal de ThingSpeak para realizar la representación de los datos en la interfaz de la aplicación, que hace referencia al apartado de ThingSpeak->Query, donde se podrán hacer consultas indicando dos intervalos de tiempo entre sí y señalando los tipos de vehículos, mostrando los datos en un tabla dentro de la interfaz de MATLAB y una gráfica donde se representan con líneas de diferentes colores en función del tipo de vehículo.
- Creé el canal de ThingSpeak²⁴ donde se suben y consultan todos los datos referentes al conteo de vehículos por parte de la aplicación desarrollada en Matlab.

- Creeé la cuenta de Twitter que posteriormente fusionaría con ThingSpeak, mediante las apps de ThingTweet y React, propias de ThingSpeak, que permiten mostrar avisos sobre los datos que entran en el canal de ThingSpeak. ThingTweet, sirve para enlazar la cuenta de Twitter con el canal deseado de ThingSpeak y React para crear una especie de disparador por el cumplimiento de las condiciones prefijadas, en cuyo caso se lanza un mensaje indicando qué campo ha sido modificado. Realicé un React por cada campo a controlar.
- Realicé varios recortes de los diferentes videos grabados en la autovía A-2, para mejorar la identificación de coches por delante, coches por detrás y asfalto. Comentar que en un principio realice 200 recortes de coches por delante y coches por detrás, pero al entrenar la CNN Alexnet⁹, se producía un sobreajuste de los coches por delante y por detrás, siendo necesario realizar un reequilibrio entre todos los modelos de datos para el entrenamiento. En cuanto al tipo asfalto, simplemente contiene imágenes de asfalto y líneas de carretera, que sirven a Alexnet para descartar posibles zonas del video con falso movimiento. Esta tarea fue realizada en colaboración con Manuel Guerrero.
- Llevé a cabo la redimensión de todas las imágenes extraídas de los videos usando la función de redimensionar implementada en la propia aplicación, subiendo las imágenes redimensionadas al repositorio drive, para posteriormente ser utilizadas en el entrenamiento del modelo de red Alexnet, ya que solo acepta imágenes con dimensiones: 227x227x3.
- Depuración del código fuente de la aplicación para el subsanamiento de los errores hallados, realización de pruebas y detección de incompatibilidades de la aplicación al ser usada en sistemas de tipo GNU/Linux. Tarea realizada en colaboración con el resto de miembros del proyecto.

Memoria:

- Revisé la memoria.
- Redacté el apartado 3.1.2, 3.1.2.1, 3.3 y 6
- Apartados redactados en colaboración con mis compañeros: 3.1, 4.3, 5, 7.

1.6.4 Manuel Guerrero Moñús

Al igual que en los casos anteriores, las tareas realizadas se estructuran como sigue:

Labores de investigación:

- Estudio de las capas, y las operaciones de éstas, del modelo de Red Neuronal Convolucional AlexNet⁹.
- Investigación sobre la aplicación específica del concepto “transferencia de aprendizaje”¹³. Qué implica y cuándo ha de usarse.
- Investigar, comprender y estudiar la forma de aplicar métodos eficientes de entrenamiento de las Redes Neuronales Convolucionales.
- Estudio del concepto y posibilidades de aplicación del método del gradiente o método de Lucas-Kanade⁴ para el cálculo del flujo óptico en imágenes digitales. Búsqueda de herramientas y métodos alternativos para el cálculo del flujo óptico en tiempo real.
- Investigación sobre la aplicación del proceso de binarización de imágenes a color.
- Comprensión del algoritmo de dos fases de Haralick y Shapiro (1992)³ para la extracción de regiones en imágenes binarizadas.
- Análisis del problema de la ubicación de un sistema de conteo de vehículos basado en el uso de las propiedades básicas de las regiones conexas extraídas de una imagen. En colaboración con el resto de miembros del equipo.

Diseño y desarrollo de software:

- Implementación del concepto “transferencia de aprendizaje”.
- Preprocesamiento de las imágenes de los conjuntos de entrenamiento y validación, así como preparación de lo que se denomina aumento de imágenes (*image augmentation*) para mejorar el entrenamiento y favorecer la clasificación posterior.
- Configuración del proceso de entrenamiento del modelo de Red Neuronal Convolucional AlexNet.

- Inclusión de un analizador de Redes Neuronales Convolucionales¹¹ que sirve para proporcionar, tanto información como ayuda con respecto al modelo de red empleado.
- Inclusión de la Matriz de Confusión¹², para verificar y validar el proceso de entrenamiento.
- Cálculo del flujo óptico sobre imágenes binarias.
- Desarrollo del proceso de binarización de una imagen a color mediante umbralización, extracción de las regiones conexas y su posterior etiquetado.
- Obtención de las propiedades (*Bounding Box, Area y Centroid*) requeridas para el reconocimiento de los vehículos.
- Implementación de un sistema de conteo de vehículos utilizando las propiedades anteriores. Tarea realizada en colaboración con el resto de miembros del equipo.
- Conexión del módulo de inteligencia artificial en el lado del cliente con sus respectivas interfaces de usuario, para que éstas incluyan lo necesario para su utilización. Tarea realizada en colaboración con Guillermo Delgado.
- Depuración del código fuente de la aplicación para el subsanamiento de los errores hallados, realización de pruebas y detección de incompatibilidades de la aplicación al ser usada en sistemas de tipo GNU/Linux. Tarea realizada en colaboración con el resto de miembros del equipo.

Gestión del proyecto software:

- Identificación de las tareas del módulo de Deep Learning²⁷ y Computer Vision⁴, del que soy el principal responsable.
- Gestión de las tareas y del Work in Progress del módulo de Deep Learning y Computer Vision, mediante el uso de un tablero Kanban.

Trabajo de campo:

- Grabación previa y procesamiento de vídeos en la parte del cliente para proporcionar datos de tráfico al lado del servidor con fines de predicción.

- Extracción de imágenes de vehículos y otros elementos presentes en las vías de circulación a partir de vídeos de tráfico y sets de imágenes gratuitos de uso no comercial, para utilizar dichas imágenes para el entrenamiento de la red. Esta tarea fue realizada en colaboración con Víctor Goicoechea.

Memoria:

- Redacción de los apartados: 1.1, 1.2, 1.3, 1.6.4, 2.1, 2.2, 3.2, 3.4, 4.1 y 4.2.
- Apartados redactados en colaboración con mis compañeros: 1.4, 5 y 7.

1.7 Introduction

1.7.1 Preliminary

Over time humans have oriented their technological development to sustainability, changing their way of life intentionally to contribute more and more for saving the energy resources, the renewable energies, the responsible consumption and the respect for the environment.

Under this approach, smart cities are a key element in this complex gear. Among them, mobility management is established as one of the priority objectives, for which it is necessary to manage vehicle traffic in an efficient and controlled manner.

On the other hand, technological development is making possible the appearance of efficient technological solutions to tackle the problems derived from these approaches. Definitively, an important contribution on this area becomes from the improvement of sensors, the processing capacity in the available systems and the development of paradigms such as the Internet of Things (IoT). Here is where this work is focused, aimed at contributing to a reduction in pollutant emissions, noise and travel times towards sustainability.

Currently, there are solutions within what are known as Advanced Traffic Management Systems (ATMS) and Advanced Traveler Information Systems (ATIS) for efficient control and management of traffic flows, with special emphasis on urban contexts²⁵. Different ATMS / ATIS systems, designed for this purpose, have been developed, including cellular sensor networks that send information to be processed on specific stations or systems with spatial location such as Waze²⁶, this last based on the movement of users.

Under these considerations, although from a different point of view, in this work a concept solution is proposed, so that by capturing color images through a camera, it is possible to apply Computer Vision⁴ techniques to determine the movement of

vehicles, and its identification through by using Deep Learning²⁷ techniques and the handling of this information with its public access and with options to be processed under the aforementioned IoT paradigm, providing access to users (personal or institutional) for knowledge and management efficient traffic or for prediction. It is obvious how this approach differs from the previous ones as it consists of a single sensor (camera) whose data are images to be processed in computerized units with sufficient capacity and whose results are made available to users for visualization purposes and traffic flow prediction.

Both, the previous systems and the proposed in this work, the final goal is aimed at the intelligent processing and analysis of data available for extracting information, and the development of models to carry out better decision-making according to efficiency and sustainability, which in the case of this project is oriented towards a better distribution of vehicle traffic flows in cities.

Within actions for efficient traffic management in smart cities, electric vehicle charging stations are considered, both with public and private usage, so as to guarantee an appropriate traffic flow according to their possibilities, thus avoiding unnecessary waiting and waste of time. The Elmec Informatic company in Varese (Italy), has fully automated its facilities in association with Everynet and using IoT technologies¹. In the car parking, seven Libelium Smart Parking nodes²² were installed for efficient management of charging points for electric vehicles. These devices report car parks' occupancy rates, so that for integrated solutions involving the proposed approach in this work, this information becomes very useful for solving mobility problems, which is the final objective.

On the other hand, in some cities such as Dordrecht (Netherlands), a research project² has been developed, proving the utility of the IoT paradigm to some of the aforementioned problems. Eight Meshlium IoT Gateways of Libelium²² equipment were installed at the intersections of several of its streets, these serve to ensure that the data received by the sensors reach a platform that allows correct information management. Meshlium devices are synchronized by an external digital clock and through a wi-fi scanner by detecting physical addresses (mac addresses) of smartphones and vehicle hands-free devices. Data collection is carried out by accessing the local database of each sensor and then being stored in a PostgreSQL database, once the persistence of data is verified, these are analyzed using Python scripts and displayed on a geographic map thanks to the QGIS tool. With the collected data and knowing the distance relationship between two sensors, the amount of movement of each detected device can be obtained.

Taking into account the above, together with the criteria for the use of each street in the city on which monitoring was being carried out; three types of users of these devices can be recognized: pedestrians, cyclists and vehicles. This has served to

study the relationship between these three categories throughout the day, which has been useful to identify the preferred streets for each type of user, as well as to know the habits of users over time.

It is clear that all the information available, and especially that related to traffic movement and density, as in this work, allows defining the most effective routes when traveling within and outside cities, avoiding traffic congestion, which justifies the proposal in the present project.

1.7.2 Objectives

The general objective of this project is the development of a conceptual application that can be implemented in the framework of the smart cities in a near future, for the detection of traffic flow (vehicles) in the access roads, it will be made through the analysis of the images detected in videos.

Most part of the processing will be made on the client side of the proposed application, by using processing and graphic units, supporting the required computational load for retraining a convolutional neural network and for detecting objects (vehicles) in images as fast as possible.

The resulting data of the processed information on the client side will be transferred to a ThingSpeak²⁴ remote server, there they will be stored for its visualization, queries and other processing tasks.

As aforementioned, a smart conceptual application is proposed, including Computer Vision⁴ and Deep Learning²⁷ techniques, integrated under an IoT context, generating, by itself, the main objective developed on this work.

The previous approach results in the following specific objectives:

- Development of techniques based on optical flow analysis of images for the detection of vehicles in movement.
- Development of image segmentation techniques to extract the regions in movement through umbralization and labeling of associated regions.
- To develop a method to compute vehicles in image sequences to upload data in the cloud.
- To develop a module for transfer learning through the use of pre-trained and re-trained convolutional neural networks.

- Development of techniques to get access to the cloud and for data processing inside the cloud.
- Development of techniques based on predictive analysis that can be applied to the traffic of vehicles in the cloud.
- Development of friendly user interfaces to capture images and for data visualization and to display the results derived from data.
- To integrate the previous techniques and modules under a software-based architecture.

1.7.3 Work plan

Specific tasks to reach the previous objectives are synthetized as follows,

- **Optical flow analysis:** study and implementation of techniques for detecting objects (vehicles) in movement from the images. An IDE (*Integrated Development Environment*) has been developed with the appropriate functionalities.
- **Umbralization, labeling and extraction of regions in movement:** study and implementation of image segmentation techniques to determine candidate regions involving vehicles in movement.
- **To identify the convolutional neural network model and to apply transfer learning:** There are a lot of pre-trained convolutional neural networks models with thousands an even millions of images, one of them is to be selected with sufficient efficiency.
- **Image processing and data augmentation for validation and training:** Considering AlexNet⁹ as one of the most promising, this task consists in the generation of a datastore to acquire the specific parameters from the images containing vehicles in movement.
- **Configuration of training process of Alexnet model:** study of the AlexNet training parameters, including: number of *epochs*, *batch size*, *gradient optimization method* and *learn rate*.
- **To develop a system for vehicle counting:** study and implementation of methods for counting vehicles crossing a specific area in the image, once vehicles have been identified.

- **Generation of training and classification reports:** the goal of this task is to provide additional information to the user about the training and validation processes.
- **Develop a client-server architecture:** focused on read and write calls. When writing, the server will store the number and types of vehicles detected. When reading, specific fields of ThingSpeak²⁴ channel is accessed to be displayed.
- **Server side programming:** to define applications and functionalities provided by ThingSpeak, including: ThingTweet to link a Twitter account; React, to send warnings on a Twitter account when new data are written in the appropriate channel.
- **To develop a cloud algorithm for traffic flow prediction:** MATLAB Analysis container of ThingSpeak will be used to configure an AR model from data on a specific point of M30 Madrid ring road.
- **Design and development of user interfaces:** study of elements and functionalities provided by the MATLAB IDE (*Integrated Development Environment*) and MATLAB App Designer. Intuitive and friendly user's interfaces is the goal.
- **Design and development of a multilayer software architecture:** study and analysis to structure the application as a multilayer scheme by applying software modeling, including class diagrams, use cases and deployment. Implementation, integration and testing phases are considered.

2. Métodos conceptuales aplicados

La aplicación desarrollada se fundamenta en dos pilares fundamentales del campo de la Inteligencia Artificial: Visión por Computador⁴ y Aprendizaje Profundo²⁷.

Las técnicas de Visión por Computador han servido para el análisis del movimiento, mientras que el Aprendizaje Profundo mediante Redes Neuronales Convolucionales ha servido para poder identificar las partes en movimiento dentro de imágenes.

2.1 Visión por Computador

Los conocimientos de Visión por Computador se aplican en el orden que se describe a continuación.

En primer lugar se aplican las técnicas de detección de movimiento basadas en el cómputo del flujo óptico para la identificación de las regiones en movimiento, que son representativas de los posibles vehículos en circulación.

Sobre estas regiones, identificadas como potenciales vehículos en movimiento, se aplican técnicas de segmentación de imágenes para poder calcular algunas de sus propiedades, tales como sus áreas, centroides y rectángulos delimitadores. Estas permiten discriminar las áreas candidatas que no son de interés, así como computar el número de vehículos que atraviesan una línea imaginaria en la vía, cuyo objetivo es determinar el número de vehículos en tránsito, es decir, el flujo de éstos en la vía pública.

2.1.1 Cálculo del flujo óptico

El flujo óptico es una técnica para el cálculo del movimiento de los objetos que tiene lugar entre dos secuencias (*frames*) de vídeo consecutivos separados por un intervalo de tiempo reducido (dt) (Pajares y Cruz, 2007)⁴. La obtención de éste permite determinar la posición, dirección y velocidad de un objeto dentro de un *frame*. Este cálculo se realiza sobre todos los píxeles de un *frame* de vídeo.

Uno de los métodos más empleados para el cálculo del flujo óptico está basado en el cómputo del gradiente, el cual representa los cambios del nivel de intensidad en una imagen.

Para llegar a su concepción se puede pensar en la obtención de la intensidad de un píxel en la posición (x,y) en un instante de tiempo (t) a través de la función $f(x,y,t)$. Si además la imagen es dinámica en el tiempo, se puede pensar en encontrar el mismo píxel pero en otro instante de tiempo muy muy cercano (dt) en el que apenas ha habido desplazamiento (dx,dy). Esto se sintetiza en la ecuación (2.1).

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

Si se desarrolla el lado izquierdo de la ecuación (2.1) como un polinomio de Taylor de primer orden, se obtiene el resultado mostrado en la ecuación (2.2).

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

En este desarrollo las derivadas se interpretan como las diferencias existentes entre las intensidades de los píxeles adyacentes.

Ahora podemos despreciar el término de orden superior, obteniendo una igualdad aproximada, luego de hacer esto y despejar la ecuación, llegamos a (2.3).

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

De esta forma se obtiene una ecuación que indica efectivamente que la diferencia de intensidad f , en una misma posición e instante de tiempo se debe a la diferencia de intensidad espacial, que es debida a la cantidad de movimiento.

Asumiendo que la intensidad de la imagen permanezca constante a lo largo del tiempo ($\delta f / \delta t = 0$) y que el vector gradiente de la ecuación (2.3) sea constante, se llega a la conclusión reflejada en la ecuación (2.4).

$$\begin{bmatrix} \delta^2 f / \delta x^2 & \delta^2 f / \delta x \delta y \\ \delta^2 f / \delta x \delta y & \delta^2 f / \delta y^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = -\frac{\delta(\nabla f)}{\delta t} \quad (2.4)$$

Si además se considera un entorno de vecindad Ω para todos los puntos, entonces la ecuación se transforma en la (2.5).

$$\begin{bmatrix} \sum_{\Omega} f_x^2 & \sum_{\Omega} f_x f_y \\ \sum_{\Omega} f_x f_y & \sum_{\Omega} f_y^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = - \begin{bmatrix} \sum_{\Omega} f_t f_x \\ \sum_{\Omega} f_t f_y \end{bmatrix} = A \mathbf{v} = \mathbf{b} \quad (2.5)$$

Aplicando mínimos cuadrados a la ecuación (2.5) se llega a la solución, que aparece en la ecuación (2.6).

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

De forma que se puede obtener el flujo óptico $\mathbf{V}(\mathbf{u}, \mathbf{v})$ para todo píxel de una imagen, cuyas componentes horizontal y vertical son \mathbf{u} y \mathbf{v} , respectivamente.

2.1.2 Detección de regiones en movimiento

A partir del cálculo del flujo óptico $\mathbf{V}(\mathbf{u}, \mathbf{v})$ en todos los puntos de una imagen, se pueden identificar los objetos en movimiento presentes en ésta, ya que estos vectores contienen y representan implícitamente la magnitud y sentido del movimiento producido en cada píxel.

El módulo o magnitud del flujo óptico se utiliza para calcular el valor medio del movimiento y la desviación estándar de éste. La suma de la media y la desviación estándar se utiliza para determinar un valor umbral, que sirve para realizar un proceso de binarización sobre una imagen, proporcionando como resultado una imagen en blanco y negro (Pajares y Cruz, 2007)⁴.

Sobre esta imagen binaria se realiza un proceso de etiquetado de las distintas componentes conexas, teniendo cada una de éstas el mismo identificador para cada uno de los píxeles que las componen.

Para realizar este proceso de etiquetado se aplica el algoritmo de Haralick y Shapiro (1992)³, un algoritmo de dos fases que emplea una tabla de equivalencias.

En la primera fase se realiza un recorrido sobre la imagen binarizada, de todas las filas de arriba hacia abajo y 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 se la puede propagar a éste.

En relación a la aplicación de este algoritmo, se distinguen varios casos:

- 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 fase anterior pero 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.

A continuación se muestra un ejemplo ilustrativo de este proceso. Se supone que se ha capturado una imagen de tráfico en la que aparecen varios vehículos cuyo movimiento ha hecho que se hayan detectado en varios píxeles un movimiento con una magnitud que es superior al valor umbral fijado. Si se aplica a dicha imagen el proceso de binarización se obtiene una imagen binaria como la de la figura 2.1.

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	1	1	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0
0	1	1	1	1	0	0	1	1	0	0	0	0	0	0	0	0	0	0	0
0	1	1	1	1	0	1	1	1	0	0	0	0	1	1	0	0	0	0	0
0	1	1	1	1	0	1	1	1	0	0	0	0	1	1	0	0	0	0	0
0	1	1	1	1	0	1	1	1	0	0	0	0	1	1	0	0	0	0	0
0	1	1	1	1	0	1	1	1	0	1	1	1	1	1	1	1	1	1	0
0	1	1	1	1	0	1	1	1	0	1	1	1	1	1	1	1	1	1	0
0	1	1	1	1	0	0	0	0	0	0	0	1	1	1	1	1	1	1	0
0	1	1	1	1	0	0	1	1	0	0	1	1	1	1	1	1	1	1	0
0	1	1	1	1	0	0	1	1	0	0	1	1	1	1	1	1	1	1	0
0	1	1	1	1	0	0	1	1	0	0	1	1	1	1	1	1	1	1	0
0	1	1	1	1	0	0	1	1	0	0	0	0	0	1	1	0	0	0	0
0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	1	1	0	0	0
0	0	0	0	0	0	0	1	1	1	0	0	0	0	0	1	1	0	0	0
0	0	0	0	0	0	0	1	1	1	0	1	1	1	1	1	1	1	1	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Figura 2.1 Ejemplo ilustrativo de etiquetado de componentes conexas.

Al realizar la primera fase del algoritmo se obtienen los resultados de la figura 2.2.

0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	1	1	1	1	0	0	2	2	0	0	0	0	0	0	0	0	0	0	0
0	1	1	1	1	0	0	2	2	0	0	0	0	0	0	0	0	0	0	0
0	1	1	1	1	0	2	2	2	2	0	0	0	0	3	3	0	0	0	0
0	1	1	1	1	0	2	2	2	2	0	0	0	0	3	3	0	0	0	0
0	1	1	1	1	0	2	2	2	2	0	4	4	3	3	3	3	3	0	0
0	1	1	1	1	0	2	2	2	2	0	4	3	3	3	3	3	3	0	0
0	1	1	1	1	0	0	0	0	0	0	3	3	3	3	3	3	3	0	0
0	1	1	1	1	0	0	0	0	0	0	3	3	3	3	3	3	3	0	0
0	1	1	1	1	0	0	5	5	0	0	3	3	3	3	3	3	3	3	0
0	1	1	1	1	0	0	5	5	0	0	3	3	3	3	3	3	3	3	0
0	1	1	1	1	0	0	5	5	0	0	3	3	3	3	3	3	3	3	0
0	1	1	1	1	0	5	5	5	5	0	3	3	3	3	3	3	3	3	0
0	0	0	0	0	0	5	5	5	5	0	0	0	0	3	3	0	0	0	0
0	0	0	0	0	0	5	5	5	5	0	0	0	0	0	3	3	0	0	0
0	0	0	0	0	0	5	5	5	5	0	6	6	3	3	3	3	3	3	0
0	0	0	0	0	0	0	0	0	0	0	6	3	3	3	3	3	3	3	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Tabla de equivalencias

$1 \rightarrow 1$
 $2 \rightarrow 2$
 $3 \rightarrow 3$
 $4 \rightarrow 3$
 $5 \rightarrow 5$
 $6 \rightarrow 3$

Figura 2.2 Final de la fase 1 del algoritmo de Haralick y Shapiro.

Tras la aplicación del proceso correspondiente a la segunda fase del algoritmo se obtiene el resultado de la figura 2.3.

Figura 2.3 Final de la fase 2 del algoritmo de Haralick y Shapiro.

Una vez finaliza el algoritmo se obtienen una serie de propiedades de la imagen, una de las más importantes es la obtención de la ubicación y el tamaño de las áreas conexas, etiquetadas y delimitadas por un rectángulo (*Bounding Box*). Su obtención se realiza a través de la comparación de aquellas zonas que tengan una magnitud de movimiento significativo determinado por el valor umbral, previamente obtenido, entre dos *frames* de vídeo consecutivos.

Una vez se obtienen todas las regiones candidatas que son representativas del movimiento, éstas se ubican y localizan sobre la imagen original, realizando un recorte sobre ésta usando las coordenadas del rectángulo delimitador (*Bounding Box*). Este recorte es el que se proporciona a la Red Neuronal Convolutacional como entrada, para que trate de identificar el tipo de vehículo en movimiento (FrontCar, BackCar, FrontMotorbike, BackMotorbike, FrontTruckVan, BackTruckVan, FrontBus, BackBus).

Otra de las propiedades de interés que también se calcula son los centroides de las áreas en las que se ha detectado un movimiento superior al valor umbral. Esto ha permitido implementar un sistema de conteo de vehículos, ya que podemos ubicar dichos centroides dentro de una franja delimitada en la imagen que se procesa.

2.2 Redes Neuronales Convolucionales

Las Redes Neuronales Convolucionales (*Convolutional Neural Networks*, CNN) están ubicadas dentro de lo que se conoce como Aprendizaje Profundo (Deep Learning)²⁷. Éstas poseen unas estructuras de rejilla, donde se ubican principalmente los pesos aprendidos de la red y que permiten obtener los mapas de características cuando se procesan los datos de entrada, imágenes en este caso.

Estos dos tipos de estructuras también son conocidas como tensores, los cuales tienen un volumen determinado pero variable en función de las capas de la red, por lo que poseen ancho, alto y profundidad.

Se llaman convolucionales por hacer uso de la operación de convolución, ésta se aborda desde el punto de vista de las redes neuronales, que no se corresponde con el mismo concepto en matemáticas o en la teoría de procesamiento de señales.

En este proyecto se ha optado por utilizar AlexNet⁹, ya que ésta hace uso de una arquitectura secuencial, más simple que otras CNN como GoogleNet. Se trata de un modelo cuya arquitectura se muestra en la figura 2.4, lo conforman 25 capas.

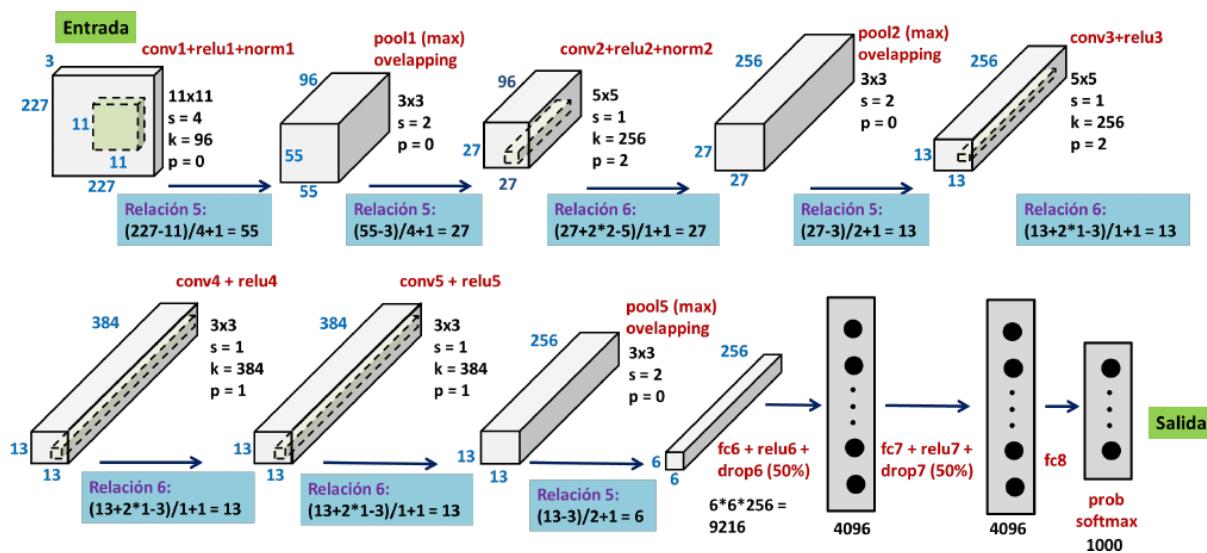


Figura 2.4 Arquitectura de la Red Neuronal Convolutacional AlexNet.

Este modelo está pre-entrenado, lo que permite aprovechar su contenido para llevar a cabo un proceso de re-entrenamiento con los datos de los vehículos específicos de nuestra aplicación. Para ello han rediseñado las capas 23 y 25 (Fully Connected y Classification) de forma que ambas cuenten con tantas neuronas como clases tiene el problema de clasificación de vehículos, sustituyendo así las 1000 clases de imágenes de la red original.

Esta técnica es lo que se conoce como “transferencia de aprendizaje”¹³ y permite reutilizar los núcleos convolucionales de la CNN AlexNet original para re-adaptarlos al problema de clasificación de vehículos en lugar de tener que entrenar la red desde cero para obtenerlos, lo cual es más costoso y requiere de mucho tiempo de cómputo.

Los recortes sobre la imagen original, obtenidos según se explica durante el proceso de detección y segmentación de imágenes, descrito previamente, constituyen las entradas a la red, que en el caso del modelo AlexNet necesita un redimensionado a 227 x 227 píxeles en las dimensiones ancho y alto, independientemente del tamaño del recorte. Esto es así debido a los requisitos del propio diseño de esta red.

A continuación se explican las capas que componen el modelo de la Red Neuronal Convolucional AlexNet y las operaciones involucradas en cada una de ellas. Éstas son básicamente: convolución (conv), ReLU (relu), *pooling* (pool), normalización (norm), *fully connected* (fc) o capa totalmente conectada, y *softmax*.

En la figura 2.4 se muestran las distintas capas de este modelo, identificadas por su tipo y un número identificativo.

