GESTIÓN INTELIGENTE DE PLAZAS DE APARCAMIENTO MEDIANTE PROCESAMIENTO DE IMÁGENES EN IoT: Redes neuronales convolucionales y predicción

INTELIGENT MANAGEMENT OF PARKING SPACES BASED ON IMAGE PROCESSING IN IoT: Convolutional Neural Networks and prediction

LUCAS SEGARRA FERNÁNDEZ

MÁSTER EN INTERNET DE LAS COSAS. FACULTAD DE INFORMÁTICA
UNIVERSIDAD COMPLUTENSE DE MADRID

Trabajo Fin Máster en Internet de las Cosas

Curso académico: 2018-2019
Presentado el 05/07/2019. Calificación obtenida: 9

Director:

Gonzalo Pajares Martinsanz

Autorización de difusión

Lucas Segarra Fernández

17 de Junio de 2019

El/la abajo firmante, matriculado/a en el Máster en Investigación en Informática de la Facultad de Informática, autoriza a la Universidad Complutense de Madrid (UCM) a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a su autor el presente Trabajo Fin de Máster: "GESTIÓN DE PLAZAS DE APARCAMIENTO MEDIANTE PROCESAMIENTO DE IMÁGENES EN IoT: Redes neuronales convolucionales y predicción ", realizado durante el curso académico 2018-2019 bajo la dirección de Gonzalo Pajares Martinsanz en el Departamento de Ingeniería del Software e Inteligencia Artificial, y a la Biblioteca de la UCM a depositarlo en el Archivo Institucional E-Prints Complutense con el objeto de incrementar la difusión, uso e impacto del trabajo en Internet y garantizar su preservación y acceso a largo plazo.

Resumen en castellano

En el presente trabajo se realiza un estudio, diseño y desarrollo de un sistema autónomo de reconocimiento del grado de ocupación de aparcamientos utilizando imágenes de los mismos tomadas en distintos instantes de tiempo. Se trata de determinar sobre cada una de las capturas el número de plazas libres sobre el total de plazas disponibles. Se proporciona una solución conceptual en el ámbito del Internet de las Cosas (IoT, *Internet of Things*) de suerte que se procesa la imagen convenientemente, procediendo a continuación a la publicación de los resultados, quedando disponibles en un repositorio específico y accesible, que en el contexto del IoT se corresponde con el concepto de nube (*cloud*). En la propia nube se realiza el procesamiento de datos con el fin de realizar previsiones a futuro, de suerte que tanto éstos como el modelo predictivo quedan disponibles para su consulta remota en la nube en cualquier momento. De esta forma se tiene información precisa e instantánea sobre la situación de ocupación de un determinado parking, lo que contribuye a la gestión eficiente de las zonas de aparcamiento y por tanto a la regulación inteligente del tráfico.

El esquema de diseño consta de dos módulos, que constituyen el núcleo de la aplicación, a saber: local y remoto. En el módulo local se llevan a cabo dos procesos, uno relativo al re-entrenamiento de la red neuronal convolucional de tipo AlexNet, la cual está convenientemente pre-entrenada. Los parámetros y el modelo resultantes del re-entrenamiento se almacenan convenientemente para un posterior reconocimiento. En este caso se determina el número de clases a identificar según el contenido de la plaza, que en esencia son plaza ocupada o libre. Un segundo proceso que incluye la segmentación de la imagen original, para realizar recortes sobre la misma, que en teoría se corresponden con el contenido de las plazas de parking. Durante el proceso de segmentación se realiza la captura de la imagen y su procesamiento, para identificar, mediante técnicas de visión por computador, las regiones de ésta que se corresponden con las diferentes plazas de parking. Para ello se aplican técnicas de binarización, seguidas de las operaciones morfológicas pertinentes para determinar posibles áreas candidatas como plazas de parking. A continuación se realiza un etiquetado de componentes conexas, que asigna a cada una de las regiones candidatas, una única etiqueta identificativa. Gracias al etiquetado, se obtienen como propiedades el área y la envoltura, que la delimita (*bounding-box*). Para cada región y su correspondiente *bounding-box* se realiza el recorte correspondiente sobre la imagen original, que previamente redimensionada se proporciona a la red neuronal para determinar el contenido de la misma, según las clases asignadas durante el re-entrenamiento. Una vez procesada la imagen original al completo, se determina el número de plazas disponibles junto con la capacidad del parking.

Los datos así obtenidos se publican convenientemente en una plataforma remota, constituyendo así el módulo del mismo nombre. Los datos se almacenan asociados con los tiempos de captura, permitiendo realizar un estudio predictivo del grado de ocupación mediante análisis de series temporales. Estos datos quedan publicados permitiendo su monitorización y consulta en cualquier momento. A nivel local, también se realiza un análisis basado en Redes Neuronales Recurrentes (*Recurrent Neural Networks*) mediante técnicas basadas en

el concepto *Long Short-Term Memory* (LSTM).

Palabras clave

Redes Neuronales Convolucionales - Modelos de predicción autoregresivos - Memoria a corto y largo plazo ($LSTM$) - Ciudades Inteligentes - Reconocimiento de Imágenes - Gestión del Tráfico - Monitorización de Plazas de Aparcamiento

Abstract

This work consists in studying, designing and developing an autonomous system of recognition of the degree of occupancy of car parks. Images taken at different moments have been used. The goal is to count the amount of free places over the total of available places. A conceptual solution within the scope of the IoT (Internet of Things) is provided.

First, each image is conveniently processed. Resulting data is published, remaining available at a specific repository. Such repository represents the concept of cloud in the IoT context. Data processing, for the purpose of forecasting occupation also runs in the cloud. Predicted occupation data and predictive model can be consulted in the cloud at any time, providing accurate and real time information about parking lots' occupation. This contributes to smart management of parking places and due to that to smart management of traffic.

Design Schema consists of two modules: local and remote, which constitute the core of the application. Local module runs two different processes: First one is related with re-training the Convolutional Neuronal Network (CNN) of type Alexnet, which has been conveniently pre-trained. The parameters and the model, resulting from re-training process, are stored for later recognition. In this case the number of classes to be identified according with the content of the place is determined (busy or free). Second process includes segmenting original image, in order to trim it according to the parking places. To achieve it, binarization techniques are applied with morphologic operations to identify candidates to parking places. Later, related components are labeled. Labeling is used to extract features such as area and delimiting wrap. For each area and it's bounding-box, the original image is trimmed. Cutouts are considered as the expected parking areas at original image, and are resized and classified by the CNN in accordance with the classes assigned during re-training. Once the whole image is processed, available places and parking's capacity are counted.

Resulting data is published on a remote platform, which constitutes so named module. Data is stored associated with picture's time, and used for forecasting occupation average by analysis of time series. Results are published for it's inquiry and monitoring at any time. Recurrent Neuronal Network (RNN) based analysis is also performed locally using techniques based on Long Short-Term Memory (LSTM).

Key Words

Convolutional Neural Networks (*CNN*)- Autoregressive Prediction Models - Short and Long Term Memory (*LSTM*) - Smart Cities - Image Recognition - Traffic Management - Monitoring of Parking Spaces

Índice general

Capítulo 1

Introducción

1.1. Planteamiento General

El advenimiento y desarrollo de dispositivos de bajo rendimiento, coste y consumo, unido al avance de tecnologías para facilitar a tales dispositivos comunicarse entre ellos o con sistemas remotos de forma eficiente y fiable, han impulsado la aparición de un modelo de soluciones basado en procesar información obtenida localmente mediante diversas tecnologías sensoriales (p.e. cámaras como las que sirven de fuente de información en el presente trabajo), en sistemas inteligentes, que generalmente gestionan datos de varios dispositivos de forma centralizada, pudiendo combinarlos entre sí o con otros procedentes de distintas fuentes, y que son los responsables de publicar todos los datos recopilados para su acceso general.

Se conoce como Internet de las Cosas (*IoT, Internet of Things*) a este tipo de soluciones.

Pertenecientes a esta categoría, se consideran relativas a las *Smart Cities* (ciudades inteligentes) las soluciones diseñadas para desplegarse en un contexto urbano, añadiendo o mejorando los servicios ofrecidos por las administraciones a los ciudadanos, facilitando su día a día y mejorando su calidad de vida.

De aquí en adelante se utilizan indistintamente los términos ciudades inteligentes y *Smart Cities*, en este último caso por la amplia difusión del término a nivel internacional.

Dentro del contexto de las *Smart Cities*, se conocen como *Smart Parkings* a los sistemas basados en sensorizar aparcamientos de vehículos para conocer y predecir el nivel de ocupación, entre otras cuestiones.

La solución de concepto que aquí se plantea consiste en procesar imágenes de parkings obtenidas con cámaras operando en el espectro visible, para detectar el porcentaje de plazas libres y ocupadas, enviar avisos a los servicios de control de tráfico, o a los conductores a nivel individual, cuando se cumplan ciertas condiciones (p.e. cierto grado de ocupación), realizar análisis predictivos de futuros niveles de ocupación, y hacer públicos los datos relativos al grado de ocupación actual, así como las predicciones generadas, para informar a los usuarios de los mismos sobre su disponibilidad.

El IoT en Smart Cities

En esencia el concepto de *Smart City* pivota sobre tres elementos. Obtener información de la ciudad, procesar esta información y actuar.[1]

El *IoT* es una de las piezas claves sobre las que funcionan las *Smart Cities*. Partiendo de la sensorización de la ciudad y de los medios disponibles en el municipio es posible conocer en tiempo real lo que sucede y reaccionar en consecuencia. Por ejemplo se puede monitorizar el tráfico de la ciudad y en caso de congestión regular a distancia los semáforos.

Para profundizar en el concepto *Smart City* y poder explicar alguno de sus objetivos hay que revisar los servicios que por ley han de prestar los municipios. La Ley 7/1985, de 2 de abril, Reguladora de las Bases del Régimen Local, establece los servicios que deben prestar los municipios.[2]

- *Seguridad en lugares públicos*
- *Ordenación del tráfico y peatones*
- *Protección civil, prevención y extinción de incendios*
- *Ordenación, gestión, ejecución y disciplina urbanística; promoción y gestión de viviendas; parques y jardines, pavimentación de vías públicas urbanas y conservación de caminos y vías rurales*
- *Patrimonio histórico-artístico*
- *Protección del medio ambiente*
- *Abastos, mataderos, ferias, mercados y defensa de usuarios y consumidores*
- *Protección de la salubridad pública*
- *Participación en la gestión de la atención primaria de la salud*
- *Cementerios y servicios funerarios*
- *Prestación de los servicios sociales y de promoción y reinserción social*

- *Suministro de agua y alumbrado público; servicios de limpieza viaria, de recogida y tratamiento de residuos, alcantarillado y tratamiento de aguas residuales*
- *Transporte público de viajeros*
- *Actividades o instalaciones culturales y deportivas; ocupación del tiempo libre; turismo*
- *Participar en la programación de la enseñanza y cooperar con la Administración educativa en la creación, construcción y sostenimiento de los centros docentes públicos, intervenir en sus órganos de gestión y participar en la vigilancia del cumplimiento de la escolaridad obligatoria*
- *Promoción de la igualdad entre hombres y mujeres*

Ofrecer un mejor servicio

El objetivo principal de los proyectos de *Smat Cities* es ofrecer un mejor servicio a los ciudadanos.[3]

Algunos ejemplos son los siguientes:

1. **Protección del medio ambiente**: A través del *IoT* se pueden conocer para diferentes puntos de la ciudad el nivel de contaminación existente, ya sea atmosférica o acústica. A partir de la información proporcionada por los sensores el municipio puede restringir el tráfico, establecer límites de velocidad, limitar los horarios para realizar obras.

2. **Gestión de parques**: Mediante los sensores adecuados se puede medir el nivel de humedad del suelo de los parques, la temperatura existente y activar los sistemas de riego.[4]

3. **Gestión de aparcamientos**: Tal y como se describe en este trabajo, la monitorización de las plazas de aparcamiento permite predecir el nivel de ocupación, proporcionando la información necesaria a los usuarios para conocer dónde encontrar plazas libres.

4. **Gestión de activos**: Un municipio realiza su actividad utilizando activos como vehículos, equipos informáticos, oficinas, instalaciones deportivas, contenedores, equipos de alumbrado y semáforos. La sensoración permite tenerlos localizados y conocer su estado de funcionamiento. Así por ejemplo se pueden conocer las condiciones de funcionamiento de un vehículo y prevenir problemas. También se pueden establecer políticas de mantenimiento preventivo a partir de la información histórica almacenada.[5]

Control de las concesiones de servicios públicos

Muchos de estos servicios son prestados por medios ajenos utilizando concesionarios de servicios. Así por ejemplo, recogida de basuras, riego y limpieza de parques públicos, tratamientos de aguas residuales. Otros servicios han de ser desempeñados con medios propios como es el caso de la seguridad en lugares públicos.

Para los prestados en régimen de concesión, los proyectos de *Smart Cities* permiten sensorizar los medios a través de los cuales las empresas concesionarias prestan los servicios para comprobar que cumplen con las especificaciones del contrato. Así por ejemplo los vehículos para riego de vías públicas al estar sensorizados permiten su georreferenciación y conocer dónde y durante cuanto tiempo han prestado el servicio de riego. Lo mismo sucede con los servicios de recogida de basuras.

1.2. Soluciones Alternativas

Se han propuesto soluciones para abordar la misma problemática bajo distintas perspectivas, entre las que se encuentran los llevados a cabo en tres ciudades pioneras en proyectos *IoT* como son Santander y Málaga en España y Montpellier en Francia.

En tales casos se puede considerar, que la solución desplegada forma parte de otra más amplia, que consiste en que potencialmente la ciudad asuma el concepto de una *Smart City* que cuenta con un ecosistema de servicios.

A continuación se proporcionan los detalles relevantes relacionados con tales proyectos.

Montpellier

Montpellier es una ciudad situada en el sur de Francia de unos 277.000 habitantes (según datos de 2015).

Con el apoyo de la compañía francesa Synox(2019)[6], la estructura intermunicipal Montpellier Mediterranean Metropolis ha desplegado una solución utilizando como nodos dispositivos de Libelium(2019) conocidos como Waspmote (2019). Aprovechando el establecimiento de una red privada metropolitana basada en *LoRaWAN*, para dar cobertura a los proyectos *IoT* de la ciudad, se ha instalado en la superficie de la calzada, una red de sensores que detectan si un vehículo se posiciona sobre ellos, y lo comunican a través de la red privada junto con otra información de interés general como la temperatura.

Dicha información puede ser accedida desde canales públicos mediante una aplicación móvil y página web. La figura 1.1 muestra el panel detallado del GUI relativo a la aplicación en Montpelier, donde pueden apreciarse diferentes visualizaciones del estado de los aparcamientos de la ciudad.

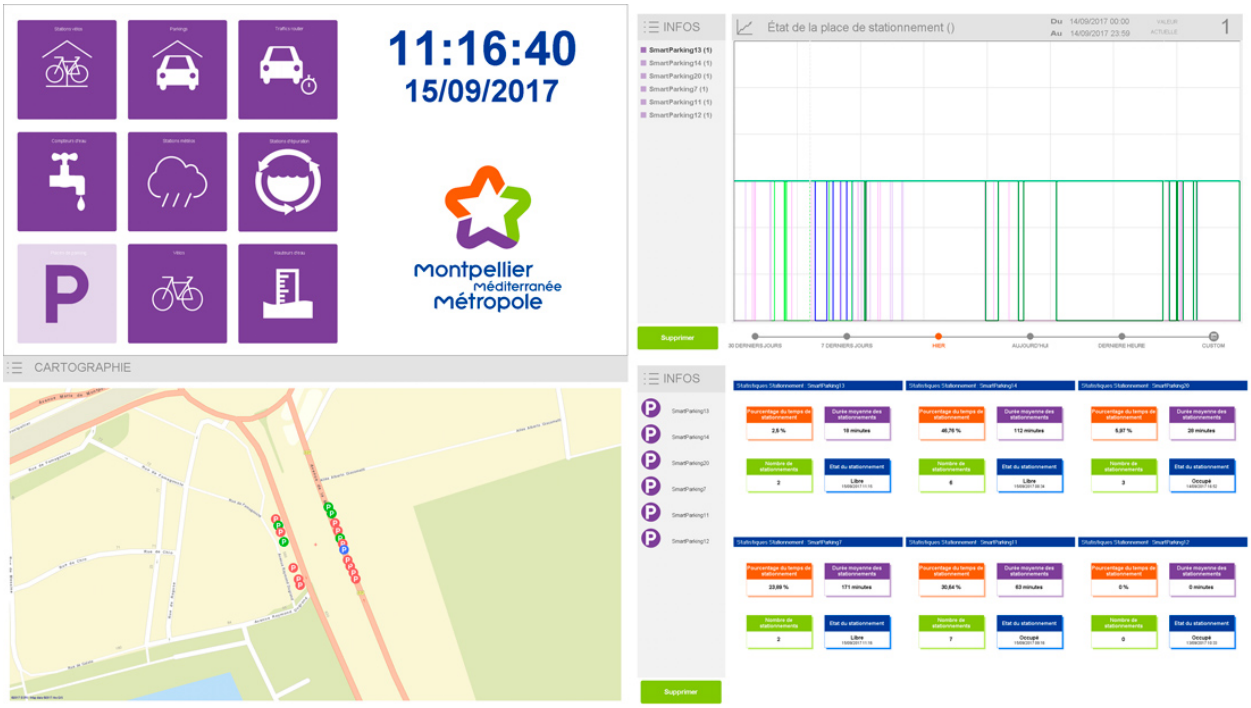


Figura 1.1: Panel de *Connected Parking*[7]

Málaga

Málaga es una ciudad de 571.026 habitantes (según datos de 2018) situada en el sureste de España. Tiene un parque de vehículos de 1.158.654 unidades (2017) de los cuales 806.408 son turismos.

El Ayuntamiento de Málaga ha apostado por el impulso de *Open Data*.[8] Entre otros datos, ha publicado *data sets* de la ocupación de 9 aparcamientos públicos subterráneos que ofrecen 2.957 plazas de aparcamiento. A partir de estos datos el grupo NEO: Networking and Emerging Optimization (2019), que es un grupo de investigación integrado en la Universidad de Málaga, presentó una propuesta de solución en un concurso de iniciativas para la reutilización de datos abiertos.

La propuesta fue seleccionada y se instalaron sensores en las plazas de los aparcamientos públicos. Se realizaron mediciones durante 250 días y se llevaron a cabo predicciones por aplicación de series temporales. En la actualidad existe un prototipo no final publicado en la web que permite conocer la probabilidad de encontrar plazas libres de aparcamiento en cada uno de los 9 aparcamientos públicos. Se ofrece en la figura 1.2 una captura de pantalla de este prototipo.

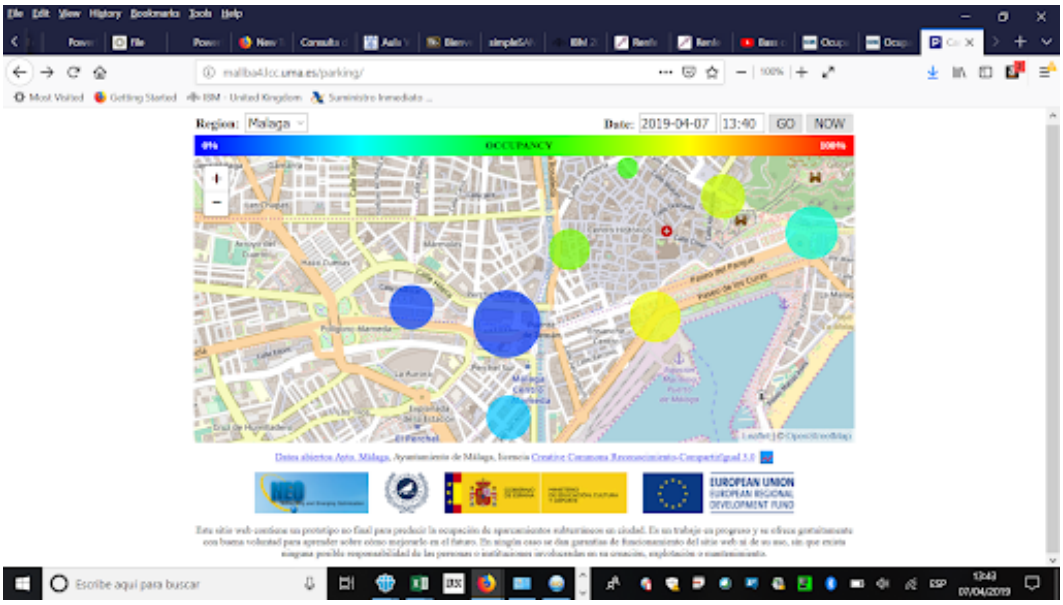


Figura 1.2: Prototipo de la aplicación para mostrar el grado esperado de ocupación en Málaga[9]

Santander

Santander es una ciudad del norte de España con 172.000 habitantes, según el censo de 2017 año en que se implantó el sistema de gestión de aparcamientos inteligente.