Por otra parte, a través de las relaciones 5 y 6, tal y como están definidas en este modelo y que podemos ver en la figura 2.4, se determinan las dimensiones de cada bloque, o mapa de características de cada capa, a partir de las dimensiones del bloque precedente y de las operaciones aplicadas. Las relaciones son las siguientes:

Para cada i (dimensión de entrada), k (tamaño del núcleo del filtro de convolución), $s > 1$ (pasos de desplazamiento del núcleo) y $p = 0$ (padding, añadido de ceros en las filas y columnas externas). La salida o , se obtiene como sigue:

$$\text{Relación 5: } o = [i - k]/s + 1 ; \text{ Relación 6: } o = [i - 2p - k]/s + 1$$

A modo de ejemplo en la primera capa se aplica un núcleo de convolución de tamaño 11×11 , por lo que $k = 11$, para una entrada (imagen) de tamaño 227×227 y por tanto $i = 227$, con $s = 4$ y $p = 0$, para obtener como salida $o = 55$, por aplicación de la Relación 5.

2.2.1 Capas del modelo AlexNet

a) Capa Convolution

Convolución en el ámbito del procesamiento de señales:

Para entender de forma más intuitiva lo que implica la operación de convolución se parte de un ejemplo ilustrativo (Romero-Pérez, J. 2019)⁸.

Supongamos que tenemos una fuente de luz variable cuya intensidad es percibida por un sensor fotoeléctrico, éste genera una salida en un instante de tiempo t , es decir $x(t)$, siendo tanto t como $x(t)$ números reales.

Debido a las variaciones de la fuente de luz, se pueden obtener diferentes lecturas en distintos instantes de tiempo, además la captura de la señal puede estar contaminada por ciertos ruidos de distinta procedencia. Por lo que para obtener una señal más limpia calcularemos el promedio de las salidas obtenidas a lo largo del tiempo.

Se realiza una ponderación dando más relevancia a las lecturas más recientes que a aquellas más distantes en el tiempo, ésto se consigue mediante la función $w(a)$, donde a representa lo que la medición se ha alejado en el tiempo.

La realización de este promedio ponderado en cada instante de tiempo puede expresarse como una nueva función como la proporcionada en la ecuación (2.7).

$$s(t) = \int x(a) w(t - a) da \quad (2.7)$$

Esta operación se denomina convolución y se suele representar como en (2.8).

$$s(t) = \int (x * w)(t) dt \quad (2.8)$$

Hay que puntualizar que w ha de ser una función de densidad de probabilidad para asegurar que la salida esté promediada y que devuelva cero cuando su valor de entrada sea negativo, con esto último evitamos la toma de valores en instantes de tiempo futuros, algo que no es posible.

Convolución en el ámbito de las Redes Neuronales Convolucionales:

Desde un punto de vista computacional se trabaja con variables discretas, es decir, sólamente con valores enteros, por lo que se define la convolución discreta dada por la ecuación (2.9).

$$s(t) = (x * w)(t) = \sum_{a=-\infty}^{\infty} x(a) w(t - a) \quad (2.9)$$

En el ámbito de las CNN a la variable x se la conoce como entrada (input), a w se la conoce como núcleo de convolución (convolution kernel) y a la salida s se le da el nombre de mapa de características (*feature map*).

La entrada es un vector o una matriz de una o más dimensiones que contiene datos. El núcleo convolucional por lo general también es un vector o matriz de una o varias dimensiones, solo que este último alberga los valores de una serie de parámetros que serán ajustados durante el proceso de entrenamiento.

Estas estructuras multidimensionales reciben el nombre de tensores y sus elementos se almacenan por separado, cualquier valor que no pertenezca a una de estas dos estructuras se considera nulo, por lo que esta operación se traduce en una suma finita de valores para un número finito de elementos.

En este ámbito las convoluciones se realizan sobre más de un eje a la vez, ya que se trabaja sobre una imagen bidimensional I que actúa como entrada, ésta será tratada por otra estructura bidimensional K , un núcleo convolucional, como vemos en (2.10).

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(m,n) K(i-m, j-n) \quad (2.10)$$

La convolución es conmutativa por lo que podemos reescribirla como en (2.11), esta forma resulta en una menor variación en los valores de ***m*** y ***n***, lo que la haría más eficiente en términos computacionales.

$$S(i,j) = (K * I)(i,j) = \sum_m \sum_n I(i-m, j-n) K(m,n) \quad (2.11)$$

La conmutación se interpreta como la reflexión del núcleo respecto de la entrada.

Otra concepción matemática que también se utiliza mucho debido a su sencillez y que suele confundirse con la convolución es la correlación cruzada (2.12), la cual tiene un efecto muy parecido a la operación de convolución.

$$S(i,j) = (K * I)(i,j) = \sum_m \sum_n I(i+m, j+n) K(m,n) \quad (2.12)$$

A continuación podemos ver en la figura (2.5) un ejemplo de correlación cruzada.

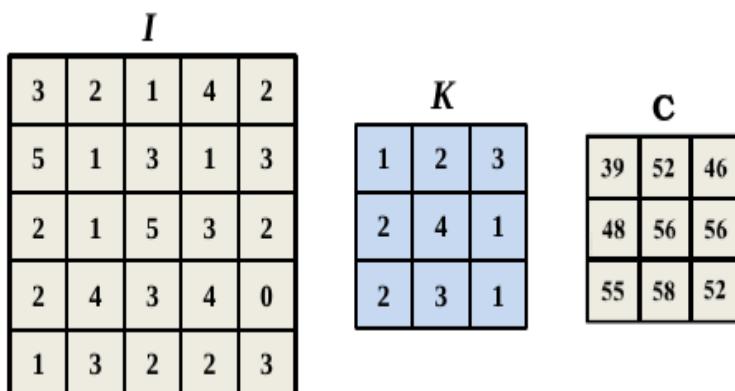


Figura 2.5 Ejemplo de aplicar correlación cruzada a la imagen I, con el núcleo K, obteniendo C.

b) Capa Pooling

Esta capa sirve para obtener una nueva representación aproximadamente invariante a partir de pequeñas traslaciones de la entrada (Romero-Pérez, J. 2019)⁸.

Existen dos tipos de operaciones de agrupación (Zhou y Chellappa, 1988)⁵:

- a) Máximo (Max-Pooling), consiste en dividir la imagen de entrada en varias ventanas, sin solapamiento entre ellas, produciendo como salida el máximo de cada ventana.

- b) Media (Average-Pooling), divide la imagen de entrada en diversas ventanas, sin solapamiento, produciendo como salida la media de cada una de ellas.

Esta operación ha demostrado que si se aplica una pequeña traslación a la imagen de entrada los valores de muchas salidas sobre las que se ha aplicado el *pooling* no cambian, esto se interpreta como que alguna función se repite sobre subconjuntos (ventanas) de la imagen de entrada (Romero-Pérez, J. 2019)⁸, lo cual es útil ya que por ejemplo si se realiza *pooling* sobre la salida de la convolución de una imagen es posible aprender qué transformaciones son invariantes para una misma imagen de entrada .

La operación de *pooling* se usa para identificar si alguna característica está presente en la imagen de entrada.

En la figura 2.8 se muestra un ejemplo práctico de cómo una imagen binaria puede dividirse en varias ventanas de igual cantidad de píxeles.

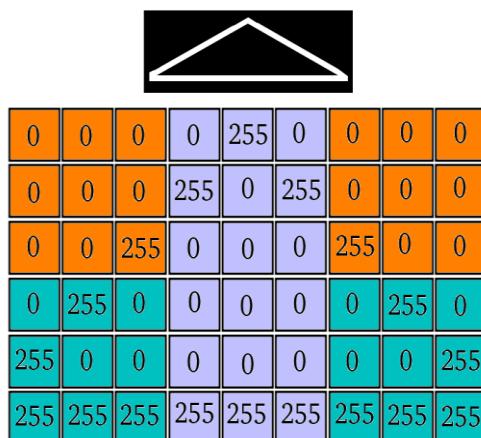


Figura 2.8 Imagen binaria y debajo de esta, ventanas de 3x3 sobre sus distintas regiones.

Aplicando sobre las ventanas las dos operaciones de pooling anteriores obtenemos los resultados de la figura 2.9

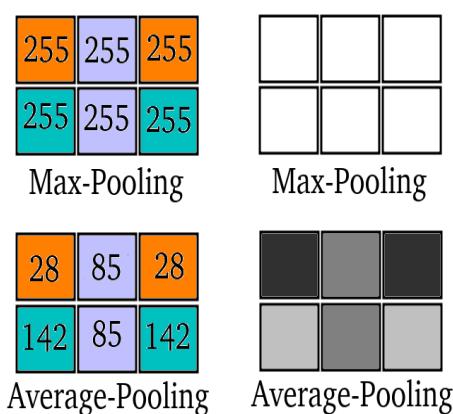


Figura 2.9 Resultado matemático y visual de aplicar Max-pooling y Average-Pooling.

c) Capa Dropout

Un problema muy típico de las redes neuronales es el “overfitting” (sobreajuste).

En las CNN se ajustan un número elevado de pesos en varias neuronas, la cuestión es cómo realizar el mejor ajuste posible de éstos. Hay varias propuestas para abordar este problema, como por ejemplo generar un modelo que pase por todos los puntos mediante el uso de un polinomio de alto grado o la generación de un modelo lineal que se ajuste lo mejor posible a todos los puntos (Romero-Pérez, J. 2019)⁸.

La figura 2.10 muestra de forma ilustrativa los problemas de cada enfoque. En el modelo lineal el ajuste a los datos no es suficientemente bueno (*underfitting*), mientras que en el modelo polinómico el ajuste a los datos es bueno, pero falla cuando hacemos uso de este para la realización de clasificaciones (*overfitting*).

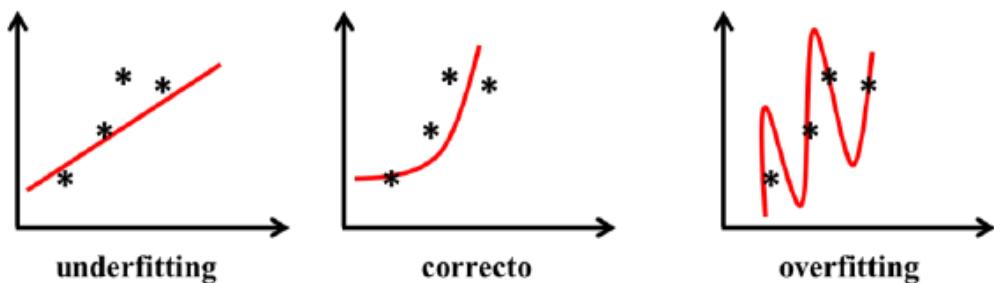


Figura 2.10 Problemas del ajuste lineal y el ajuste polinómico, se requiere algo intermedio.

Para paliar el efecto del *overfitting* se propone desconectar o anular (*dropout*) determinado número de neuronas para prevenir que éste se produzca al ajustar los pesos (Srivastava y col., 2014)⁶ durante el proceso de entrenamiento.

La selección de las neuronas a desconectar se realiza de forma aleatoria, de esta forma la CNN queda definida por las neuronas que sobreviven al efecto de dropout. Podemos ver un ejemplo ilustrativo de esto en la figura 2.11.

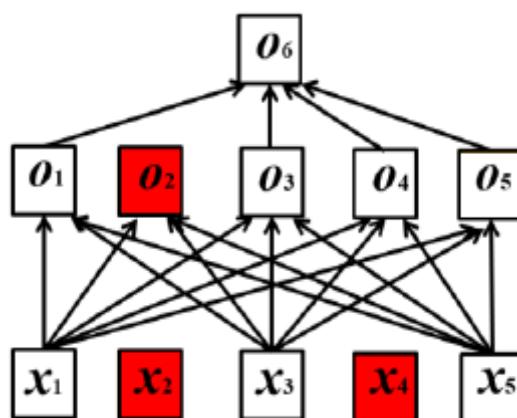


Figura 2.11 Ejemplo visual de la desconexión neuronal aleatoria que produce dropout.

Otra estrategia que se podría usar es añadir un hiperparámetro a las neuronas, éste se usaría para asignar una probabilidad que regule la influencia de sus pesos (Romero-Pérez, J. 2019)⁸.

d) Capa totalmente conectada (Fully Connected)

Todas las neuronas de esta capa reciben una entrada de todas las neuronas de la capa anterior, éstas son representadas por la función mostrada en la ecuación (2.13), que sigue un modelo lineal.

$$y = \sum_i^n (x_i * w_{ij}) + b_j \quad (2.13)$$

Donde x_i representa el valor de una característica obtenida en una capa anterior, w_{ij} es el peso o la importancia que se le concede al valor obtenido de la neurona i de la capa anterior en la neurona actual j , la constante b es lo que se conoce como “bias” o umbral de disparo y sirve para controlar si la neurona debe activarse (Brío y Sanz-Molina, 2006)⁷.

El objetivo de las neuronas en estas capas es la de aplicar funciones lógicas (AND, OR, NOT, etc) sobre las características de la entrada que le son pasadas para así lograr la extracción de características a distintos niveles. La capa de entrada busca obtener las características primarias de una imagen, como los bordes. En las capas ocultas se obtienen características de otros rasgos tales como los contornos, y finalmente la capa de salida está orientada a describir el objeto de forma general.

En el modelo de Red Neuronal Convolutacional AlexNet⁹ se dispone de tres de estas capas puestas de forma casi consecutiva ya que necesitan ser seguidas inmediatamente por una capa de tipo función de activación que aplique una modificación no lineal a la salida de la neurona, ya que de lo contrario, estas capas colapsarían y serían equivalentes a una única neurona, lo que sería desastroso, pues con una única neurona no se puede realizar el suficiente número de operaciones lógicas como para poder definir una imagen.

e) Capa ReLU (Rectified Linear Unit)

Las funciones de activación son utilizadas para indicar de forma matemática el grado en que se posee una característica, muy presente o poco presente.

Una de las funciones de activación más clásica es la función sigmoide, ésta se define matemáticamente como se indica en la ecuación (2.14) cuya representación gráfica se muestra en la figura 2.6.

$$f(a, x, c) = \frac{1}{1 + e^{-a(x-c)}} \quad (2.14)$$

La función sigmoide proyecta, para todo valor real de su entrada, un valor de salida en el intervalo $[0, 1]$, si bien posee dos problemas:

1. La rápida saturación del gradiente. El valor de la función de activación tiende a los extremos, mayoritariamente a 0, lo que afecta al ajuste de los pesos.
2. La continua obtención de pesos positivos, ya que el valor medio no es 0.

Otra función de activación que también es muy utilizada es la tangente hiperbólica (\tanh), cuyas salidas son números reales en el rango $[-1, +1]$. Esta función es una variante de la función sigmoide, ya que esta se incluye en su definición, la cual se describe como: $\tanh(x) = 2 * \text{sigmoid}(2x) - 1$. También presenta el problema de la saturación del gradiente.

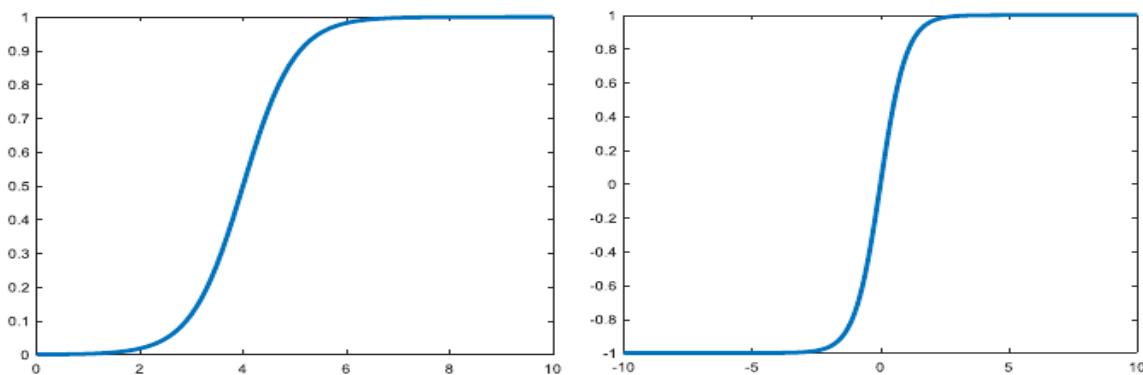


Figura 2.6 Función sigmoide a la izquierda, función tangente hiperbólica a la derecha.

Actualmente, la función que más se usa en este ámbito es la función ReLU (Rectified Linear Unit), definida como $f(x) = \max(0, x)$ y representada en la figura 2.7. Las ventajas de esta función son (Romero-Pérez, J. 2019)⁸:

- a) Gradiente no saturado para cualquier valor positivo.
- b) Es computacionalmente simple.

Las desventajas de la función ReLU, proviene del hecho de que si todos los datos que pasan por esta capa resultan con valor cero debido a la configuración de los pesos de una neurona anterior, entonces esta neurona resulta ser un problema, ya que al intentar aplicar el método del descenso del gradiente, se obtiene un gradiente nulo, lo que no permite realizar ajustes en los pesos hacia atrás durante la aplicación del algoritmo de retropropagación. Cuando esto le ocurre a una neurona ReLU se dice que ésta está muerta. Para evitar esta situación se puede usar la función Leaky ReLU (ReLU con fugas), definida como: $f(x) = \max(0.001x, x)$.

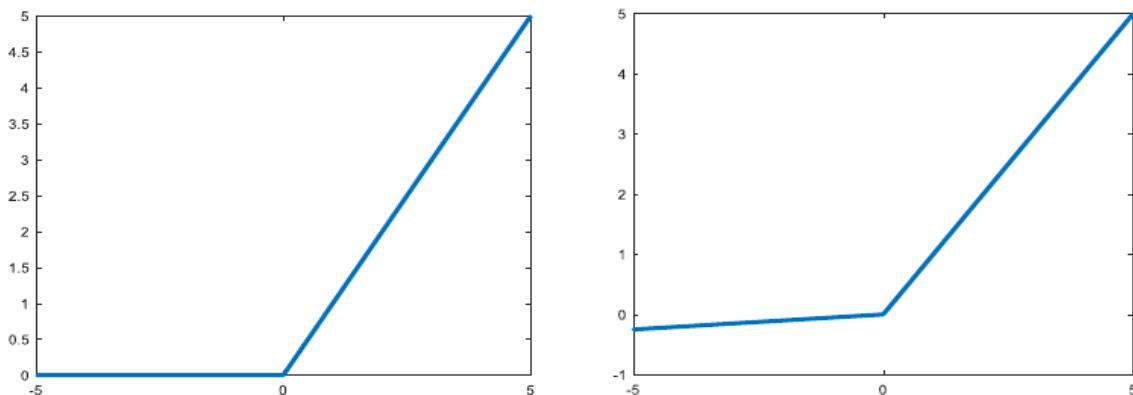


Figura 2.7 Gráficas de ejemplo, función ReLU a la izquierda y función Leaky Relu a la derecha.

f) Capa Normalization

El objetivo de esta capa es la de normalizar la salida de la capa ReLU para mejorar la convergencia hacia el error mínimo durante el entrenamiento. Para lograrlo aplica una normalización de valores a nivel de canal en el mapa de características. Por lo general, la normalización ajusta los valores de los canales al intervalo [0,1] o [-1,1].

Esto es importante ya que hay casos en los que las neuronas reciben valores muy grandes, y al mismo tiempo otros demasiado pequeños, lo que resulta perjudicial pues los valores demasiado grandes influyen demasiado sobre el resultado que se va a obtener mientras que los valores pequeños prácticamente no tienen ningún efecto sobre éste.

g) Capa Softmax

La función softmax o función exponencial normalizada aparece por lo general en las últimas capas ocultas de una Red Neuronal Convolutiva, ésta se define según la ecuación (2.15).

$$\text{softmax}(x)_i = \frac{\exp(x_i)}{\sum_{j=1}^n \exp(x_j)} \quad \forall i = 1 \dots n; x_j = (x_1, \dots, x_n) \in \mathbb{R}^n \quad (2.15)$$

Esta función se aplica a cada elemento x_j de un vector de entrada n -dimensional x , normalizando los valores de éste en el rango [0,1] mediante la aplicación de la función exponencial y dividiendo entre la suma de las exponenciales. Se genera de esta forma un nuevo vector n -dimensional $\text{softmax}(x)_i$ cuya suma de elementos es 1, pudiendo interpretarse este hecho de tal forma que la suma de todos los grados de pertenencia de una imagen a todas las clases del problema de clasificación ha de ser exactamente 1.

2.3 Series temporales: Autorregresión

Una serie temporal se define como la sucesión de datos medidos en momentos determinados y ordenados cronológicamente en el tiempo. Los datos que conforman la serie se toman teniendo en cuenta la misma duración e intervalos. A modo de ejemplo, los que se utilizan en este trabajo, descritos en los resultados, son datos que reflejan el flujo de vehículos en un punto kilométrico de la carretera de circunvalación M30 en Madrid durante una franja horaria y calculados todos los días, sólo los lunes, etc.

La ecuación (2.16) define formalmente una serie temporal (Mauricio, 2007)¹⁶.

$$y_t = y_t, y_{t-1}, \dots, y_N; t = 1, \dots, N \quad (2.16)$$

Donde N es el número de observaciones, y cada uno de los valores y_t es la observación realizada en instante t de la serie.

Las series temporales ofrecen la posibilidad de poder realizar predicciones en el tiempo, es decir estimar valores futuros de la variable y_t en un tiempo t futuro. Para ello, se definen modelos matemáticos parametrizados, siendo los parámetros los valores a estimar a partir de los datos disponibles, permitiendo realizar la predicción. Uno de tales modelos es el denominado ARMA (AutoRegressive Moving Average), como su nombre indica de naturaleza autorregresiva. A esta categoría pertenece el modelo autorregresivo AR(p), donde p indica el orden del modelo, a partir del cual se determina el número de parámetros a estimar. Este modelo se define según la ecuación (2.17).

$$y_t = c + \Phi_1 y_{t-1} + \Phi_2 y_{t-2} + \dots + \Phi_p y_{t-p} + \varepsilon_t \quad (2.17)$$

Donde c es una constante, $\Phi_1, \Phi_2, \dots, \Phi_p$ se corresponden con los mencionados parámetros a estimar del modelo y ε_t es ruido blanco. En el modelo de predicción finalmente utilizado, el término de ruido blanco se obvia.

Una vez definido el modelo, el siguiente paso consiste en encontrar el mejor valor del orden p del modelo y por tanto los valores de los parámetros, para lograrlo es práctica habitual aplicar las conocidas ecuaciones de Yule-Walker (Walker, 1931; Hochreiter y Schmidhuber, 1997)^{17 18} ya que existe una relación entre los parámetros que se quieren averiguar y la función de covarianza del proceso. Se aplica la regla de derivación en cadena a las ecuaciones para obtener los parámetros del modelo.

Una vez estimados los parámetros del modelo, el proceso de predicción con un horizonte temporal dado “n”, que expresa el número de predicciones a futuro, se realiza estableciendo una predicción base, que sirve para la siguiente y estas dos

para una tercera y así sucesivamente hasta conseguir el límite de n , según la ecuación (2.18). El número de valores de la serie a utilizar durante la predicción depende del número de parámetros del modelo.

Así por ejemplo, para un modelo con $p = 2$, se tienen dos parámetros, de forma que se necesitan siempre dos valores de datos. Así, en fase de predicción para el primer valor predicho se utilizan los dos reales anteriores, para el segundo valor predicho intervienen el último real y el predicho, para la tercera predicción se utilizan los últimos predichos y a partir de aquí siempre los dos últimos valores predichos.

$$y_{t+n} = c + \Phi_1 y_{t-1+n} + \Phi_2 y_{t-2+n} + \dots + \Phi_p y_{t-p+n} \quad (2.18)$$

3. Diseño y análisis de la plataforma

Una vez establecidos los objetivos y el marco teórico para su desarrollo, a continuación se describe el diseño de la plataforma, que permite su implementación. En el esquema de la Figura 3.1, se muestra un esquema general de dicho diseño con los elementos que componen la arquitectura del sistema global y las comunicaciones que permiten el trasvase de datos e información entre ellos.

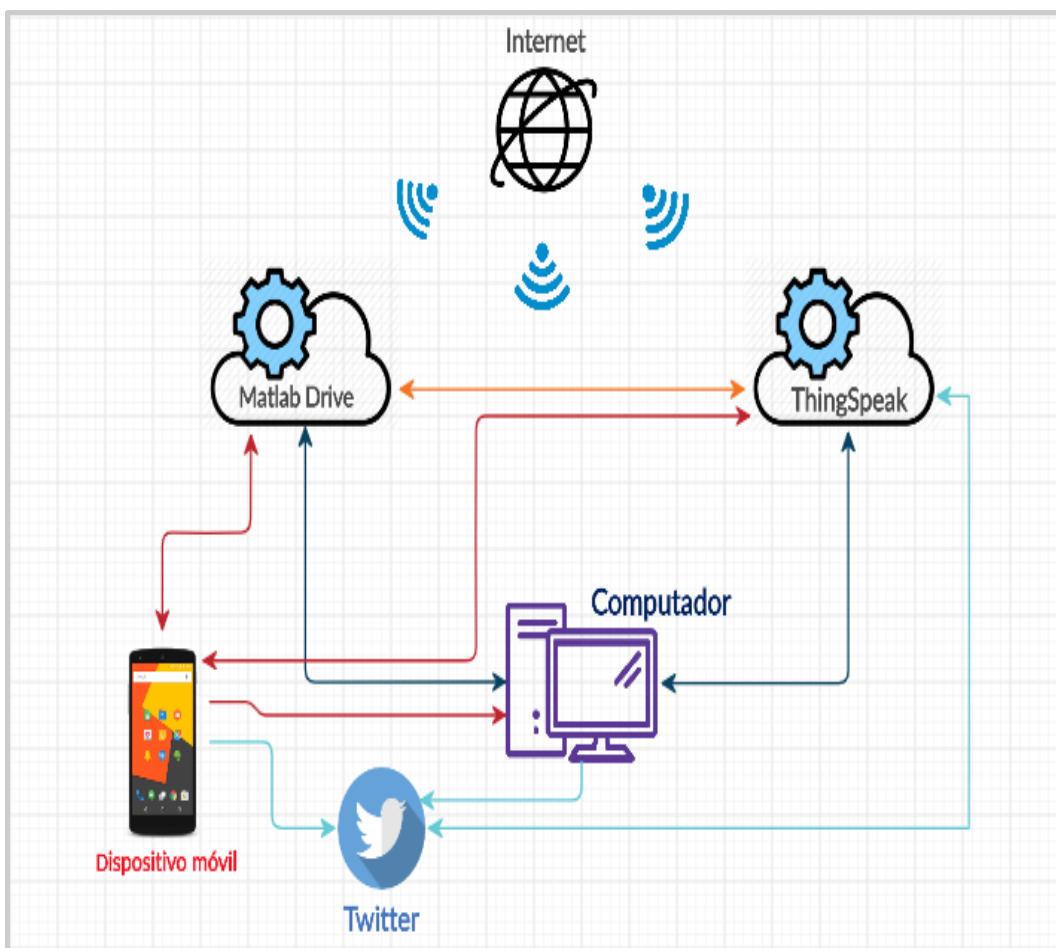


Figura 3.1 Esquema de los componentes de la arquitectura del sistema.

- **Computador:** Plataforma para la ejecución de procesos con una alta carga computacional, tal como el procesamiento de vídeos, así como para el entrenamiento de la Red Neuronal Convolutinal, cuyo proceso no puede realizarse en el dispositivo móvil.
- **Dispositivo móvil:** Utilizado para la captura de datos (secuencias de imágenes) a procesar y como conexión con ThingSpeak²⁴. Recibe mensajes vía Twitter cada vez que se publiquen datos en el canal de ThingSpeak. No obstante, en la solución que se presenta, la opción de captura no está habilitada por la imposibilidad de desplegar un equipamiento de esta naturaleza en un entorno real, sustituyéndose por secuencias de imágenes (vídeo) que permiten la simulación de una operativa real. En caso de implantar este sistema en un entorno real, este dispositivo de captura puede sustituirse por una cámara con capacidad de transmisión de imágenes en tiempo real para su procesamiento remoto.
- **ThingSpeak:** Plataforma para servicios de MATLAB en la nube, bajo el paradigma IoT. Permite tanto el almacenamiento de datos procesados como procesamientos propios, así como la programación de otros servicios tales como envío de información ante determinados eventos, ofreciendo además la posibilidad de visualización de los mismos por la naturaleza pública de los canales definidos en la plataforma.
- **MATLAB Drive:** Servicio remoto de MATLAB para el procesamiento de datos en la nube y con conexión a ThingSpeak. Se trata de un servicio al que se accede, mediante conexión internet, tanto desde el computador como desde el dispositivo móvil de forma compartida. Se realiza una sincronización automática entre los dispositivos con acceso permitido de forma que su contenido está permanentemente actualizado.

El flujo y procesamiento de datos propuesto para el módulo de entrenamiento y detección es el siguiente:

1. Con el dispositivo móvil se graban vídeos en entornos con tráfico de vehículos para la extracción de los datos necesarios para el entrenamiento de la Red Neuronal Convolutinal. Estos datos son en realidad recortes de imágenes conteniendo distintos tipos de vehículos en diferentes posiciones y tamaños, que sirven como entradas a la red. También, permite la grabación de vídeos, distintos a los de entrenamiento, para la detección de vehículos en movimiento durante la fase de clasificación, y con el modelo de red re-entrenado.

2. Mediante el computador se realiza el re-entrenamiento de la red y el procesamiento de los vídeos, mencionados previamente, para la detección de vehículos.
3. Una vez procesado el video, también mediante el computador, en ThingSpeak²⁴ se almacena el número de vehículos detectados de cada tipo, de la misma manera, se envía un aviso con los datos resultantes a la cuenta de Twitter asociada al canal de ThingSpeak habilitado y convenientemente configurado.

El flujo y procesamiento de datos propuesto para la predicción del flujo de vehículos, y por tanto para determinar el tránsito a futuro, es el siguiente:

1. Utilizando los datos del DataSet proporcionados por el Ayuntamiento de Madrid, a través de su portal de datos abiertos¹⁵, se almacena la información relevante (útil para la predicción) en el canal de ThingSpeak destinado y configurado específicamente para ello.
2. Desde el computador se introducen en el canal de ThingSpeak correspondiente los datos necesarios para realizar una predicción. Automáticamente se activa la función de Matlab Analysis que realiza la predicción, almacenando el resultado obtenido en ThingSpeak.

En el diagrama de casos de uso mostrados en la Figura 3.2 aparecen reflejadas las operaciones que pueden realizar los usuarios según su rol en la aplicación. Se distinguen dos tipos de usuarios, a saber: administrador y común.

El administrador se entiende desde la perspectiva de manejo experto del sistema en el sentido de su capacidad para realizar las operaciones siguientes: añadir imágenes al set de entrenamiento, redimensionar imágenes, entrenar la red CNN, en este caso el modelo AlexNet, procesar datos y enviar éstos al servidor en la nube, consultar datos en la nube, realizar predicciones sobre el flujo de tráfico, publicar resultados en Twitter. Los resultados publicados en esta red social, pueden ser consultados por usuarios con el perfil común, que también puede realizar predicciones y ver los resultados.

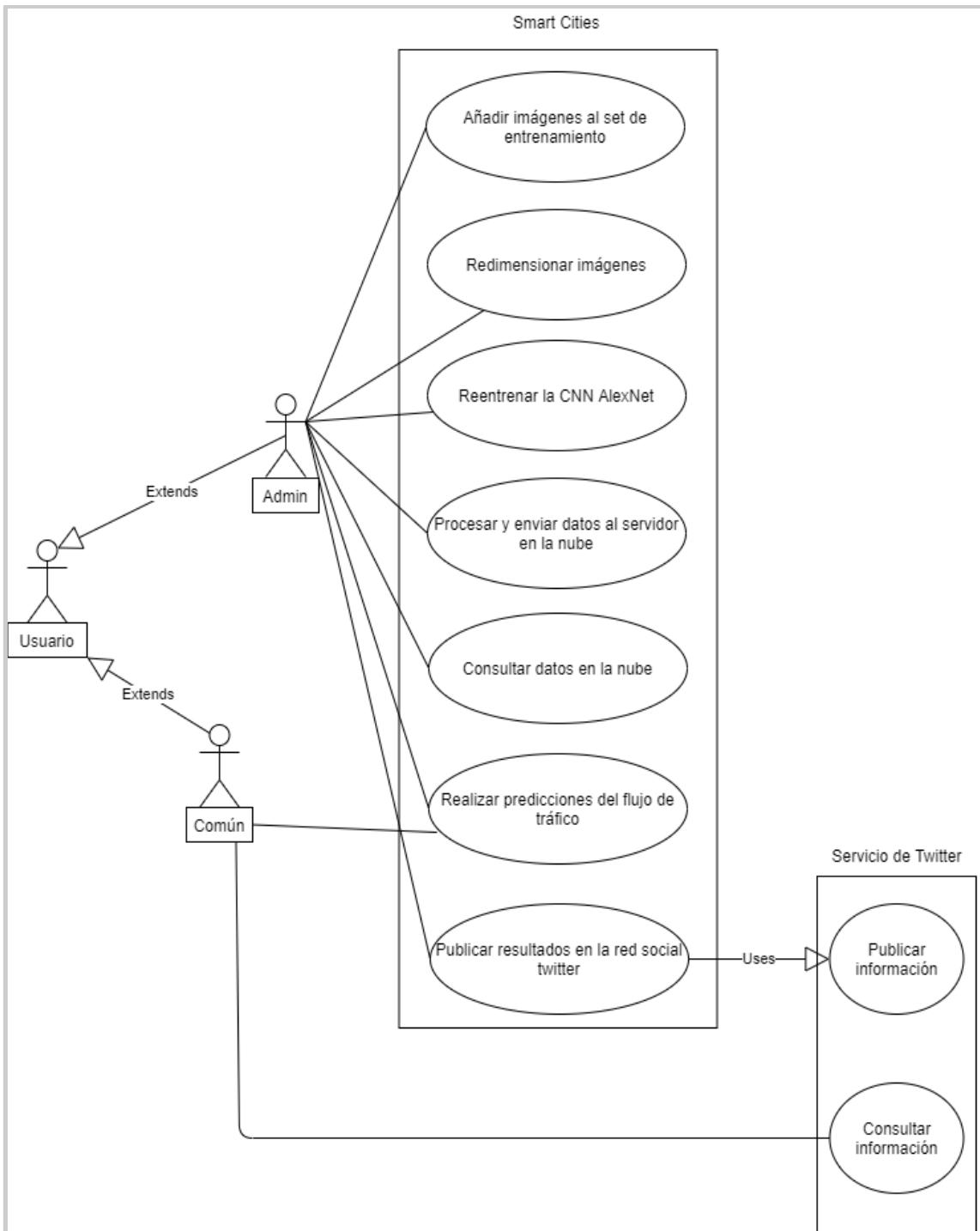


Figura 3.2 Diagrama de casos de uso.

3.1 Análisis arquitectónico

3.1.1 Lado del cliente

En el lado del cliente se llevan a cabo las siguientes funcionalidades que aparecen en la figura 3.2:

- Añadir imágenes al set de entrenamiento.
- Redimensionar imágenes.
- Reentrenar la CNN AlexNet.
- Procesar los vídeos sin publicar los resultados a ThingSpeak²⁴.

La arquitectura modelada³¹ y posteriormente utilizada sigue un modelo multicapa. Este tipo de sistemas software está organizado, como su nombre indica, en varias capas, cada una de las cuales contiene un conjunto de clases con responsabilidades relacionadas con la capa a la que pertenecen. Las tres capas que componen un sistema software multicapa son:

- **Capa de presentación**, encargada de interactuar con el usuario de la aplicación mediante una interfaz de usuario.
- **Capa de negocio** (lógica de la aplicación), usualmente implementada utilizando un modelo orientado a objetos del dominio de la aplicación, es la responsable de realizar las tareas para las cuales se diseña el sistema.
- **Capa de integración** (de acceso a los datos), encargada de gestionar el almacenamiento de los datos, generalmente en un sistema gestor de bases de datos relacionales.

En este caso, como el almacenamiento de datos se realiza en la nube a través de ThingSpeak, no necesitamos de una capa de integración, ya que al utilizar un software de terceros nos facilita el control de los datos sin necesidad de gestionar la concurrencia o el acceso a los datos. ThingSpeak está completamente integrado con MATLAB, por lo que la integración de ambas herramientas es sencilla y no requiere de ningún otro servicio o herramienta.

Para el modelado de la aplicación se han utilizado los siguientes patrones de diseño de software:

- **Transfer:** Es un objeto que transporta datos entre procesos. Se utiliza cuando se quiere transmitir datos con múltiples atributos de una sola vez.^{28 30}
- **Singleton:** Es un patrón de diseño que permite restringir la creación de objetos pertenecientes a una clase. Su intención consiste en garantizar que una clase sólo tenga una instancia y proporcionar un punto de acceso global a ella.²⁹

- **Command:** Este patrón permite solicitar una operación a un objeto sin conocer realmente el contenido de esta operación, ni el receptor real de la misma.²⁹
- **Context:** Es un patrón que nos ayuda a encapsular los eventos producidos por las vistas y los datos para ejecutar los comandos.²⁸
- **Factory:** Consiste en utilizar una clase dedicada a la construcción de objetos del mismo tipo.²⁹
- **Dispatcher View:** Este patrón permite crear un objeto que se encarga únicamente de recibir un evento y redirigir a la vista correspondiente.²⁸
- **Application Controller:** Gestiona los eventos producidos en las interfaces.²⁸
- **Application Service:** Implementa la lógica del sistema.²⁸

La capa de presentación se muestra en el diagrama de clases de la Figura 3.3. El proceso se realiza de la siguiente manera:

1. Ante cualquier acción en la interfaz, se crea un objeto contexto, que contiene el evento (acción a realizar) y el transfer (datos necesarios para realizar la acción), enviándole así ambos al controlador.
2. El controlador gestiona los eventos como sigue:
 - Se crea el comando correspondiente mediante la factoría de comandos. Por ejemplo, si el evento es executeCarDetection, se crea el comando CommandCarDetection.
 - Se ejecuta el comando creado con los datos del contexto.
 - El comando devuelve un contexto y el controlador se encarga de dárselo al Dispatcher, para que éste genere la vista.
3. El Dispatcher redirige a la vista correspondiente y ésta se actualiza para mostrar la información resultante.

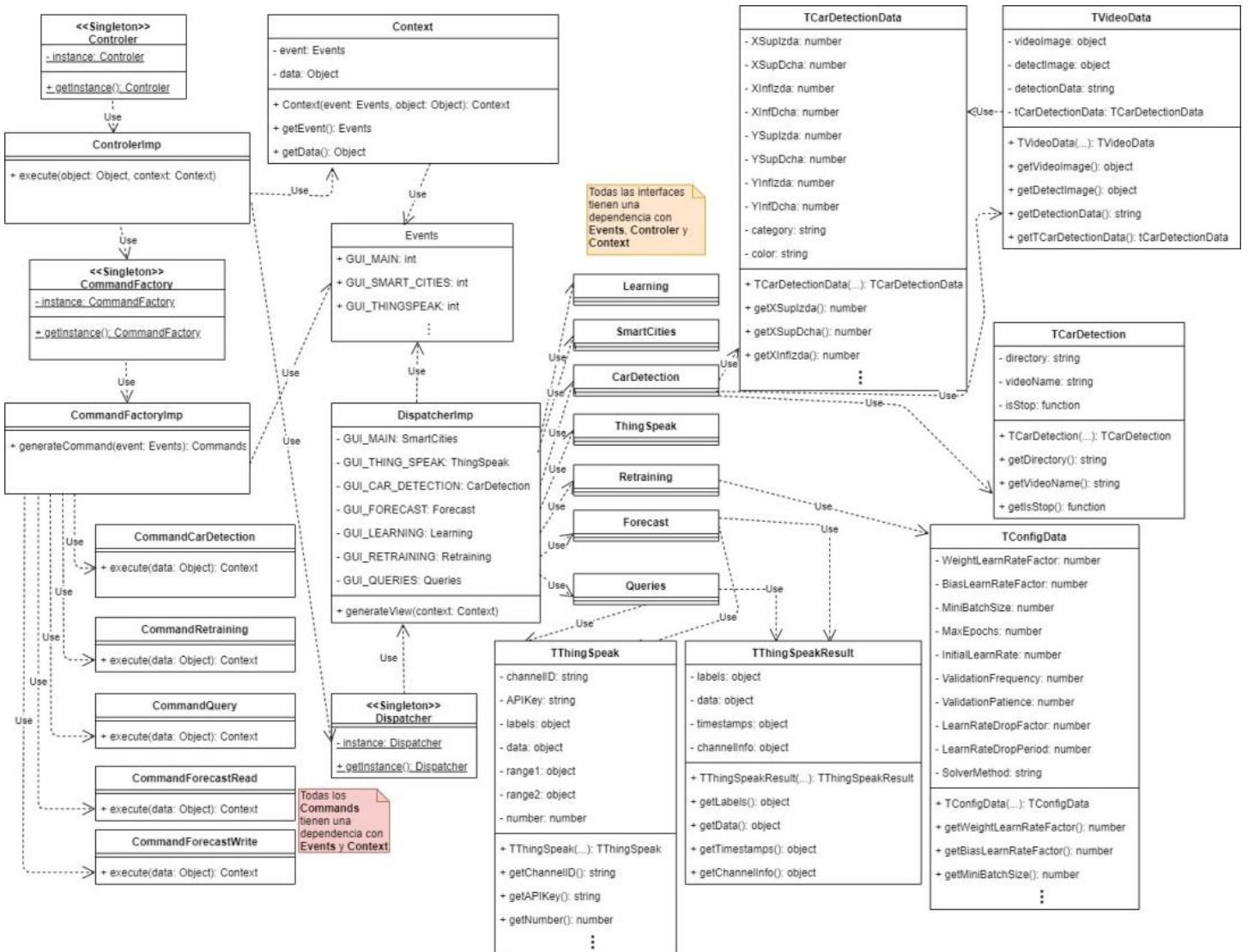


Figura 3.3 Diagrama de clases de la capa de presentación.

La capa de negocio está representada por el diagrama de clases de la Figura 3.4. El proceso se realiza de la siguiente manera:

1. El comando usa la clase ASFactory que se encarga de gestionar la creación de los diferentes servicios de aplicación.
2. Se ejecuta la función del servicio de aplicación correspondiente con el comando a realizar. Por ejemplo el comando CommandCarDetection ejecutará la función detection de ASSmartCities.
3. El resultado de la ejecución se devolverá al controlador.

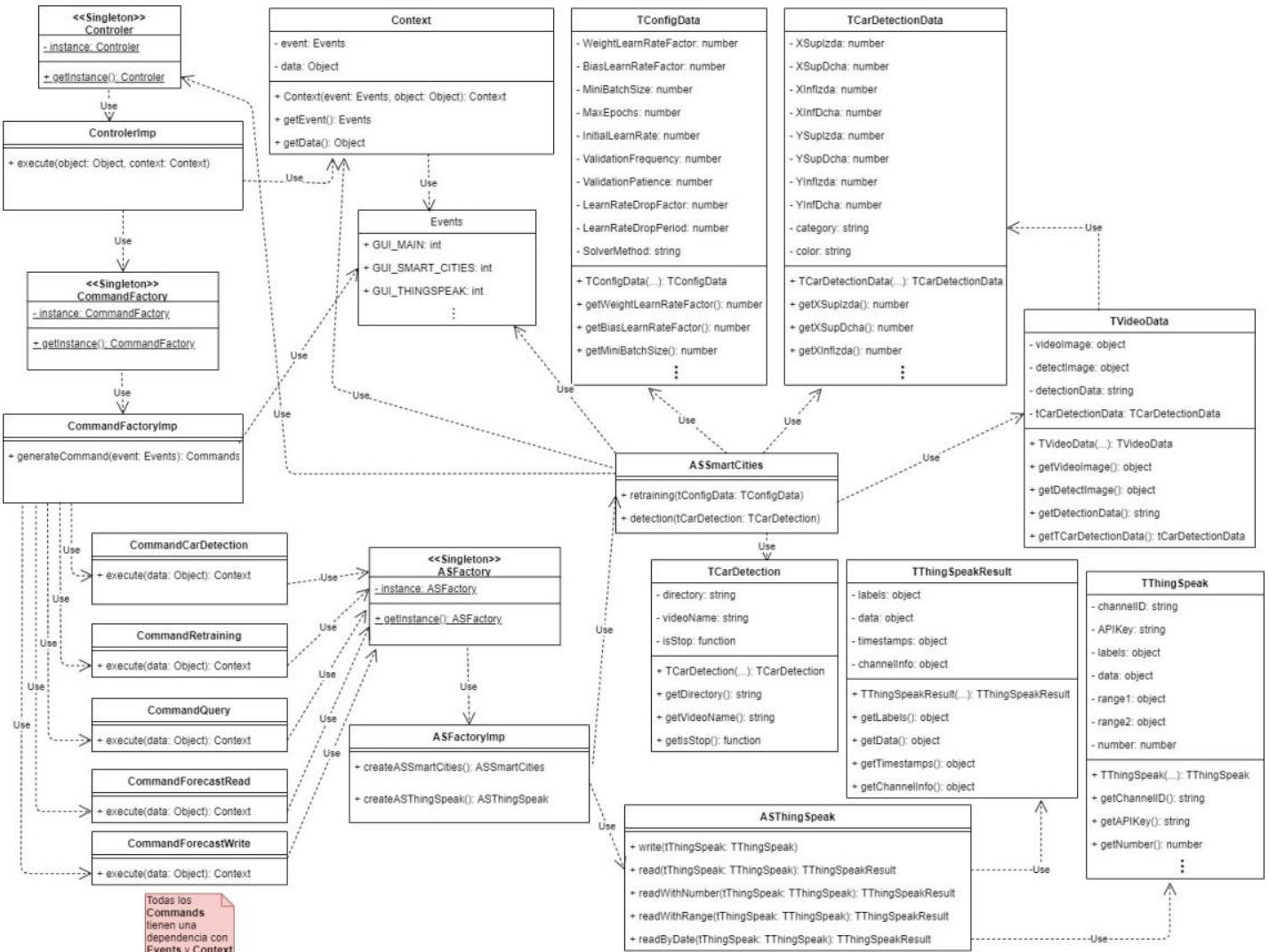


Figura 3.4 Diagrama de clases de la capa de negocio.

Para cada una de las funcionalidades siguientes se describe el flujo de la aplicación, considerando las funcionalidades descritas en las figuras 3.3 y 3.4:

- Añadir imágenes al set de entrenamiento:
 - La clase Learning es la interfaz en la que se pulsa el botón para abrir el set de entrenamiento. Como la funcionalidad consiste en abrir un directorio se realiza directamente desde la interfaz.
- Redimensionar imágenes:
 - La clase Learning es la interfaz en la que se pulsa el botón para redimensionar las imágenes. Se abre un selector de archivos en el que se seleccionan una o varias imágenes que se redimensionan desde la propia interfaz.

- Reentrenar la CNN AlexNet:
 - La clase Learning es la interfaz en la que se pulsa el botón para reentrenar la CNN.
 - Se abre entonces la interfaz Retraining en la que seleccionan los parámetros del reentrenamiento.
 - Se envía la petición al controlador. Éste crea el comando CommandRetraining que llama a la función retraining de la clase ASSmartCities.
 - CommandRetraining y devuelve el resultado de la operación al controlador. El controlador usa el Dispatcher para devolver el resultado a la vista correspondiente, en este caso Retraining.
- Procesar los vídeos sin publicar los resultados a ThingSpeak:
 - La clase CarDetection es la interfaz en la que se pulsa el botón para leer un video y procesarlo.
 - Se envía la petición al controlador. Éste crea el comando CommandCarDetection que llama a la función detection de la clase ASSmartCities.
 - CommandCarDetection devuelve el resultado de la operación al controlador. El controlador usa el Dispatcher para a su vez devolver el resultado a la vista correspondiente, en este caso CarDetection.

Más información a cerca de la arquitectura disponible en los Anexos I^I y II^{II}.

3.1.2 Lado del servidor

En el lado del servidor se llevan a cabo las siguientes funcionalidades de las que aparecen en la figura 3.2:

- Publicar los resultados del procesamiento de videos en ThingSpeak.
- Realizar predicciones del flujo de tráfico
- Publicar resultados en la red social Twitter
- Consultar datos en la nube

En cuanto a la arquitectura del servidor decir que está basado en ThingSpeak²⁴, con la particularidad de que facilita considerablemente el trabajo de diseño ya que se trata de una herramienta muy completa. A continuación se describe la estructuración y configuración aplicada en el servidor con dicha herramienta.

ThingSpeak permite la creación de canales específicos, de forma que durante su creación es posible ajustar campos tales como, nombre del canal, breve descripción del canal, número de campos y los nombres asignados a éstos, haciendo que se pueda diseñar la arquitectura del servidor en función de las necesidades del proyecto.

El orden de los campos es importante ya que son utilizados por defecto en el orden creado a la hora de insertar datos mediante procesos de escritura y hacer consultas de lectura de los datos almacenados, que son los que luego se pueden visualizar en el interfaz de la aplicación.

Para ello, ThingSpeak proporciona las denominadas “api keys” que sirven para que cuando se quiera acceder a ThingSpeak en modo lectura o escritura baste con utilizar dichas claves en la llamada correspondiente desde Matlab. Estas “api keys” son generadas automáticamente por ThingSpeak en la creación del canal, pudiéndose modificar mediante la opción correspondiente. Por motivos de seguridad no se especifican las api keys en este documento, lo único indicar que es una palabra formada por números y letras con 17 caracteres.

También es importante el identificador del canal “channel ID” o ID de canal, que es la clave para conectar ThingSpeak con el teléfono móvil, Figura 3.5, permitiendo visualizar el contenido del canal si está en modo público. Conviene señalar que ThingSpeak ofrece la opción de canal público o privado. El primer caso con acceso a cualquier usuario y en el segundo restringido al creador. El identificador de canal es necesario para los accesos de lectura y escritura.

Dado que conocidas las claves de acceso e identificador del canal, es posible el acceso múltiple e incluso por diferentes usuarios al mismo, se hace necesario el control de los accesos concurrentes en escritura para evitar posibles fallos de actualización de datos y sincronización cuando se intenta escribir al mismo tiempo. ThingSpeak controla estos casos de concurrencia obligando a que haya un intervalo de 15 segundos entre cada acceso de escritura, de forma que en el caso de que lleguen al mismo tiempo dos o más accesos de escritura o sin respetar el intervalo de 15 segundos, ThingSpeak devolverá el siguiente mensaje: "accesos de escritura demasiados frecuentes", aceptando únicamente el que llegue primero y denegando el resto.

Como se ha indicado previamente, la figura 3.5 muestra la visualización de datos contenidos en el canal en los diferentes campos mostrados, utilizando la aplicación

ThingView instalada en el dispositivo móvil. En este caso concreto se muestran los ocho campos de un mismo canal (Field 1 a Field 8) mostrando el tipo de vehículos detectados en cada uno de ellos a lo largo del tiempo, cuya descripción y configuración se describe posteriormente.

Por ejemplo, en el diagrama Field 1 aparece el número de coches vistos por delante y detectados como tal por la aplicación, esto es por la Red Neuronal Convolutacional, en distintos momentos del mes de Abril de 2020, según aparece en el eje horizontal. En el resto de gráficos aparecen representadas las detecciones de otros tipos de vehículos, también con indicación del momento de su detección en el tiempo.

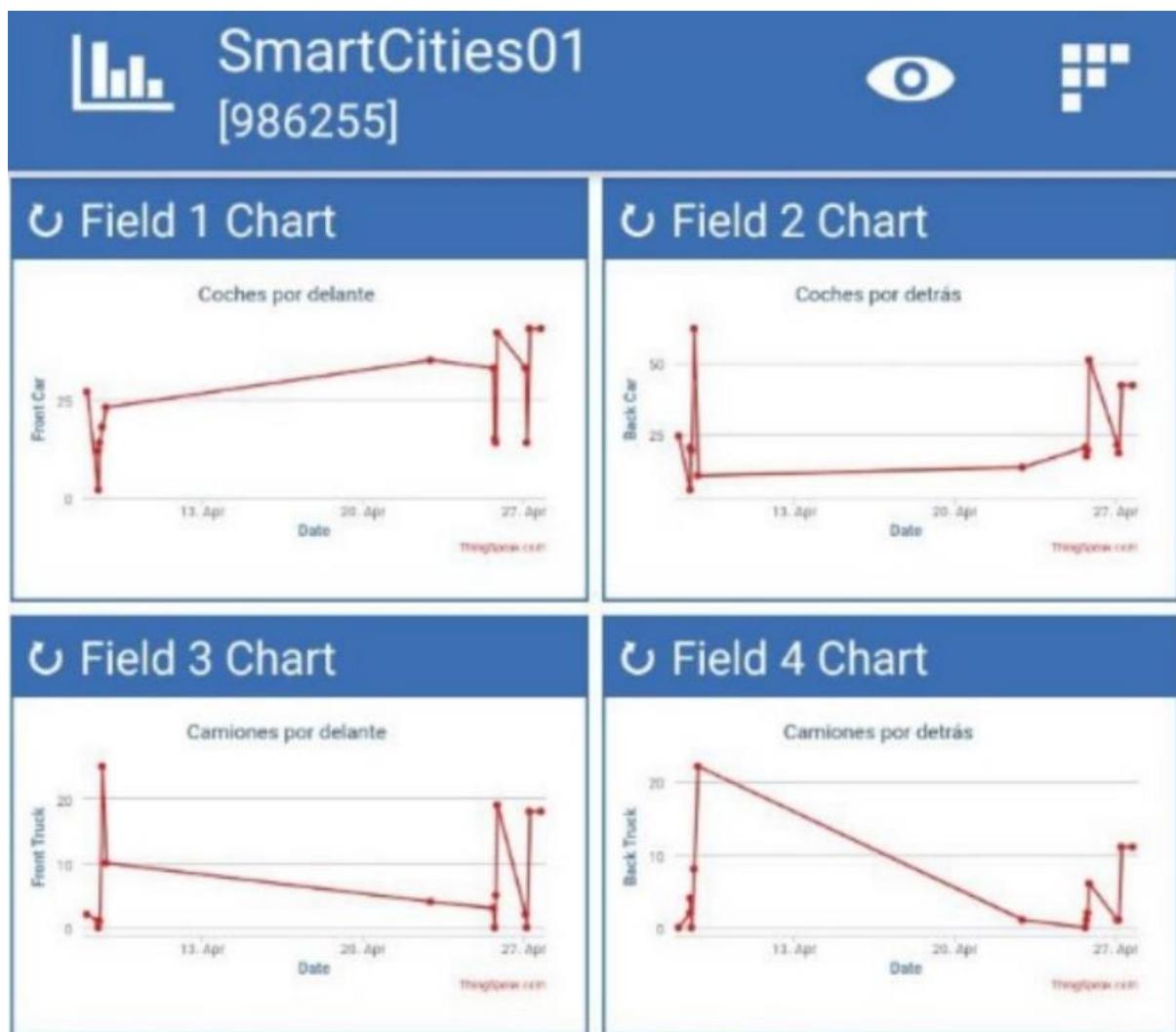


Figura 3.5 Imagen de cómo se ve el canal de ThingSpeak que contabiliza los diferentes tipos de vehículos en la aplicación para móviles ThingShow.

3.1.2.1 Canal de detección de vehículos

El canal de detección de vehículos, tiene por título “SmartCities”, siendo configurado con los ocho campos que permite como máximo la licencia de ThingSpeak²⁴.

Cada uno de los campos identifica el tipo de vehículo detectado, guardando dentro del campo el número de vehículos detectados de esa categoría durante la fase de clasificación llevada a cabo por la CNN.

La Figura 3.2 muestra la definición y estructuración del canal con los campos siguientes:

- Campo 1 - Front Car (coches por delante).
- Campo 2 - Back Car (coches por detrás)
- Campo 3 - Front Truck (camiones por delante)
- Campo 4 - Back Truck (camiones por detrás)
- Campo 5 - Front Motorbike (motos por delante)
- Campo 6 - Back Motorbike (motos por detrás)
- Campo 7 - Front Bus (buses por delante)
- Campo 8 - Back Bus (buses por detrás)

The screenshot shows the ThingSpeak channel configuration interface. At the top, there's a navigation bar with links for 'Channels', 'Apps', 'Support', 'Commercial Use', 'How to Buy', and a 'VG' logo. Below the navigation bar, it says 'Access: Public'. The main area has tabs for 'Private View', 'Public View', 'Channel Settings' (which is selected), 'Sharing', 'API Keys', and 'Data Import / Export'. On the left, under 'Channel Settings', there are fields for 'Percentage complete' (50%), 'Channel ID' (986255), 'Name' (SmartCities01), and 'Description' (Canal para realizar pruebas en la interfaz de consultas de ThingSpeak). There are also eight 'Field' input boxes labeled Field 1 through Field 8, each containing a vehicle category: 'Front Car', 'Back Car', 'Front Truck', 'Back Truck', 'Front Motorbike', 'Back Motorbike', 'Front Bus', and 'Back Bus'. Each field has a checked checkbox next to it. On the right, there's a 'Help' section with a detailed description of channels and a 'Channel Settings' section with a bulleted list of configuration options and their descriptions.

Help

Channels store all the data that a ThingSpeak application collects. Each channel includes eight fields that can hold any type of data, plus three fields for location data and one for status data. Once you collect data in a channel, you can use ThingSpeak apps to analyze and visualize it.