El proyecto *SmartSantander* es un estudio de investigación a escala ciudad de soporte de servicios y aplicaciones típicos de una *Smart City*. Una parte del proyecto consiste en monitorizar y señalar las plazas de aparcamiento disponibles en la ciudad. [10]

La ciudad se ha dividido en 27 zonas, cada una provista de un *gateway*, que acumula mediciones de hasta 5 tipos de sensores: temperatura, luminosidad, monóxido de carbono, sonido y ocupación de las plazas de aparcamiento en la vía pública.

La ocupación de las plazas de aparccamiento está sensorizada mediante dispositivos situados bajo el pavimento, que miden el campo magnético utilizando tecnología ferromagnética. Existen 2.000 sensores que realizan esta medición.

La Universidad de Cantabria participó en el proyecto, que utiliza tecnología *OTAP* (*Over the air Programming*) para la programación remota de los nodos. Al igual que en la solución de Montpellier, los nodos son dispositivos WaspMote (2019).

Como se puede observar en la imagen de la figura 1.3, este caso además de disponer de la información en sus dispositivos móviles u ordenadores, los ciudadanos pueden encontrar la información de plazas libres en paneles situados en las calles que indican el número de plazas de aparcamiento cercanas disponibles en cada dirección de modo que se pueden consultar mientras se conduce.

Figura 1.3: Señal indicando plazas de aparcamiento libres en Santander

Las soluciones anteriores están basadas en el uso de diferentes nodos conectados para cada plaza a monitorizar, mientras que la solución propuesta en este trabajo ofrece la opción de utilizar un único dispositivo conectado, en este caso la cámara. Además, ofrece la oportunidad de utilizar esas mismas imágenes para otros fines, tales como videovigilancia o incluso ampliar su uso para determinar el tránsito peatonal en los aparcamientos. Aunque estas últimas opciones quedan fuera del ámbito del presente trabajo, se ha querido reseñar sus posibilidades de uso, que mejoran las soluciones anteriores desde este punto de vista.[11]

En cualquier caso y al margen de los casos indicados previamente, la gestión inteligente de parkings en ciudades se encuentra en continua evolución y con diferentes ofertas y oportunidades, tal es el caso de las propuestas ofrecidas por Nwave o IntelliVision (2019), que que además de las tecnologías sensoriales conectadas, ya con soluciones 5G (Alfa y col., 2018)[12] ofrece alternativas de procesamiento masivo en el ámbito del Big Data.[13]

1.3. Motivación

De forma análoga a como se entiende que el contexto tecnológico nunca ha sido tan propicio para el florecimiento de soluciones *IoT*, se puede observar que tanto los niveles de contaminación en ciertas ciudades, como la proporción del tiempo de su vida que pasa una persona en atascos de circulación o buscando aparcamiento, así como la cantidad de gente que se desplaza a determinadas horas del día, se encuentran también en máximos históricos.

Según estudios de Parking Network, IBM y Cisco, buscar aparcamiento incrementa en un 16 % de media el consumo de combustible de los conductores españoles, y el 30 % de los atascos están provocados por conductores que buscan aparcamiento.[14]

Ciudades como Madrid, restringen el tráfico o el aparcamiento en la vía pública, en algunas zonas, a determinadas horas del día, en función de los niveles de contaminación.

Las soluciones del tipo *Smart Parking* suponen un claro beneficio para el ciudadano que puede ir directamente al aparcamiento en el que sabe que va a encontrar plaza, consiguiendo un ahorro de tiempo y de combustible a la vez que se minimizan los efectos contaminantes.

Para la ciudad en su conjunto, la principal ventaja consiste en que haya menos coches en circulación, provocando una reducción de la contaminación y una mejora en la circulación.

A la Administración Local le proporciona información muy valiosa para conocer zonas de saturación de los aparcamientos. Podrían utilizar la información para aplicar medidas de restricción de circulación, tenerla en cuenta para tomar decisiones como invertir en aparcamientos públicos, reforzar o ampliar líneas de transporte público, y fomentar o facilitar que las empresas de transporte compartido operen en determinadas áreas.

Desde un punto de vista tecnológico, la motivación viene suscitada por la oportunidad que el concepto y la tecnología *IoT* ofrece para dar solución al manejo eficiente de información relativa al grado de ocupación de los aparcamientos.

Este planteamiento se sustenta por soluciones tecnológicas ya desarrolladas o ideas de futuro.[15][16]

1.4. Objetivos

El objetivo general de presente proyecto plantea el desarrollo de una solución IoT para la gestión y monitorización eficiente de plazas de parking en ciudades inteligentes. Se distinguen dos aspectos fundamentales en función del hito del que forman parte, constituyendo dos sub-objetivos generales.

1.4.1. Generales

1) Aplicar técnicas avanzadas de procesamiento y clasificación de imágenes

En la aplicación se contempla la inclusión de un sistema inteligente a nivel local en el que se realiza un procesamiento de imágenes mediante técnicas de Visión por Computador para identificar las plazas de parking, seguido de un proceso de identificación para determinar si la plaza está libre u ocupada, y en este último caso incluso distinguiendo el elemento que la ocupa. Para dicho reconocimiento se plantea el uso de Redes Neuronales Convolucionales (CNN, *Convoluction Neural Networks*) en el ámbito del aprendizaje profundo (deep learning). Los resultados así obtenidos, relativos a la determinación del grado de ocupación del parking, se publican en la nube. Este planteamiento se enmarca próximo al paradigma conocido como *edge computing*, que está tomando cierta relevancia y cuya finalidad es realizar procesamientos a nivel local, y en la medida de lo posible a nivel sensorial, para enviar a la nube los datos necesarios filtrados.

2) Predicción de tasa de ocupación de un parking mediante análisis de series temporales

Con los datos publicados en la nube, se realiza un análisis predictivo del grado de ocupación de cada parking en función de una unidad de tiempo convenientemente establecida.

1.4.2. Específicos

A partir de los objetivos generales, a continuación se enumeran los objetivos específicos, junto con una breve descripción motivada que permite su justificación. Este planteamiento define también el plan de trabajo propuesto.

1) Realizar una revisión de soluciones *IoT* existentes en la literatura

Con la emergencia del *IoT*, se despliegan soluciones para una amplia gama de sectores de negocio. La mayoría de estas soluciones tienen algunos componentes comunes:

- Uno o varios **sensores** que toman alguna información de utilidad para el sistema, normalmente se trata de dispositivos de muy bajo coste. Para clasificar plazas de aparcamiento en libres y ocupadas se ha identificado el uso de dispositivos situados en cada plaza que detectan si hay un coche o no (p.e. detectando el campo magnético)

como el caso de WaspMote (2019) utilizado en Montpellier y Santander. Como alternativa a esta solución, en el presente trabajo se plantea el uso de cámaras que toman imágenes de un parking con una serie de plazas para posteriormente analizar dichas imágenes. Se trata de un planteamiento avanzado en este contexto.

- Una **unidad de procesamiento** (p.e. *CPU, SoC*) **local** con respecto a almenos un sensor, del que ingesta la información que éste capta, proporcionándosela al resto del sistema. En algunos casos este componente también aplica algún tipo de preprocesado a la información antes de compartirla (proximidad al *edge computing*). Para el presente estudio se plantea el uso de un ordenador portátil procesa las regiones tomadas de los aparcamientos aplicando técnicas de Visión por Computador. En ese mismo *host* se realiza la clasificación de las imágenes mediante técnicas basadas en CNN utilizando modelos pre-entrenados, como AlexNet (ImageNet, 2019), sobre las que se realiza un re-entrenamiento de acuerdo con las necesidades del proyecto. Esta clasificación en tiempo real podría realizarse en algún tipo de servicio *XaaS*, facilitando que se utilizasen dispositivos con menor capacidad de procesamiento y por tanto más económicos para aplicar las técnicas de procesamiento de imágenes. Se plantea así una posible ampliación de futuro.

- Uno o varios **servicios remotos** en forma de ***cloud*** (o ***fog***) que sacan partido a la información obtenida, en bruto o preprocesada. El caso que se menciona en el punto anterior relativo a utilizar un servicio de este tipo para realizar la clasificación, y el estudio predictivo que se hace en remoto de las series temporales del nivel ocupación de cada aparcamiento, son ejemplos de este tipo de componentes.

2) Diseñar la arquitectura de la aplicación, incluyendo sendos módulos local y remoto

De entre las dos alternativas más típicas para soluciones de tipo *SmartParking*, mencionadas en el punto anterior, se plantea una aproximación local para el procesamiento de las imágenes y la clasificación mediante CNN, con modelos pre-entrenados del tipo AlexNet, por su excelente y demostrado desempeño. Los datos procesados en el tiempo se envían a un servicio remoto (nube) para análisis de series temporales con predicción de ocupación de plazas. Los parámetros estimados de la serie junto con los datos de la propia serie quedan disponibles en la nube para consulta y descarga, lo que permite predicciones locales.

3) Aplicar técnicas de segmentación de imágenes orientadas a extracción de regiones

Para cada parking elegido para el estudio se crean plantillas base. Tales plantillas contienen regiones diferenciadas cromáticamente que se corresponden con las zonas esperadas de estacionamiento. El apartado 2.2 se explican las técnicas empleadas para obtener las plantillas.

La aplicación principal, utiliza estas plantillas para identificar las zonas que posteriormente se clasifican individualmente como libres u ocupadas mediante la CNN.

4) Aplicar técnicas de redes neuronales convolucionales

Utilizar una CNN pre-entrenada del tipo Alexnet, que se reentrena para el reajuste de los pesos en las capas de convolución mediante el correspondiente re-entrenamiento utilizando imágenes de vehículos estacionados y plazas de aparcamiento vacías.

Cada región etiquetada, se clasifica y se identifica como libre u ocupada. Posteriormente se contabiliza el total de plazas clasificadas como ocupadas y se publica en un servicio remoto.

5) Desarrollar un sistema de análisis predictivo en la nube

El análisis de series temporales permite la predicción de los niveles de ocupación de plazas de parking. En remoto se plantea un análisis predictivo mediante técnicas basadas en un modelo de auto-regresión, de forma que, como se ha indicado previamente, los parámetros estimados del modelo quedan disponibles en la nube con carácter público. Con la descarga de los parámetros estimados del modelo de auto-regresión y los propios datos, se realizan predicciones en local. Además se plantea, también en local, el uso de técnicas de predicción basadas en Redes Neuronales Recurrentes del tipo *LSTM* (*Long Short-Term Memory*).

1.5. Organización de la Memoria

La memoria está organizada en capítulos, de forma que el contenido del resto es como sigue. En el capítulo dos se realiza una descripción de los métodos y técnicas relativas al procesamiento de imágenes, incluyendo el modelo de red neuronal convolucional. Además se describen los métodos de predicción utilizados. El capítulo tres describe el diseño de la solución IoT planteada, donde se describen de forma detallada los módulos que la componen. El capítulo cuatro presenta los resultados obtenidos, junto con las discusiones pertinentes. Finalmente, el capítulo cinco contiene las conclusiones generales y trabajo futuro.

Capítulo 2

Descripción de métodos y técnicas aplicados

2.1. Introducción

En este capítulo se realiza una descripción de métodos y técnicas aplicados en el desarrollo de este trabajo, todas ellas enmarcadas en el ámbito de la Inteligencia Artificial con propuestas avanzadas, actualmente en pleno auge y de vanguardia. Se comienza con el estudio del procesamiento de las imágenes para la identificación de las regiones que delimitan las plazas de parking. Una vez delimitadas éstas, se procede a clasificar el contenido de cada una de ellas mediante el modelo de red neuronal AlexNet ubicada en la categoría de las CNN. Se incluye aquí tanto la fase de entrenamiento como de clasificación. El proceso de clasificación permite determinar el número de plazas ocupadas y libres, de forma que esta información pueda publicarse en la nube con el fin de disponer de esta información con acceso público. El análisis de la información se plantea bajo la perspectiva del análisis de series temporales de manera que a lo largo del tiempo se almacena dicha información convenientemente organizada a través de unidades de tiempo.

2.2. Procesamiento de imágenes

El planteamiento inicial se realiza considerando que la captura de las imágenes se realiza con una cámara que proporciona una vista panorámica de un parking con visualización de las plazas libres y ocupadas, así como otros posibles elementos sobre ellas, tales como sombras u oclusiones debidas a la presencia de árboles u otras estructuras. Obviamente, para una mayor eficiencia lo ideal es que las plazas aparezcan libres de este tipo de obstáculos no deseados. Así pues, para una mayor eficacia en el ámbito de las *Smart Cities* lo ideal es evitar todo tipo de situaciones de este tipo.

Las imágenes utilizadas se obtienen de repositorios públicos principalmente a través de Google Maps, de forma que en el momento de su captura contienen los elementos proporcionados por dicho repositorio.

Desde el punto de vista del procesamiento de imágenes, el planteamiento conceptual que se formula en el presente trabajo consiste en disponer de imágenes de parking vacíos con las líneas delimitadoras de las plazas bien identificadas, incluyendo las plazas destinadas a minusválidos. De esta forma y siguiendo el ejemplo mostrado en la imagen de la figura 2.1, correspondiente a una imagen de una zona de parking en el campus de la Universidad Rey Juan Carlos (URJC) en Móstoles (Madrid), es posible identificar las líneas de forma automática mediante el siguiente procedimiento, cuyos fundamentos teóricos se describen en Pajares y Cruz (2007)[17].

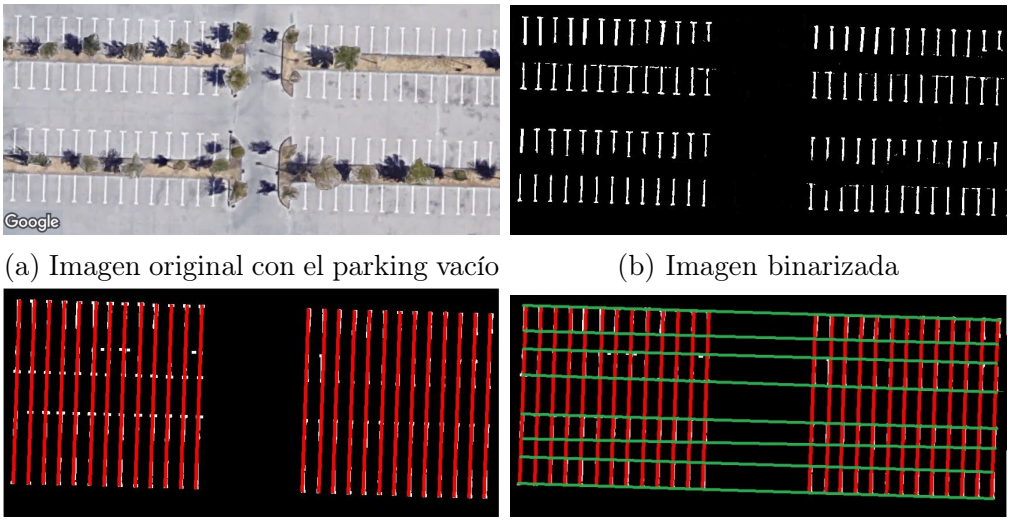
1. Transformar la imagen original en el modelo de color RGB, figura 2.1a, al modelo de color HSI.

2. Sobre la imagen de intensidad (I) segmentar ésta de forma que las líneas en blanco son tal que sus niveles de intensidad superan un valor de umbral, que se establece en 200. De esta forma se obtiene la imagen binaria mostrada en la figura 2.1b con el objetivo de ajustar las líneas verticales que aparecen en esta imagen.

3. A partir de la imagen binaria se identifican las líneas verticales mediante la transformada de Hough, consistente en ajustar una línea recta estimando los parámetros ρ y θ según la ecuación $x\cos\theta + y\sin\theta = \rho$, teniendo en cuenta que las líneas objeto de detección son prácticamente verticales, por lo que θ tiene valores próximos a $\pm 90°$. El resultado es el que se muestra en la figura 2.1c.

4. Con las líneas verticales identificadas, se determinan las líneas cuasi horizontales que delimitan las plazas. Para ello, se determina para cada bloque el inicio y finalización de cada tramo (x_i, y_i) estimando los valores promedio de las coordenadas, que definen las mencionadas líneas horizontales, figura 2.1d.

Con el procedimiento anterior, se dispone de un conjunto de celdas que definen cada una de las ubicaciones de las plazas de parking. No obstante, dado que no se dispone de una configuración de cámaras donde se puedan tomar imágenes con el parking vacío y ocupado total o parcialmente, se ha optado por una solución alternativa, con similar validez para

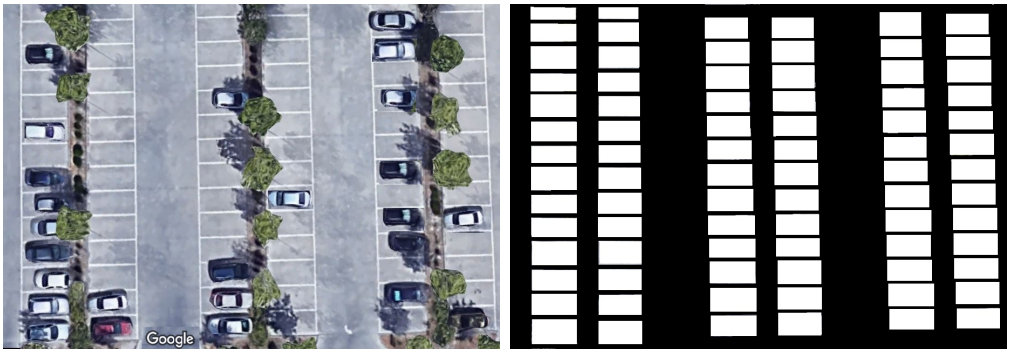


(a) Imagen original con el parking vacío (b) Imagen binarizada

(c) Trazado de líneas rectas (d) Trazado de líneas rectas(2)

Figura 2.1: Imágenes relativas al Parking de la URJC

este tipo de situaciones. En este sentido, la propuesta que se formula consiste en generar para cada parking, objeto de la monitorización, una máscara binaria, que define, como en el caso anterior, las celdas en la imagen que definen las plazas de parking. En la figura 2.2a se muestra una imagen con plazas libres y ocupadas, que también pertenece al mismo campus de la URJC y en 2.2b se muestra la correspondiente máscara binaria, que se genera manualmente mediante la visualización y asociación de las plazas en la imagen original. Así pues, las plazas aparecen identificadas con valores de intensidad 255, que sería el mismo valor asignado a las plazas identificadas por el procedimiento automático descrito previamente.



(a) Instantánea del parking de la URJC (b) Máscara del parking de la URJC

Figura 2.2: Imagen del parking de la URJC con su máscara asociada

El siguiente paso se lleva a cabo mediante el siguiente procedimiento:

- Aplicar el algoritmo de etiquetado de componentes conexas propuesto por Haralick y Shapiro (1992)[18]. De esta forma a cada una de estas regiones se le asigna una etiqueta

identificativa.