Channel Settings

- **Percentage complete:** Calculated based on data entered into the various fields of a channel. Enter the name, description, location, URL, video, and tags to complete your channel.
- **Channel Name:** Enter a unique name for the ThingSpeak channel.
- **Description:** Enter a description of the ThingSpeak channel.
- **Field#:** Check the box to enable the field, and enter a field name. Each ThingSpeak channel can have up to 8 fields.
- **Metadata:** Enter information about channel data, including JSON, XML, or CSV data.
- **Tags:** Enter keywords that identify the channel. Separate tags with commas.
- **Link to External Site:** If you have a website that contains information about your ThingSpeak channel, specify the URL.
- **Show Channel Location:**
 - **Latitude:** Specify the latitude position in decimal degrees. For example, the latitude of the city of London is 51.5072.
 - **Longitude:** Specify the longitude position in decimal degrees. For example, the longitude of the city of London is -0.1275.
 - **Elevation:** Specify the elevation position meters. For example, the elevation of

Figura 3.6 Visualización de las opciones de configuración del canal de ThingSpeak.

A través del enlace [SmartCities](#) se puede acceder a la versión web del canal de ThingSpeak²⁴, donde se encuentran almacenados todos los datos referentes al conteo de los diferentes tipos de vehículos, tal y como se puede observar en la Figura 3.6. El identificador de este canal de ThingSpeak es “986255”.

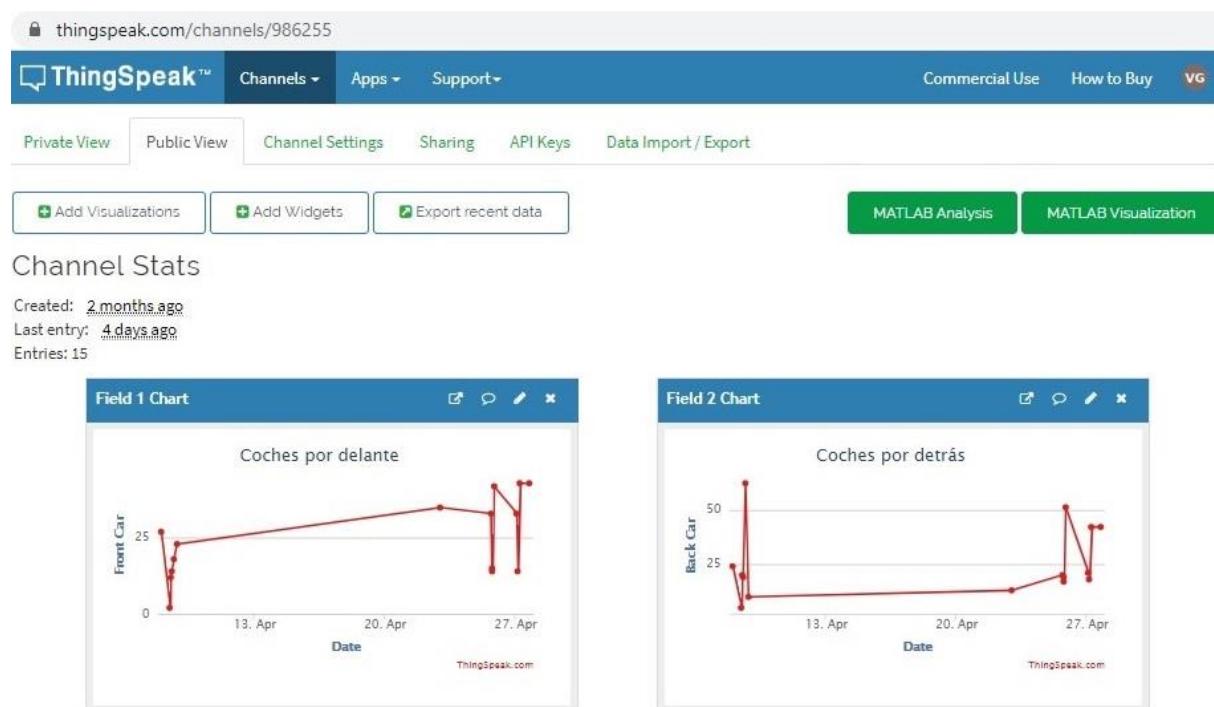


Figura 3.7 Visualización gráfica de los diferentes campos del canal de ThingSpeak.

ThingSpeak proporciona Apps específicas para su interconexión con otras plataformas. En el diseño propuesto se conecta a una cuenta de Twitter, de forma que ésta lance avisos del número de vehículos de un determinado tipo que son almacenados cada vez que se complete el análisis de un video. Para ello se hace uso de la app ThingTweet, que conecta la cuenta de Twitter con el canal de ThingSpeak y una segunda app de tipo React que sirve para explorar los valores de todos los campos y enviar el tweet correspondiente cada vez que se actualice un determinado valor de campo en el canal de ThingSpeak correspondiente. El enlace a la cuenta de Twitter utilizada para lanzar los avisos es [Cuenta Twitter](#), donde se puede ver cómo se muestran los mensajes.

3.1.2.2 Canales de predicción

Para la tarea de predicción del flujo de vehículos se cuenta con tres canales de ThingSpeak. En el primero se almacenan los datos parseados del DataSet del Ayuntamiento de Madrid¹⁵. Se tienen datos desde enero del 2017 hasta febrero de 2020 (no se han considerado los meses de Marzo y Abril dada la situación anómala en cuanto al flujo de vehículos por las restricciones derivadas de COVID-19, ya que

alterarían los valores de predicción en situaciones normales). Este primer canal consta de los siete campos que se muestran en la figura 3.7 de forma que sobre el eje horizontal se muestra el día del mes correspondiente al día de la semana, que aparece sobre el eje vertical.

- Campo 1 - Lunes
- Campo 2 - Martes
- Campo 3 - Miércoles
- Campo 4 - Jueves
- Campo 5 - Viernes
- Campo 6 - Sábado
- Campo 7 - Domingo

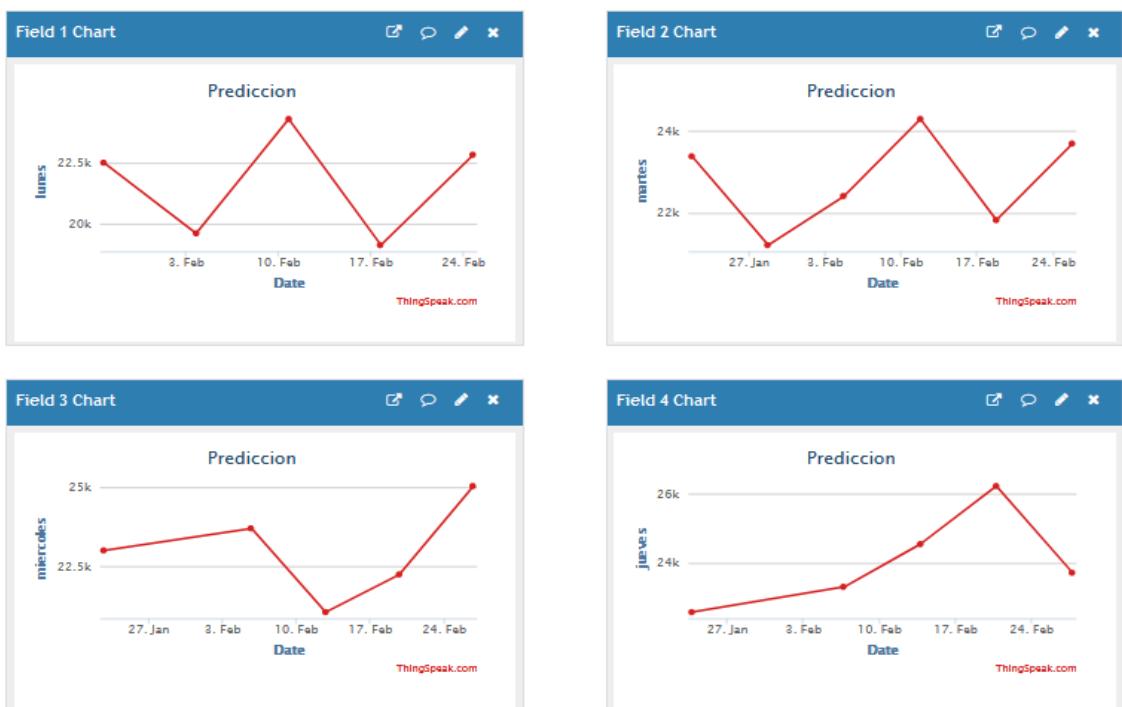


Figura 3.7: Representación gráfica del conteo de vehículos.

En el segundo de los canales de predicción se almacenan los datos de entrada para realizar la predicción propiamente dicha. Para hacer el cálculo se necesita el día de la semana y el número de predicciones que se quieren obtener. El cálculo se realiza en el servidor, usando la funcionalidad de MATLAB Analysis.

Como no se puede comunicar directamente la aplicación con MATLAB Analysis, lo que se hace desde el punto de vista del diseño es guardar en un campo cada parámetro de entrada, de forma que al detectar una nueva entrada en el canal, automáticamente se lanza la función que realiza la predicción. Esta función lee los datos recién introducidos desde la aplicación y realiza la predicción.

El canal está configurado con los dos campos siguientes:

- Campo 1: Horizonte - Indica el número de días que se quieren predecir.
- Campo 2: Día de la semana - Indica el día de la semana sobre el que se realiza la predicción.

La figura 3.8 muestra sendas representaciones gráficas del contenido de los dos campos anteriores, de forma que en la parte izquierda aparece el resultado de la predicción según el horizonte indicado mediante el modelo AR y en el eje X aparece el número de vehículos predichos según el día del mes. En la parte derecha aparece un gráfico con indicación del campo 2, que no puede representarse por ser los datos de tipo texto.

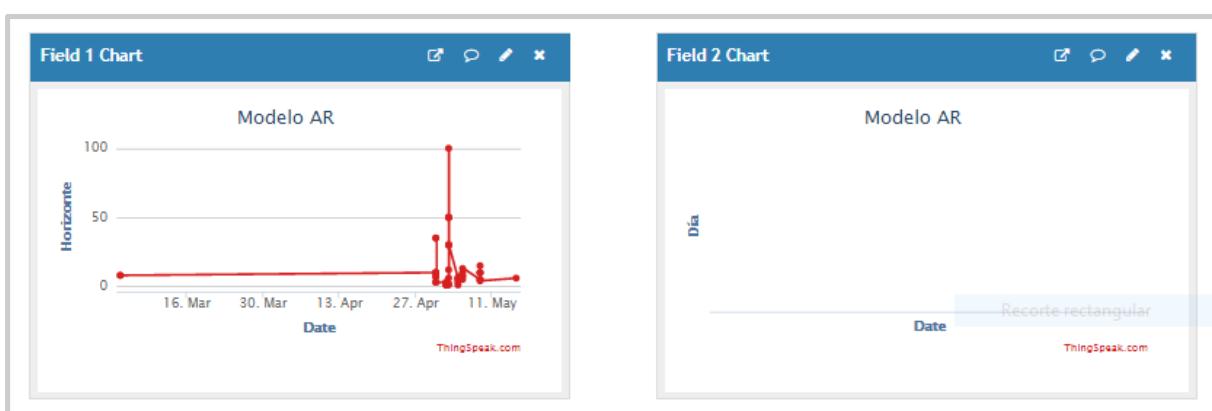


Figura 3.8 Representación gráfica de los campos del canal modelo AR.

Por ejemplo si se introduce un 10 como horizonte y sábado como día de la semana, se obtienen datos de los próximos 10 sábados comenzando desde el último sábado del que se tienen datos reales.

En el tercer canal de predicción se almacena el resultado de la predicción realizada. Tiene los siguientes siete campos, cuya visualización se muestra gráficamente en la figura 3.9 con la misma interpretación que en el caso anterior.

- Campo 1 - Lunes
- Campo 2 - Martes

- Campo 3 - Miércoles
- Campo 4 - Jueves
- Campo 5 - Viernes
- Campo 6 - Sábado
- Campo 7 - Domingo



Figura 3.9 Representación gráfica de los campos del canal resultado de predicciones.

3.2 Metodología de desarrollo empleada

Para el desarrollo de la aplicación se ha elegido la metodología ágil Scrumban ya que, como sostiene la Agile Aliance, ésta combina las mejores características de otras dos metodologías de desarrollo ágiles: Scrum y Kanban¹⁴. Sus características principales se resumen a continuación:

- Permite una cómoda visualización del flujo de trabajo.
- Revisión de un tablero Kanban en lugar de realizar las Daily Scrum.
- Work In Progress regulado por el desarrollador de cada módulo.

- No existen roles, ya que únicamente están el profesor y los alumnos.
- Favorece la especialización del trabajo y la autoorganización del equipo.
- La documentación se limitada únicamente a la aclaración del código fuente.
- Reuniones de revisión con el profesor a corto plazo, cada 2 - 3 semanas.
- Reuniones de retrospectiva en el equipo a muy corto plazo, cada semana.
- Sin Sprints ni estimaciones, trabajo continuo y bajo la demanda del profesor.

3.3 Herramientas y recursos software

A continuación se enumeran las diferentes herramientas usadas para el desarrollo e implementación de la aplicación.

- **MATLAB:** Para el desarrollo del proyecto se ha recurrido principalmente a la tecnología del software de MATLAB, concretamente a la versión R2019a. Este software ofrece un entorno de desarrollo propio con un lenguaje de programación pensado para ser eficiente en el cómputo numérico, un diseñador de interfaces de usuario sencillo e intuitivo, motivo por el cual se ha utilizado para los desarrollos de la parte local o lado del cliente.

Ha sido necesaria la instalación de algunos add-ons (toolboxes) que amplían la funcionalidad del IDE (*Integrated Development Environment*), éstos son los siguientes:

- Deep Learning Toolbox: incluye las funcionalidades necesarias para poder trabajar con Redes Neuronales Convolucionales.
- Image Processing Toolbox: provee las funciones necesarias para visualizar y procesar imágenes.
- Computer Vision Toolbox: proporciona los algoritmos y funciones necesarias para trabajar con aspectos relacionados con la Visión por Computador⁴, tales como el análisis del flujo óptico, la detección y el seguimiento de objetos, etc.
- Deep Learning Toolbox Model for AlexNet Network: proporciona un diseñador gráfico para configurar una Red Neuronal Convolutional existente como AlexNet o permite crear una Red Neuronal nueva de carácter secuencial.

- Parallel Computing Toolbox: permite poder hacer un uso eficiente de la GPU mediante el empleo de sus operaciones en paralelo, con el objetivo de agilizar aquellas operaciones que son computacionalmente más lentas. Gracias a este Toolbox, el proceso de entrenamiento de AlexNet es más rápido.

La elección de esta herramienta se hizo en base al soporte que proporciona la empresa suministradora (MathWorks), junto con la potente comunidad de desarrolladores con la que cuenta, además de su potencialidad, con licencia soportada por la Universidad Complutense de Madrid (UCM), tanto para el núcleo base como para los diferentes Toolboxes.

- **ThingSpeak:** Servidor donde se almacenan todos los datos resultantes de los análisis de las imágenes de los vídeos, se utiliza también para realizar consultas de los datos guardados. También constituye el pilar fundamental que hará posible que los datos sean representados en múltiples plataformas, ya sea desde un ordenador (página web o desde la propia interfaz de la aplicación), un teléfono móvil o por una cuenta de Twitter creada expresamente para este trabajo. La elección de esta herramienta se debió a que es el entorno natural con MATLAB para desarrollos IoT, que permite la creación de los canales correspondientes mediante la cuenta oficial de la UCM.
- **Github:** Para el control de versiones se optó por una herramienta bastante conocida en el mundo del desarrollo, la cual permite una organización en diferentes ramas, donde cada una es completamente independiente y facilita el trabajar al mismo tiempo en dos ubicaciones distintas de la aplicación, sin perder ningún cambio. También decir que es totalmente gratuita y si se registra con el correo de la UCM se accede a una versión PRO, la cual permite tener proyectos ocultos al público para evitar copias. La elección frente a otras herramientas de control, como puede ser el caso de GitLab y Bitbucket, es por la familiaridad que el equipo tiene con Github y por la facilidad de llevar a cabo el control de las diferentes versiones con su herramienta para escritorio Github Desktop, la cual sustituye la interfaz de comandos de Git, por una interfaz mucho más atractiva y amigable para el usuario.
- **Mockflow:** Herramienta que ofrece una licencia de prueba, permitiendo llevar a cabo el diseño de las diferentes interfaces de la aplicación, que posibilita visualizar la interfaz final, con anterioridad a su validación definitiva. Fue elegida debido a la diversidad de opciones que proporciona para realizar bocetos de interfaces y la experiencia previa del grupo con la herramienta.
- **Draw.io:** Herramienta utilizada para la creación de los diagramas de flujo. Elegida por su sencillez y licencia gratuita, además de ser una herramienta web que permite guardar los diagramas tanto en un ordenador local, como en el

drive, posibilitando su acceso a todos los desarrolladores y, por tanto, a los miembros del equipo.

- **Google Drive:** Herramienta para almacenar las imágenes y videos de vehículos circulando, utilizados para el entrenamiento de la CNN. También es donde se lleva a cabo la realización de la memoria del proyecto, gracias a los documentos de Google que permiten el acceso simultáneo de varios usuarios.
- **Telegram:** Herramienta utilizada para la comunicación entre los diferentes miembros del equipo, ayudando a organizar reuniones, resolver dudas puntuales y determinar los avances y la problemática, en su caso, en relación a alguna tarea o actividad específica. La herramienta fue elegida por su similitud con la aplicación WhatsApp y por su gratuidad.
- **Trello:** Para la organización del equipo y tener claro qué tareas realiza cada integrante, también para llevar un control de los avances conseguidos con el tiempo y así poder conocer en qué punto del desarrollo se encuentra el proyecto. Se eligió esta herramienta frente a otras, como Jira, IceScrum o ScrumDo, por el alto grado de familiaridad para los integrantes del equipo y por su sencillez y claridad. Además es una herramienta con licencia gratuita.
- **Herramienta recortes:** Es una herramienta que se usará para la obtención de imágenes de los diferentes tipos de vehículos que tratará la CNN (FrontCar, BackCar, FrontMotorbike, BackMotorbike, FrontTruckVan, BackTruckVan, FrontBus, BackBus) y de esta forma poder entrenarla. Se eligió esta herramienta por la sencillez que proporcionaba al usuario y por venir integrada en Windows 10.
- **Skype:** herramienta para la realización de videollamadas, fundamental para mantener la comunicación con los miembros del equipo y poder organizar de forma efectiva todas las tareas relacionadas con el trabajo, útil debido al estado de excepcionalidad provocado por la pandemia del Covid-19.

4. Resultados

En esta sección se exponen los resultados obtenidos por la aplicación desarrollada, tanto a nivel individual de los diferentes módulos que constituyen el sistema software como de la integración de los mismos.

Para el análisis se han utilizado 48 vídeos de tráfico cuya duración media es de 1 minuto, capturados a una frecuencia de 30 *frames* por segundo (fps) con un teléfono móvil Meizu Pro 6, por lo que se han capturado aproximadamente 84.600 imágenes en las que se pueden apreciar distintas densidades de tráfico. Los vídeos han sido

tomados en un tramo de la autovía del noreste de Madrid (A-2), aproximadamente en el PK 7.

El ordenador empleado para el procesamiento de vídeos ha sido un DELL XPS 15 con un procesador Intel Core i9 de 8 núcleos a 2,40GHz, 2 memorias RAM DDR4 de 8 GB a 2666 MHz, una tarjeta gráfica NVIDIA GeForce GTX 1650, una unidad de estado sólido de 512 GB y Ubuntu como sistema operativo.

Los resultados se muestran en el siguiente orden. En primer lugar se muestra el interfaz de la aplicación con todas sus funcionalidades. En segundo lugar se describen los resultados derivados de la aplicación de las funciones de Visión por Computador⁴ sobre las imágenes de los vídeos. En tercer lugar, se analizan los resultados relativos a los módulos concernientes al aprendizaje automático en las fases de entrenamiento y clasificación mediante una CNN. Finalmente, se muestran los resultados relativos a los procesos (predicción) y almacenamiento de datos en la plataforma ThingSpeak²⁴.

4.1 Interfaz de la aplicación

Al ejecutar la aplicación aparece un menú inicial como el de la Figura 4.1 que ofrece las siguientes opciones:

- **Learning:** Módulo para realizar el entrenamiento de la Red Neuronal Convolutinal y el conjunto de imágenes.
- **Car Detection:** Módulo para la introducción de vídeos y la detección de vehículos.
- **ThingSpeak:** Módulo para consultar los contadores de vehículos y las estadísticas del canal de ThingSpeak en el que se guardan.
- **Forecast:** Módulo para consultar las predicciones.

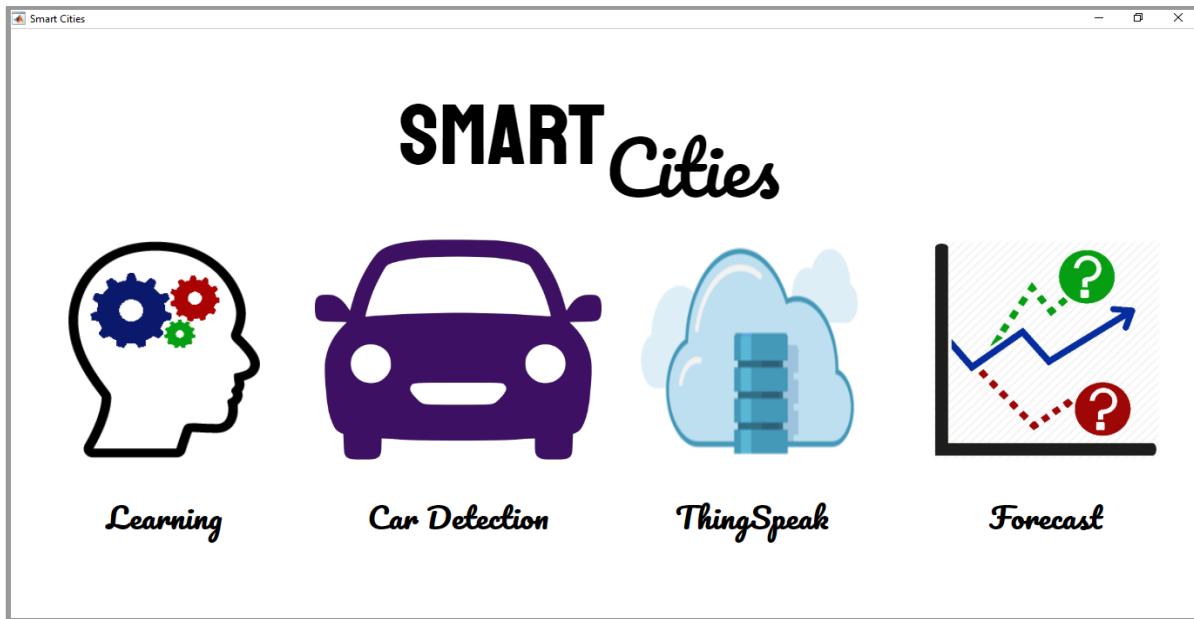


Figura 4.1 Interfaz del menú principal.

4.1.1 Learning

Al pulsar en el botón Learning de la Figura 4.1 aparece el menú mostrado en la Figura 4.2, con las siguientes opciones:

- **Image Resizer:** Permite subir una o varias imágenes para proceder a su redimensión en las dimensiones requeridas por la CNN para realizar el entrenamiento.
- **Update Training Data:** Abre la carpeta en la que se guardan los sets de imágenes del entrenamiento para poder modificarlos.
- **Retraining:** Sigue un entrenamiento de la CNN, o re-entrenamiento en caso de que se utilice un modelo pre-entrenado.

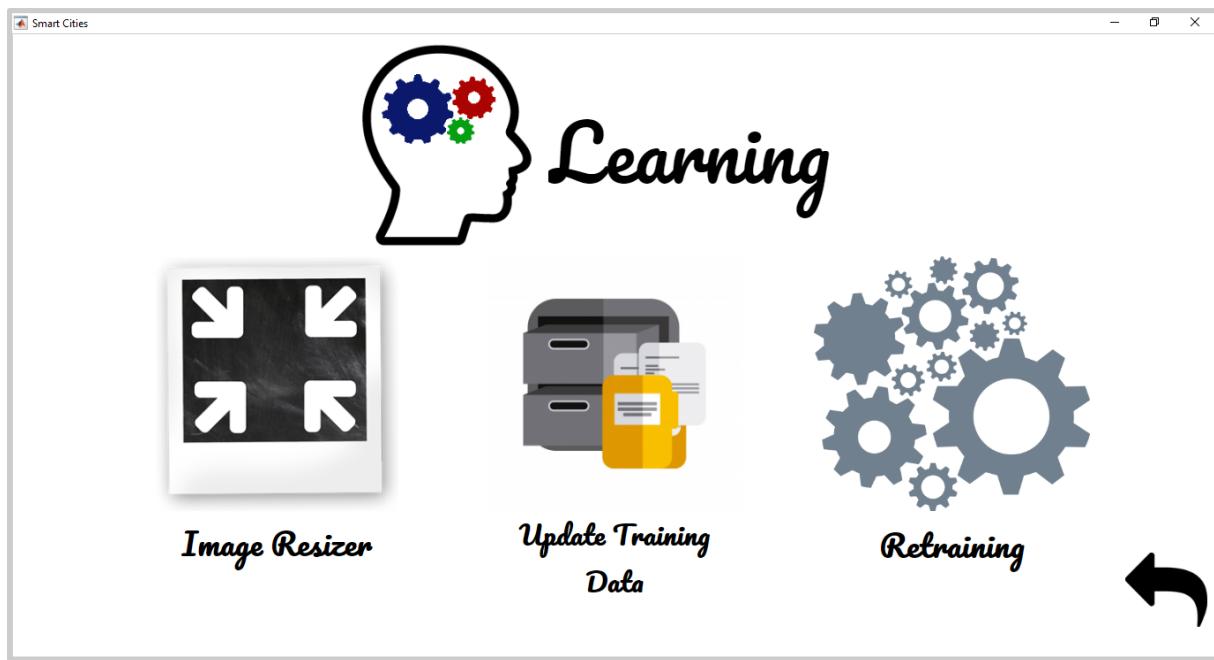


Figura 4.2 Interfaz del menú Learning.

4.1.1.1 Retraining

Al pulsar en el botón Retraining de la Figura 4.2 aparece una ventana como la mostrada en la Figura 4.3 a la izquierda, de forma que en la parte superior contiene una pestaña para acceder al menú mostrado en la misma figura de la derecha. Ambas permiten seleccionar los parámetros que tendrá el entrenamiento.

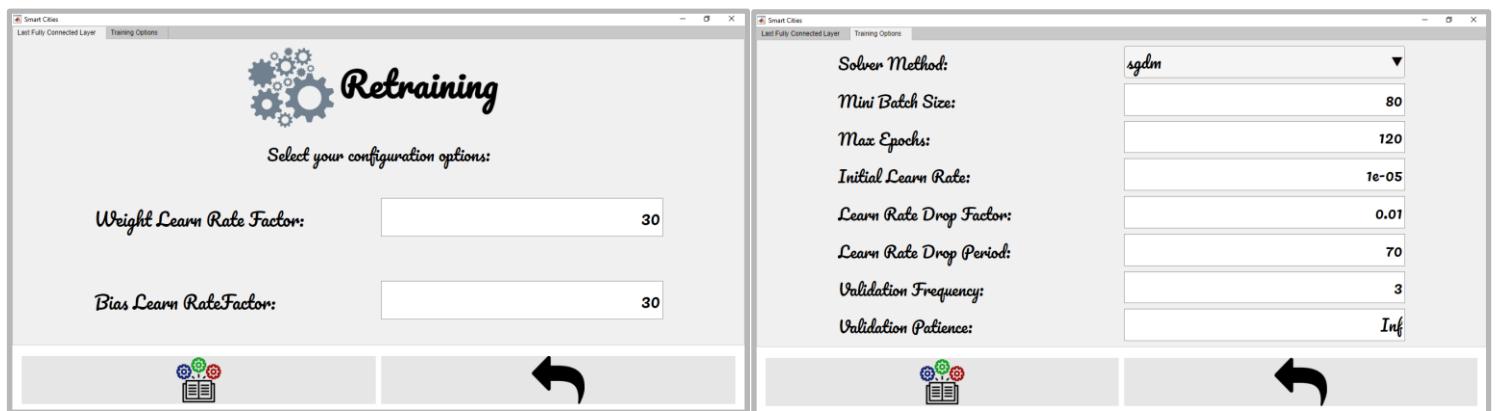


Figura 4.3 Interfaces del menú de Retraining.

4.1.2 Car Detection

Al pulsar en el botón Car Detection de la Figura 4.1 aparece una ventana como la mostrada en la Figura 4.4, con los siguientes elementos:

- En el panel izquierdo se muestra el video frame a frame con las etiquetas de identificación de los vehículos detectados.
- En el panel superior derecho aparecen las imágenes en las que se ha detectado movimiento.
- En el panel inferior derecho se muestra el resultado de la matriz de detección.
- En el panel de botones inferior se permite seleccionar el vídeo y detener la ejecución del vídeo actual.



Figura 4.4 Interfaz de Car Detection.

4.1.3 ThingSpeak

Al pulsar sobre el botón ThingSpeak²⁴ de la Figura 4.1 aparece el menú representado por la Figura 4.5, con las siguientes opciones:

- **Queries:** Permite lanzar consultas al canal para la realización de comparativas y estadísticas.
- **ThingSpeak Web:** Acceso directo a la web del canal de ThingSpeak en el que se guardan los datos de los contadores de vehículos detectados por la aplicación.

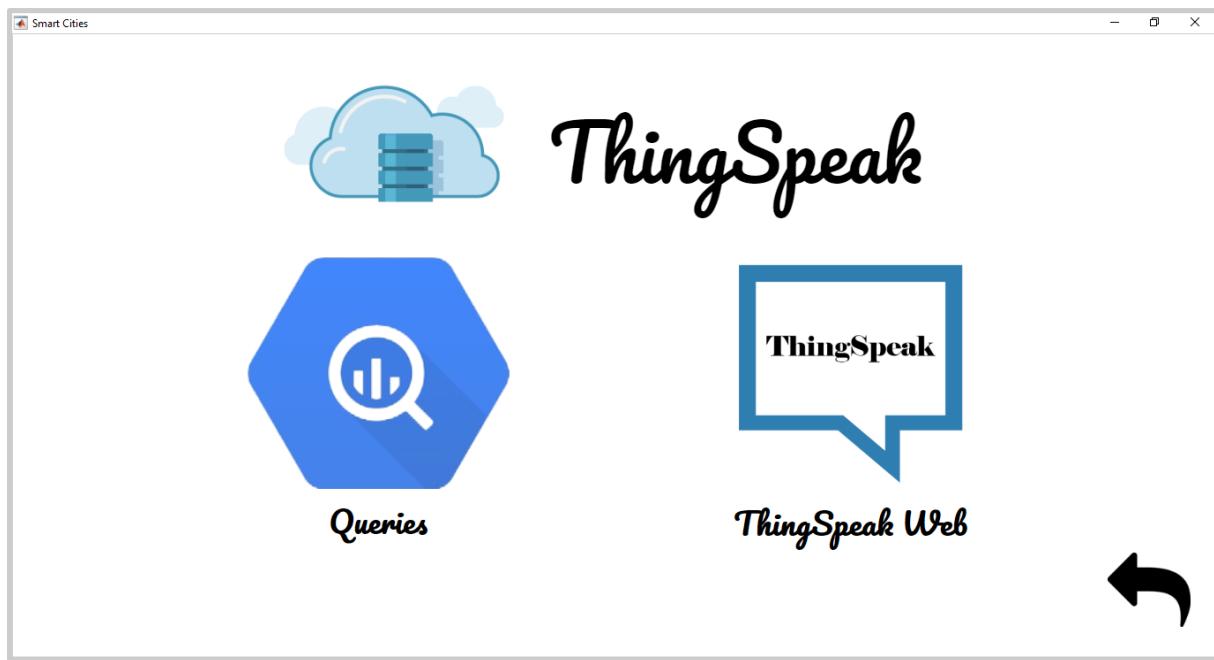


Figura 4.5 Interfaz de ThingSpeak.

4.1.3.1 Queries de ThingSpeak

Al pulsar en el botón Queries de la Figura 4.6 aparece el menú representado por la Figura 4.7, con los siguientes elementos:

- En el panel superior izquierdo se seleccionan las fechas entre las que se quiere realizar la consulta.
- En el panel inferior izquierdo se seleccionan los tipos de vehículos que se quieren buscar.
- En el panel superior derecho se muestra una tabla con el resultado de la consulta realizada.
- En el panel inferior derecho se muestra una representación gráfica mostrando los datos consultados.



Figura 4.6 Interfaz de Queries de ThingSpeak.

4.1.4 Forecast

Al pulsar sobre el botón Forecast de la Figura 4.1 aparece el menú mostrado en la Figura 4.7, con los siguientes elementos:

- En el panel superior izquierdo se selecciona el día de la semana y cuántos días se quieren predecir.
- En el panel superior derecho se muestra una tabla con el resultado de la predicción realizada.
- En el panel inferior se muestra una representación gráfica usando los datos predichos.

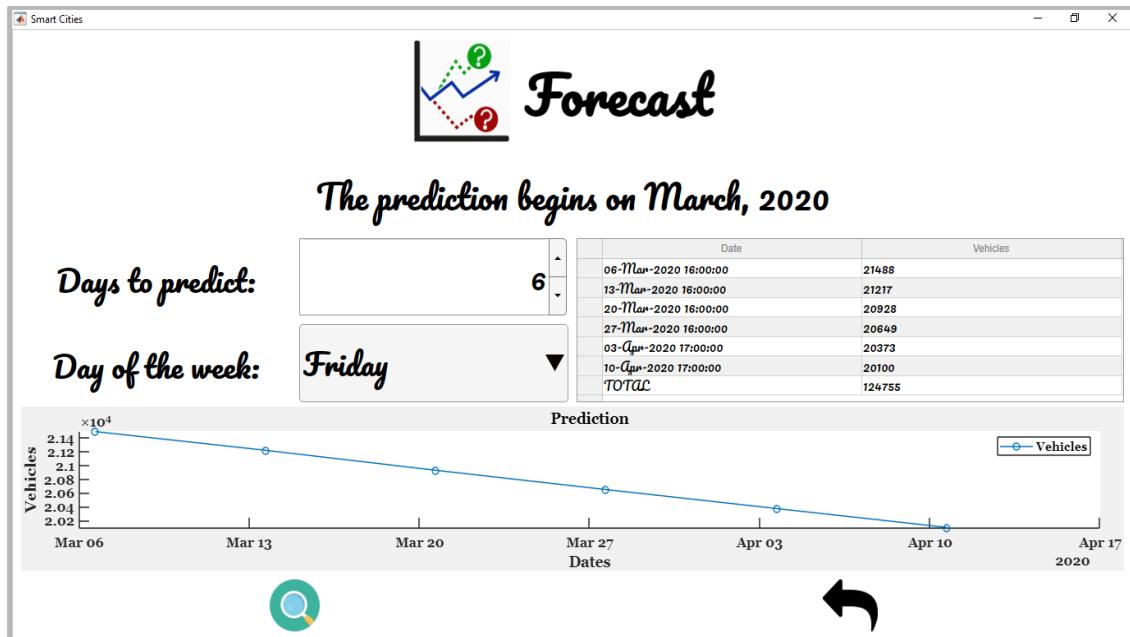


Figura 4.7 Interfaz de Forecast.

4.2 Visión por computador

A continuación se describen los resultados obtenidos al aplicar los métodos teóricos para la detección del movimiento en imágenes, la extracción de dichas regiones y la identificación de los diferentes vehículos.

4.2.1 Detección de movimiento

La primera acción que se realiza en este sistema inteligente es el cálculo del flujo óptico, es decir, para todo píxel de una imagen, se calcula un vector que indica la dirección y el sentido en el que se mueve un objeto determinado dentro de un *frame* de vídeo. Podemos ver un ejemplo de la realización de este proceso a través de la figura 4.8.

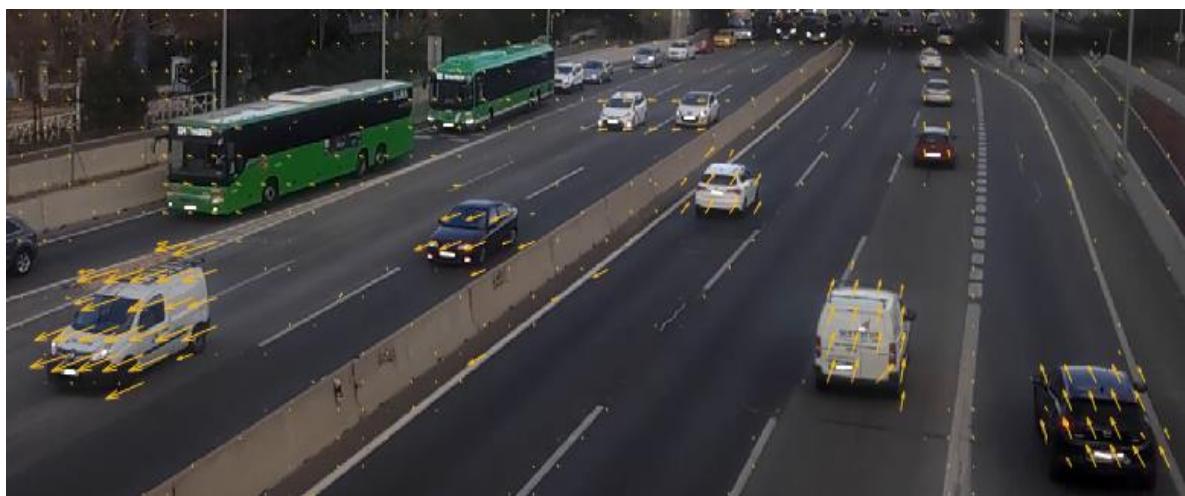


Figura 4.8 Representación del flujo óptico en una imagen de tráfico.

Como se observa en líneas generales, la detección del movimiento es muy fiel a lo que cabría esperar, pues refleja correctamente la dirección y el sentido de la marcha de los objetos (vehículos) en movimiento.

Un problema que cabe mencionar con respecto a este proceso es que, a nivel de efectividad, los métodos para el cálculo del flujo óptico de Farneback¹⁰ o Lucas-Kanade⁴ no se ejecutan lo suficientemente rápido bajo el ecosistema de MATLAB, ya que éste emplea un lenguaje de programación interpretado, lo que hace que la ejecución del programa se ralentice durante el procesamiento de los *frames*.

4.2.2 Proceso de binarización

Una vez se ha identificado la magnitud de movimiento detectado en una imagen, tal y como se muestra en la figura 4.9, el sistema tendrá que identificar en qué ubicación del frame de vídeo se ha detectado este movimiento. El primer paso para conseguir este objetivo es aislar la zona de movimiento, para ello se procede a binarizar esta imagen.

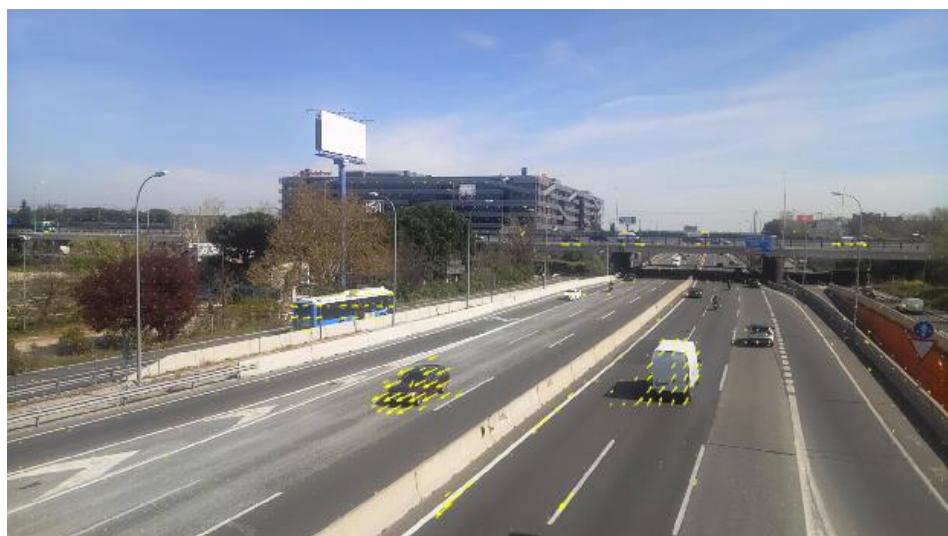


Figura 4.9 Ejemplo del cómputo del flujo óptico previo al proceso de binarización.

Recordemos que el proceso de binarización es un proceso que genera una nueva imagen en blanco y negro a partir de una imagen en color. Para generarla se recorren todos los píxeles de la imagen original. Cuando uno de ellos cumpla que la magnitud del flujo óptico en dicho punto sea superior a la suma de la media del flujo de todo el *frame* y la desviación típica de éste, lo que se conoce como valor umbral, tal y como se ha descrito previamente, entonces a dicha región se le asigna el valor lógico uno (blanco). En caso contrario, a dicho píxel se le asignaría el valor lógico cero (negro). La figura 4.10 muestra un ejemplo de este proceso a partir de la imagen mostrada en la figura 4.9, sobre la que se detectó el flujo óptico.

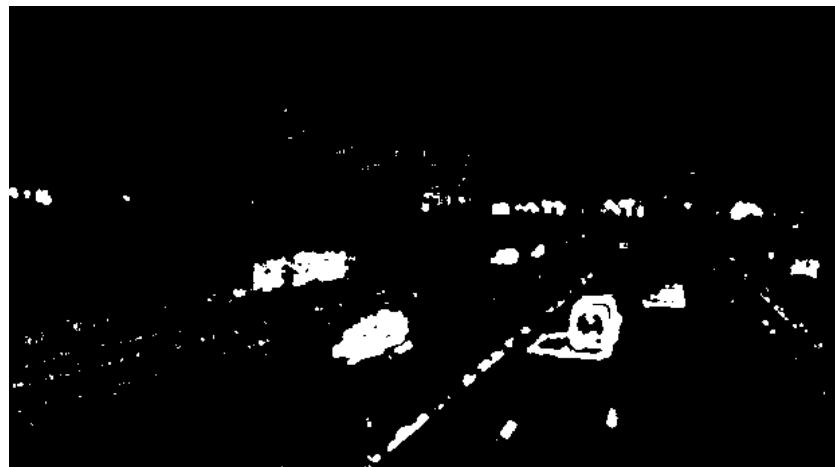


Figura 4.10 Resultado del proceso de binarización aplicado a la figura 4.9.

4.2.3 Proceso de segmentación

En este punto, a partir de la imagen binaria se procede al etiquetado de las componentes conexas, con el fin de detectar el número de objetos en movimiento y otras propiedades como su área, centroide y *Bounding Box*. Para ello se utiliza el algoritmo de etiquetado de componentes conexas de Haralick y Shapiro (1992)³ descrito en la sección 2.1.2. Cada región recibe una etiqueta numérica, cuya representación en etiquetas de color se muestra en la figura 4.11, como un ejemplo.



Figura 4.11 Segmentación de las regiones de la figura 4.10.

4.2.4 Clasificación del flujo de tráfico

Ahora que se han obtenido todas las regiones candidatas a representar un vehículo en movimiento, se extraen una serie de propiedades de éstas, una muy importante es su ubicación y su tamaño (*Bounding Box*). Esta propiedad nos permitirá extraer la imagen del *frame* de vídeo mediante la realización de un recorte para poder pasar la

imagen al modelo AlexNet⁹, así este determinará si efectivamente la imagen que le ha sido pasada como entrada es un vehículo o si bien esta debe de ser descartada.

La figura 4.12 muestra un ejemplo de como AlexNet ha clasificado varios vehículos y los ha enmarcado dentro de su área limítrofe.



Figura 4.12 Ejemplo de clasificación de vehículos y enmarcado dentro de su *Bounding Box*.

Los resultados obtenidos en la clasificación del flujo de tráfico también entrañan una serie de problemas, éstos se deben a que se ha partido del cálculo del flujo óptico para el reconocimiento de vehículos.

Uno de los problemas más notables es lo que hemos denominado “el problema de la sombra”, ya que si un objeto proyecta una sombra, ésta se mueve en la misma dirección que el objeto en movimiento y a la misma velocidad. Esto es perjudicial, ya que en el cómputo del flujo óptico no se aplica ningún método para distinguir que las cantidades de movimiento detectadas no son un mismo objeto en movimiento, lo que hará que en procesos posteriores ya no se pueda distinguir qué parte pertenece a la sombra y qué parte pertenece al objeto que se encuentra en movimiento.

Finalmente se observó que “el problema de la sombra” se generaliza para concluir que en cualquier situación en la que se detecten dos o más objetos en movimiento que se encuentren muy próximos entre sí, tras el cómputo del flujo óptico se hace imposible la distinción individual de éstos.

En la figura 4.13 podemos ver dos ejemplos ilustrativos, observándose cómo aparece un solapamiento de dos vehículos y uno sólo, en ambos casos con sus “sombras” asociadas, que hacen que el rectángulo delimitador rebase a los objetos a los que debería representar. Esto conduce a la posibilidad de que la Red Neuronal Convolucional cometa errores de detección, lo que hará que posteriormente también falle el sistema de de cómputo de vehículos para el cálculo del flujo de tráfico, ya que se contabiliza un único vehículo en lugar de los dos.

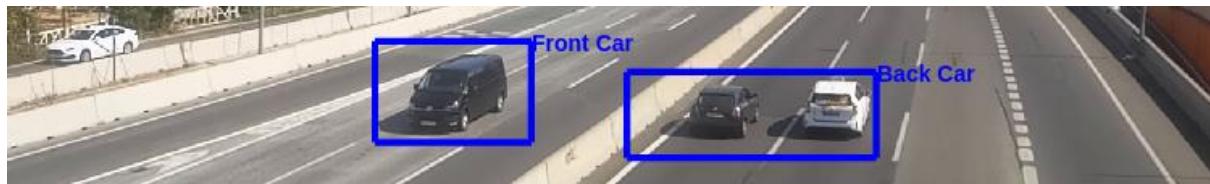


Figura 4.13 Problema de la segmentación regiones a partir de del cálculo del flujo óptico.

Otro de los problemas detectados es la sensibilidad del método del gradiente para el cálculo del flujo óptico.

Este problema proviene del hecho de que si entre dos *frames* de vídeo consecutivos se produce un ruido en la imagen debido al movimiento del dispositivo digital que graba el flujo de transporte, éste se detecta como si realmente se hubiera producido un movimiento real. Por lo que en este caso la detección y ubicación de los vehículos es muy propensa a fallos e incluso se puede llegar a identificar erróneamente la categoría de una imagen o, lo que es peor, es posible que no se reconozca ninguna característica del vehículo debido a la cantidad de ruido presente. Podemos ver un ejemplo de esta situación en la figura 4.14.

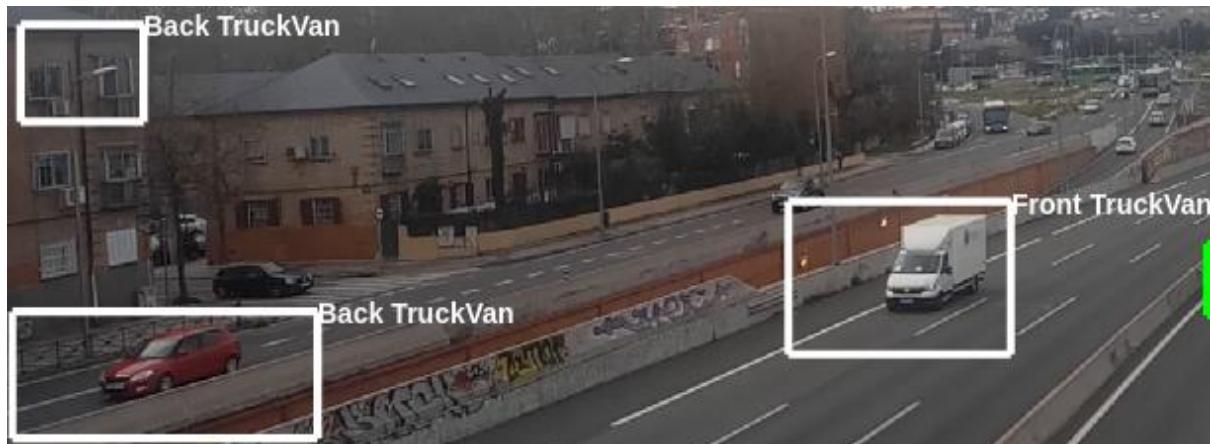


Figura 4.14 Problema de la sensibilidad del cálculo del flujo óptico.

4.2.5 Cómputo del flujo del tráfico

Para la implantación de un sistema de cómputo del flujo de tráfico se ha optado por fijar sobre la imagen una franja imaginaria, de forma que cuando los centroides de las regiones asociadas a imágenes que han sido clasificadas como vehículos en movimiento por la Red Neuronal Convolucional, se encuentren dentro de esta franja, se contabilice el paso de un vehículo.

En la figura 4.15 se muestra una representación de la franja posicionada sobre una localización fija, para contabilizar los coches que pasan por una autopista.



Figura 4.15 Posicionamiento de una franja de conteo que actúa como un dispositivo sensor.

La franja se ha ubicado sobre la imagen considerando lo siguiente:

- 1) Cuanto más cerca está un vehículo de la parte inferior de la imagen, y por tanto del dispositivo de captura, mayor es su tamaño, lo que tal vez provoque que el vehículo sea contabilizado más de una vez.
- 2) Algo parecido ocurre en la parte superior de la imagen, debido al reducido tamaño de los vehículos en esta zona.

El hecho de seguir estas directrices permitió minimizar parte del error por defecto y por exceso del conteo del flujo.

A continuación se muestran dos ejemplos ilustrativos de los resultados obtenidos a partir de vídeos reales de tránsito del tráfico. Éstos fueron tomados manualmente en la autopista del nordeste de Madrid (A-2) en el tramo que pasa por Canillejas.

En la figura 4.16 (Vídeo 1) se muestra una gráfica con los resultados obtenidos en relación a la detección de vehículos. En el eje horizontal se muestra el número de aciertos y fallos totales, mientras que en el vertical el tipo de vehículo detectado. La interpretación de la misma es como sigue: para cada uno de los vehículos en tránsito el sistema lo detecta, de forma que la observación a través del experto humano determina si se trata de un acierto o fallo.

A continuación se describe el análisis de los errores cometidos en este primer vídeo:

- El 42% de los fallos se deben a que el modelo AlexNet⁹ no fué capaz de clasificar correctamente la imagen que se le pasó como entrada a partir de un recorte del frame de vídeo. Esto se debió a que algunas clases son a veces confundidas con otras, como por ejemplo *BackTruckVan* con *BackCar* o bien al revés.
- Un 33% de los fallos cometidos por el sistema en el procesamiento de este vídeo se debió a que se contabilizó más de una vez un vehículo al quedar su vector centroide atrapado más de un instante de tiempo en la franja de conteo.

- Apenas un 16% de los errores se debieron al tamaño y al posicionamiento de la franja de conteo.
- Finalmente el 9% del error cometido se ha debido al denominado “problema de la sombra”, lo que ha hecho que apenas algún vehículo sea ignorado.

Resultados (Vídeo 1)

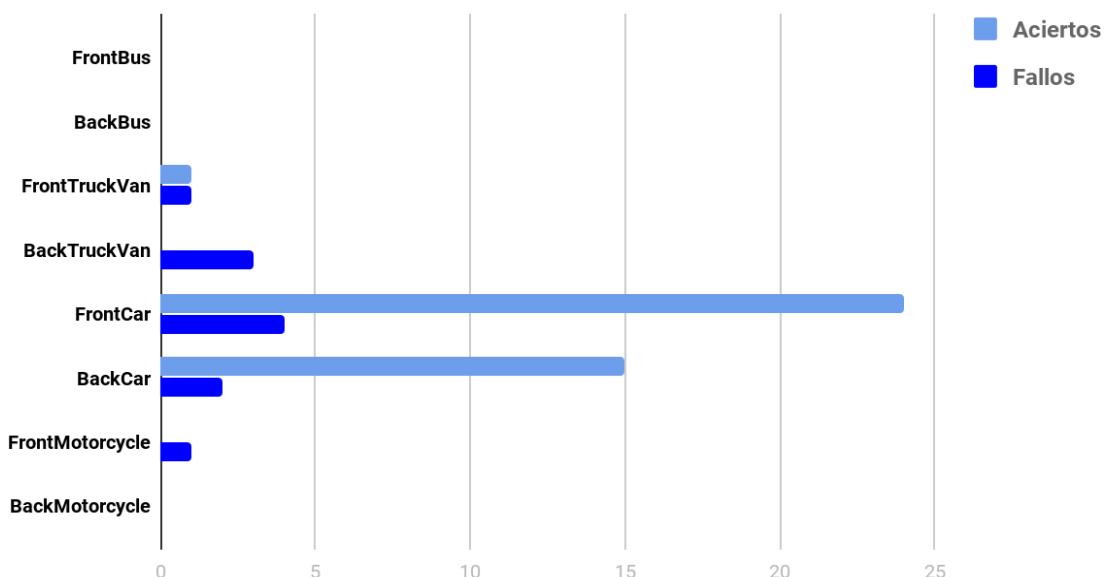


Figura 4.16 Resultados de detección en Vídeo 1 para el tránsito de vehículos.

La figura 4.17 (Vídeo 2) representa un vídeo en el que ha aumentado el flujo de tráfico en comparación con el que se puede intuir de la figura 4.16.

Los errores cometidos en este segundo vídeo se describen a continuación:

- En primer lugar, el 48% de los fallos cometidos se debió a que el modelo AlexNet⁹ no fué capaz de clasificar correctamente los vehículos detectados, confundiendo sobre todo las categorías *BackCar*, *BackTruckVan*, *FrontCar*, *FrontTruckVan*. Por lo que aunque la detección de vehículos fue correcta, su clasificación no.
- En segundo lugar, el 34% de los fallos ocurrieron debido a que la franja de detección no contó los vehículos que pasaron sobre ella, lo que ha penalizado sobre todo en este caso a las categorías de vehículos *BackCar* y *FrontCar*.
- En tercer lugar, el 10% de los fallos se debieron a que el vector centroide de las imágenes clasificadas se quedó atrapado más de un instante de tiempo en la franja de conteo.

- Por último, el 8% restante de los fallos aconteció debido al “problema de la sombra”, que como se mencionó anteriormente, puede hacer que dos o más vehículos sean computados como un único vehículo de otra categoría, que puede llegar a ser completamente diferente a la de los originales.

Resultados (Vídeo 2)

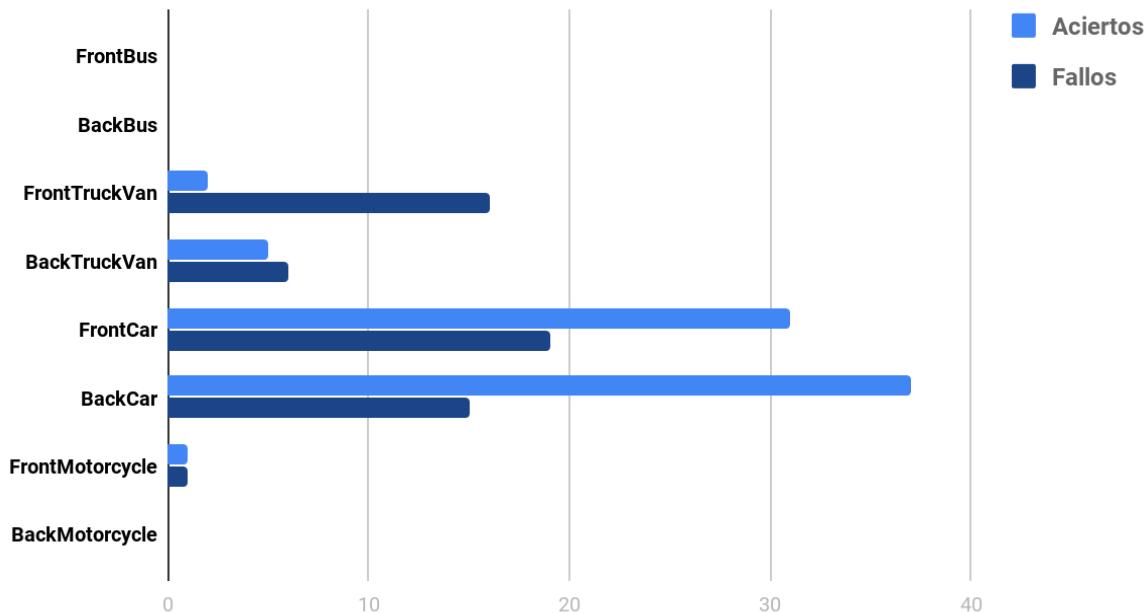


Figura 4.17 Resultados de detección en Vídeo 2 sobre el tránsito de vehículos.

Las conclusiones obtenidas a partir de los vídeos anteriores han sido las siguientes:

- a) Los resultados indican que a medida que aumenta el número de vehículos de una clase concreta, mayor es el número de aciertos del sistema inteligente a la hora de clasificar, mientras que para un número pequeño de vehículos en tránsito, el porcentaje de aciertos disminuye.
- b) Los centroides de los coches que no han sido detectados por la franja de conteo, debido a que su velocidad era demasiado alta, se debe a que a veces la velocidad de grabación del dispositivo empleado, 30 fps, puede resultar un poco lenta, pues al producirse un salto entre 2 frames de vídeo consecutivos, éstos quizás no puedan ser capturados.
- c) Los resultados son variables si tenemos en cuenta la ubicación del dispositivo de grabación, pues para diferentes ángulos es posible obtener resultados distintos en lo que respecta a la detección de vehículos y a la clasificación de éstos. Lo que sugiere que la mejor práctica que se puede realizar en este aspecto es tener el dispositivo de grabación totalmente alineado frontalmente

con respecto a la vía de tráfico en la que se realice la grabación, de forma que esta pueda visualizarse completa y claramente.