- Para cada una de las regiones y en base a su etiqueta, se obtienen dos propiedades identificativas: área y polígono que las delimita (*bounding box*).
- Se descartan regiones con áreas relativamente pequeñas, menores que un determinado umbral, puesto a un valor de 100 en el presente estudio. El objetivo es eliminar posibles áreas espurias. Con las áreas finalmente seleccionadas, cumpliendo los criterios del paso anterior, y teniendo en cuenta el correspondiente bounding box se realiza un recorte sobre la imagen original, obteniéndose así muestras como las mostradas en la figura 2.3 (dos coches y asfalto). Estas imágenes son las que se proporcionan a la red CNN, tanto para la fase de entrenamiento como de clasificación.

Figura 2.3: Imágenes con el contenido de tres celdas (coches y asfalto)

2.3. Redes Neuronales Convolucionales

Las redes neuronales convolucionales tienen multitud de aplicaciones entre las que se encuentra la clasificación natural de objetos (*Natural Object Classification*). Inspiradas en cómo identifican objetos naturales, mediante sus organos visuales, ciertos mamíferos, las redes diseñadas con este objetivo, se basan en una seria de pasos secuenciales entre capas de neuronas, asimétricas en cuanto a que producen distintos efectos sobre la entrada que reciben de la capa anterior y propagan a la capa siguiente.[19]

Una CNN es un tipo específico de red neuronal, cuya base se sitúa precisamente en lo que se conocen como capas de convolución, de ahí su nombre. Estas capas de convolución están formadas por filtros y éstos contienen sus respectivos pesos asociados, siendo éstos los que verdaderamente se ajustan durante el proceso de aprendizaje de la red con los ejemplos procesados. Conviene señalar en este sentido que, como en el resto de sistemas basados en redes neuronales, se distingue entre fase de *aprendizaje* y *clasificación*.

Como se ha indicado previamente, las CNN, durante la fase de aprendizaje, ajustan los pesos de las capas de convolución y a partir de éstos se determinan las estructuras señaladas previamente sobre los ejemplos suministrados. Estas capas de convolución simulan el comportamiento de los modelos biológicos en el campo de la visión. En este sentido, como se verá posteriormente en el capítulo cuatro durante el análisis de resultados, las capas de convolución son similares a los conocidos como filtros de Gabor[20], que son los modelos establecidos en los campos receptivos visuales biológicos.

Una CNN es una red neuronal multicapa diseñada para reconocer patrones visuales. Uno de los modelos ampliamente usados es el definido por ALEXNET[21], esta red ganó la competición ILSVRC-2012 con una tasa de error de un 15,3 % frente al segundo competidor que obtuvo un 26,2 %. Para comprender este tipo de red, a continuación se describen los elementos básicos de su estructura. En la siguiente figura se muestran las diferentes capas que usa la red AlexNet.

AlexNet posee veinticinco capas, siendo un modelo pre-entrenado con anterioridad para clasificar mil objetos pertenecientes a distintas categorías. La configuración de la red se describe posteriormente.

En cada capa se aplican distintas operaciones, incluyendo las de convolución con núcleos de distintos tamaños especificados mediante la nomenclatura del tipo $3x3$ u otra dimensión. Además, los símbolos k, p y s que aparecen en cada una de las capas, se refieren respectivamente al número de filtros (con su correspondiente núcleo de convolución) aplicados, al número de ceros añadidos (*padding*) y unidades de desplazamiento del núcleo (*stride*). Este modelo recibe como entrada una imagen de dimensión $227 * 227 * 3$.

En la figura 2.4 se muestra el esquema general de Alexnet con su correspondiente configuración, que se describe seguidamente.

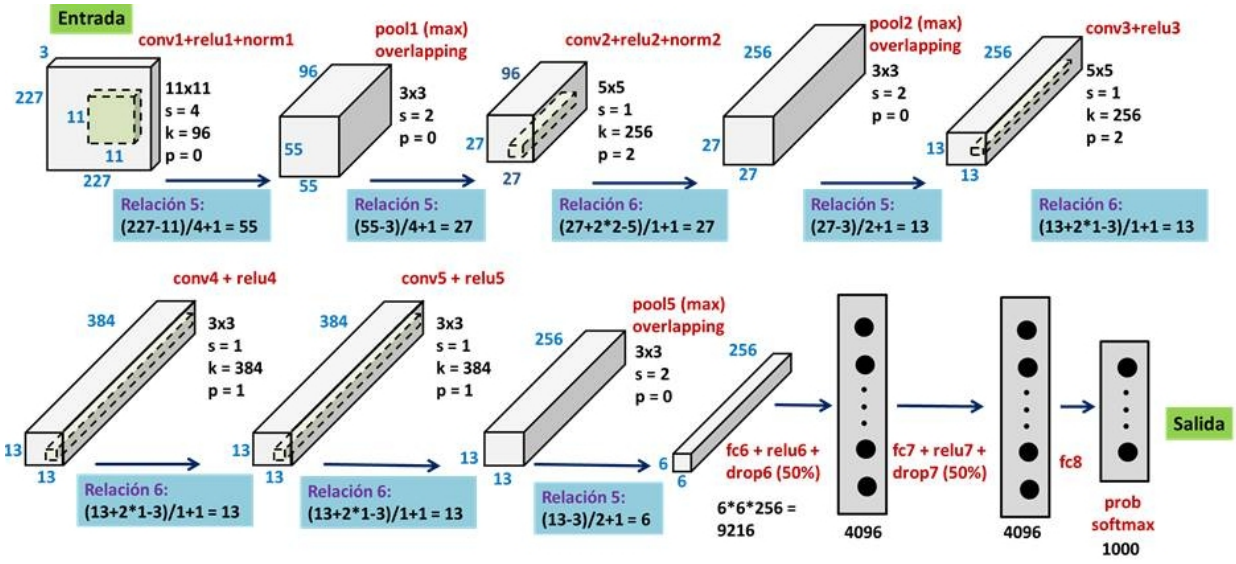


Figura 2.4: Esquema de la red AlexNet

En dicha figura se muestra la secuencia de capas de la red empleada para este trabajo. A continuación se resume brevemente el efecto que tiene cada una de las mismas:

- **Entrada** *(I)* son imágenes de dimensión $227 * 227 * 3$, correspondientes a las distintas categorías, similares a las mostradas en la figura 2.3, tanto en la fase de aprendizaje como de clasificación.

- **Convolución (conv)**: esta operación matemática combina dos funciones para generar

una tercera, básicamente superpone una función sobre otra modificada para obtener una serie de características.

$$S(i,j) = (I * K)(i,j) \tag{2.1}$$

El parámetro K, denominado filtro (kernel) de convolución, es un vector o matriz de parámetros ajustables por el proceso de aprendizaje. Es lo que se conoce como tensor. La salida S se define generalmente como mapa de características (*feature map*).

- **ReLU (relu)**, es una función de activación, de forma que por cada resultado generado por la convolución, la rectifica linealmente devolviendo 0 para valores negativos y el propio valor de entrada para valores positivos, según se define a continuación, con x siendo la entrada a la neurona.

$$f(x) = \text{máx}(0, x) \tag{2.2}$$

- **Normalización (norm)**, si se denota por $a^i_{x,y}$ a la actividad de una neurona obtenida por aplicación del núcleo i en la posición (x, y) y luego se aplica la nolinealidad ReLU, la actividad de la respuesta normalizada $b^i_{x,y}$ viene dada por la expresión,

$$b^i_{x,y} = \frac{a^i_{x,y}}{\left(k + \alpha \sum_{j=\text{máx}(0,i-n/2)}^{\text{mín}(N-1,i+n/2)} (a^i_{x,y})^2\right)^\beta} \tag{2.3}$$

donde la suma se extiende sobre n mapas adyacentes generados por los núcleos en la misma posición (x, y) y N es el número total de núcleos en la capa. El ordenamiento de los mapas es arbitrario y se determina antes de que comience el aprendizaje. Esta clase de respuesta normalizada implementa una forma de inhibición lateral inspirada por el tipo encontrado en neuronas reales, creando una competición para actividades elevadas frente a salidas de neurones obtenidas usando diferentes núcleos. Las constantes k, n, α y β son hiperparámetros cuyos valores se determinan usando un conjunto de validación. En Krizhevsky y col. (2012)[22] se proponen los siguientes valores $k = 2$, $n = 5$, $\alpha = 10^{-4}$ y $\beta = 0{,}75$. En el mismo trabajo, se indica que esta normalización se realiza en ciertas capas tras la aplicación de ReLU.

- **Pooling (pool)**, las capas denominadas de pooling o agrupaciones proporcionan una importante invarianza a pequeñas traslaciones de la entrada. Existen varias operaciones de este tipo, una de ellas es el máximo (max), que consiste en dividir la entrada en ventanas, generalmente sin solapamiento, produciendo como salida el máximo de la ventana. Otra de las operaciones es la media (*average*, *mean*) de la ventana en la que la salida es el resultado de esta operación sobre la ventana[23]. Desde el punto de vista de la convolución, esta operación puede interpretarse bajo la suposición de que alguna función se repite sobre subconjuntos (ventanas) de la entrada. Por tanto, teniendo en

cuenta las relaciones previas, el *pooling* es una operación sin *zero padding*, de forma que la relación que describe el caso general y que sirve para cualquier tipo de *pooling*, es como sigue, teniendo en cuenta que i es el tamaño de la imagen o estructura de entrada, k el tamaño del filtro, s es el desplazamiento (*stride*) y o el tamaño de la imagen o estructura resultante.

$$o = \left[\frac{i-k}{s}\right] + 1 \tag{2.4}$$

El pooling ayuda a hacer la representación aproximadamente invariante a pequeñas traslaciones de la entrada, lo que significa que si se traslada la entrada con un pequeño desplazamiento, los valores de muchas salidas sobre las que se ha aplicado el pooling no cambian.

- **Dropout (drop)**, sirve para no sobresaturar la red neuronal dotando soluciones de polinomios de grado demasiado alto. Para ello se propone anular determinado tipo de neuronas de forma aleatoria mediante un hiperparámetro que indique la probabilidad de supervivencia de cada neurona. Las operaciones de este tipo aparecen indicadas como *drop*, correspondiendo en ambos casos al 50 % de las neuronas anuladas.

- **Softmax**, la función sofmax o función exponencial normalizada, que aparece generalmente en las últimas capas ocultas, definida en la ecuación 2.5, se emplea para proyectar un vector n-dimensional, x de valores reales en un vector n-dimensional $softmax(x)$ de valores reales en el rango $[0, 1]$.

$$softmax(x)_i = \frac{exp(x_i)}{\sum_{j=1}^{n}} exp(x_j) : i = 1..n \wedge x = \left(x_1, x_2, .., x_n\right) \in i^n \tag{2.5}$$

- **Gradiente descendente**, Es la técnica de minimización que se usa para el ajuste de los pesos de la red durante el proceso de retro-propagación. La función a minimizar se conoce como función objetivo o criterio y cuando se está minimizando, se le denomina también cost function, *loss function* o textiterror function. En el ámbito del aprendizaje automático, el problema consiste en minimizar una función objetivo que tiene la forma,

$$J(w) = \frac{1}{n}\sum_{i=1}^{n} J_i(w) \tag{2.6}$$

donde el parámetro w que minimiza $J(w)$ debe estimarse. J_i se asocia con la i-ésima observación en el conjunto de datos utilizados para el entrenamiento (ajuste). El término estocástico proviene del hecho relativo a la selección de las muestras (observaciones) para el ajuste de forma aleatoria. $Ji(w)$ es el valor de la loss function en el i-ésimo ejemplo y $J(w)$ es el riesgo empírico. El gradiente descendente se usa para minimizar esta función de forma iterativa, con iteraciones t,

$$w(t+1) = w(t) - \epsilon \frac{1}{n} \sum_{i=1}^{n} \nabla J_i(w) \tag{2.7}$$

De esta forma iterativa el método recorre el conjunto de entrenamiento, realiza la actualización anterior para cada muestra de entrenamiento. Se pueden realizar varios pasos sobre el conjunto de entrenamiento hasta la convergencia. Si se hace esto, los datos se pueden seleccionar aleatoriamente (criterio estocástico) en cada paso para evitar ciclos. Estas muestras así seleccionadas constituyen lo que se denomina batch. Las implementaciones típicas pueden usar una razón de aprendizaje adaptativa para que el algoritmo converja. En pseudocódigo, el método de gradiente descendente estocástico es como sigue,

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

La red AlexNet original genera una salida con capacidad para clasificar mil categorías, si bien en el presente trabajo, éstas se han limitado considerablemente a cuatro, a saber: vehículo, asfalto, que se corresponde con plaza libre, árbol, que oculta una plaza y que puede considerarse libre y sombras, proyectadas de los árboles, cuya interpretación también es plaza libre. Tras la capa softmax se espera obtener un valor, en función de las cuatro clases, que variará entre 0 y 1 para establecer así una probabilidad de pertenencia a cada una de las clases. En función del error obtenido a la salida, con respecto a la salida esperada, se realiza un proceso de propagación hacia atrás (retropropagación) para ajustar los pesos de la red en proceso de aprendizaje. El objetivo consiste en minimizar al máximo la función de pérdida, momento en el que se produce lo que se denomina convergencia de la red.

2.4. Series temporales

2.4.1. Definición

Las series temporales se refieren a valores de datos muestreados en el tiempo, generalmente a intervalos regulares. En realidad la evolución en el tiempo de un evento, por ejemplo el grado de ocupación de un parking en franjas horarias a lo largo de los días de la semana, tal sería el caso de los viernes de 8 a 10 de la mañana durante un año o el número de vehículos de entrada a una cuidad en esa misma franja horaria por mencionar sólo dos ejemplos significativos. El análisis de series temporales conlleva implícitamente los siguientes tres procesos:

- Patrones de la serie
- Modelado
- Predicción

Una serie temporal consta de N observaciones o datos, también denominandos patrones secuencialmente ordenados en el tiempo relativos a una determinada variable (univariante) o varias (vectorial o multivariante) observable en distintos instantes de tiempo[24].

$$y_t = y_1, y_2, ..., y_N : t = 1, ..., N \tag{2.8}$$

donde y_t es la observación en el instante $t(1 \leq t \leq N))$ de la serie.

Si se expresan las N observaciones mediante un vector, se tiene $y \equiv (y_1, y_2.., y_n)$ en el caso univariante que constituye el objeto de este trabajo.

El modelado de la serie se refiere a la estructuración del esquema para captar la información subyacente en los datos de la serie. La predicción se realiza a partir del modelo y de los parámetros estimados o aprendidos según dicho modelo. A continuación se exponed dos modelos específicos relacionados con este tipo de datos estructurados en el tiempo.

2.4.2. Modelo autorregresivo

Un proceso autorregresivo de orden p, $AR(p)$, es un proceso $\left[Y_t\right]_{\infty}^{-\infty}$ que cumple,

$$Y_t = c + \phi_1 Y_{t-1} + \phi_2 Y_{t-2} + ... + \phi_p Y_{t-p} + \varepsilon_t \tag{2.9}$$

donde $c, \phi_1, \phi_2, ..., , \phi_p$ son parámetros y $\left[\varepsilon_t\right]_{\infty}^{-\infty}$ ruido blanco.

Un proceso $AR(1)$ se escribe como,

$$Y_t = c + \phi_1 Y_{t-1} + \varepsilon_t \tag{2.10}$$

Considérese una serie temporal $\left[Y_t\right]_1^N$ descrita por un modelo $AR(p)$ de la forma expresado en la ecuación 2.11 como sigue con sus correspondientes coeficiente ϕ_j,

$$Y_t = c + \phi_1 Y_{t-1} + \phi_2 Y_{t-2} + ... + \phi_p Y_{t-p} + \varepsilon_t \tag{2.11}$$

En este caso, el objetivo consiste en determinar el orden p del modelo, así como los correspondientes coeficientes. Para ello existen diversos procedimientos, siendo uno de ellos el que utiliza las ecuaciones de Yule-Walker[25][26]. Bajo consideraciones estadísticas, y realizando un análisis de varianzas y covarianzas se pueden obtener los mencionados coeficientes. En términos generales las funciones de predicción de procesos autorregresivos puros, se obtienen aplicando la regla de la cadena según la ecuación previa. Así para un horizonte h, se obtiene primeramente la siguiente predicción Y_t, que sirve para la estimación de Y_{t+1}, ésta para Y_{t+2}

y así sucesivamente hasta conseguir el horizonte indicado h, siempre teniendo en cuenta que en la predicción intervienen los p términos del modelo AR.

$$\begin{aligned} Y_t &= c + \phi_1 Y_{t-1} + \phi_2 Y_{t-2} + ... + \phi_p Y_{t-p} \\ Y_{t+1} &= c + \phi_1 Y_t + \phi_2 Y_{t-1} + ... + \phi_p Y_{t-p+1} \\ Y_{t+2} &= c + \phi_1 Y_{t+1} + \phi_2 Y_t + ... + \phi_p Y_{t-p+2} \\ &\cdot \\ &\cdot \\ &\cdot \\ Y_{t+h} &= c + \phi_1 Y_{t-1+h} + \phi_2 Y_{t-2+h} + ... + \phi_p Y_{t-p+h} \end{aligned} \tag{2.12}$$

2.4.3. Modelo basado en redes neuronales

Dentro de las redes neuronales recurrentes (RNN, *Recurrent Neural Networks*), destaca el modelo conocido como Long Short-Term Memory (LSTM). Este tipo de redes fueron introducidas por Hochreiter y Schmidhuber (1997)[27] con el fin de resolver el problema que surge con otros modelos relativos a que sólo son eficientes para series con poca dependencia temporal.

Con estas redes se procesa una secuencia de pares de entrada salida $\left(x_t, y_t\right)_{t=1}^{n}$. Para cada par (x_t, y_t) , la celda LSTM toma una nueva entrada x_t y el valor oculto h_{t-1} obtenido en el último paso y produce una estimación $\hat{y}_i$ para la salida objetivo y_t dada la secuencia de entrada previa $x1, x2, \ldots, xt$ también con un nuevo valor oculto h_t y un nuevo valor de memoria m_i. La figura 2.5 muestra la estructura de una celda LSTM[28][29][30].

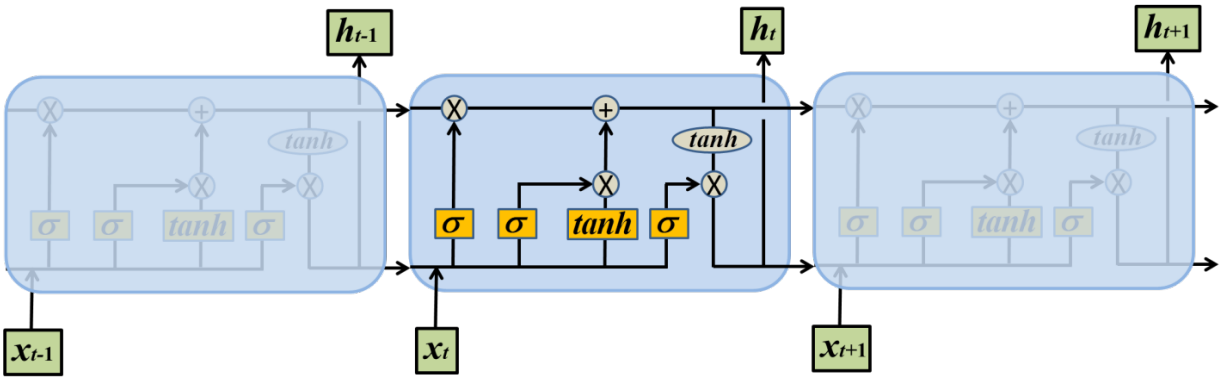


Figura 2.5: Celda LSTM

En la figura anterior aparecen operaciones del tipo sigmoide (σ) y tangente hiperbólica (tanh) que operan sobre las entradas x_t, que en definitiva se encargan de determinar qué información se mantiene o no en el estado de la celda, para producir una salida determinada. Con las entradas se asocian los correspondientes pesos, que son realmente los parámetros

que aprende la red durante la fase de entrenamiento, con los que realizar predicciones una vez integrados en el modelo.

Capítulo 3

Diseño de la aplicación

3.1. Planteamiento del diseño

El presente capítulo describe en primer lugar el diseño conceptual modular de la arquitectura que se plantea, indicando los recursos utilizados en cada caso. En segundo lugar se analiza el flujo de datos presente entre cada uno de los módulos del sistema. Finalmente, se describe el modelo de simulación utilizado, que sustituye en este caso al diseño real. Esto es necesariamente así debido a la imposibilidad material de implantar un sistema de estas características en una o varias ubicaciones físicas.

3.2. Arquitectura del sistema

La figura 3.1 se corresponde con un diagrama de la topología de la solución propuesta. Para facilitar su interpretación, se describen conceptualmente los diferentes agentes o módulos que integran la arquitectura:

- *Parking*: Superficie de acceso público que dispone de plazas, generalmente señalizadas, destinadas al estacionamiento de vehículos.
- *Vehicle*: Vehículo susceptible de estacionar en las superficies destinadas a ello, que constituyen el objeto dinámico de monitorización.
- *Free Place*: Parte del parking habilitada y disponible para que aparque un vehículo.
- *Bussy Place*: Parte del parking habilitada y no disponible para que aparque un vehículo, principalmente por existir un vehículo ya estacionado en ese mismo lugar.
- *Camera*: Dispositivo que registra imágenes, con al menos un parking a su alcance y enfocada hacia éste. La cámara envía imágenes hacia el host según su capacidad de captura, establecida en un determinado número de imágenes por segundo.

- *Parking's Caption*: Imagen instantánea tomada de un Parking en un instante de tiempo dado. En el caso planteado en el presente trabajo, por las razones expresadas previamente, estas imágenes se toman de panorámicas ofrecidas por Google Maps (2019), tal y como se describe en el capítulo cuatro.

- *Solution Host*: Unidad de procesamiento que recibe las imágenes de la cámara situada en cada uno de los parkings, se encarga de gestionar los servicios de procesamiento de imágenes y su clasificación, la transferencia de datos con la nube y el procesamiento de datos con fines de predicción en el tiempo mediante análisis de series temporales. En el capítulo cuatro se identifica el tipo de procesador utilizado en este caso, así como la implementación de los procesos en Matlab (2019).

- *Classification*: Clasificación individual de las plazas de parking que aparecen en la imagen (Parking's Caption) en libres y ocupadas. Lo cual se lleva a cabo en el Host mediante el modelo de red entrenada y almacenado convenientemente.

- *Number of places*: Cantidad de las plazas libres y ocupadas según la clasificación realizada.

- *Ocupation Average*: Porcentaje de ocupación del parking.

- *Current Time*: Hora a la que se toma la imagen.

- *Trained CNN*: Bajo este agente genérico se ubican los siguientes servicios: a) procesamiento de la imagen recibida; b) Entrenamiento de la CNN, c) Modelado y predicción basada en series temporales. En este módulo se almacenan los resultados derivados del entrenamiento de las redes. En la solución propuesta, esta tarea igualmente podría realizarse de forma remota.

- *Canal*: Hace mención a los servicios ubicados a nivel remoto en la nube, donde se almacenan datos y se llevan a cabo procesamiento de datos cuya conveniencia así se recomienda. Estos servicios se han ubicado en la plataforma ThingSpeak (2019), que se describe posteriormente con detalle.

- *Aministration*: Administración pública con acceso a los datos de la nube tanto para recabar información con fines de gestión y regulación de las plazas de parking o monitorizar su estado al objeto de llevar a cabo tareas tales como regulación del tráfico rodado en función de los niveles de ocupación del parking. También puede recibir alertas instantáneas cuando se cumplan determinadas condiciones para tener información puntual del estado y evolución del grado de ocupación del parking.

- *People*: Habitantes de la ciudad, visitantes y demás interesados en la información relativa al grado de ocupación de los parkings en la ciudad elegida con el fin de llevar a cabo un proceso de auto-gestión.

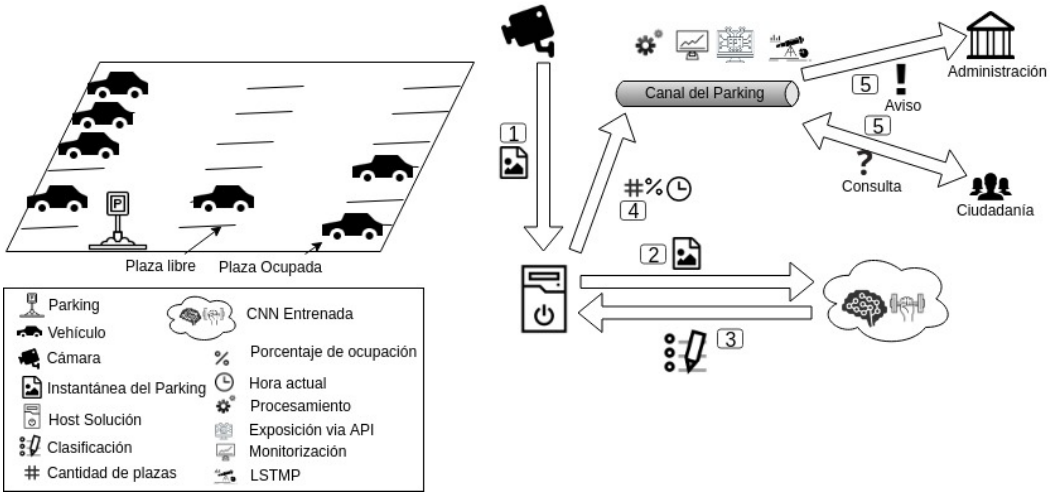


Figura 3.1: Topología

3.3. Procesamiento de datos

Seguidamente se describe el flujo y procesamiento de datos según el esquema de la arquitectura mostrada en la figura 3.1. Obsérvese, en dicho esquema la existencia de las flechas que indican el sentido de transferencia de los datos junto con un número de orden que define de alguna manera la jerarquía del proceso. Dicha numeración se corresponde además con los puntos siguientes:

1. La cámara, captura imágenes de forma continua a razón de un número determinado por segundo, según sus especificaciones. Se selecciona una imagen de la secuencia, que queda a disposición de la unidad de procesamiento (Host Solución).

2. Mediante el proceso descrito en la sección 2.2, la unidad de procesamiento (Host Solución) selecciona las regiones de la imagen donde se espera que haya vehículos estacionados.

3. Posteriormente se clasifica individualmente cada una de las estas regiones mediante la red re-entrenada (CNN Entrenada) para la aplicación que se propone, y sobre la que ya se han ajustado los pesos. Esta red, como se ha descrito en la sección 2.3, parte del modelo pre-entrenado de Alexnet. Para cada una de las regiones segmentadas en el procesamiento de imágenes previo, se realiza la clasificación individual de cada región para identificar entre plazas libres u ocupadas. Una vez se han procesado todas ellas, y teniendo en cuenta el número total de plazas existente en el parking, se determina el número de plazas disponibles y ocupadas, teniendo en cuenta que una plaza está ocupada cuando la región correspondiente se clasifica como perteneciente a la categoría vehículo, según se describe en el capítulo cuatro. Para la clasificación se utiliza el mismo host (Host Solución) que para el el re-entrenamiento y el reajuste de los pesos.

4. El Host Solución, en el tiempo programado, contabiliza el número de plazas libres, de forma que conociendo el total de plazas disponibles en el parking se determina el número de plazas ocupadas. El número de plazas totales, libres y ocupadas se publican en la correspondiente plataforma en la nube, en este caso ThingSpeak (2019) dedicada a tal propósito. Cada uno de estos datos tiene asociada la fecha de publicación con indicación del año, mes, día, hora, minuto y segundo correspondiente a la publicación del dato, que puede corresponderse con la captura de la imagen (Instantánea del Parking) o con un promedio de ocupación durante un tramo especificado, publicando el dato a la finalización del periodo considerado. Por ejemplo, en el presente trabajo se realiza una simulación considerando un tramo horario de una hora (de 9 a 10h de la mañana) durante el último año y durante los viernes de cada semana, de tal manera que a las 10h de cada viernes se publica el dato. De esta forma los datos quedan estructurados en la nube, a través de la correspondiente interfaz (Exposición via API) constituyendo una secuencia (serie) temporal para su análisis con fines de predicción. En la solución que se plantea, se propone el análisis temporal de la seria mediante un modelo autorregresivo, descrito en el capítulo dos.

5. La secuencia de datos queda disponible en la nube para su consulta, bien por parte de las administraciones públicas o cualquier usuario que requiera de la información y cuando se considere oportuno. También se pueden habilitar servicios de aviso cuando se produzcan determinados eventos, tal como la publicación de un dato, a una hora determinada o cuando el porcentaje de ocupación alcance un cierto nivel. En la nube también están disponibles los parámetros que definen el modelo resultante del análisis de la serie temporal, con ellos se pueden hacer predicciones en cualquier dispositivo con acceso a ellos o en la propia nube. Estos mismos datos se pueden descargar para realizar estimaciones a nivel local cuando los procesos requeridos para estimar el modelo no sean soportados por los servicios ofrecidos por la plataforma en la nube. Esto es lo que ocurre en el planteamiento que se presenta con respecto a la estimación del modelo LSTM, que se realiza en el host (Host Solución) dada su capacidad de proceso.

3.4. Simulación

Como se ha indicado reiteradamente, este trabajo estudia la viabilidad y la utilidad de una solución IoT, que de forma autónoma realice una serie de tareas para facilitar el acceso público a los niveles de ocupación, actuales y previstos, de los parkings de una ciudad, todo ello dentro del paradigma de las ciudades inteligentes. Estas tareas se han desarrollado parcialmente en concepto de prueba con las consideraciones indicadas previamente. A lo largo de este capítulo cuando se menciona la solución desarrollada, se hace referencia a este desarrollo conceptual, y cuando se habla de la solución propuesta se refiere al despliegue e instalación de todas las metodologías que se plantean en el presente trabajo.

No obstante, como se ha indicado previamente, la implantación de este sistema a nivel físico no resulta factible por su naturaleza. Por tal motivo, a continuación se explica el

proceso de simulación, indicando las diferentes tareas asociadas, acompañadas de las consideraciones respectivas en cada una de ellas desde el punto de vista de dicha simulación y con vistas al análisis de su viabilidad, cuyos resultados en los distintos niveles se muestran en el capítulo cuatro. El proceso de simulación se estructura según los pasos siguientes:

Paso 1: *Configuración de un canal en ThingSpeak*

Obviamente, y con carácter previo, es necesario dar de alta la correspondiente cuenta en ThingSpeak, creando a continuación el correspondiente canal, con su identificador (ID) y su configuración tal y como aparece en la figura 3.2. En ella se definen siete campos con la descripción que se indica. Así en *Field1* se almacenan los datos de la serie. En *Field2* el número de plazas disponibles en el parking. En *Field3* las plazas libres. En *Field4* el porcentaje de ocupación. *Field5* contiene los datos estimados que definen el modelo autorregresivo de orden p, AR(p). *Field6* es un flag que se activa cuando los datos se han cargado desde el Host con fines de simulación. Finalmente, *Field7* contiene un segundo flag para indicar cuándo ha finalizado el programa de estimación del modelo $AR(p)$.

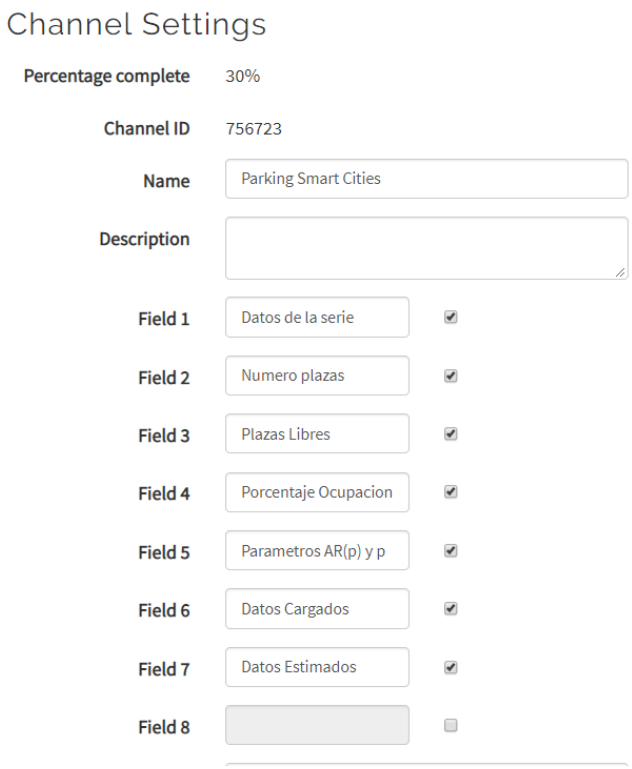


Figura 3.2: Configuración de canal en ThingSpeak

Paso 2: *Carga de datos en ThingSpeak*

Se ha mencionado previamente, que para la simulación se supone que todos los viernes de cada semana a las 10h se carga el correspondiente valor medio de ocupación del parking entre las 9 y las 10h mediante el procesamiento y clasificación de las imágenes con la CNN, anotando los datos correspondientes en los Field1 a 4 con su correspondiente asociación de

fecha. Esta acción se realiza con el procedimiento descrito en el programa *EjemploCNNClasificacionParking.m*, cuyo entrenamiento se realiza con el programa *EjemploCNNTraining.m*. Dado que no se dispone de datos reales correspondientes a un año, se genera una serie de datos aleatorios con 52 valores correspondientes a un año de captura de datos. Una vez cargados los datos el valor de Field 6 se activa. El procedimiento de carga se realiza con el programa *Cargar.m*. En este punto es cuando se puede activar un control de cualquiera de los definidos en ThingSpeak para enviar un mensaje, por ejemplo a las Administraciones para indicar

Paso 3: *Estimación del modelo AR(p)*

Una vez realizada la carga de datos, se levante el flag de Field6 para activar el control de tipo React tal y como aparece en la figura 3.3. Esto produce la ejecución inmediata del programa cargado en ThingSpeak identificado como Modelo AR(p), tal y como se muestra en la figura 3.4, este mismo programa se contiene en Estimar.m. Este modelo es el que se describe en la sección 2.4, cuyos resultados se muestran en el capítulo cuatro. El modelo AR de la figura 3.4 es de orden p. Tras la ejecución de este programa se almacenan los tres parámetros del modelo en Field5, junto con el valor de p. Además se levanta el flag de Field7 para indicar, a quienes accedan al canal, que los parámetros del modelo están estimados y actualizados.

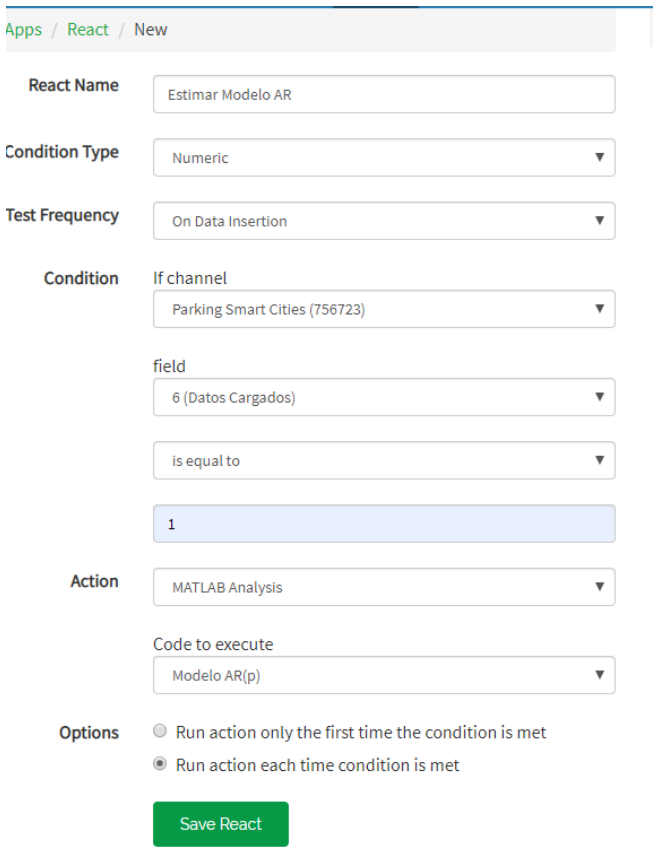


Figura 3.3: Configuración de canal en ThingSpeak

Modelo AR(p)

MATLAB Code

```
%% Parte II: Código a ejecutar en ThingSpeak: estimación usando la funcionalidad de Matlab.
% Este código se ejecuta en ThingSpeak, donde se realiza la estimación de los parámetros
% En Field2 se almacena (se hace así para evitar accesos de escritura frecuentes al ThingSpeak
% p: orden del modelo
% p+1 parámetros: (c, phi_1, phi_2,...,phi_p)

% Recuperación de los datos de la serie temporal
% Canal Parking: Identificación y claves de acceso
ChannelIDParking = 756723;
readAPIKeyParking = '3LFMC1XUNP83H9MH';
writeAPIKeyParking = 'CJ3J78VICY7IMSFV';

%delay(10); %esperamos un tiempo para evitar accesos de lectura muy frecuentes

% Datos leídos del campo Field1 donde se han almacenado las variables de
% porcentaje. Se leen los datos comprendidos en el rango de tiempo almacenado un poco antes y
t1 = datetime(2018,04,6,9,0,0); t2 = datetime(2019,03,29,11,0,0);
datos_Field1 = thingSpeakRead(ChannelIDParking, 'Fields', 1, 'DateRange',[t1,t2],'Readkey',rea

p = 2; % orden del modelo
modeloAR = ar(datos_Field1,p);

dataField2(1) = p;
dataField2(2) = modeloAR.A(1);
for k=3:1:p+2
  dataField2(k) = modeloAR.A(k-1);
end
% Generar timestamps para los parámetros estimados
tStamps = (datetime('now')-minutes(p+1):minutes(1):datetime('now'))';
% Crear timetable
dataTable = table(tStamps,dataField2');
% Escribir en el campo 5
respuesta = thingSpeakWrite(ChannelIDParking,dataTable,'Fields',5,'Writekey',writeAPIKeyParkin

% Flag en Field 7 para indicar que se han estimado ya los datos
dataField7 = 1;
respuesta = thingSpeakWrite(ChannelIDParking,dataField7,'Fields',7,'Writekey',writeAPIKeyParki

% predicción de los 10 siguientes datos
Horizonte = 10;
DatosPredichos = forecast(modeloAR,datos_Field1,Horizonte)

% función para esperar accesos (de escritura) a datos del canal
function t = delay(segundos)
  % esperar un tiempo mínimo en segundos
  a = datetime('now');
  b = datetime('now');
  t = second(b)-second(a);
  while t < segundos
    b = datetime('now');
    t = second(b)-second(a);
  end
end
```

Figura 3.4: Programa Modelo $AR(p)$ en ThingSpeak

La ejecución del programa Modelo AR(p) además de los parámetros realiza una predicción con un horizonte temporal de 10, que se corresponde con los resultados de ocupación previstos para las diez siguientes semanas. Los resultados aparecen como salida del programa y son los que se muestran en la sección 4.3.1. Desde el punto de vista ejecutivo, hay que tener en cuenta que ThingSpeak, asigna una ventana de tiempo para la ejecución completa del programa, además de que los accesos de escritura deben realizarse tras esperas de 15s, lo que dificulta la programación al respecto.

Paso 4: *Estimación del modelo LSTM*

Una vez que la serie temporal se encuentra almacenada en ThingSpeak, tal y como se ha indicado previamente, se puede descargar en el Host para realizar la estimación mediante el modelo LSTM descrito en la sección 2.4.3, cuyos resultados se muestran en la sección 4.3.3. El programa correspondiente se contiene en *Predecir.m*

Paso 5: *Visualización de los datos en ThingSpeak*

Los datos almacenados en el correspondiente canal se pueden visualizar mediante el acceso al canal a través de cualquier dispositivo con acceso al mismo, generalmente internet. Dentro de las opciones de visualización ThingSpeak proporciona una APP para móviles,

denominada ThingView Free, con esta capacidad.