- d) Se ha de evitar que las grabaciones sean realizadas directamente por personas, ya que el pulso de éstas se podría traducir en que los vídeos quedan llenos de ruido, lo que dificulta la clasificación a la CNN. También puede derivar en que si el usuario mueve el vídeo, por poco que sea, la franja de conteo puede quedar situada en una posición que no resultaría adecuada para realizar el conteo de los vehículos en tránsito.

4.3 Aprendizaje profundo

En esta sección se describen los resultados obtenidos durante el proceso de entrenamiento y el ajuste de éste para el modelo de Red Neuronal Convolutacional AlexNet⁹. También se analizan los resultados obtenidos a partir de la clasificación de las diferentes imágenes de vehículos en movimiento.

4.3.1 Ajustes del entrenamiento e interpretación de los resultados

Debido a que hemos construido un set que cuenta con 100 imágenes por cada una de las 10 clases del problema de clasificación de vehículos, se ha determinado que el tamaño de los lotes de imágenes a procesar por la red (batch size) sea de 80, esto reduce el tiempo de ejecución ya que prácticamente se procesa una categoría completa de una pasada.

El número de veces que todas las imágenes son pasadas por la red (*iterations*) durante el aprendizaje se calcula automáticamente en base al tamaño del lote (*batch size*). En este caso, para un valor de 80, son 10 iteraciones.

El número de veces que se repite el número de iteraciones anterior se conoce como ciclos de entrenamiento (*epochs*), siendo un factor determinante para establecer el tiempo que dura el entrenamiento. Para un conjunto de imágenes como el utilizado en los experimentos, que resulta elevado para un computador de potencia medio-alta, es necesario una gran cantidad de ciclos de entrenamiento para que AlexNet pueda aprender las características más representativas de cada clase de vehículo o elemento de la vía.

Desde un punto de vista técnico, se han necesitado un total de 120 ciclos de entrenamiento para realizar el proceso de aprendizaje, con ello el proceso se completa en aproximadamente unos 10 minutos, aunque este tiempo puede variar en función de la potencia de la unidad de procesamiento gráfico (GPU) del computador empleado.

El método seleccionado para que AlexNet ajuste el valor de los pesos ha sido SGDM (*stochastic gradient with momentum*), éste consiste en moverse ligeramente en la dirección de menos el gradiente de la función de error. La capacidad de aprendizaje se regula mediante la variable de factor de aprendizaje (*learn rate factor*), este valor sirve para indicar la magnitud del desplazamiento que se realizará en la dirección de menos gradiente de la función del error cometido para ajustar el valor de los pesos.

Es un valor muy pequeño, fijado a 10^{-5} en los experimentos realizados. El factor de aprendizaje puede permanecer constante a lo largo del entrenamiento, sin embargo es posible establecer que éste disminuya una cierta cantidad cada cierto número de ciclos de entrenamiento (*learn rate drop period*). Cuando se alcanza dicho número de ciclos o un múltiplo de éste, el factor de aprendizaje es multiplicado por un factor constante (*learn rate drop factor*), normalmente muy pequeño, que disminuye el factor de aprendizaje total.

Esto hace que los pesos de la Red Neuronal se ajusten más lentamente. Sin embargo, al movernos más despacio hacia el mínimo de la función de error lograremos que a la larga disminuya el error cometido al ajustar los pesos. Los valores establecidos para estas variables han sido fijados en 70 y 0,01.

Es necesario comprobar durante el entrenamiento que AlexNet⁹ está aprendiendo realmente y no memorizando las características de las imágenes que componen el conjunto de entrenamiento. Para evitar esto se realiza la clasificación de un determinado número de las imágenes que componen el conjunto de validación cada cierto número de iteraciones (*validation frequency*), para comprobar que la eficacia del aprendizaje de la red durante el entrenamiento sea menor que la eficacia de la capacidad de la red para clasificar imágenes, ya que de lo contrario AlexNet estaría aprendiendo a extraer características que solo se encuentran en las imágenes de entrenamiento, por lo que sus clasificaciones serían muy inexactas con cualquier otra imagen que no esté en ese conjunto.

Es bueno realizar estas validaciones a intervalos de tiempo relativamente pequeños, por lo que se ha establecido que la validación del aprendizaje tenga lugar cada 3 iteraciones.

Es posible utilizar el proceso de validación para detener el entrenamiento cuando transcurra un cierto número de validaciones sin haber conseguido disminuir el error que se comete en las clasificaciones (*validation patience*). Esta opción suele requerir una gran experiencia en el entrenamiento de las Redes Neuronales Convolucionales por lo que, en los experimentos realizados, este valor se ha fijado en infinito, ya que de esta forma no se detendrá el entrenamiento aunque existan varias iteraciones en las que el proceso de validación indique qué resultado de la clasificación es inferior al

aprendizaje. No por ello se ha dejado de obtener unos buenos resultados en el entrenamiento sin caer en el problema del sobreajuste.

Como se ha aplicado la técnica de la “transferencia de aprendizaje”¹³, también hay que indicar los valores del factor de aprendizaje y de la variable “bias” de la primera capa del tipo totalmente conectada, *Fully Connected*.

Estos valores, a diferencia de lo que ocurre en los pasos anteriores, no son el valor real de dichas variables para esa capa, sino que son escalares positivos que multiplican al factor de aprendizaje global, establecido anteriormente.

Para explicar los resultados obtenidos en el entrenamiento haciendo uso de esta configuración, se toma como referencia las gráficas de las figuras 4.18 y 4.19.

El eje horizontal de ambas gráficas se expresa en función del número de iteración, cada una de ellas en un instante de tiempo diferente. En la figura 4.18, el eje vertical sirve para expresar la eficacia alcanzada por la red al realizar sus clasificaciones en una iteración concreta. En la gráfica 4.19, el eje vertical indica la semejanza entre la salida que se desea obtener y la salida obtenida, cuanto más semejantes sean, los valores en la vertical se aproximarán más a cero.

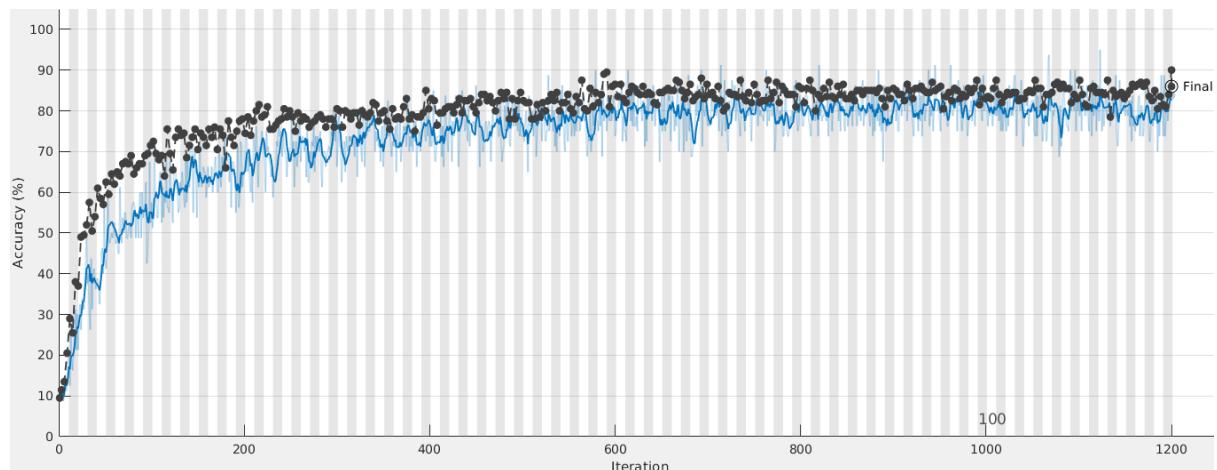


Figura 4.18 Variación de la eficacia de las clasificaciones a lo largo de las iteraciones.

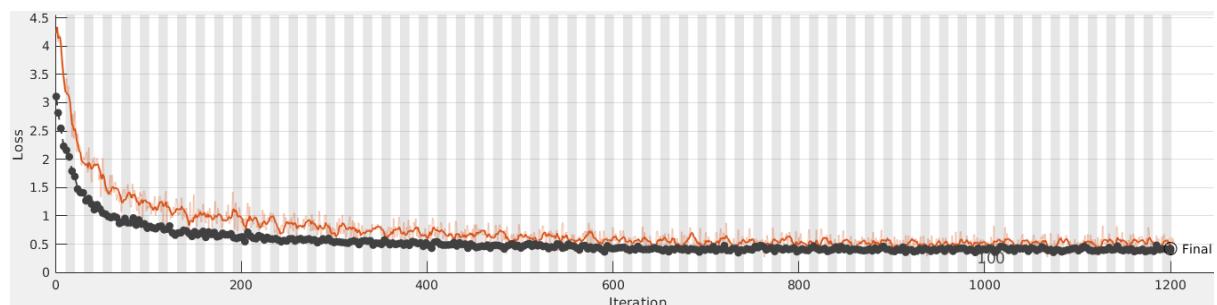


Figura 4.19 Variación del error cometido en las clasificaciones a lo largo de las iteraciones.

Los resultados observados en la figura 4.18 indican que a medida que se han ido sucediendo los ciclos de entrenamiento, la eficacia de las clasificaciones con respecto del total de imágenes clasificadas ha mejorado sin que apenas tenga lugar algún caso en el que la eficacia de las clasificaciones del aprendizaje supere a la del proceso de validación.

Esto conduce a la conclusión de que tal vez AlexNet⁹ tenga un ligero problema al clasificar alguna clase de imagen, sin embargo, ante la ausencia de una gran cantidad de picos que rebasen los puntos de validación, es probable que para esa clase o clases el fallo de las predicciones ronden aproximadamente en torno al 25%.

En la figura 4.19 los resultados indican que según han transcurrido los *epochs* la diferencia entre la salida generada y la salida deseada ha disminuido.

Observando las figuras también se concluye que el aprendizaje tiende a converger en torno al ciclo de entrenamiento 120, por lo que si se prolonga el proceso de entrenamiento más allá de este ciclo, es posible que entonces AlexNet tendiera a sobreajustarse ya que comenzaría a buscar características demasiado específicas de las imágenes del conjunto de entrenamiento, en lugar de generalizar el conocimiento que tiene de ellas.

El modelo AlexNet⁹ obtenido a partir de este entrenamiento y con la configuración de los parámetros anteriormente descritos, ha dado lugar a un modelo entrenado que posee una tasa de acierto en sus clasificaciones del 86%, un valor aceptable si se considera el número total de imágenes analizadas.

En total, este sistema inteligente es entrenado para identificar hasta 10 clases de imágenes, compuestas tanto por vehículos como por otros elementos presentes en las vías de circulación: *FrontCar*, *BackCar*, *FrontBus*, *BackBus*, *FrontTruckVan*, *BackTruckVan*, *FrontMotorbike*, *BackMotorbike*, *Asphalt* y *Wall*. En la figura 4.20 mostraremos algunas de estas imágenes a modo de ejemplo.



Figura 4.20 Collage de imágenes del proceso de entrenamiento.

4.3.2 Clasificación post-entrenamiento

Para profundizar en el análisis y con carácter más general que el realizado durante el entrenamiento, se procede a volver a clasificar todas las imágenes que componen el conjunto de imágenes disponibles, es decir, tanto las que fueron seleccionadas para el proceso entrenamiento, como las que fueron seleccionadas para el proceso de validación, aún conscientes de que en los procesos de aprendizaje no es habitual probar la red con las mismas imágenes utilizadas durante el entrenamiento.

Todos los resultados obtenidos se anotan, indicando cuántas predicciones fueron correctas e incorrectas con respecto de cada clase, también se contabilizará el total de aciertos y fallos obtenidos con respecto al total de imágenes de cada clase.

Esta información se muestra en forma de matriz y sirve para identificar de forma más clara qué clases necesitan una mayor cantidad de imágenes para que la red pueda aprender a generalizar sus características. Esta matriz se muestra en la figura 4.21 y es conocida como matriz de confusión¹².

92			1				1		6
	81	1		1	15	2			
		90				10			
		3	77				20		
	1	5		81	2	2		9	
	5	2			90	1		1	1
		2				96		2	
	1		9			1	89		
		3			1	4		92	
7	1	4		2			3		83

Figura 4.21 Matriz de confusión representando la eficacia del proceso de entrenamiento.

Como se puede observar, esta matriz es cuadrada (clases x clases), y cada fila representa la clase real a la que pertenece un objeto determinado, mientras que las distintas columnas de dicha fila indican las distintas clases en las que la red puede clasificarlo. Por lo tanto, los números de la diagonal principal se han de interpretar como el total de imágenes o vehículos que han sido correctamente clasificados para una categoría concreta. En este caso, como cada clase posee un máximo de 100

imágenes, éste es el acierto máximo que la Red Neuronal Convolutinal puede tener con respecto al total de imágenes de una categoría.

Para una mejor interpretación de la figura 4.21, diremos que la cuarta línea de la matriz representa la clase *BackMotorbike*. Partiendo de aquí se puede reconocer que el elemento de esta línea, que coincide con la diagonal principal, es el número de motos vistas por detrás que se han clasificado correctamente sobre un total de 100 motos vistas desde atrás, en este caso, se tendría un 77% de aciertos para esta clase. El resto de elementos numéricos ubicados en esa misma fila son predicciones incorrectas con respecto a esta clase, por ejemplo, en 20 ocasiones se ha confundido esta imagen como si fuera un una moto por delante, e incluso en otras 3 ocasiones como si fuera un coche por detrás.

Se concluye pues, que esta matriz proporciona una ayuda de cara al análisis de los resultados del entrenamiento, de forma que es posible determinar qué clases necesitan un mayor número de imágenes para que el modelo AlexNet pueda generalizar el conocimiento que tiene sobre ellas y así evitar caer en el problema del sobreajuste (*overfitting*).

4.4 Procesamiento y predicción en ThingSpeak

La herramienta ThingSpeak²⁴ ha sido de gran utilidad para la realización de la aplicación, ya que ha facilitado el manejo de los datos en un servidor en la nube, generados por el análisis de los diferentes tipos de vehículos, estando disponible para cualquier integrante del equipo y pudiendo facilitar su consulta de una forma rápida y sencilla, mostrando todos los datos recopilados en gráficas para una mejor visualización y entendimiento de los mismos.

También proporciona un control de la concurrencia que ayuda a que el servidor gestione las peticiones de escritura de los datos que le llegan en varias peticiones al mismo tiempo.

Decir que ThingSpeak proporciona diferentes apps para conectar el canal a otros sitios web, por ejemplo, permite conectar el canal a una cuenta de Twitter para poder monitorear cuándo se producen escrituras en el canal de ThingSpeak seleccionado y poder interactuar con los usuarios de la red social, haciendo uso de IoT.

Los datos utilizados en los experimentos con fines de predicción provienen del repositorio de Tráfico de la comunidad de Madrid, un histórico de datos del tráfico desde 2013 hasta 2020. Concretamente se ha realizado un conteo de los vehículos que pasan entre las las 16:00hs y las 18:00hs por un mismo punto kilométrico de la carretera de circunvalación M30 en Madrid.

4.4.1 Resultados obtenidos

En la figura 4.22 se muestran los resultados de las predicciones realizadas para los siguientes días:

- **Lunes:** Se realizó una predicción de los 5 lunes posteriores al día 24 de Febrero de 2020. Con resultados variando entre 22800 y 21200 vehículos.
- **Martes:** Se realizó una predicción de los 10 martes posteriores al 25 de Febrero de 2020. Se obtuvieron resultados ligeramente decrecientes variando entre 22700 y 22200 vehículos.
- **Miércoles:** Se realizó una predicción de los 15 miércoles posteriores al 26 de Febrero de 2020. Los resultados muestran un decrecimiento inicial desde 24100 hasta 22900, con un posterior crecimiento hasta los 23200 y finalmente un decrecimiento hasta los 22100 vehículos.
- **Jueves:** Se realizó una predicción de los 9 jueves posteriores al 27 de Febrero de 2020. Los resultados en general son decrecientes, desde 23900 hasta 23200.

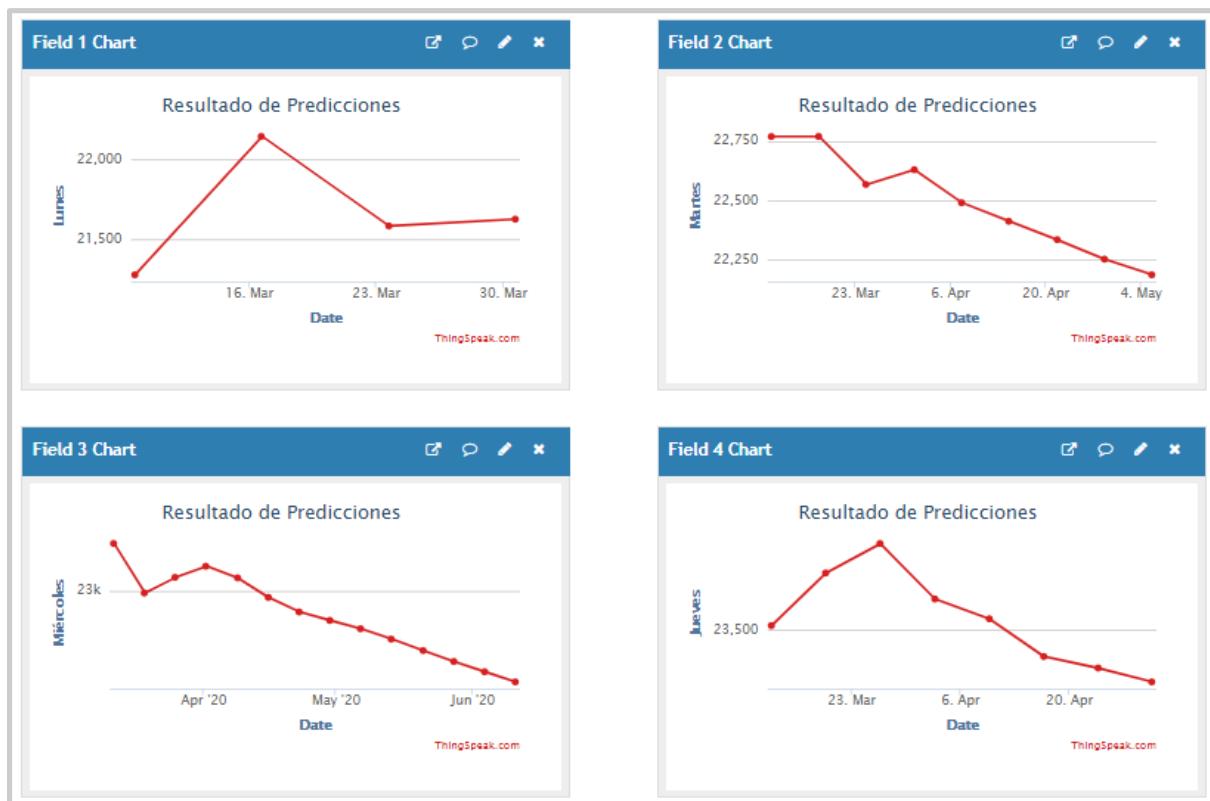


Figura 4.22 Representación resultados de predicción de lunes – jueves.

En la figura 4.23 se muestran los resultados de las predicciones realizadas para los días que se indican a continuación:

- **Viernes:** Se realizaron 7 predicciones de los viernes posteriores al 28 de Febrero de 2020. Los valores obtenidos variaron entre 22700 y 21700 vehículos.
- **Sábado:** Se realizaron 11 predicciones de los sábados posteriores al 29 de Febrero de 2020. Los valores oscilaban entre 18500 y 17300 vehículos.
- **Domingo:** Se realizaron 8 predicciones de los domingos posteriores al 23 de Febrero de 2020. Los resultados obtenidos muestran una tendencia decreciente oscilando entre 18600 y 17700 automóviles.

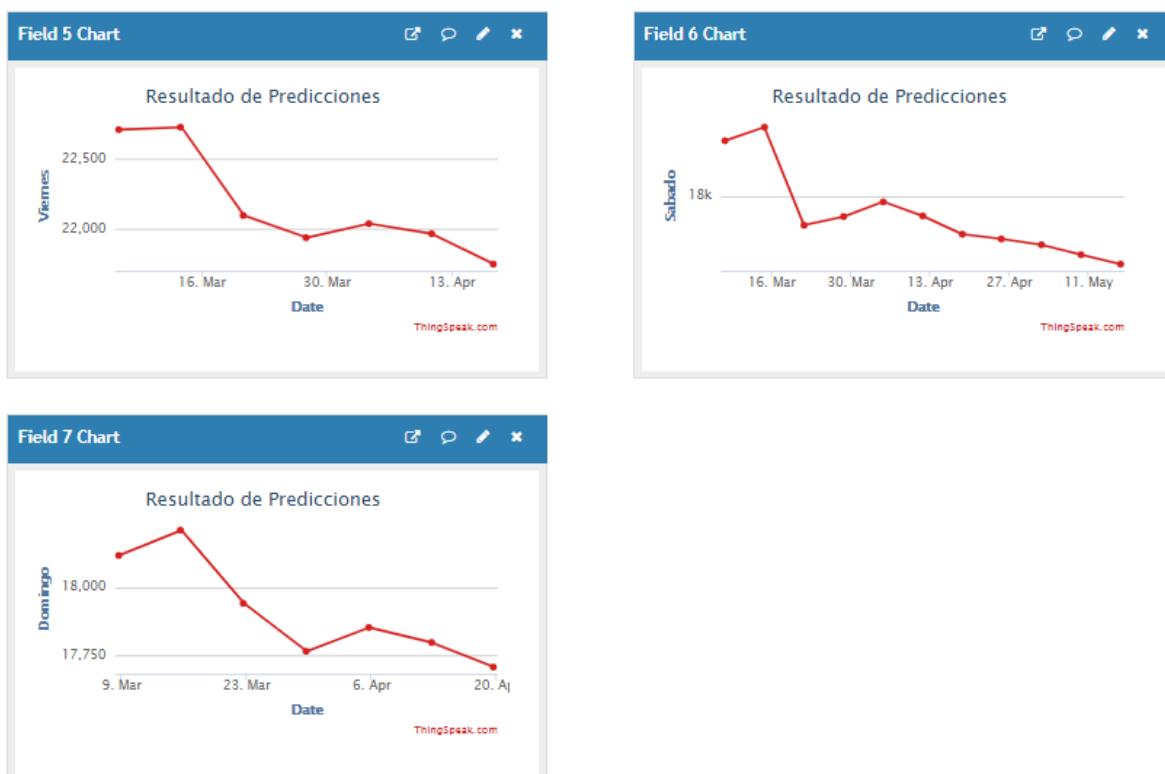


Figura 4.23 Representación resultados de predicción de viernes – domingo.

Los días de la semana con menor flujo de vehículos son el sábado y el domingo con una media de 18000 vehículos, mientras que el día con mayor flujo es el jueves. El resultado más alto se corresponde con el jueves 5 de Marzo de 2020, con más de 23900 vehículos.

El resultado más bajo se corresponde con el sábado 16 de Mayo de 2020 con algo más de 17350 vehículos. En el Anexo III^{III} se muestran las tablas detalladas con todos los resultados obtenidos.

No es posible derivar conclusiones de utilidad real informativa, dado que no se conocen al detalle las características de los datos en relación a variables asociadas con la meteorología u otros factores determinantes, no siendo el objetivo del trabajo dicho análisis, sino la integración de los modelos de predicción en la plataforma IoT para consultas externas.

4.5 Resultados del contador de vehículos de ThingSpeak

A partir del análisis de resultados mostrados en la tabla del Anexo IV^{IV}, se observa una gran afluencia de vehículos del tipo coche, tanto vistos por delante como por detrás, seguidos de un menor número de furgonetas y camiones, dejando en un segundo plano a las motocicletas y autobuses, con números bastante bajos.

Estas cifras admiten la posibilidad de errores, ya que la CNN Alexnet falla a la hora de identificar los diferentes vehículos, pudiendo afirmar que resulta más fácil identificar vehículos del tipo coche, debido a su gran número registrado en el canal de ThingSpeak²⁴, y que resulta más difícil visualizar el tránsito de vehículos de tipo buses ya que hay un menor número de ellos contabilizados en el canal.

En las figuras figuras 4.24, 4.25, 4.26 y 4.27 se muestran las gráficas de cada tipo de vehículo contabilizado en el canal ThingSpeak.

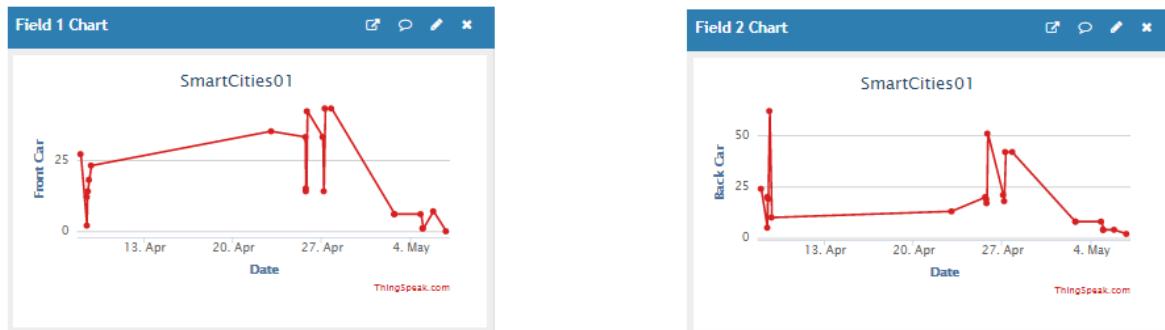


Figura 4.24 Representación resultados del conteo de tipos de vehículos de Coches por delante y por detrás.

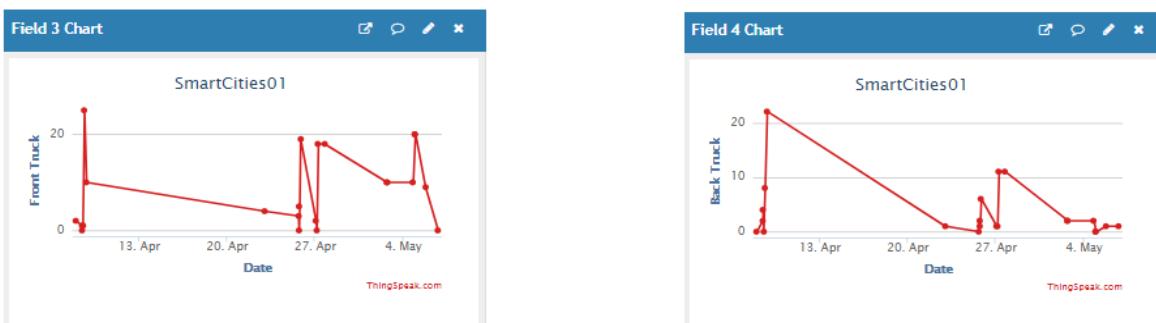


Figura 4.25 Representación resultados del conteo de tipos de vehículos de Furgonetas y Camiones por delante y por detrás.

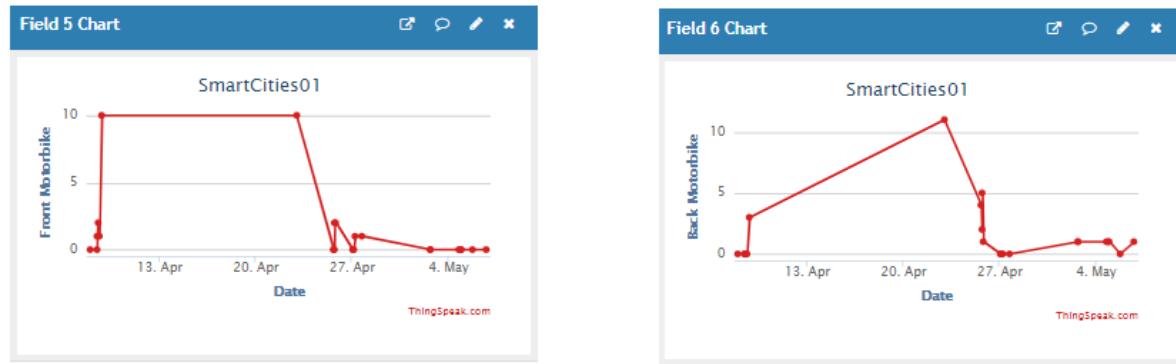


Figura 4.26 Representación resultados del conteo de tipos de vehículos de Motos por delante y por detrás.