En la figura 3.5 se muestran los datos en los correspondientes campos de un canal en dicha plataforma.

Figura 3.5: Visualización de los datos en los campos de ThingSpeak

Se ha subido a subido el código desarrollado al siguiente repositorio con permisos de lectura (y descarga) para cualquier usuario de la Univerisdad Complutense de Madrid para ponerlo a disposición del tribunal.

Capítulo 4

Resultados

4.1. Introducción

El presente capítulo refleja los resultados obtenidos mediante los desarrollos realizados. En primer lugar se describe el tipo de datos utilizados, siendo esencialmente imágenes, así como las herramientas para su procesamiento.

A continuación se presentan los resultados relativos al procesamiento de imágenes, distinguiendo en este sentido, los siguientes aspectos: a) pre-procesamiento de las imágenes, para la segmentación de las mismas con el fin de proporcionar las entradas a la red CNN; b) entrenamiento de la CNN y c) clasificación de las imágenes para identificación del estado de las plazas de parking y el cómputo de plazas libres y ocupadas.

Teniendo en cuenta cómo se obtienen los datos relativos a la ocupación, se procede a la generación de datos simulados con el fin de obtener series temporales para su análisis con fines de predicción según un horizonte temporal definido. En este sentido, se proporcionan los resultados correspondientes tanto al modelo auto-regresivo como al LSTM, que son los dos propuestos para tal finalidad.

4.2. Recursos: imágenes y herramientas

Como se ha mencionado previamente, y a falta de imágenes reales para el planteamiento de la solución propuesta, se han utilizado imágenes proporcionadas por Google Maps (2019) con la siguiente referencia, incluyendo el proveedor: Imágenes ©2019 Google, Datos del mapa ©2019 Google, Inst. Geogr. Nacional, España. Son imágenes procedentes de parkings públicos abiertos. Concretamente, se han utilizado dos zonas de parking en la URJC localizados aproximadamente sobre las siguientes coordenadas geográficas $40°20'10{,}7''N 3°52'42{,}5''W$. También se han utilizado otras dos zonas de parking ubicadas en el campus de la Universidad Autónoma de Madrid con localización aproximada en las coordenadas geográficas siguientes: $40°32'38{,}5''N 3°41'39{,}4''W$. A partir de ellas se ha realizado una composición con nuevos elementos en los correspondientes parkings hasta completar un total de 30 imágenes para análisis

Desde el punto de vista de plantear una solución viable en IoT, con las imágenes disponibles para cada parking se crean artificialmente nuevas imágenes que simulan la evolución dinámica del grado de ocupación de los parkings. El objetivo final es crear situaciones variables en el tiempo para simular distintos estados de ocupación de un determinado parking a lo largo del tiempo con la perspectiva puesta en el análisis de series temporales. Aunque, ésta no es la alternativa finalmente adoptada, se plantea como una posibilidad de cara a simulaciones posibles.

Para el procesamiento de los datos se utiliza Matlab R2018b (Mathworks, 2019) con los Toolboxes siguientes: Image Processing, Computer Vision, Deep Learning, Thing Speak Support. Se ha utilizado un procesador Intel Core i7 2.0 GHz (4th generation) con 8 GB de RAM y Debian 9 como sistema operativo de 64-bits.

Por otra parte, para la gestión y procesamiento de la información en la nube se utiliza ThingSpeak (2019), junto con todos sus recursos disponibles.

4.3. Procesamiento de imágenes

Como se ha indicado previamente, el procesamiento de imágenes comprende la segmentación de las mismas, como paso necesario para proporcionar a la CNN los recortes de la imagen original para determinar si su contenido se corresponde con una plaza ocupada por un vehículo o está libre.

En relación a la CNN, las imágenes se procesan atendiendo a las respectivas fases de entrenamiento y clasificación.

4.3.1. Segmentación de imágenes

Mediante el proceso descrito en la sección 2.2, a partir de la máscara definida para un parking en concreto se lleva a cabo un primer paso para determinar el número de plazas disponibles mediante el etiquetado de componentes conexas. El resultado de esta primera fase es el que se muestra en la figura 4.1, donde las plazas de parking disponibles aparecen marcadas en verde, una vez se ha determinado el *bounding box* como parte del procedimiento de segmentación.

Tras el proceso previo se procede a realizar el recorte de cada una de las celdas identificadas sobre la imagen original, las cuales constituyen las entradas a la CNN, tanto durante la fase de entrenamiento como de clasificación.

4.3.2. Entrenamiento de la red

El proceso de entrenamiento se lleva a cabo considerando el modelo de red AlexNet con su capa de salida reorganizada para la clasificación de cuatro categorías de plazas según su contenido: a) ocupada con un vehículo; b) libre con un fondo de asfalto; b) sombras proyectadas por la existencia de árboles y c) árboles que cubren total o parcialmente la plaza.

Figura 4.1: Segmentación de imágenes

Se han utilizado 60 imágenes para cada categoría, redimensionadas al tamaño de $227 * 227 * 3$ tal y como exige la entrada del modelo AlexNet. En la figura 4.2 se muestran ejemplos representativos de cada una de las categorías indicadas. Estos recortes se han obtenido manualmente a partir de las imágenes globales disponibles, tratando de incluir la más amplia variedad de situaciones, tales como vehículos de distintas marcas, categorías, modelos y colores, y ubicados en distintas posiciones (frontal, trasera) con respecto a la situación de la plaza, así como diferentes tipos de asfalto, sombras y árboles con ocupación total o parcial de la plaza. Además, la base de imágenes se ha completado con cincuenta imágenes reales tomadas a diferentes alturas, también redimensionadas, y bajo similares condiciones de perspectiva de la cámara, completando así el conjunto de imágenes procedentes de Google Maps.

(a) Vehículo (b) Asfalto (c) Sombra (d) Árbol

Figura 4.2: Ejemplos ilustrativos de imágenes para el entrenamiento

Son dos los aspectos relevantes a considerar en relación a las imágenes utilizadas para el entrenamiento. Por un lado, el número de imágenes utilizadas de 60 en cada clase fren-

te al aproximadamente 1,2 millones de imágenes utilizadas para clasificar 1000 categorías diferentes en AlexNet. Se trata de un porcentaje insignificante, con el que, no obstante, se consiguen resultados significativos, como se verá posteriormente. Este hecho confirma la validez del modelo propuesto. Cabe reseñar en este sentido, que con otras configuraciones de redes se han obtenido resultados similares, más concretamente con los modelos VGG-16 y VGG-19 (2019) según Russakovsky y col., (2015)[31] y Simonyan y Zisserman (2014)[32], así como con GoogleLeNet (2019)[33] y ResNet-18 (2019)[34]. Por otro lado, tal y como se desprende de la observación de la figura 4.2, la baja resolución en términos cualitativos, de las imágenes de muestra utilizadas no resulta un inconveniente insalvable de cara a la obtención de los resultados.

A continuación se detallan algunos aspectos relativos al proceso de entrenamiento, describiendo y definiendo los parámetros correspondientes al modelo y configuración de la red:

1. *Muestras de entrenamiento y validación.* Indican respectivamente el número de muestras utilizadas para el entrenamiento y la validación, obtenidas a partir del total disponible. En este trabajo, del total de 240 muestras disponibles, el 70 % (168) se utilizan para el entrenamiento y el 30 % para la validación (72).

2. *Epochs e iteraciones.* Representa el paso de todas las muestras disponibles para el entrenamiento y por tanto las 168 inidcadas previamente. Se han establecido 4 *epochs* con 4 iteraciones por epoch, resultando un total de 16 iteraciones.

3. *Precisión en la validación (validation accuracy).* Precisión en el conjunto de imágenes utilizadas para validación, esto es en las 72 imágenes dedicadas a tal fin.

4. *Dimensión por lotes (mini-batch size).* Define la cantidad de imágenes que se utilizan en cada iteración. En este caso se ha fijado a 12.

5. *Razón de aprendizaje (learning rate).* Determina la rapidez con la que la red aprende según el método de actualización de los parámetros que constituyen el objetivo del aprendizaje. Se ha fijado a un valor constante de 10^{-4}. Se han realizado diversos experimentos en este sentido disminuyendo la razón de aprendizaje del orden del 0,2 % cada epoch sin lograr mejoras significativas.

6. *Método de optimización.* Que define el método utilizado para realizar la minimización de la función de coste. En este caso *gradiente descendente.*

La figura 4.3 muestra una etapa intermedia del proceso de entrenamiento de la red CNN, cuando se han ejecutado 8 iteraciones de 16, esto es el 50 %. El proceso completo consta de 4 epochs con 4 iteraciones cada una, hasta completar un total de 16 iteraciones, que se corresponde con el máximo. A medida que se progresa se muestran los resultados correspondientes a la precisión y al ajuste relativo a la función de coste o *loss function.* No se ha fijado ningún límite del tipo patience, que está puesto a infinito. Cuando se fija este parámetro a un valor determinado, lo que realmente se indica es que el proceso se detenga

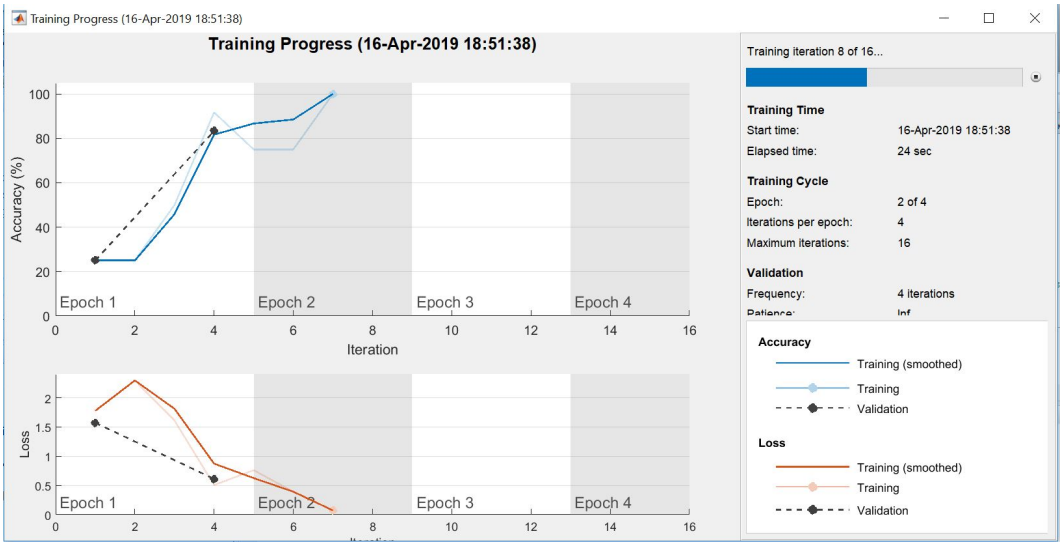


Figura 4.3: Evolución del proceso de entrenamiento de la red CNN

después de que tras el número especificado de *epochs* no se consigan mejoras significativas o relevantes en la precisión.

El proceso de entrenamiento se completa según se indica en la figura 4.4, observándose una precisión en cuanto a validación de las muestras de test del 100 %, lo que permite concluir que el modelo utilizado resulta ciertamente eficiente. No obstante, como se verá más adelante, durante la fase de clasificación todavía se producen errores en la clasificación de las plazas de parking, lo que hace pensar que a pesar de este resultado el modelo es mejorable, desde el punto de vista de la utilización de más muestras de entrenamiento.

En relación a esta misma figura se observa que el tiempo total empleado en el proceso ha sido de 54 segundos con indicación de que la ejecución se ha llevado a cabo en una CPU simple, sin procesamiento en GPU. Se trata de un tiempo razonable bajo las consideraciones especificadas.

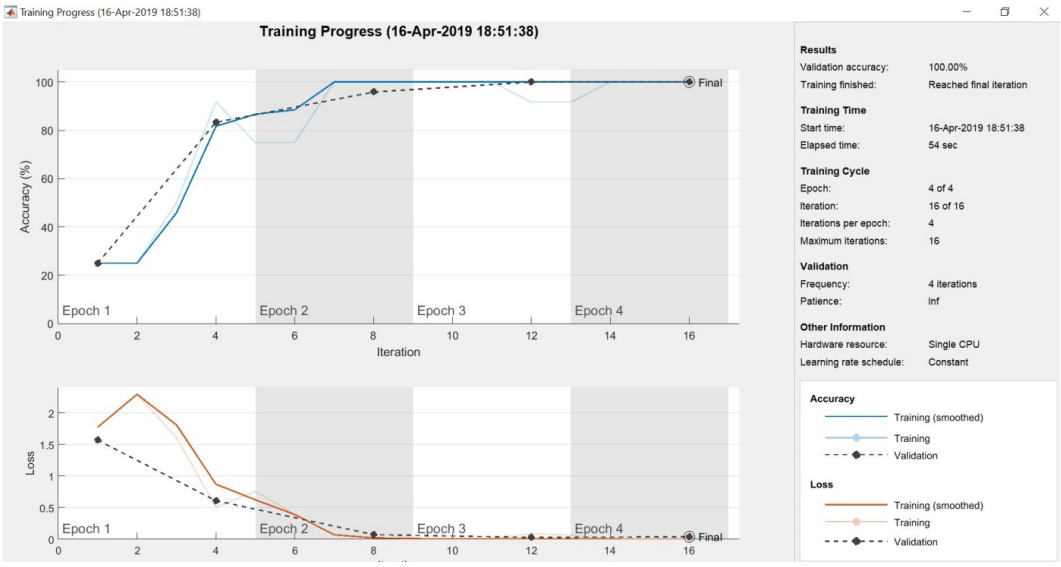


Figura 4.4: Finalización del proceso de entrenamiento de la red CNN

En la figura 4.5 se muestran los resultados obtenidos tras el aprendizaje en la primera capa de convolución (*conv1* en la figura 2.4) para la arquitectura de red del modelo AlexNet. Son 96 filtros en total para esta capa con tres canales (R,G,B), de ahí su visualización en color, tal y como queda especificado en la mencionada figura donde se detalla su dimensión.

Resulta bien conocido el hecho de que este modelado coincide con bastante aproximación con la representación de los denominados filtros de Gabor (Daugman, 1980, 1985; Goodfellow y col., 2016), que simulan el comportamiento biológico desde una perspectiva computacional.

En cualquier caso, conviene reseñar que dicha representación no difiere grandemente de los pesos que la propia red AlexNet posee antes del proceso de re-entrenamiento, lo que significa de alguna manera que los 1,2 millones de imágenes utilizadas han caracterizado suficientemente los pesos de la red original. De hecho, los resultados que se obtienen durante la fase de clasificación con el modelo sin reentrenar llegan a ser aceptables, apreciándose una mejora tras el entrenamiento de aproximadamente tan sólo un 5 %.

Todas las capas de convolución contienen los correspondientes pesos aprendidos, así en la capa de convolución cuatro (conv4) se tienen $3 * 3 * 192 * 384$ pesos, correspondientes a los 384 núcleos representados en la figura 2.4, dos de los cuales son los que se muestran a modo de ejemplo en la tabla 4.1 con sus correspondientes núcleos (K). Obsérvese cómo los valores de los núcleos no tienen un significado interpretativo directo.

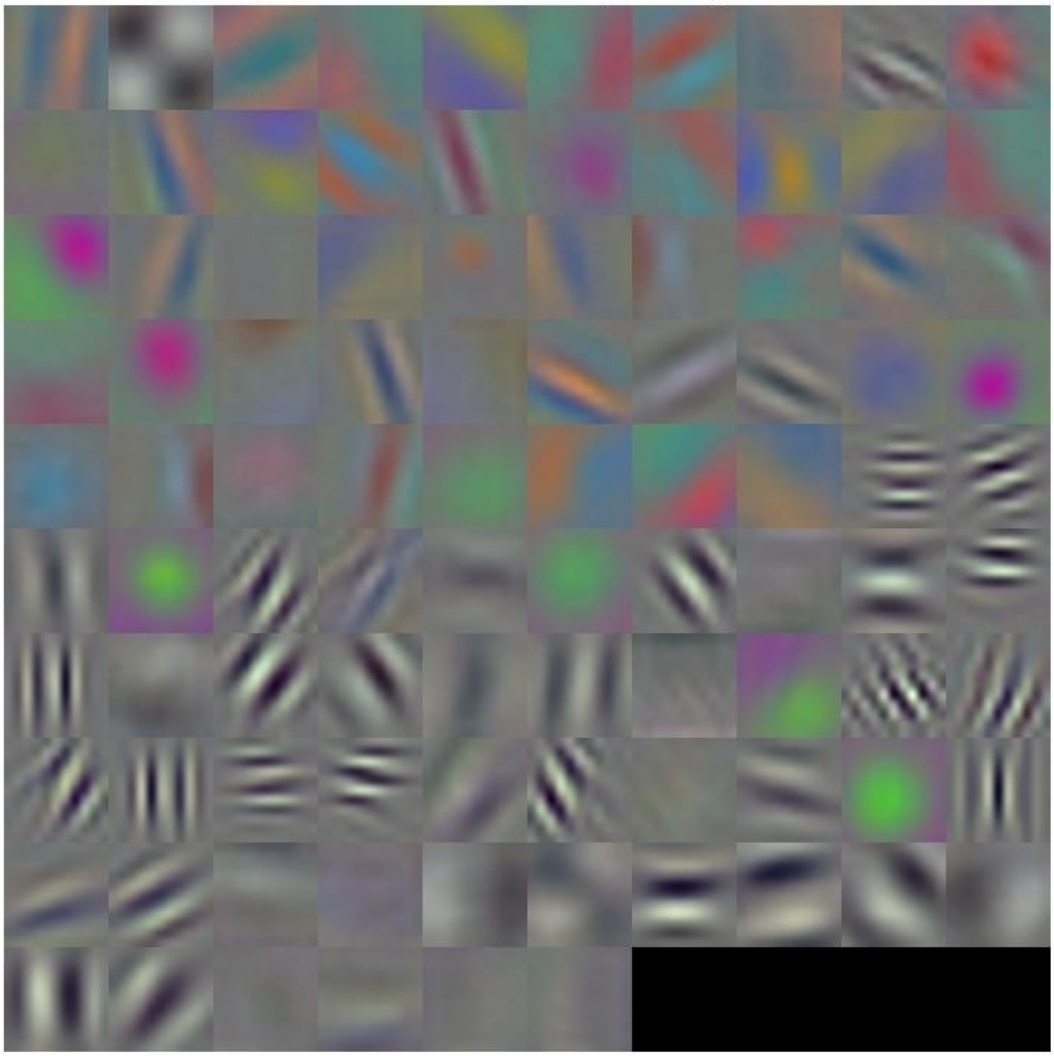


Figura 4.5: Pesos aprendidos en la primera capa de convolución (conv1)

Un aspecto relevante a destacar en relación a este tipo de redes, es el hecho de que los valores de los núcleos de convolución se configuran en función del contenido de las imágenes que se utilizan para entrenar la red. De esta forma no es necesaria su definición previa, al contrario de lo que ocurre en procesamientos clásicos. Esto constituye por una parte una gran ventaja a la vez que un avance considerable en cuanto a la concepción de este tipo de redes capaces de captar las características relevantes de las imágenes.