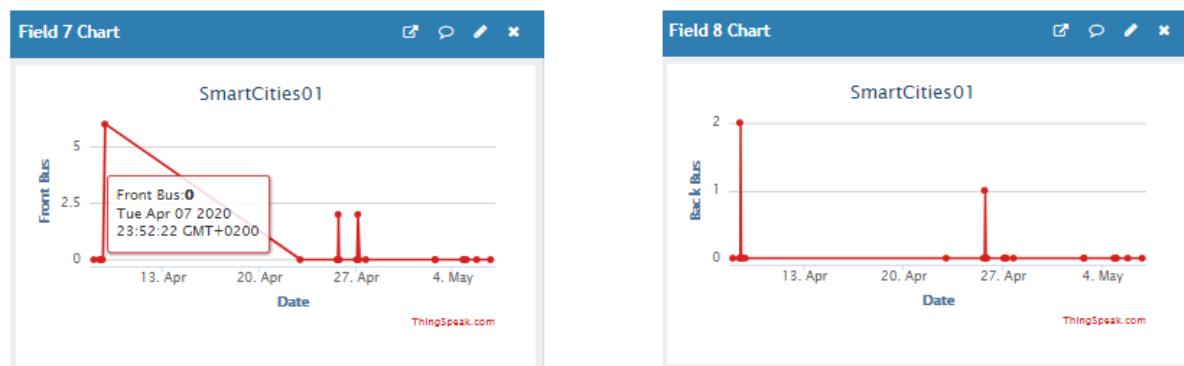


Figura 4.27 Representación resultados del conteo de tipos de vehículos de Buses por delante y por detrás.

En la figura 4.28 se muestra la gráfica en la que se representa de forma unificada el número de vehículos contabilizados en el canal ThingSpeak²⁴ dedicado a tal fin. La visualización de esta gráfica es posible gracias a la aplicación desarrollada específicamente con tal finalidad en ThingSpeak, que permite realizar consultas con visualización conjunta del número de vehículos en cada uno de los días representados en el eje horizontal.

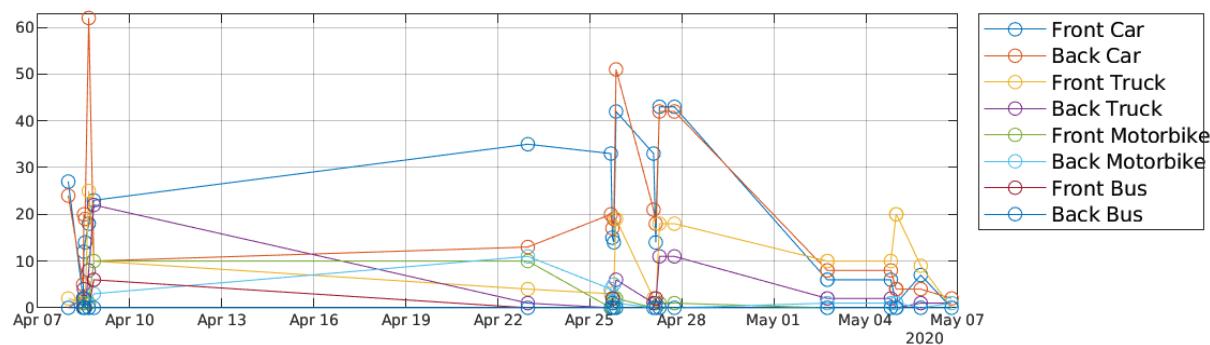


Figura 4.28 Representación comparativa de todos los diferentes tipos de vehículos.

ThingSpeak²⁴ resulta ser una opción de gran utilidad, que permite mantener un registro histórico de los vehículos detectados por el sistema. De esta forma, tal y como

se puede visualizar y deducir a partir de la representación gráfica de la figura 4.28 y las 4.24 a 4.27, el hecho de disponer de esos datos permite poder realizar análisis de los mismos, bien con fines de predicción o para otro tipo de estudios tales como por ejemplo determinar el tipo de vehículos que circulan y su nivel contaminante o capacidad del transporte público de viajeros.

Si desea saber como hacer uso de la aplicación, vaya al Anexo V⁵.

5. Conclusiones y trabajo futuro

Conclusiones

En el presente trabajo se ha desarrollado una aplicación para la identificación de diferentes tipos de vehículos a partir de imágenes de vídeo, circulando en ambos sentidos de una vía mediante el uso de diversas tecnologías en el ámbito de IoT, incluyendo técnicas de Visión por Computador⁴ para análisis del movimiento y segmentación de las imágenes, con el objetivo de proporcionar a la Red Neuronal Convolutacional Alexnet⁹ las entradas necesarias para la identificación del tipo de vehículos. Junto con las técnicas mencionadas se ha desarrollado un procedimiento para contar los vehículos en circulación de la forma más precisa posible para evitar duplicidades, de forma que un mismo vehículo sea contabilizado más de una vez. El diseño global de las técnicas implementadas hace posible que el sistema funcione correctamente con una amplia tasa de acierto.

El objetivo principal ha sido la puesta en funcionamiento de un sistema software para poder determinar la cantidad y el tipo de vehículos que pasan por una determinada zona de una carretera, para más adelante poder hacer predicciones en base a esos datos y poder proporcionar al usuario, en tiempo real, qué es lo que está sucediendo.

La estructura está formada por una cámara que graba los vehículos en movimiento y actúa como sensor generador de datos, enviando la información a un servidor para analizar el tipo de vehículos presentes en cada frame. Si bien, dado que la solución aportada no es real, sino una solución de concepto, se ha sustituido la captura de imágenes en tiempo real por secuencias de vídeo pregrabadas.

La estrategia que se utiliza para el reconocimiento de los vehículos se basa en la aplicación de técnicas de Aprendizaje Profundo²⁷, concretamente mediante el uso de una CNN convenientemente entrenada, habiendo elegido el modelo Alexnet pre-entrenado para mil objetos, que es rediseñada y re-entrenada para adaptarse a la clasificación de vehículos del tipo previsto en la aplicación.

De esta forma se ha realizado lo que se conoce como “transferencia de aprendizaje”¹³, aprovechando el mencionado modelo pre-entrenado la Red Neuronal recibe imágenes para llevar a cabo la clasificación de ocho tipos de vehículos definidos (FrontCar, BackCar, FrontMotorbike, BackMotorbike, FrontTruckVan, BackTruckVan, FrontBus, BackBus).

Para obtener en las imágenes de la secuencia de vídeo las zonas donde se ha producido el movimiento y por tanto la determinación de las regiones correspondientes a los vehículos, se utiliza una combinación de técnicas de visión computacional basadas en la detección del flujo óptico y de segmentación de regiones, que permite identificar la posición de los vehículos en cada una de las imágenes pertenecientes a la secuencia recibida en función del movimiento detectado. Los resultados del análisis se almacenan en la plataforma pública de ThingSpeak²⁴, utilizando así el concepto de ubicación en la nube, quedando a disposición de cualquier persona o institución.

Además de los datos obtenidos relativos al cómputo de los vehículos mediante las técnicas de Visión por Computador⁴ y el uso de del Aprendizaje Profundo²⁷, también se ha desarrollado un módulo de predicción, basado en el tratamiento de datos públicos del Ayuntamiento de Madrid¹⁵, en cuanto al número de vehículos que se desplazan a lo largo de los años 2017, 2018, 2019 y parte de 2020, este último con datos de los primeros meses, que son los disponibles a la hora de realizar el presente trabajo. A partir de estos datos, que conforman las correspondientes series temporales¹⁶, se lleva a cabo la estimación de los parámetros del modelo y la posterior predicción bajo demanda a ThingSpeak. El modelo utilizado para la predicción es de regresión lineal de primer orden del tipo AR. El hecho de utilizar estos datos, se debe a la imposibilidad de utilizar datos reales a lo largo de distintos años con fines predictivos. De esta forma la idea consiste en determinar la validez de la propuesta de forma que en el futuro, y ante el posible despliegue de un sistema real, no haya más que sustituir estos datos por los obtenidos a partir de la aplicación y a lo largo del tiempo en varios años.

Se han realizado las pruebas necesarias, con resultados satisfactorios, para validar la aplicación integrada en su conjunto, así como de los diferentes módulos que conforman la misma a nivel individual.

Trabajo futuro

- Detectar aquellos vehículos que no siguen el sentido correcto de la marcha, es decir, los llamados “kamikazes”, de esta forma se podría enviar una notificación tanto a los conductores como a los equipos de emergencias, avisando de la

existencia de peligro en la vía con fines de actuación inmediata ante un escenario de semejante naturaleza.

- Enriquecer el entrenamiento de la Red Neuronal con imágenes de vehículos en condiciones climatológicas adversas, tales como: lluvia, nieve, etc.
- Obtener más videos para enriquecer el canal de ThingSpeak²⁴ encargado del conteo de los diferentes tipos de vehículos y poder hacer predicciones en base a los datos obtenidos de los análisis de los vehículos siendo mucho más amplios y precisos.
- Aplicar técnicas que permitan el procesamiento del flujo óptico de los *frames* de vídeo en tiempo real.
- Inclusión de un algoritmo que evite que las “sombras” de los vehículos no formen parte del movimiento asociado a los vehículos, sino como otros elementos que se mueven por sí mismos.
- Dar la posibilidad al usuario de usar su propia Red Neuronal Convolucional en lugar de trabajar siempre con AlexNet⁹, para ello se diseñarán otros modelos basados en GoogleNet, ResNet u otros ya pre-entrenados.
- Se dejará que sean los usuarios quienes positionen la franja de detección de vehículos, ya que así podrá ajustarla mejor al emplazamiento donde se ha realizado la grabación. También se le permitirá establecer de forma manual el tamaño de la franja.
- Estudiar nuevos modelos de predicción para comparar con el modelo AR e incluir la posibilidad de técnicas basadas en redes neuronales recurrentes.

La solución conceptual propuesta se ha centrado en el cómputo del flujo de varios tipos de vehículos. Esta misma estrategia se puede plantear en otros ámbitos tanto dentro del concepto de las ciudades inteligentes del futuro como en otras aplicaciones, sin más que realizar las correspondientes adaptaciones en los diferentes módulos y en su integración.

6. Conclusions and future Work

Conclusions

In the present work an application has been developed for the identification of different types of vehicles, from video images, driving in both directions of a road through the

use of several Computer Vision⁴, Deep Learning²⁷ and IoT technologies for analysis of movements and segmentation of the images, with the objective of providing the Alexnet Convolutional Neural Network with the necessary inputs to identify the type of vehicle.

Along with the mentioned techniques, a procedure has been developed to count the number of vehicles in circulation as precisely as possible to avoid duplication, so that the same vehicle is counted more than once. The global design of the implemented techniques makes it possible for the system to work correctly with a wide hit rate.

In the present work a solution has been proposed to achieve the identification of different types of vehicles. The different techniques for detecting and classifying the traffic of vehicles passing through a road have been explained in detail, as well as the structure that makes it possible for the system to work correctly with a wide hit rate.

The main objective has been the implementation of a software system in order to determine the number and type of vehicles that cross through a defined area on the road. Finally, predictions techniques have been also developed considering these data in the future.

The structure is made up of a camera that captures images with moving vehicles and acts as a data-generating sensor, sending the information to a server to analyze the type of vehicles present at each frame. Although, since the solution provided is not real, but a concept solution, the capture of images in real time has been replaced by prerecorded video sequences.

The strategy used for vehicle recognition is based in the application of Deep Learning²⁷ techniques, specially through the use of properly trained CNN, having chosen the pre-trained Alexnet model for a thousand objects, which is redesigned and re-trained to adapt it to the classification of the vehicles of the type provided in the application. In this way, what is known as learning transfer has been carried out, taking advantage of the aforementioned pre-trained model. The neural network receives images to carry out the classification of eight types of defined vehicles (FrontCar, BackCar, FrontTruckVan, BackTruckVan, FrontMotorcycle, BackMotorcycle, FrontBus, BackBus).

To obtain in the sequence of video images the areas where the movement has occurred and therefore the determination of the regions corresponding to the vehicles, it is used a combination of computational vision techniques based on the detection of optical flow and segmentation of regions that allows identifying the position of the vehicles at each of the images belonging to the received sequence based on the detected movement. Results of the analysis are stored on the public ThingSpeak²⁴

platform, using the concept of cloud location, making it available to personal or institutional users.

In addition to the data obtained concerning to the computation of vehicles using computer vision techniques and the use of Deep Learning²⁷, a prediction module has also been developed, based on the treatment of public data from the Madrid City Council, from years 2017, 2018, 2019 and part of 2020, the latter with data from the first months, which are those available at the time of carrying out this work. From these data, which make up the corresponding time series, the estimation of the model parameters and the subsequent prediction on demand to ThingSpeak²⁴ are carried out. The model used for prediction is a first-order linear AR regression type. The fact of using these data is due to the impossibility of using real data over different years for predictive purposes.

In this way, the idea is to determine the validity of the proposal so that in the future, for possible deployment of a real system, all that is necessary is to replace these data with those obtained from the application and over time in several years.

The required tests have been carried out, with satisfactory results, to validate the integrated application as a whole, as well as the different modules that make it up at the individual level.

Future work

- Detect those vehicles that do not follow the correct direction of travel, that is, so-called “kamikazes”. In this way a notification could be sent to both, drivers and emergency teams, warning them about the existence of danger situations on the road for immediate action in an adverse scenario.
- Improve the neural network training process with images of vehicles in adverse weather conditions, such as rain, snow, fog, etc.
- Get more videos to improve the ThingSpeak²⁴ where the different types of vehicles are stored and to be able to make predictions based on the data obtained from the number of vehicles detected, being much more extensive and accurate.
- Apply techniques to calculate the optical flow of video frames in real time.
- Inclusion of an algorithm to prevent shadows of vehicles which are part of the movement associated with vehicles, but a different element that moves by itself, in this way it will be avoided as part of vehicle identification.

- Provide the user with the possibility of using their own convolutional neural network instead of always using Alexnet⁹. Explore other pre-trained models such as GoogleNet, ResNet or Region-based approaches.
- Users will be left to position the vehicle detection strip, as this will allow them to better adjust it to the location where the recording was made. Users will also be allowed to manually set up the stripe size.
- Study new prediction models to compare with the AR model and including the possibility of techniques based on recurrent neural networks.

The proposed conceptual solution has been focused on the computation of the flow of various types of vehicles, this same strategy can be considered to detect other elements in movement (pedestrian, runners, walkers, bikes) in different scenarios, both within the concept of the smart cities of the future and in other applications, simply by making the corresponding adaptations in the different modules and in their integration.

7. Bibliografía

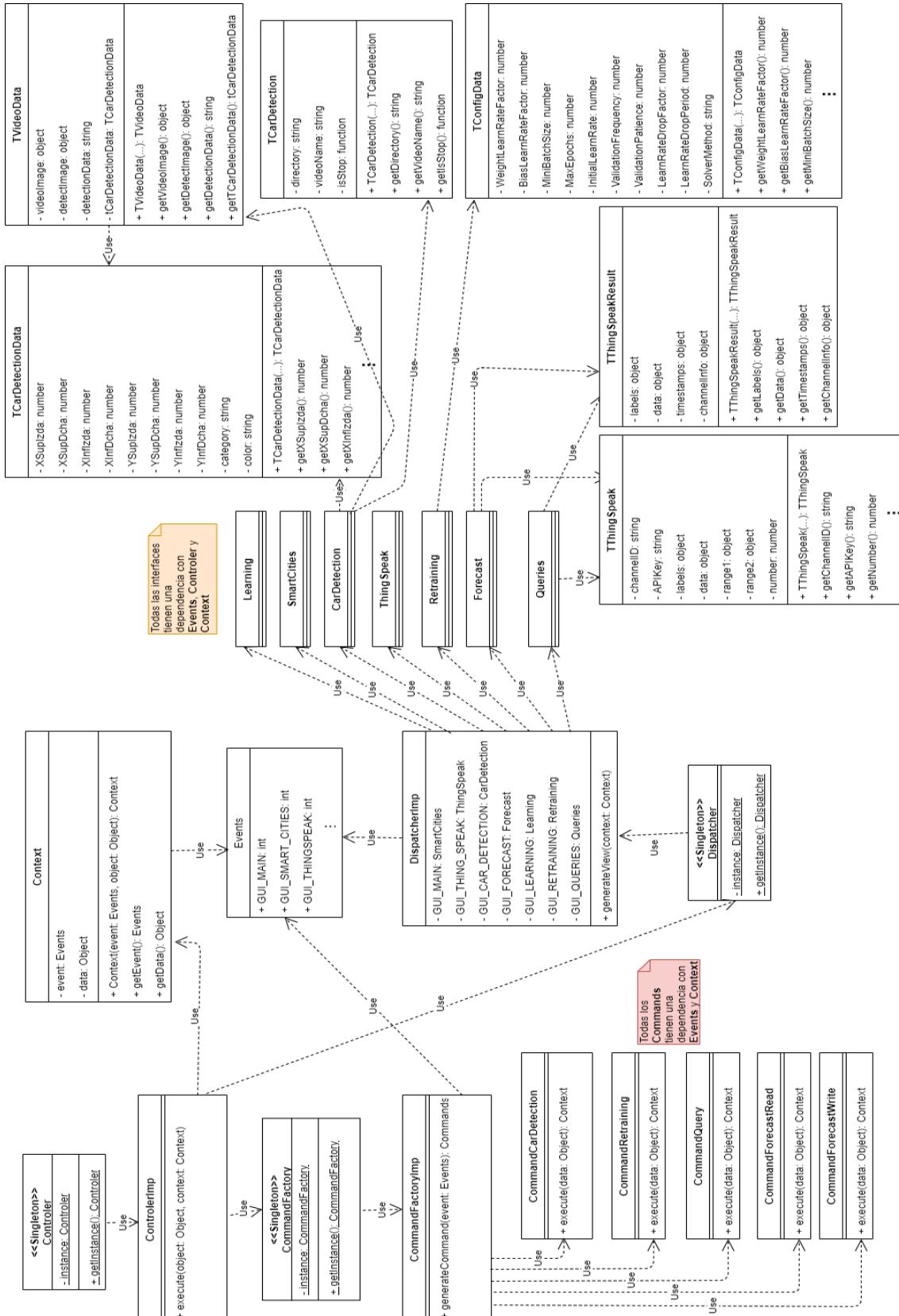
- [1] Smart Parking IoT platform to increase the efficiency of electric car recharging station: <http://www.libelium.com/smart-parking-iot-platform-to-increase-the-efficiency-of-electric-car-recharging-station> (accedido abril 2020).
- [2] Planificación Urbana mediante IoT y sensores de monitorización de tráfico de peatones y vehículos en la ciudad:
<https://www.esmartcity.es/2017/08/11/planificacion-urbana-mediante-iot-sensores-monitorizacion-trafico-peatones-vehiculos-ciudad> (accedido abril 2020).
- [3] Haralick, R.M., Shapiro, L.G. (1992) Computer and Robot Vision. Addison-Wesley, Boston.
- [4] Pajares, G., Cruz, J.M. (2007) Visión por Computador: Imágenes digitales y aplicaciones. RA-MA, Madrid.
- [5] Zhou, Y. and Chellappa, R. (1988). Computation of optical flow using a neural network. Proc. IEEE International Conference on Neural Networks, 1988, pages 71-78.
- [6] Srivastava, N., G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov (2014) Dropout: A Simple Way to Prevent Neural Networks from Overfitting. Journal of Machine Learning Research, 15, 1929-1958.
- [7] Brío, B.M., Sanz-Molina, A. (2006). Redes Neuronales y Sistemas Borrosos, RA-MA, Madrid.
- [8] Romero-Pérez, J. (2019). IoT para monitorización de tránsito peatonal mediante técnicas de Aprendizaje Profundo. TFM de la UCM (e-prints). Disponible en línea: <https://eprints.ucm.es/57468/> (accedido abril 2020)
- [9] Krizhevsky, Alex, Ilya Sutskever, and Geoffrey E. Hinton. "ImageNet Classification with Deep Convolutional Neural Networks." *Advances in neural information processing systems*. 2012.
- [10] Farneback, G. (2003) Two-Frame Motion Estimation Based on Polynomial Expansion. Disponible en línea:
https://www.researchgate.net/publication/225138825_Two-Frame_Motion_Estimation_Based_on_Polynomial_Expansion (accedido abril 2020).

- [11] Visualización de las activaciones neuronales en redes de tipo neuronal convolucional. Disponible online:
<https://es.mathworks.com/help/deeplearning/examples/visualize-activations-of-a-convolutional-neural-network.html> (accedido abril 2020).
- [12] Visualización de la matriz de confusión como ayuda a la interpretación de los resultados del entrenamiento de una red de tipo neuronal convolucional. Disponible online: <https://es.mathworks.com/help/stats/confusionchart.html> (accedido abril 2020).
- [13] Implementación de la técnica “transferencia de aprendizaje” en la Red Neuronal Convolutacional AlexNet. Disponible online:
<https://www.mathworks.com/help/deeplearning/ug/transfer-learning-using-alexnet.html> (accedido abril 2020).
- [14] Metodología de desarrollo ágil Scrumban. Disponible online:
<https://www.agilealliance.org/what-is-scrumban/> (accedido abril 2020).
- [15] DataSet del flujo de tráfico del Ayuntamiento de Madrid:
<https://datos.madrid.es/sites/v/index.jsp?vgnextoid=33cb30c367e78410VgnVCM10000000b205a0aRCRD&vgnextchannel=374512b9ace9f310VgnVCM10000171f5a0aRCRD> (accedido abril 2020).
- [16] Mauricio, J.A. (2007) Introducción al Análisis de Series Temporales. Disponible on-line: <https://www.ucm.es/data/cont/docs/518-2013-11-11-JAM-IAST-Libro.pdf/> (accedido abril 2020).
- [17] Walker, G. (1931). On Periodicity in Series of Related Terms, Proc. Of the Royal Society of London, Ser. A, Vol. 131, 518–532. Disponible on-line: <http://visualiseur.bnf.fr/Visualiseur?Destination=Gallica&O=NUMM-56224> (accedido abril 2020).
- [18] Hochreiter, S., Schmidhuber, J. (1997). Long short-term memory. Neural Computation, 9(8), 1735–1780.
- [19] Canal de ThingSpeak del parseo de los datos del DataSet del Ayuntamiento de Madrid: <https://thingspeak.com/channels/990311> .
- [20] Canal de ThingSpeak que guarda los resultados de la predicción: <https://thingspeak.com/channels/1005368>.
- [21] Canal de ThingSpeak que guarda los parámetros de entrada para realizar la predicción: <https://thingspeak.com/channels/1005862>.

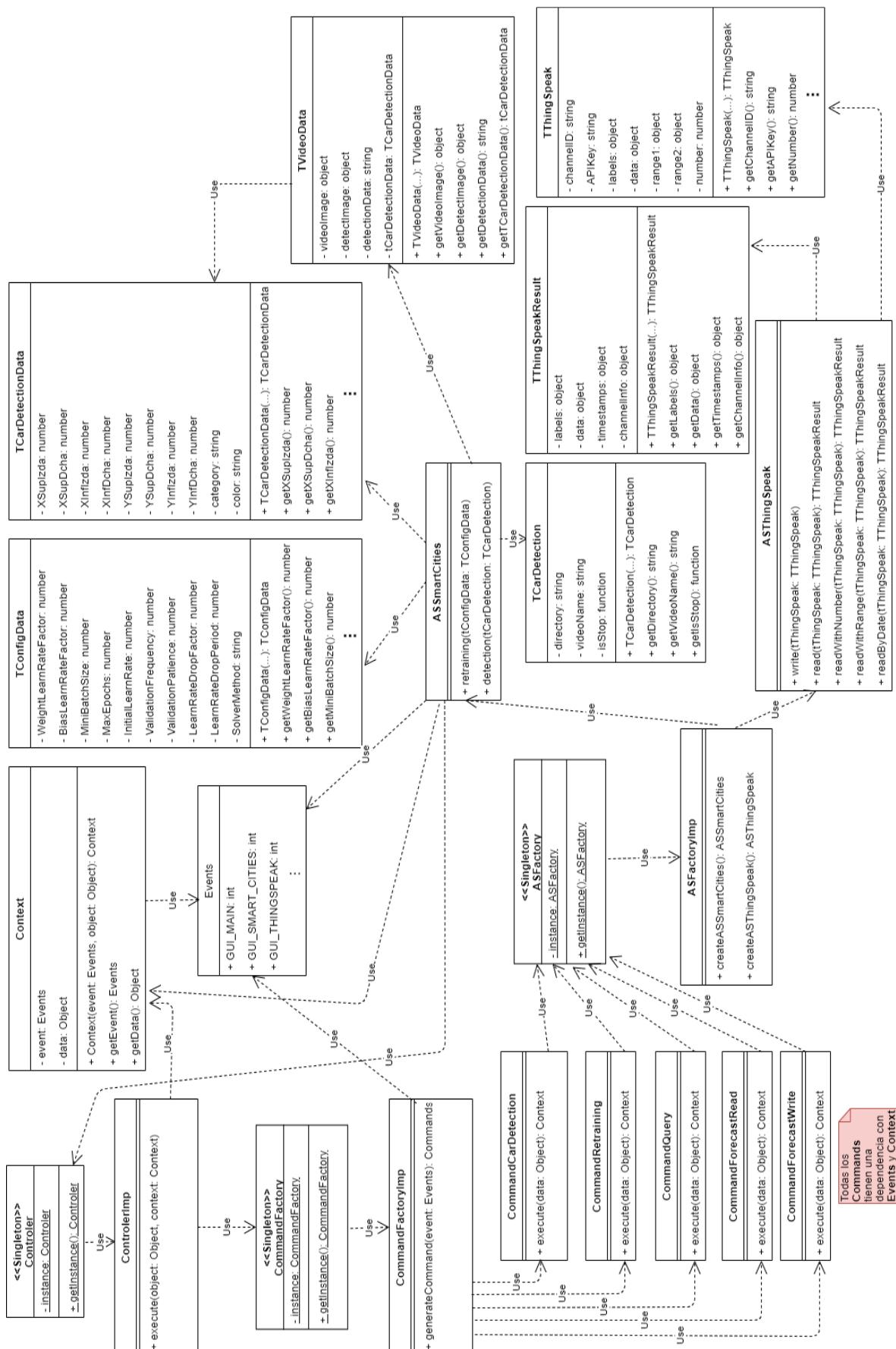
- [22] Multinacional tecnológica Libelium: <http://www.libelium.com/> (accedido mayo 2020).
- [23] Smart Parking project in Montpellier to relieve traffic congestion and reduce car parking search:
<http://www.libelium.com/smart-parking-project-in-montpellier-to-relieve-traffic-congestion-and-reduce-car-parking-search/> (accedido mayo 2020).
- [24] Plataforma IoT ThingSpeak: <https://thingspeak.com/> (accedido mayo 2020).
- [25] Allström A., Barceló J., Ekström J., Grumert E., Gundlegård D., Rydergren C. (2017) Traffic Management for Smart Cities. In: Angelakis V., Tragos E., Pöhls H., Kapovits A., Bassi A. (eds) Designing, Developing, and Facilitating Smart Cities. Springer, Cham.
- [26] Gaze (2020). Waze - GPS, Mapas, Alertas de Tráfico y Navegación. Disponible on-line:
<https://play.google.com/store/apps/details?id=com.waze&hl=es> (accedido abril, 2020).
- [27] Goodfellow, I., Bengio, Y., Courville, A. (2016). Deep Learning. MIT Press, Cambridge, MA, USA.
- [28] Alur, D., Crupi, J., Malks, D. (2003). Core J2EE Patterns: Best Practices and Design Strategies, Prentice-Hall, Upper Saddle River, NJ, USA.
- [29] Gamma, E., Vlissides, J., Helm, R., Johnson, R. (1994). Desing Patterns: Elements of Reusable Object-Oriented Software, Addison-Wesley Professional, Boston, MA, USA.
- [30] Fowler, M. (2002). Patterns of Enterprise Application Architecture, Addison-Wesley Professional, Boston, MA, USA.
- [31] Arlow, J., Neustadt, I. (2006). UML 2, Anaya Multimedia, Madrid.

8. Anexos

Anexo I : Diagrama de clases de la capa de presentación.



Anexo II : Diagrama de clase de la capa de negocio.



Anexo III: Resultados ThingSpeak Prediction.

En la tabla siguiente se muestran los datos de flujo de tráfico, según la cantidad de vehículos detectados y suministrados por el Ayuntamiento de Madrid en la vía de circunvalación M30 para los días de la semana y fecha que se indican, que por otra parte son los utilizados para estimar los parámetros del modelo de predicción AR.