4.3.3. Clasificación de imágenes

Una vez que se dispone del modelo de red re-entrenado, se recorre la imagen completa seleccionando cada una de las regiones proporcionadas por el procesamiento previo, tal y como se muestra en la figura 4.1. Estas regiones se recortan convenientemente sobre la

Núcleos	Pesos
$K(:,:,12,1)$	$\begin{bmatrix} -0{,}0116 & 0{,}0054 & -0{,}0029 \\ -0{,}057 & 0{,}0027 & 0{,}0021 \\ -0{,}0191 & 0{,}0154 & -0{,}0325 \end{bmatrix}$
$K(:,:,192,3)$	$\begin{bmatrix} -0{,}026 & -0{,}0093 & -0{,}0164 \\ -0{,}0236 & 0{,}0011 & 0{,}0049 \\ -0{,}0088 & -0{,}0178 & -0{,}0101 \end{bmatrix}$

Cuadro 4.1: Pesos aprendidos en la capa de convolución cinco (conv4)

imagen original. Por ejemplo en la figura 4.6 se muestra una imagen de estas características con una resolución en píxeles de $41 * 92 * 3$. Esta misma imagen se redimensiona hasta conseguir las dimensiones de $227 * 227 * 3$ tal y como requiere el modelo AlexNet como imagen de entrada (figura 2.4), observándose cómo cambia la fisonomía del vehículo como consecuencia de tal redimensionado, si bien se preservan las características relevantes tales como bordes, contrastes y por supuesto el color. Son por tanto tales características las que definen de alguna manera la naturaleza de la imagen de cara a su posterior clasificación, que se gestiona a través de las diferentes capas que conforman la arquitectura general del modelo de red.

A medida que esta imagen pasa por las distintas capas de la red se van obteniendo los mapas de características correspondientes. Así a la salida de la primera capa de convolución (*conv1* en la figura 2.4) se obtienen 96 imágenes filtradas, que se muestran en la figura 4.7. A partir de estos resultados se deduce claramente que no todos los filtros producen el mismo resultado, observándose cómo varios de ellos resaltan las características más relevantes de la imagen de entrada, mientras otros no surten efecto alguno o éste es mínimo. Esto sugiere claramente que durante el proceso de clasificación los filtros sin efecto podrían eliminarse, ahorrando el consiguiente cómputo computacional derivado de los filtros inefectivos, lo cual resulta interesante para conseguir una mayor efectividad, particularmente en aplicaciones de tiempo real, donde el tiempo constituye un aspecto relevante. De los 96 filtros mostrados se puede seleccionar el más efectivo, esto es el que presenta el mayor nivel de activación, resultando el que se muestra en la figura 4.8, observándose los diferentes niveles de contraste que caracterizan la imagen.

Avanzando hacia las capas más profundas y llegando a la capa de convolución conv5, los resultados se muestran en la figura 4.9a, observándose cómo en estos 256 resultados producidos por otros tantos filtros, como corresponden a esta capa, no se aprecian detalles significativos. Queda por tanto puesto de manifiesto que es en las primeras capas donde se resaltan los detalles, de forma que a medida que se profundiza en la red, éstos se difuminan. En la figura 4.9b se muestra el resultado obtenido con el filtro más activo del conjunto de los 256. En la figura 4.9c se muestra el resultado obtenido por uno de los filtros en la capa ReLU tras la convolución en la capa 5, esto es relu5, observándose cómo los detalles no son significativos a este nivel.

(a) (b)

Figura 4.6: Imágenes original y redimensionada para la entrada a la red AlexNet

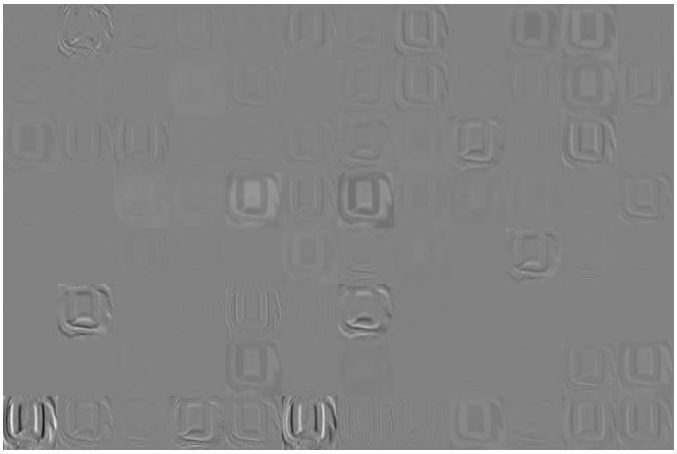
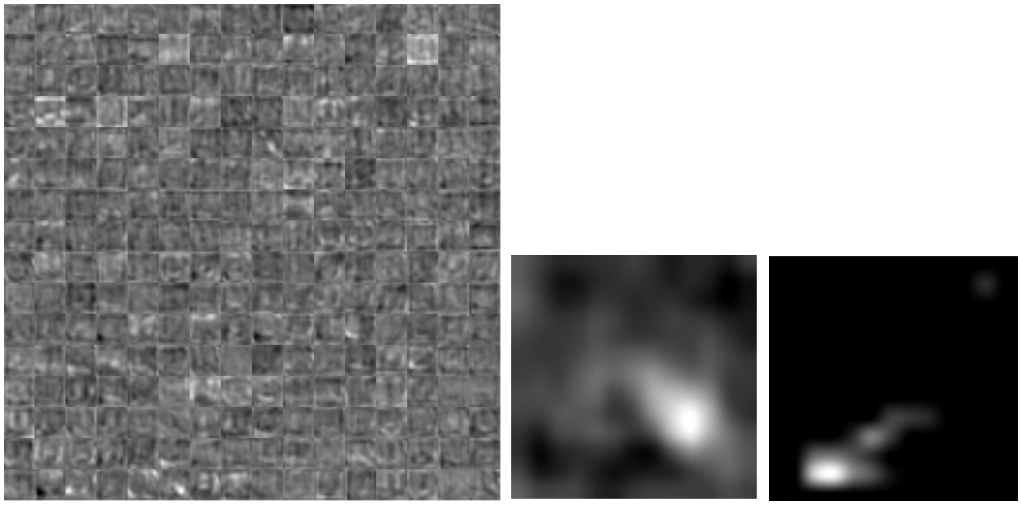


Figura 4.7: Resultados del filtrado con los 96 núcleos de la capa conv1

Figura 4.8: Resultado del canal con el máximo nivel de activación en la capa conv1



(a) (b) (c)

Figura 4.9: Resultados del filtrado con los 256 núcleos de la capa conv5

Durante el proceso de clasificación cada sub-imagen procedente de un recorte se proporciona como entrada a la red de forma que se procede a su clasificación, obteniendo como resultado el porcentaje de acierto tras las salida proporcionada por la función softmax. En la figura 4.10 se muestran una serie de resultados relativos a la clasificación junto con sus correspondientes porcentajes de acierto, expresados en el rango $[0, 1]$ y la sub-imagen que los genera. En la figura 4.10a se obtienen dos porcentajes respecto de las clases Árboles y Asfalto, con una ligera predominancia de la primera, a pesar de que aparentemente no parece ser exactamente el caso representado por la correspondiente sub-imagen. En la figura 4.10b el 100 % corresponde a la clase Asfalto. En la figura 4.10c y la figura 4.10d se muestran los resultados para la clasificación de vehículos, en ambos casos de distintas características y con diferentes porcentajes, siendo que en los dos casos el mayor porcentaje se corresponde con la clase vehículo. Se observa cómo en el segundo caso las características inherentes del vehículo están mejor reflejadas en la red. En 4.10e aparece una distribución de porcentajes con diferentes valores en cada clase sin una distinción clara, en particular en lo que respec-

ta a las clases Asfalto y Sombras, en ambos casos con porcentajes inferiores al 50 %. De hecho la propia imagen resulta ciertamente confusa como se aprecia directamente en ella. Finalmente, en (f) la clase predominante es Asfalto sobre Sombras, estando en ambos casos relativamente equilibradas en porcentaje, lo cual es coherente en relación a la imagen que representa.

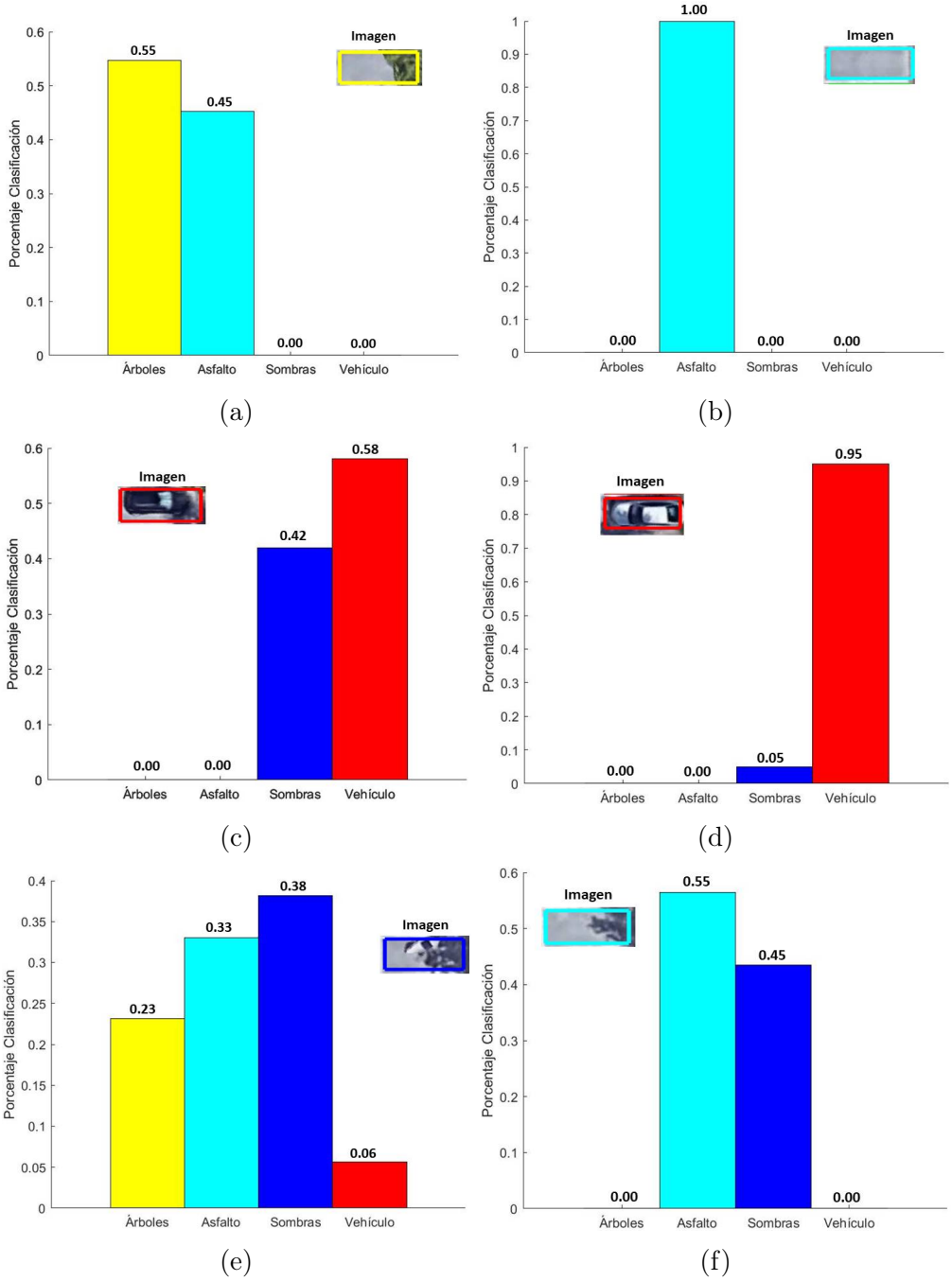


Figura 4.10: Resultados del filtrado con los 256 núcleos de la capa conv5

Dada la imagen segmentada, mostrada en la figura 4.1, se procede a la clasificación de cada una de las sub-imágenes correspondientes a las regiones etiquetadas. De esta forma se obtiene la clasificación global mostrada en la figura 4.11. Donde las distintas clases se identifican por los correspondientes recuadros en color: azul para Sombras, cyan para Asfalto, amarillo para Árboles y rojo para Vehículos. El criterio utilizado para la clasificación consiste en elegir el mayor porcentaje y asignar a la sub-imagen la etiqueta correspondiente. Bien es cierto, que se podría haber considerado que el porcentaje superara un determinado umbral, tal vez el 50 %, si bien al no tener como objetivo el análisis de desempeño de la red se ha optado por el criterio señalado.

Con respecto a la imagen mostrada en la figura 4.11, los resultados obtenidos pueden considerarse aceptables, habiendo conseguido porcentajes de acierto del orden del 96,20 % para el conjunto de las imágenes utilizadas, que ascienden a un total de 20. Así pues en este caso el porcentaje de plazas ocupadas resulta ser del 20,24 % o equivalentemente el 79,76 % de plazas libres, que se corresponde con la existencia de 84 plazas de las cuales 17 están ocupadas por un vehículo (enmarcadas en rojo). El porcentaje de plazas ocupadas, junto con el número de plazas y las plazas libres son los datos que se suben a la nube en los correspondientes campos de ThingSpeak según se han definido en la figura 3.2.

Por otra parte conviene reseñar que el tiempo medio de procesamiento de cada una de las sub-imágenes con las características del procesador reseñado previamente es de 63 milisegundos, de forma que para el caso de la imagen mostrada en la figura 4.11 resulta un tiempo total de proceso de 5.29s.

Figura 4.11: Resultado final de la clasificación

4.4. Series temporales

4.4.1. Generación de datos de la serie

Desde el punto de vista operativo, los datos que configuran la serie temporal deben obtenerse a lo largo del tiempo. Con tal propósito se debe establecer una unidad de tiempo (t) para caracterizar la serie, tal y como se define en la sección 2.4. Como se ha indicado previamente, el planteamiento que se formula en el presente proyecto consiste en analizar el grado de ocupación de un determinado parking en el tiempo. Con tal propósito, y bajo el punto de vista de lo que esto implica en el ámbito de las ciudades inteligentes, se considera ciertamente efectivo el hecho de monitorizar, con el propósito de realizar las predicciones pertinentes, la situación de los parkings en determinados intervalos de tiempo. Resulta bien conocido, al menos en las grandes ciudades, situaciones complejas de tráfico a determinadas horas del día y la semana, particularmente en días laborables. Como quiera que la monitorización del grado de ocupación de un determinado parking estratégico puede, y de hecho, influye en el control y regulación del tráfico por parte de los gestores pertinentes, a la vez que proporciona información valiosa a los conductores que deben transitar por las zonas afectadas, y muchas veces congestionadas, en este trabajo se plantea un estudio bajo esta perspectiva, que sirve de ejemplo para la realización de planteamientos similares teniendo en cuenta otras unidades de tiempo y otras situaciones.

Así pues, con el fin de mostrar la efectividad de la propuesta, se realiza el planteamiento que se explica a continuación. Se trata de monitorizar el estado de ocupación de un determinado parking en una franja horaria de 9 a 10h de la mañana, todos los viernes y durante el último año. Se dispone así de una serie temporal como la definida en la ecuación 2.8 donde las variables de la serie $y_t \equiv y_1, y_2, ..., y_N$ representan el porcentaje de ocupación de un determinado parking en la mencionada franja horaria, con la secuencia temporal, representando las semanas, definida por los sub-índices t según el instante de tiempo, de forma que en el caso objeto de estudio, N es igual a 52, expresando el número de semanas de un año, que constituyen el objeto de observación.

Dado que para el desarrollo del presente trabajo, no se dispone de datos reales con carácter operativo y con cámaras instaladas físicamente en ubicaciones específicas, la secuencia temporal se genera, para un determinado parking, de forma aleatoria hasta conseguir una serie con los 52 datos indicados, que se suben a la nube (ThingSpeak) para su almacenamiento y disponibilidad. Conviene reseñar en este sentido, que el planteamiento realizado es susceptible de variación de forma que la serie puede extenderse en el tiempo, a la vez que cabe la posibilidad de considerar otras unidades de tiempo (horarias, diarias, semanales, mensuales, anuales, etc.).

Dado el carácter aleatorio de los datos generados, éstos pueden variar según el momento de su generación. A modo de ejemplo, en la tabla 4.2 se muestra una secuencia de datos, tal y como se almacenan en la nube con indicación del día y asumiendo que la hora de captura del dato, en todos los casos, es a las 10:00:00 horas, entendiendo que se trata del promedio de ocupación entre las 9 y las 10 horas de la mañana. La figura 4.12 muestra la representación gráfica de los mismos datos de la serie.

Tiempo(t)	%	Tiempo(t)	%	Tiempo(t)	%	Tiempo(t)	%
06-Apr-2018	0,81	06-Jul-2018	0,49	05-Oct-2018	0,74	04-Jan-2019	0,03
13-Apr-2018	0,91	13-Jul-2018	0,80	12-Oct-2018	0,39	11-Jan-2019	0,44
20-Apr-2018	0,12	20-Jul-2018	0,14	19-Oct-2018	0,66	18-Jan-2019	0,38
27-Apr-2018	0,91	27-Jul-2018	0,42	26-Oct-2018	0,17	25-Jan-2019	0,77
04-May-2018	0,63	03-Aug-2018	0,92	02-Nov-2018	0,71	01-Feb-2019	0,0,80
11-May-2018	0,10	10-Aug-2018	0,79	09-Nov-2018	0,03	08-Feb-2019	0,19
18-May-2018	0,28	17-Aug-2018	0,96	16-Nov-2018	0,28	15-Feb-2019	0,49
25-May-2018	0,55	24-Aug-2018	0,65	23-Nov-2018	0,05	25-Feb-2019	0,45
01-Jun-2018	0,96	31-Aug-2018	0,04	30-Nov-2018	0,10	01-Mar-2019	0,65
08-Jun-2018	0,96	07-Sep-2018	0,85	07-Dec-2018	0,82	08-Mar-2019	0,71
15-Jun-2018	0,16	14-Sep-2018	0,93	14-Dec-2018	0,69	15-Mar-2019	0,75
22-Jun-2018	0,97	21-Sep-2018	0,68	21-Dec-2018	0,32	22-Mar-2019	0,28
29-Jun-2018	0,96	28-Sep-2018	0,76	28-Dec-2018	0,95	29-Mar-2019	0,68

Cuadro 4.2: Serie temporal con indicación del tiempo y porcentaje de ocupación

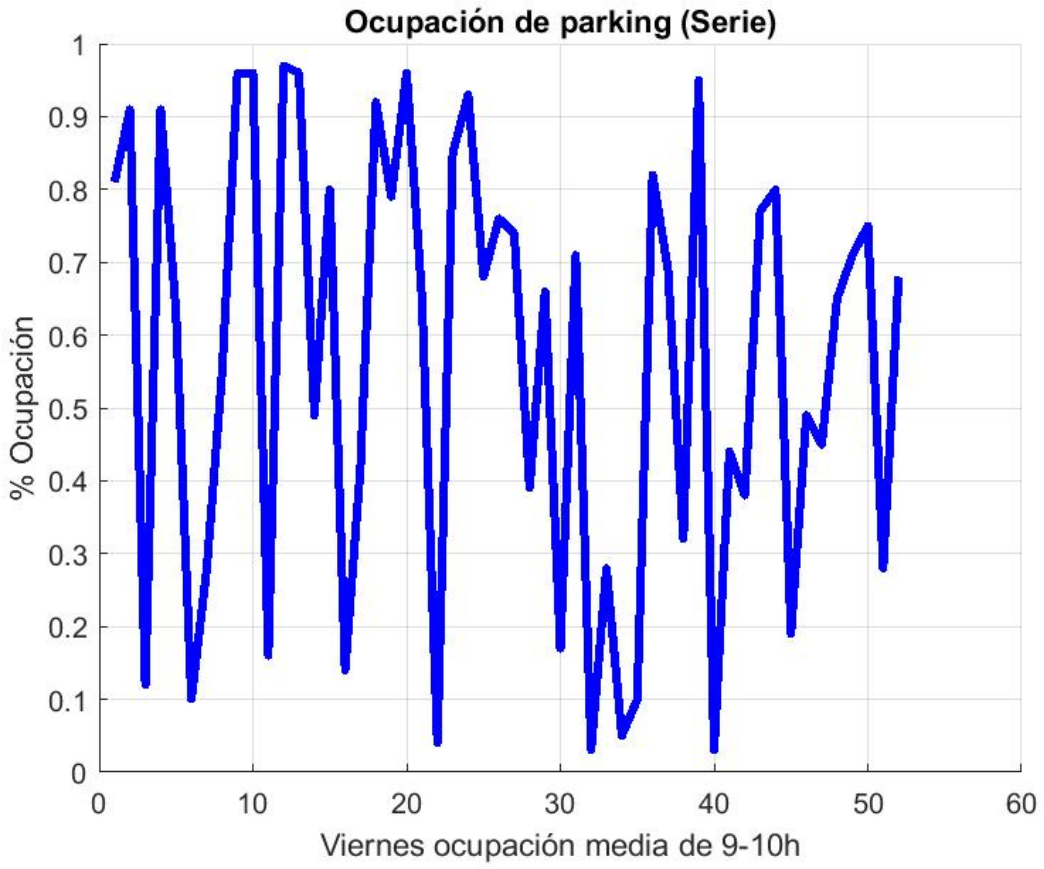


Figura 4.12: Representación gráfica de los datos de la serie

4.4.2. Modelo autorregresivo

Una vez se dispone de los datos convenientemente almacenados en la nube, se procede a estimar los parámetros correspondientes para un modelo $AR(p)$, con $p = 2$. Resolviendo la ecuación 2.11 en la nube se obtienen los siguientes parámetros: $c = 1; \phi_1 = 0{,}4320$ y $\phi_2 = 0{,}4308$, que quedan almacenados en la propia nube.

Con los parámetros estimados y aplicando la ecuación 2.12 para un horizonte h = 10, se predicen (realizando su procesamiento en la nube) los datos que aparecen representados en rojo sobre la figura 4.13, cuyos valores son los siguientes: 0.4144, 0.4719, 0.3824, 0.3685, 0.3239, 0.2986, 0.2685, 0.2447, 0.2214, 0.2010.

En este caso se corresponderían con las predicciones para las 10 semanas siguientes a la finalización de la serie (29-Mar-2019) y a la misma hora.

Dada la aleatoriedad de los datos y por tanto la falta de realidad de los mismos, no es posible deducir consecuencias lógicas respecto de los valores obtenidos mediante la predicción.

Dado el carácter público de los datos y de los parámetros estimados, siempre cabe la posibilidad de descargar los datos de la nube, junto con los parámetros estimados para realizar predicciones en el nodo local.

Figura 4.13: Representación gráfica de los datos de la serie y predicción con h = 10

4.4.3. Modelo LSTM

Considerando los mismos datos de la serie que en el caso anterior, se realiza la estimación según el modelo LSTM definido en la sección 2.4.3. Por tanto, con los 52 datos disponibles se procede a su normalización previa, de forma que la serie tenga media (ν) cero y desviación típica (σ) la unidad, mediante la siguiente expresión,

$$y_n^t = \frac{y_t - \nu}{\sigma} \tag{4.1}$$

De los 52 datos disponibles normalizados, el 90 % se utiliza para el entrenamiento de la red recurrente y el 10 % restante se reserva para la prueba de validación. Los parámetros y características de la red son los siguientes:

1. Número de capas ocultas: 200
2. Número máximo de *epochs*: 500
3. Razón de aprendizaje inicial: 0.05
4. Disminución de la razón de aprendizaje: un factor de 0.2 cada 25 *epochs*
5. Método de optimización: *gradiente descendente*

De esta forma, se inicia el aprendizaje de la red para ajustar los pesos involucrados en la misma, concretamente los que afectan a las entradas que constituyen la serie x_t y h_t, según la figura 2.5. La figura 4.14 muestra un ejemplo de evolución durante la fase de entrenamiento de la red LSTM. Se observa cómo en los primeros pasos se produce una alta variabilidad tanto en lo que respecta a la raíz del error cuadrático medio (RMSE, *Root Mean Square Error*) como a la función *loss*, estabilizándose a partir de la iteración 50 aproximadamente y manteniéndose así hasta el final, tal y como se aprecia en la figura 4.15. En ambos casos la evolución es similar, observándose cómo las dos curvas evolucionan de forma similar. Con los parámetros utilizados y con las 500 iteraciones realizadas se emplean 4 minutos y 48 segundos con un valor RMSE próximo al 0.5, que es un valor relativamente pobre, en términos generales, lo cual viene motivado principalmente por el tipo de datos utilizados y generados de forma aleatoria. No obstante, bajo el planteamiento de la solución IoT que se presenta, la mejora de los resultados vendrían por la utilización de datos reales, no siendo el objetivo del presente trabajo determinar la mejora y comportamiento de la red.

Los datos utilizados para el entrenamiento han sido generados aleatoriamente y cargados en la nube, siendo diferentes a los utilizados en el modelo autorregresivo, para luego proceder a su descarga y al entrenamiento de la red en modo local, así como a las predicciones que también se realizan localmente. Esto es así porque actualmente la herramienta ThingSpeak utilizada para gestionar la nube no admite el procesamiento de datos con este tipo de redes.

En la figura 4.16 se muestran gráficamente los datos de la serie en color azul y la predicción sobre un horizonte h de valor 5, esto es, la predicción sobre las cinco semanas siguientes

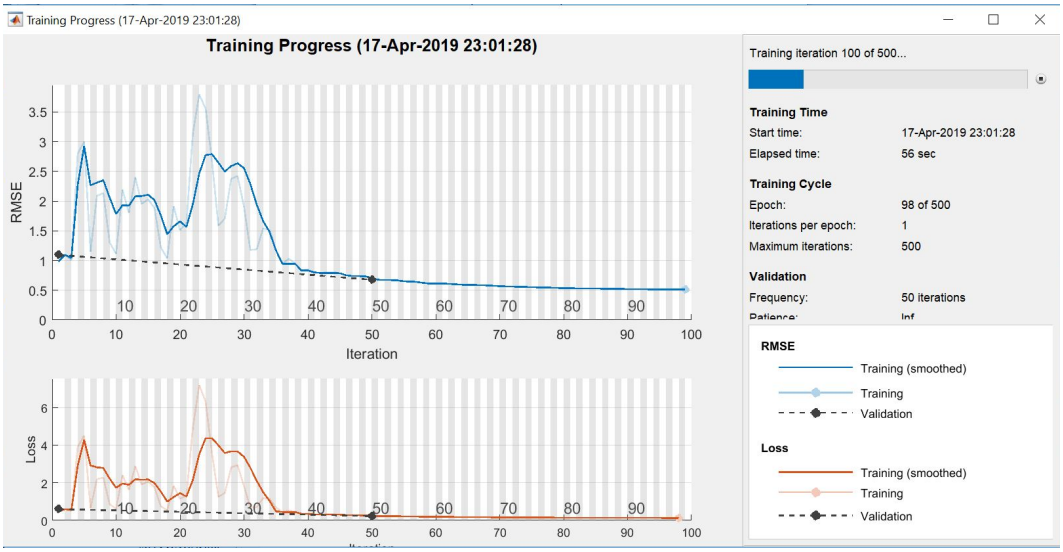


Figura 4.14: Progreso de la evolución durante la fase de entrenamiento

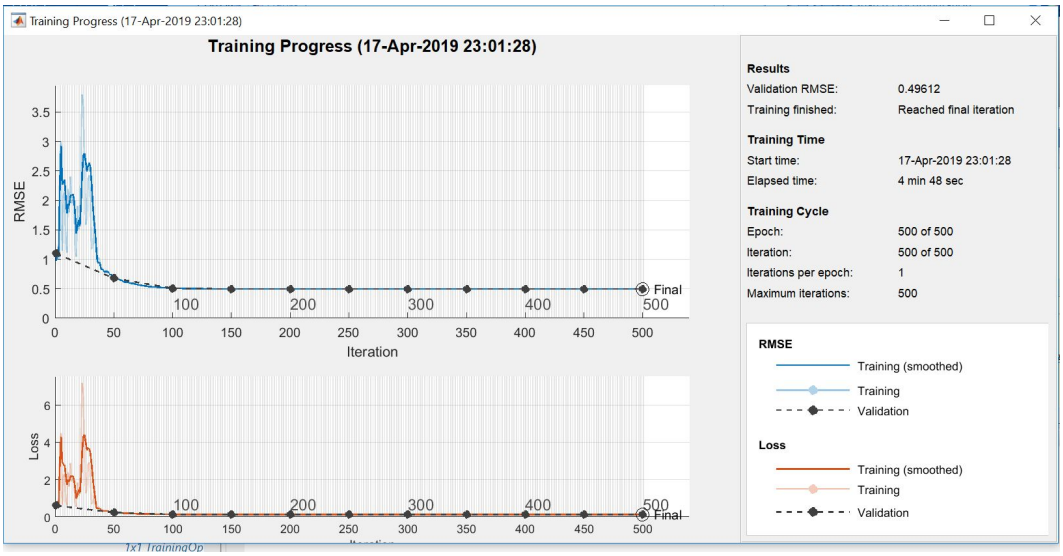


Figura 4.15: Finalización de la fase de entrenamiento

en relación al último dato de la serie. En las representaciones siguientes los datos se encuentran desnormalizados aplicando la ecuación 4.1 a la inversa.

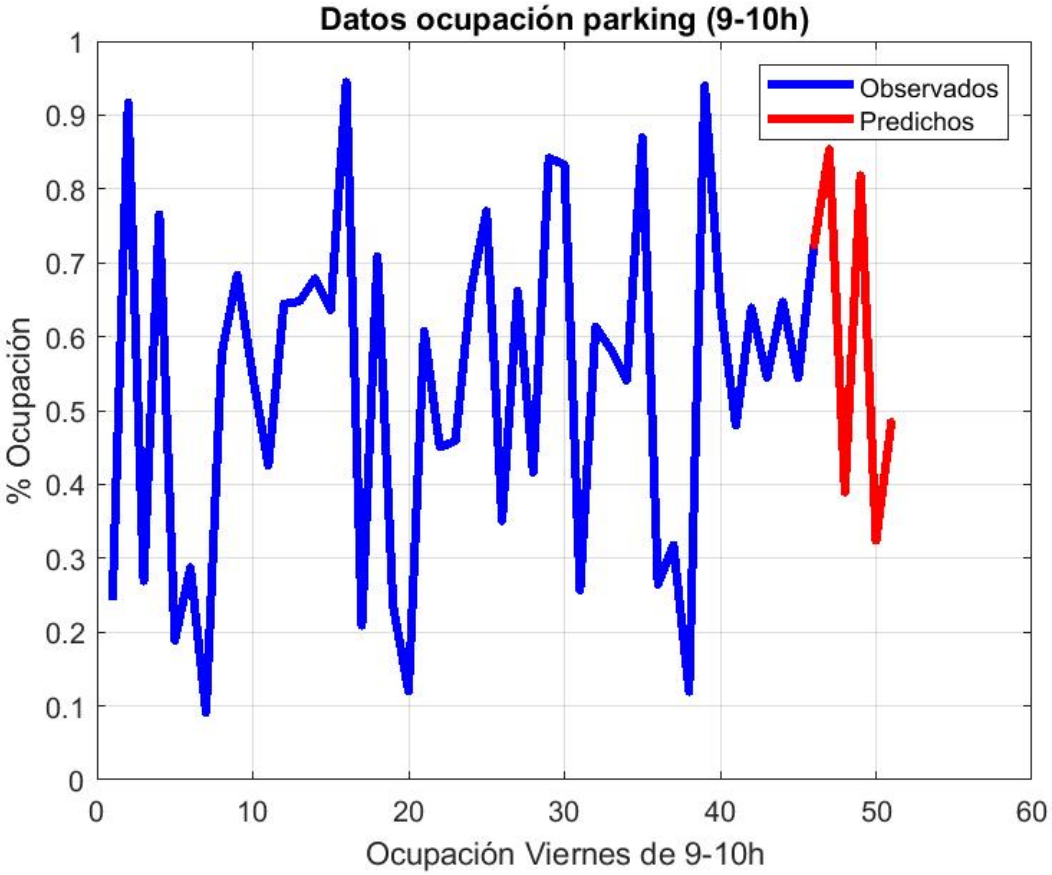


Figura 4.16: Finalización de la fase de entrenamiento

En la figura 4.17 se muestra gráficamente la comparación sobre cinco datos observados frente a los predichos por la serie, comparando las diferencias mediante el cómputo del RMSE. Los datos observados son los cinco últimos de la serie y las predicciones se hacen sobre la base del modelo de red aprendido.

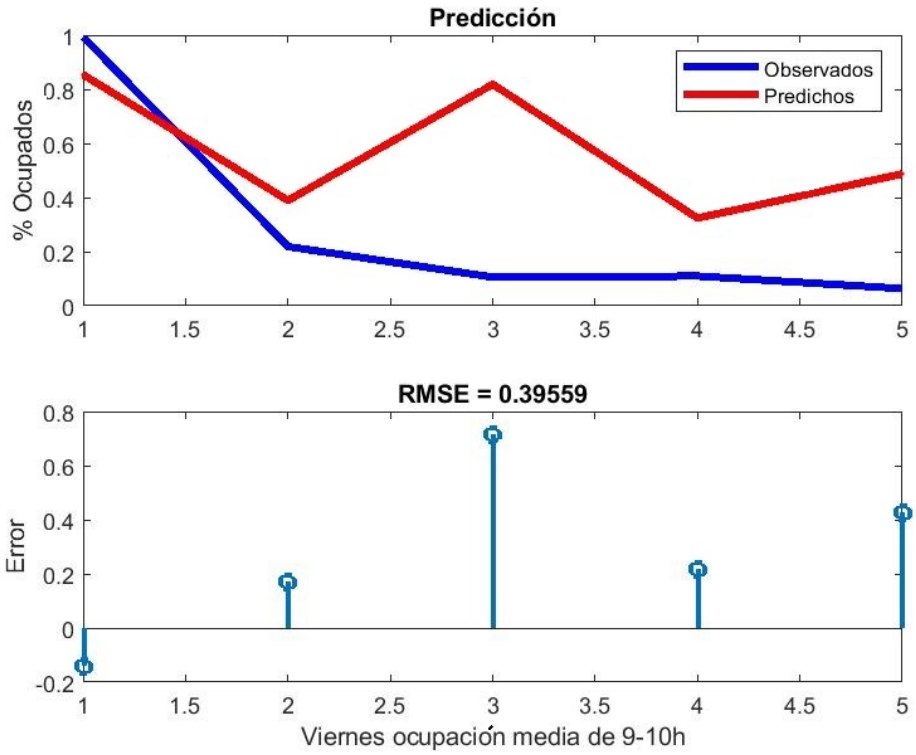


Figura 4.17: Finalización de la fase de entrenamiento

Capítulo 5

Conclusiones y trabajo futuro

En el presente trabajo se proporciona una solución conceptual con la que es posible comprobar las posibilidades de aplicación del paradigma IoT en el ámbito de las ciudades inteligentes, más concretamente para la gestión eficiente de plazas de parking. Se ha realizado una revisión sobre algunas soluciones en el mismo campo, que ya se están aplicando de forma operativa. Básicamente, lo que se propone en estos casos es la instalación de nodos individualizados en cada plaza de parking intercomunicados, que detectan la presencia de un vehículo posicionado sobre el sensor. Como una solución alternativa, en este trabajo se propone el uso de una cámara con capacidad de observación del parking completo o una gran parte de él, en cualquier caso la imagen capturada por la cámara incluye el máximo número de plazas posibles.

Las ventajas que ofrece este sistema se pueden sintetizar como sigue:

1. Con un único sensor se pueden visualizar y gestionar varias plazas de parking.

2. Además de la gestión del parking, el mismo sensor se puede utilizar para otras finalidades, tales como análisis del tránsito peatonal en el parking o para video-vigilancia.

3. La solución sensorial no exige un coste añadido importante a la hora de implantar el sistema, ya que incluso se puede pensar en el uso de tecnologías baratas, tales como *webcams*, sin más que especificar resoluciones de tamaño y calidad suficientes, pero sin demasiada exigencia computacional.

4. Al hilo de lo anterior, se ha demostrado que incluso con imágenes de calidad no necesariamente alta, los resultados del análisis de las imágenes, mediante la CNN, son satisfactorios.

El diseño arquitectónico realizado ha consistido en el uso de la cámara como sensor (cámara operando en el espectro visible), que envía datos a un servidor local para el análisis de las imágenes. Con tal propósito se ha propuesto una estrategia basada en deep learning, más específicamente utilizando una CNN pre-entrenada y re-entrenada (con ejemplos concretos de parkings con plazas ocupadas y libres) para adaptarla al caso concreto de determinación de las plazas de parking libres en un momento determinado y a lo largo del tiempo. Se ha

desarrollado una estrategia basada en técnicas de visión por computador que identifica la ubicación de las plazas de parking, las etiqueta y realiza el correspondiente recorte sobre la imagen original con el contenido de las plazas en un momento determinado con identificación del instante de tiempo que le corresponde. Estos recortes son los que se proporcionan a la red para su clasificación en una de los cuatro tipos de plazas especificadas (vehículo, asfalto, sombra, árbol). La red proporciona resultados suficientes para concluir que su uso está suficientemente justificado. El resultado del análisis de cada imagen se publica en la nube, con carácter público, con indicación de la fecha de obtención de dicho resultado.

Como parte de la solución se propone la generación de series temporales de datos sobre la ocupación de un parking en un momento determinado, más específicamente a lo largo de un año, esto es 52 semanas, marcando como unidad de tiempo la semana, de forma que todos los lunes a las 10h se obtiene el promedio de ocupación durante la hora precedente. Se han diseñado dos procedimientos para el análisis de las series temporales, uno basado en el modelo de auto-regresión de orden p y otro mediante LSTM.

Los datos correspondientes a la serie temporal se suben a la nube para su almacenamiento y visualización desde cualquier dispositivo con acceso a la nube, siendo internet uno de los más habituales. Con tal propósito se ha diseñado una estructura soportado por la plataforma ThingSpeak de carácter público, que además permite el procesamiento de datos en la nube. De hecho en esta plataforma se lleva a cabo la estimación de los parámetros correspondientes al modelo de auto-regresión (AR), cuyos valores quedan también almacenados en la nube. La disponibilidad de los datos en la nube con el mencionado carácter público permite su análisis en cualquier nodo, previo acceso y descarga de los mismos, en el presente trabajo esta funcionalidad se ha desarrollado mediante LSTM. De esta forma se cierra un ciclo completo para la solución IoT propuesta.

Respecto de las acciones de futuro identificadas durante el desarrollo del presente trabajo, cabe mencionar las siguientes:

1. Como se ha indicado previamente, las imágenes proceden de Google Maps, dada la imposibilidad de implantación de un sistema real en ubicaciones reales, por lo que como una acción clara sería la implantación de un sistema real para saltar del concepto a la realidad.

2. Siguiendo con la realidad del sistema, otra de las cuestiones requeridas es la captura de datos reales secuenciados en el tiempo para evitar la generación aleatoria de los mencionados datos.

3. Un análisis más en profundidad sobre nuevos modelos de redes CNN o de análisis de series temporales queda también pendiente como acción de futuro con el fin de determinar el mejor desempeño en cada caso.

4. El trazado de líneas de forma automática mediante la transformada de Hough, tal y como se describe en la sección 2.2, queda igualmente pendiente como una acción de futuro. De esta forma se podría determinar el alcance de la propuesta en entornos de exterior, donde el procesamiento de imágenes resulta ciertamente complejo.

Finalmente, en la línea de lo anterior cabe plantearse la posibilidad de señalizar las plazas de parking con algún distintivo identificador, con el fin de aplicar posteriormente técnicas específicas basadas en reconocimiento de color o de agrupamiento, también en el ámbito del aprendizaje automático. Un ejemplo sobre esta posibilidad se muestra en la figura 5.1.