Lunes

Fecha	Cantidad
02/03/2020	22.867
09/03/2020	21.280
16/03/2020	22.141
23/03/2020	21.584
30/03/2020	21.626

Martes

Fecha	Cantidad
03/03/2020	22.766
10/03/2020	22.769
17/03/2020	22.769
24/03/2020	22.567
31/03/2020	22.629
07/04/2020	22.489
14/04/2020	22.411
21/04/2020	22.333
28/04/2020	22.333
05/05/2020	22.185

Miércoles

Fecha	Cantidad
04/03/2020	24.121
11/03/2020	23.436
18/03/2020	22.975
25/03/2020	23.121
01/04/2020	23.225
08/04/2020	23.116
15/04/2020	22.935
22/04/2020	22.799
29/04/2020	22.720
06/05/2020	22.643
13/05/2020	22.546
20/05/2020	22.440
27/05/2020	22.338
03/06/2020	22.243
10/06/2020	22.148

Jueves

Fecha	Cantidad
05/03/2020	23.939
12/03/2020	23.523
19/03/2020	23.781
26/03/2020	23.926
02/04/2020	23.653
09/04/2020	23.556
16/04/2020	23.371
23/04/2020	23.314
30/04/2020	23.246

Viernes

Fecha	Cantidad
06/03/2020	22.708
13/03/2020	22.726
20/03/2020	22.097
27/03/2020	21.939
03/04/2020	22.038
10/04/2020	21.967
17/04/2020	21.750

Sábado

Fecha	Cantidad
07/03/2020	18.507
14/03/2020	18.632
21/03/2020	17.732
28/03/2020	17.811
04/04/2020	17.946
11/04/2020	17.817
18/04/2020	17.650
25/04/2020	17.606
02/05/2020	17.553
09/05/2020	17.461
16/05/2020	17.375

Domingo

Fecha	Cantidad
01/03/2020	18.664
08/03/2020	18.115
15/03/2020	18.208
22/03/2020	17.939
29/03/2020	17.764
05/04/2020	17.851
12/04/2020	17.796
19/04/2020	17.706

Anexo IV: Resultados ThingSpeak CarDetection.

En la siguiente tabla se puede observar el número de los diferentes tipos de vehículos detectados en los análisis de video y que fueron subidos a ThingSpeak, éstos están ordenados por fecha y hora.

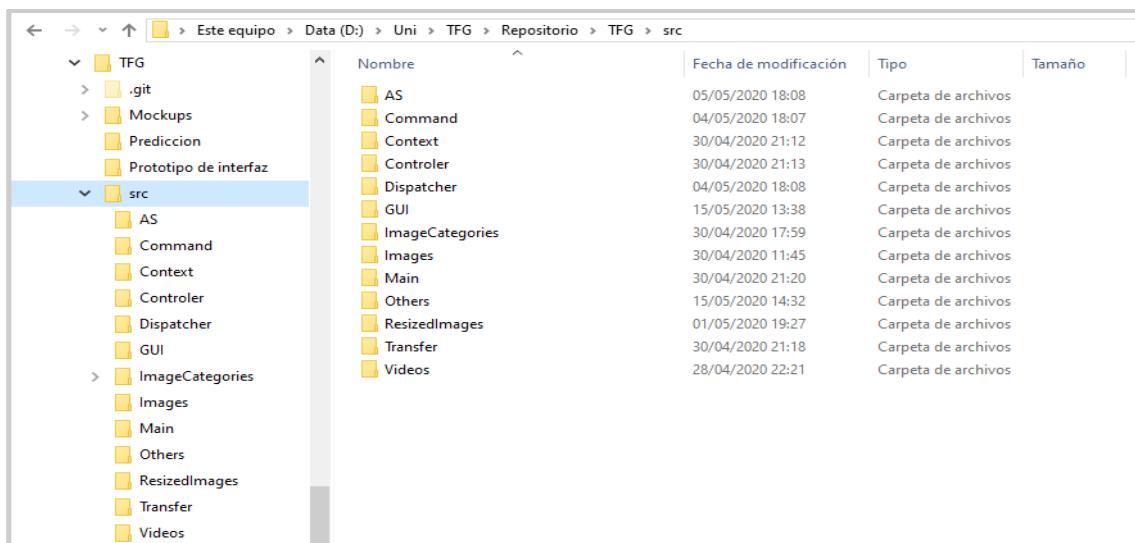
FECHA	HORA	ID_CONSULTA	FRONT CAR	BACK CAR	FRONT TRUCK VAN	BACK TRUCK VAN	FRONT MOTORCYCLE	BACK MOTORCYCLE	FRONT BUS	BACK BUS
07/04/2020	21:52:22	1	27	24	2	0	0	0	0	0
08/04/2020	9:28:16	2	2	5	1	2	0	0	0	0
08/04/2020	10:05:18	3	12	20	0	4	1	0	0	2
08/04/2020	11:00:23	4	14	19	1	0	2	0	0	0
08/04/2020	13:53:32	5	18	62	25	8	1	0	0	0
08/04/2020	17:52:58	6	23	10	10	22	10	3	0	0
22/04/2020	21:27:31	7	35	13	4	1	10	11	0	0
25/04/2020	14:27:13	8	33	20	3	0	0	4	0	0
25/04/2020	15:39:36	9	15	17	0	1	0	2	2	1
25/04/2020	16:47:00	10	14	19	5	2	2	5	0	0
25/04/2020	18:30:09	11	42	51	19	6	2	1	0	0
26/04/2020	23:51:08	12	33	21	2	1	0	0	0	0
27/04/2020	1:34:33	13	14	18	0	1	0	0	2	0
27/04/2020	4:26:38	14	43	42	18	11	1	0	0	0
27/04/2020	16:11:17	15	43	42	18	11	1	0	0	0
02/05/2020	15:42:19	16	6	8	10	2	0	1	0	0
02/05/2020	15:54:56	17	6	8	10	2	0	1	0	0
04/05/2020	17:32:09	18	6	8	10	2	0	1	0	0
04/05/2020	21:36:45	19	1	4	20	0	0	1	0	0
04/05/2020	21:57:04	20	1	4	20	0	0	1	0	0
05/05/2020	16:54:59	21	7	4	9	1	0	0	0	0
06/05/2020	17:03:52	22	0	2	0	1	0	1	0	0

Anexo V: Manual de usuario.

1. Descarguese el código del repositorio: <https://github.com/GDelga/TFG>

2. Para poder ejecutar la aplicación se necesita tener instalado el IDE MATLAB R2019a con los siguientes add-ons (toolboxes):
 - a. Image Processing Toolbox
 - b. Deep Learning Toolbox
 - c. Computer Vision Toolbox
 - d. Deep Learning Toolbox Model for AlexNet Network
 - e. Parallel Computing Toolbox

3. Asegúrese de que la estructura de carpetas es como la mostrada a continuación. Obligatoriamente deben existir las siguientes carpetas.
 - src:
 - ImageCategories: debe contener todos los directorios e imágenes para realizar el entrenamiento.
 - Images: contiene las imágenes usadas en las interfaces de la aplicación.
 - Others: directorio en el que se guarda y lee la Red Neuronal.
 - ResizedImages: carpeta en la que se guardan las imágenes redimensionadas.



The screenshot shows a Windows File Explorer window with the following details:

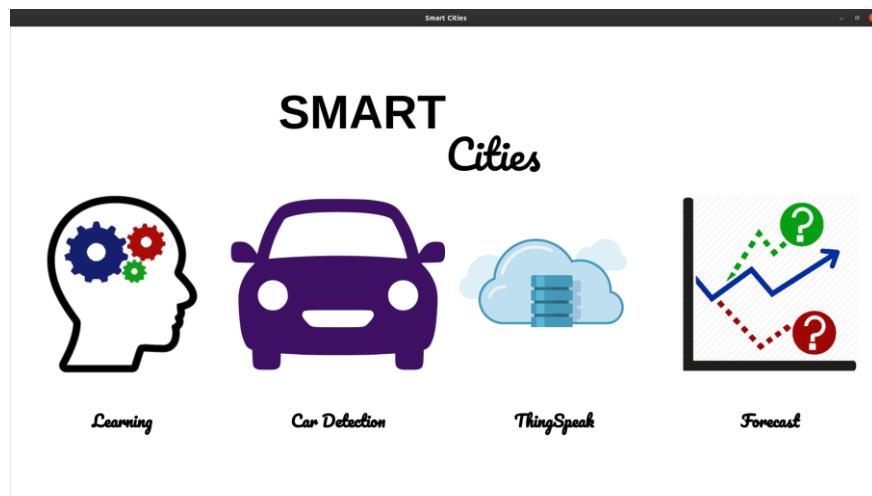
Left pane (File tree):

- Este equipo > Data (D:) > Uni > TFG > Repositorio > TFG > src
- Contains: TFG, .git, Mockups, Predicción, Prototipo de interfaz, src (selected), AS, Command, Context, Controller, Dispatcher, GUI, ImageCategories, Images, Main, Others, ResizedImages, Transfer, Videos.

Right pane (File list):

Nombre	Fecha de modificación	Tipo
AS	05/05/2020 18:08	Carpeta de archivos
Command	04/05/2020 18:07	Carpeta de archivos
Context	30/04/2020 21:12	Carpeta de archivos
Controller	30/04/2020 21:13	Carpeta de archivos
Dispatcher	04/05/2020 18:08	Carpeta de archivos
GUI	15/05/2020 13:38	Carpeta de archivos
ImageCategories	30/04/2020 17:59	Carpeta de archivos
Images	30/04/2020 11:45	Carpeta de archivos
Main	30/04/2020 21:20	Carpeta de archivos
Others	15/05/2020 14:32	Carpeta de archivos
ResizedImages	01/05/2020 19:27	Carpeta de archivos
Transfer	30/04/2020 21:18	Carpeta de archivos
Videos	28/04/2020 22:21	Carpeta de archivos

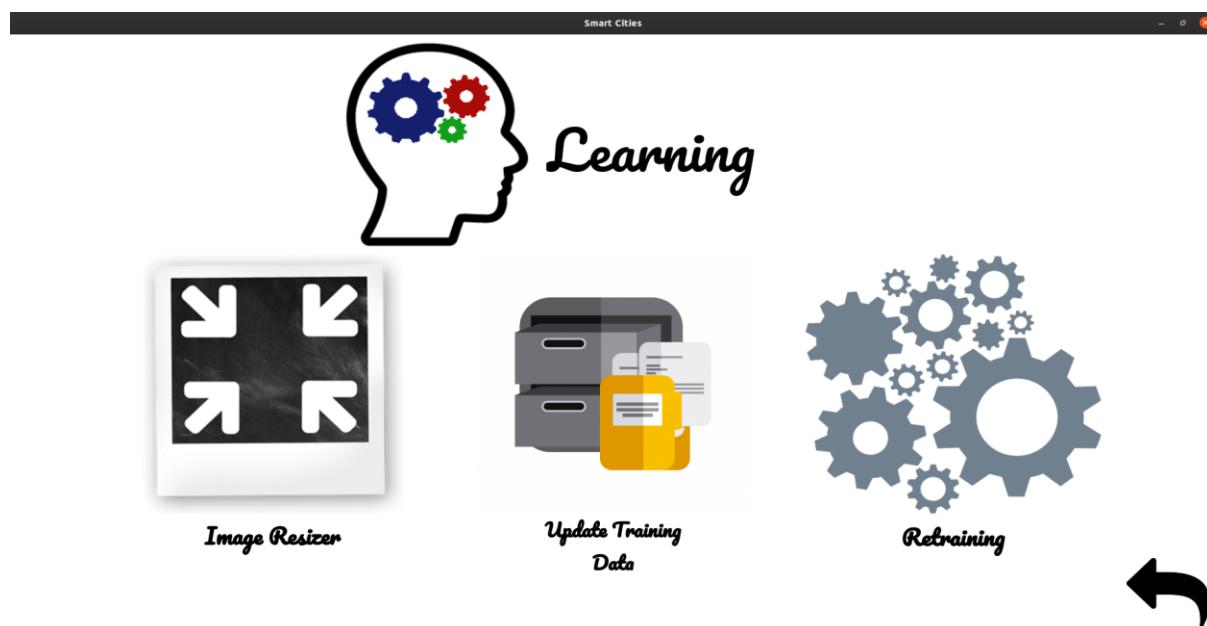
4. Al iniciar la aplicación nos encontraremos con la siguiente interfaz.



5. Para entrenar la red se selecciona la opción “Learning”.



Se mostrará el siguiente interfaz de usuario:



6. Seleccionaremos la opción “Retraining”.



A continuación se selecciona la pestaña “Training Options” que muestra la siguiente vista donde son cargados los valores por defecto para entrenar la red. Al terminar de ejecutarse esta vista se genera el archivo AlexNet.mat que contiene el modelo de la Red Neuronal Convolutacional AlexNet ya entrenada y por tanto con los pesos ajustados según las imágenes con las que ha sido entrenada la red.

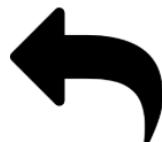
A screenshot of a software window titled "Smart Cities". The window has tabs at the top: "Last Fully Connected Layer" and "Training Options", with "Training Options" being the active tab. The main area contains several configuration options with their current values:

- Solver Method: sgdm
- Mini Batch Size: 80
- Max Epochs: 120
- Initial Learn Rate: 1e-05
- Learn Rate Drop Factor: 0.01
- Learn Rate Drop Period: 70
- Validation Frequency: 3
- Validation Patience: Inf

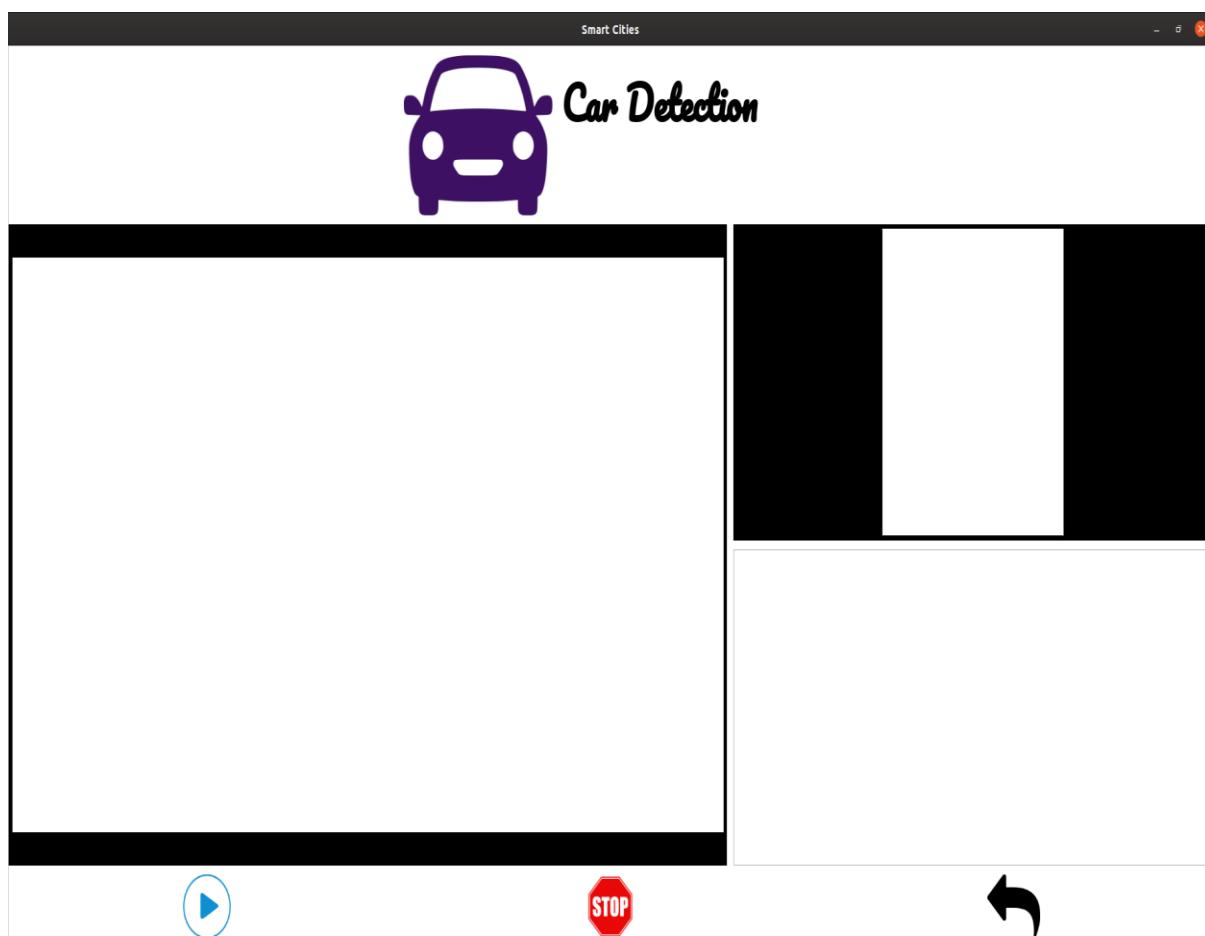
Below these settings is a toolbar with two icons: a gear and a document with three colored circles (blue, green, red) on it, followed by a large black curved arrow pointing left.

AVISO: Las funcionalidades “Image Resizer” y “Update Training” Data se explican en el apartado 4.1 de la memoria.

7. Tras el entrenamiento de la red se puede utilizar la funcionalidad de detección de vehículos. Para ello se vuelve al menú principal pulsando la flecha de retroceso. Y a continuación se selecciona la opción “Car Detection”.



Se habrá de mostrar el siguiente interfaz de usuario:

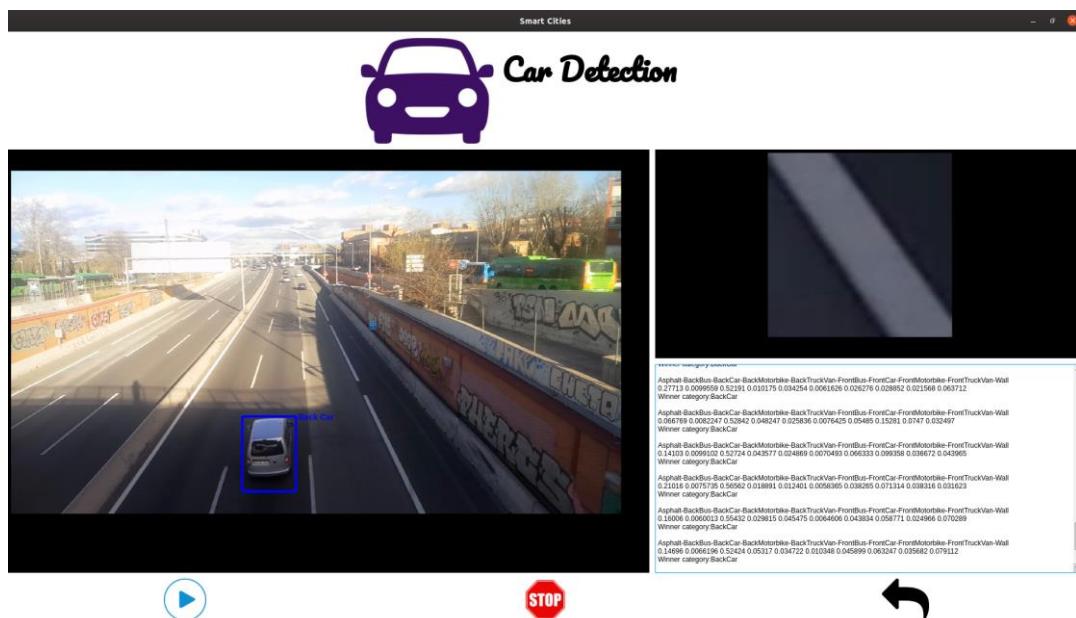


1. Al seleccionar la opción “Play” se abrirá un explorador de ficheros para seleccionar el video que se desea procesar.

2. Tanto el comienzo como la finalización del procesamiento del video se avisa con un mensaje informativo.
3. La opción “Stop” sirve para detener el procesamiento del video, saltando un mensaje informativo cuando esta es seleccionada.



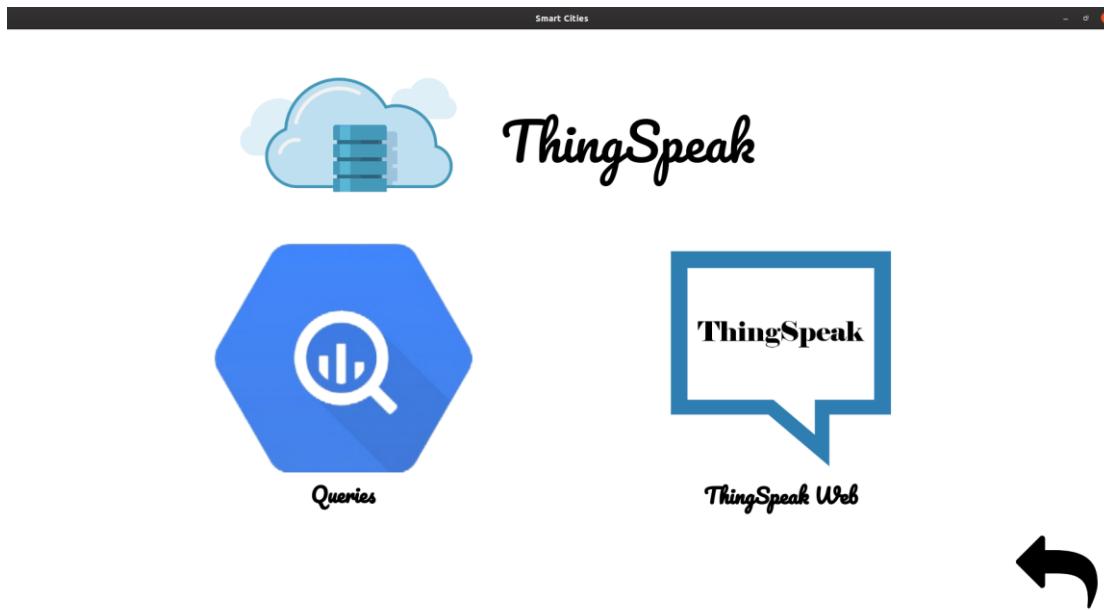
A continuación se muestra el procesamiento del vídeo presentándose en una interfaz como la mostrada a continuación que se actualiza tras cada imagen procesada.



8. Volviendo a pulsar al botón retroceso se volverá al menú principal, donde se selecciona la opción “ThingSpeak” para la consulta de los datos obtenidos en el apartado anterior.



Tras lo cual se visualiza el siguiente interfaz de usuario:



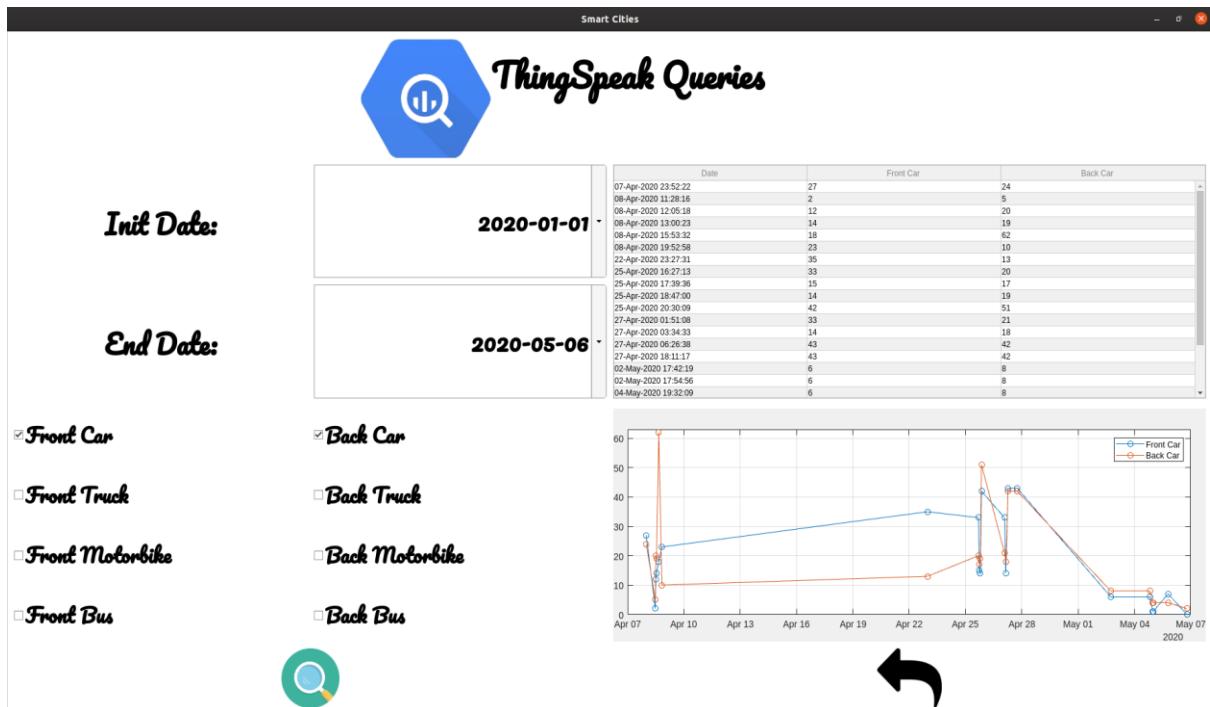
Si ahora se selecciona la opción Queries:



Se mostrará el siguiente interfaz de usuario:

The screenshot shows the 'ThingSpeak Queries' interface. It features a blue hexagonal icon with a magnifying glass over a bar chart. Below it are two input fields for 'Init Date' and 'End Date', both set to 'yyyy-mm-dd'. To the right is a table with columns: Date, Front Car, Back Car, Front Truck, Back Truck, Front Motorbike, Back Motorbike, Front Bus, and Back Bus. At the bottom left are checkboxes for vehicle types: Front Car, Front Truck, Front Motorbike, and Front Bus. To the right is a list of checkboxes for Back variants: Back Car, Back Truck, Back Motorbike, and Back Bus. A line graph is displayed at the bottom right.

En esta interfaz se habrá de seleccionar la fecha de inicio y fecha de fin, así como cualquier tipo de categoría de vehículo con la que trabaja el identificador. Una vez se hayan seleccionado las opciones deseadas, se pulsará el botón de exploración para ejecutar la query, mostrándose un resultado similar al siguiente:

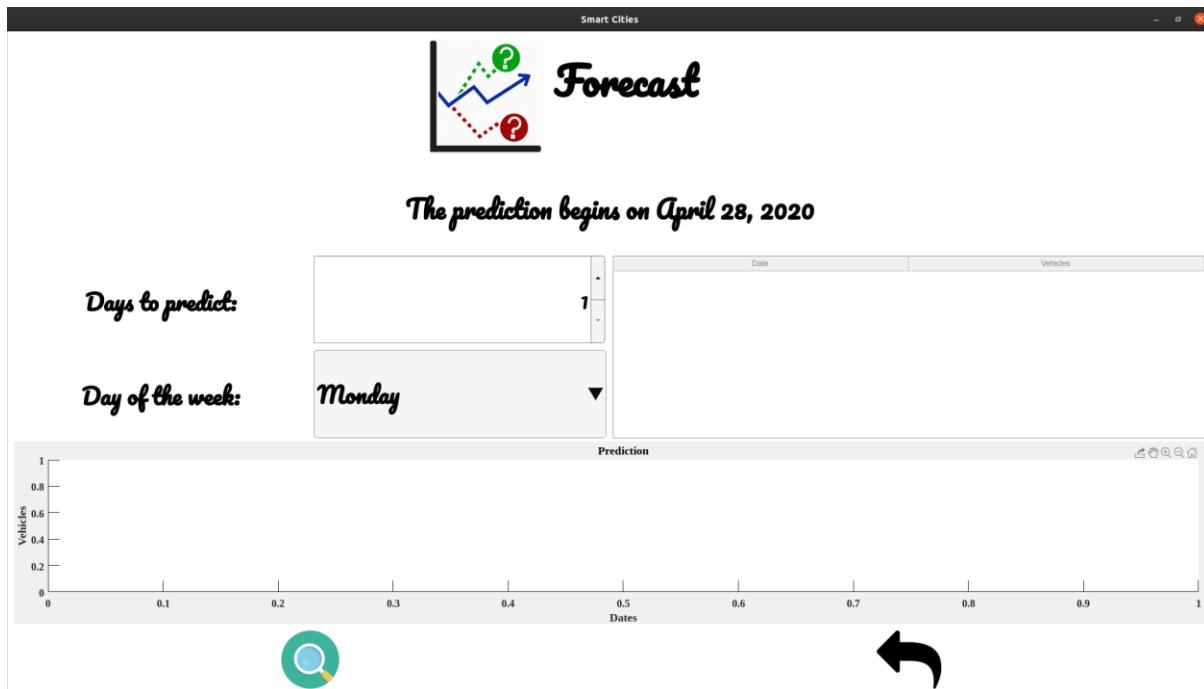


AVISO: La opción ThingSpeak Web se encuentra explicada en el punto 4.1 de la memoria.

9. Volviendo al menú principal mediante la pulsación del botón de retroceso, se seleccionará la opción “Forecast” para acceder a la sección en la que se realizan las predicciones del flujo vehículos.



Se mostrará el siguiente interfaz de usuario:



En el campo “Day of the week” se seleccionará el nombre del día de la semana, mientras que en el campo “Days to predict” se selecciona el número de días de la semana sobre los que se quiere realizar la predicción. En el apartado 4.1 de la memoria se encuentra un ejemplo de esta acción.