Figura 5.1: Finalización de la fase de entrenamiento

Capítulo 6

Introduction

6.1. General Approach

The boost of low performance, cost and consumption devices, along with the advancement of technologies that allow those devices communicate between them and with remote systems in an efficient and reliable manner, have promoted the emergence of a solutions model based on processing information obtained locally by various sensory technologies (i.e. cameras as the source of information for the present work), in intelligent systems, which generally manage data from several devices in a centralized way. Moreover, they are able to combine them with each other or with others from different sources. They are also responsible for publishing all the data collected for general access. This model of solutions is known as Internet of Things (IoT).

Smart City Solutions, which belong to IoT, are those designed to be deployed within a urban context. They may improve the services Public Administrations offer as well as add new ones, improving citizens quality of life. Smart Parking Solutions, which are present in several Smart Cities Solutions, use sensors in parking areas in order to monitor and forecast occupation average among other things.

The Prove of Concept posed, consists in taking images with cameras (during sunlight time), processing this images in order to extract occupation average, warning traffic control services, and particular drivers when certain conditions are accomplished (i.e. some occupation rate), performing predictive analytics of future occupation levels, and publishing current and forecasted occupation information.

IoT in Smart Cities

Smart City Solutions are based on three points:

- Gathering information from the city
- Analyzing gathered information
- Taking some action

[1]

Smart Cities rely on IoT. Sensorization and municipality resources allow monitoring the city and acting accordingly. For example city's traffic can be measured, and in case of clogging adjust the semaphores remotely.

The law 7/1985 acknowledges a list of municipal competences, some of them can be included in the concept of Smart City.[2]

- *Security in public places*
- *Traffic and Pedestrians management*
- *Civil protection, prevention and firefighting*
- *Planning, management, execution and urban discipline; housing promotion and management; parks and gardens, paving of urban public roads and conservation of roads and rural roads*
- *Historical-artistic heritage*
- *Environmental Protection*
- *Supplies, slaughterhouses, fairs, markets and defense of users and consumers*
- *Protection of public health*
- *Participation in the management of primary health care*
- *Cemeteries and funeral services*
- *Provision of social services and promotion and social reintegration*
- *Water supply and public lighting; street cleaning services, waste collection and treatment, sewerage and wastewater treatment*
- *Public Transport*
- *Cultural or sports activities or facilities; occupation of free time; tourism*
- *Participate in the programming of education and cooperate with the Educational Administration in the creation, construction and maintenance of public educational centers, intervene in their management bodies and participate in the monitoring of compliance with compulsory schooling*
- *Promotion of equality between men and women*

Improve offered service
The main goal of Smart Cities Projects is to improve servicces offered to citizens.[3] These are some examples:

1. **Environmental Protection**: IoT facilitates to know pollution (both atmospheric and acoustic) level in different pooints of the city. Based on information collected by sensors, traffic of private vehicles may be restricted, as well as speed limits may be established, or work hours for noisy works may be limited.

2. **Parks Management**: Parks' ground humidity level and current temperature can be measured and consequently activate watering systems.

3. **Parkings Mangement**: As this work describes, monitoring of parking places allows forecasting occupation level and provides useful information to drivers in order to find free parking spaces.[4]

4. **Asset Management**: Assets such as vehicles, computers, offices, sport facilities, light equipments and semaphores, are used by municipal Administrations in order to perform it's activity. Sensorization allows to keep them located and to monitor their functional state. For example vehicles' performance may be measured in order to avoid faults as well as to apply preventive maintenance policies.[5]

Control Service Concessions
Several of these services are provided by alien means sewage treatment in terms of concessions. For example garbage collection, watering and cleaning of public parks and sewage treatment. Others such as security in public places are usually outsourced to private companies.

For services delivered by third parties, Smart City Projects make possible to monitor concessionaires in orther to check whether they fulfill the agreed terms and conditions. For example, sensorizing vehicles used for watering in public roads enables georeferencing them, and therefore monitoring where and for how much time watering service has been provided. Garbage collecting is a similar case.

6.2. Alternative solutions

Solutions for the same problem have been proposed, from different perspectives, such as those deployed in Santander and Malaga in Spain and Montpellier in France. In these cases deployed Solution is a part of converting the city in a Smart City with an ecosystem of services.

Details of these solutions are listed bellow.

Montpellier

Montpellier is a city of the south of France with a population of 2227.000 (according to 2015).

Supported by the french company Synox(2019)[6], the intermunicipial structure Montpellier Mediterranean Metropollis has deployed a Solution using as nodes devices of Libelium(2019) known as Waspmote(2019). Taking advantage of the establishment of a private metropolitan network based on *LoRaWAN* to give coverage to city's IoT projects. Along the surface of the road, a net of sensors which are able to detect whether a vehicle its parked over them or not has been installed, and to communicate it through the private network with other information of general interest like temperature. Such information can be consulted in public channels using a mobile or a web application. Figure 1.1 shows the Dashbord GUI of the web application.

Malaga

Malaga is a city of 571.026 citizens (2018) located at the south-west of Spain. Town Hall has bet for Open Data, among other data, occupation data sets of 9 public underground parking lots has been published. Using this data, the group Neo (Network Emerging Optimization) (2019), a research group integrated in the Universidad de Malaga, presented a propsal in a initiatives for open data utilization competition. Proposal was selected, and sensors were instaled in public parking places. During 250 days measurements were taken and forecasts based on temporal series were performed. At this time a prototype is available at the web that allows to know the probability of finding free parking places in each of the 9 parking lots. Figure 1.2 shows a caption of the prototype.

Santander

Santander is a city with a population of 172.000 according to 2017, located in the north of Spain. during that year a Smart Parking system was implanted in the city. The project Smart Santander [10] is a city-scale research study of typical Smart Cities' services and applications support. A part of it, consists on monitoring available parking places over the city.

City was divided in 27 areas, each one provided with a gateway that stores measurements up to 5 different kinds of sensors: temperature, brightness, carbon monoxide, sound and occupation of parking spaces on the public road.

Parking spaces occupation is sensorized by 2.000 devices located under the ground, which measure magnetic field using ferrmagnetic technology. The Universidad de Cantabria participated in the project which uses Over the Air Programming (OTAP) for programming the nodes remotely. Like in Montpellier's Solution, nodes area WaspMote (2019) devices.

As figure 1.3 shows, in this case in addition to being able to access to information using computers or mobile devices, information of free parking places is shown in panels along all the city indicating the amount of free places and in what direction they are.

Whereas the solutions just explained are based on the use of several connected nodes in order to monitor each place, the solution proposed in this work suggests to use a single

devices for each parking area (sevaral places, in this case a camera. It also offers the chance of using these images for other purposes such as surveillance.

Smart Parking Management is in continuous evolution. Several offers and chances are emerging, such as those offered by Nwave and IntelVision (2019), that besides of sensory technologies connected with 5G solutions[12] offers massive processing alternatives within Big Data scope.

6.3. Motivation

Likewise to how it is understood that the technological context has never been so propitious for the success of IoT solutions, it can be observed that the levels of pollution in certain cities, as the proportion of the time of their life that a person spends in traffic jams, or looking for parking, as well as the number of people traveling at certain times of the day, are also at historical maxima.

According to studies conducted by Parking Network, IBM and Cisco related to parking places, growth of fuel consumption of Spanish drivers (by an average of 16 %), and traffic jams (by an average of 30 %) are caused by drivers seeking free parking places.[14] Some cities like Madrid, restrict traffic or parking on public roads, in some areas, at certain times of the day, depending on pollution levels.

Smart Parking Solutions represent a clear benefit for the citizen who can go directly to the car park where knowing that a place will be available, saving time and fuel whereas minimizing polluting effects.

For the whole city , the main advantage is that there are fewer cars in circulation, causing pollution reduction and traffic improvement.

Municipalities are provided with very valuable information that can be used to find out areas of the city experiencing shortages of parking places. This information could be considered for taking decissions like investing in public parking lots, reinforcing public transports, or allowing shared transport companies to operate in certain areas.

From a technological point of view, motivation is aroused by the opportunity that the concept of IoT offers in order to ease information relative to parking lots occupation rate smart management. This approach is based on technological solutions already developed or ideas for the future.[15][16]

6.4. Objetives

The main goal of this work is to implement an IoT Solution for parking places management.

6.4.1. General Objectives

1) Application of advanced image processing and classification techniques

The Solution considers the local inclusion of a smart system that performs image processing through Computer Vision techniques in order to identify parking places. Followed by an identification process in order to recognize whether each place is busy or free. For such recognition the use of Convolutional Neuronal Netwokes is suggested. Obtained results, concerning to the occupation rate recognition are published in the cloud. This approach is framed next to the paradigm of Edge Computing.

The Solution considers the local inclusion of a

2) Occupation rate forecasting through temporal series analyasis

Using data published in the cloud, forecasting analysis of the occupation rate of each parking in function of a time unity conveniently established is performed.

6.4.2. Specific Objectives

These are the specific objectives of the present work:

1) Review existing IoT Solutions

Most of IoT Solutions have some common components:

- One or more sensors, usually of low cost, that measure any information useful for the system. The use of devices installed at every place, that detect whether a car is parked or not, has been identified, this is the case of WaspMote (2019) devices, used in Montpellier and Santander. As an alternative to this solution, this work proposes the use of cameras that take images of each parking, and the further analysis of captured images.

- A processing unit, local to at least one sensor from which it ingests the information the sensor measures. This information is provided to the system. Sometimes, this component also applies some kind of pre-processing to the information before sharing it. For the present study, the use of a laptop, in order to process images taken from parking lots, and the use of Computer Vision techniques has been chosen. Image classification through CNN based techniques is performed in this same host. Pre-trained models like AlexNet (ImageNet 2019), which are later re-trained according to project's needs, are used. This real time classification could be done in a service by XaaS means, easing the use of lower capacity devices. This enables a possible future expansion.

- One or more remote cloud based services, that take advantage of the collected information, either raw or processed. The cases mentioned before relative to the use of this

kind of services for performing the classification and predictive study, as well as for the remote forecasting study, remotely performed, of temporal series of occupation rate, are examples of these kind of components.

2) Design solution's architecture, including local and remote modules

Among the two most typical alternatives for solutions of type textit SmartParking, mentioned in the previous point, a local approach is proposed for the processing of images and classification by CNN, with pre-trained models of the AlexNet type, given Its excellent and proven performance. The data processed over time is sent to a remote service (cloud) for analysis of time series with prediction of occupancy of parking places. The estimated parameters of the series together with the data of the series itself are available in the cloud for consultation and download. This allows local predictions.

3) Apply image segmentation techniques focused on extracting regions

For each chosen parking, base templates are created. They contain chromatically differentiated regions that correspond with expected parking areas. Section 2.2 explains techniques used for getting the templates. Core application uses the templates in order to identify those regions which are individually classified as busy or free.

4) Apply Convolutional Neuronal Network techniques

Pre-trained CNN, Alexnet, is re-trained in order to fix convolutional layers weights through correspondent re-trainment using images of parked vehicles and free parking places.

Each labeled region is classified and indentified as busy or free. Later the amount of places classified as busy is counted and published to a remote service.

5) Develop a cloud hosted predictive analysis system

Temporal series analysis allow forecasting of parking places occupation averages. Predictive analysis through techniques based on an autoregressive model are planted remotely. Estimated parameters remain available in the cloud. Downloading them and using obtained data, local forecasting is performed, the local use of forecasting techniques based of Recurrent Neuronal Networks of the type LSTM (Long Short-Term Memory) has also been planted.

6.5. Memory's Structure

Memory is organized in chapters. Chapter 2 describes methods and techniques relating to image processing. Chapter 3 describes raised IoT Solution. Chapter 4 presents obtained results. Chapter 5 contains general conclusions and future work.

Capítulo 7

Conclusions and Future Work

In the this work a conceptual solution is provided with which it is possible to verify the possibilities of application of the IoT paradigm within the scope of smart cities. More specifically to find out whether it can be used for the efficient management of parking spaces. A review has been made of some solutions in the same field, which are already being applied operationally. Basically, what is proposed in these cases is the installation of individualized nodes in each intercommunicated parking space, to detect the presence of a vehicle positioned near (over) the sensor. As an alternative solution, this work proposes the use of a camera with full observation capacity of the parking lot or a large portion of it. In any case the image captured by the camera includes the maximum number of parking places.

The advantages offered by this system are the following:

1. Several parking places can be managed with a single sensor.

2. In addition to parking's management, a sensor may be used for other purposes such as pedestrian transit analysis

3. The proposed sensory solution does not involve an additional excessive cost when implementing or scaling the system. Low cost technologies like webcams can be used. Only certain conditions of image resolution size and quality are needed.

4. It has been shown that even with images of non high-quality, the results of the analysis of the images, through CNN, are satisfactory.

Architectural design consists in the use of a camera, which sends data to a local server where images are analyzed. With such purpose, a pre-trained CNN has been retrained with samples of both free and busy parking places, in order to fix it to the scope of this work. Computer Vision techniques have been used in order to identify the position of the expected parking places in the image. Places are labeled and correspondingly trimmed at original image, which consists in a capture of the situation of the parking at a given time. Trims are provided to the net for their classification. The network provides sufficient results to

conclude that its use is sufficiently justified. The result of the analysis of each image is published along with the date when the results were obtained.

The generation of temporal series of data relating to the occupation rate of a parking in a certain moment is proposed as part of the solution. Each Monday at 10 a.m. , the occupation rate during the previous hour is obtained. Two procedures have been designed in order to analyze temporal series, one based on auto-regressive order p and other through LSTM.

Data corresponding to temporal series is uploaded to the cloud, where it is stored and remain available for it's visualization by any device with Internet access. The estimation of the parameters relative to auto-regressive (AR) model is performed within the public cloud platform Thinkspeak. Estimation parameters remain also available in the cloud, This allows it's analysis from any node, previous access and download. In the present work this functionality has been developed using LSTM. This completes the cycle of the proposed IoT Solution.

Regarding future actions identified during the development of this work, the following should be mentioned:

1. As previously indicated, the images come from Google Maps, given the impossibility of implementing a real system in real locations, Therefore clear action would be the implementation of a real system to jump from the conceptual to the operational.

2. Following the reality of the system, another of the required questions is the capture of real data sequenced in time to avoid the random generation of the aforementioned data.

3. A more in-depth analysis of new models of CNN networks or analysis of time series is also pending as a future action in order to determine the best performance in each case.

4. Line drawing automatically using the Hough transform, as described in the section 2.2, is also pending as a future action. In this way, the scope of the proposal could be determined in outdoor environments, where the processing of images is certainly complex.

Finally, in line with the above, it is worth mentioning the possibility of signaling the parking spaces with some identifier, in order to subsequently apply specific techniques based on color recognition or grouping, also in the field of machine learning. An example of this possibility is shown in figure 5.1.

Bibliografía

[1] Red.es, Ciudades e islas inteligentes. https://www.red.es/redes/es/que-hacemos/ciudades-inteligentes/plan-nacional-de-ciudades-inteligentes. Accessed: 2019-4-10.

[2] Ley 7/1985,de 2 de abril, Reguladora de las Bases del Régimen Local. https://www.boe.es/buscar/pdf/1985/BOE-A-1985-5392-consolidado.pdf. Accessed: 2019-4-10.

[3] ElPais Digital, Smart Cities. http://www.paisdigital.org/PD/smart-cities/. Accessed: 2019-4-11.

[4] IBM, Acuerdo parques Barcelona area metropolitana. https://www-03.ibm.com/press/es/es/pressrelease/51446.wss. Accessed: 2019-4-11.

[5] Mint, Madrid inteligente. http://www.madridforyou.es/mint-madrid-inteligente. Accessed: 2019-3-10.

[6] Synox (20019). Synox: innovate togheter. Disponible on-line:. https://www.synox.io/. Accessed: 2019-03-28.

[7] 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/. Accessed: 2019-03-08.

[8] Open Data Malaga. https://datosabiertos.malaga.eu/. Accessed: 2019-4-25.

[9] Prototipo App Malaga. http://mallba4.lcc.uma.es/parking/. Accessed: 2019-4-25.

[10] Smart Santander. http://www.smartsantander.eu/. Accessed: 2019-4-12.

[11] Al-Turjman, F., Malekloo, A. (2019). Intelligent Parking solutions in the IoT-based Smart Cities. In Intelligence in IoT-enabled Smart Cities (Al-Turjman, F., ed), CRC Press, Taylor and Francis Group. Boca Raton, FL.

[12] Alfa, A.S., Maharaj, B.T., Ghazaleh, H.A., Awoyemi, B. (2018). The Role of 5G and IoT in Smart Cities. Handbook of Smart Cities Software Services and Cyber Infrastructure (M., Badidi, E. Maheswaran, eds.). Springer.

[13] The Future Of Smart Parking. https://www.nwave.io/?gclid=CjwKCAjwkcblBRB_EiwAFmfyy3m68e_npMC216XRG96t0IoMzdk-opbxG-226LBbYEIYwcvrRw258RoCf20QAvD_BwE. Accessed: 2019-04-14.

[14] Smart Parking, la medida que no puede faltar en una Smart City. https://www.eoi.es/blogs/merme/smartparking/. Accessed: 2019-03-10.

[15] The Digitalization Transformation in the Digital Economy. http://www.infiniteinformationtechnology.com/iot-smart-city-what-is-smart-parking. Accessed: 2019-04-14.

[16] Sierra Wireless - IoT Blog. https://www.sierrawireless.com/iot-blog/iot-blog/2017/09/smart_city_parking_harnessing_the_power_of_iot_to_make_urban_life_easier/. Accessed: 2019-05-14.

[17] Pajares, G.; de la Cruz, J.M (2007). Visión por Computador: imágenes digitales y aplicaciones. RA-MA, Madrid.

[18] Haralick, R. M.; Shapiro, L.G. (1992). Computer and Robot Vision, Volume I, Addison-Wesley, 1992, pp. 28-48.

[19] Deep Learning with Tensor Flow. https://www.youtube.com/watch?v=SSoY8uQwDW4. Accessed: 2019-3-10.

[20] Daugman, J.G. (1980). Two-dimensional spectral analysis of cortical receptive field profiles. Vision Res., 20(10), 847–56.

[21] ImageNet, 2019; Russakovsky y col., 2015; Krizhevsky y col., 2012; BVLC AlexNet Model, 2019.

[22] Krizhevsky, A., Sutskever, I., Hinton, G.E. (2012) ImageNet Classification with Deep Convolutional Neural Networks. In Proc. 25th Int. Conf. on Neural Information Processing Systems (NIPS'12), vol. 1, pp. 1097-1105.

[23] ImageNet, 2019; Russakovsky y col., 2015; Krizhevsky y col., 2012; BVLC AlexNet Model, 2019.

[24] Mauricio, J.A. (2007). Introducción al Análisis de Series Temporales. Universidad Complutense de Madrid. Disponible on-line:. https://www.ucm.es/data/cont/docs/518-2013-11-11-JAM-IAST-Libro.pdf/. Accessed: 2019-4-22.

[25] Walker, G. (1931). On Periodicity in Series of Related Terms, Proc. Of the Royal Society of London, Ser. A, Vol. 131, 518–532.

[26] Yule, G. U. (1927). On a Method of Investigating Periodicities in Disturbed Series, with Special Reference to Wolfer's Sunspot Numbers, Philosophical Transactions of the Royal Society of London, Ser. A, Vol. 226, 267–298.

[27] Hochreiter, S., Schmidhuber, J. (1997). Long short-term memory. Neural Computation, 9(8), 1735–1780.

[28] Graves, A. (2013) Generating sequences with recurrent neural networks, Computer Science.arXiv:1308.0850.

[29] Gers, F., Schraudolph, N., Schmidhuber, J. (2002). Learning precise timing with LSTM recurrent networks. Journal of Machine Learning Research, 3,115–143.

[30] Olah, C. (2019). Understanding LSTM Networks. Disponible on-line:. http://colah.github.io/posts/2015-08-Understanding-LSTMs/. Accessed: 2019-4-21.

[31] Russakovsky, O., Deng, J., Su, H. Krause, J., Satheesh, S., Ma, S. Huang, Z., Karpathy, A., Khosla, A., Bernstein, M., Berg, A.C., Fei-Fei, L. (2015). ImageNet Large Scale Visual Recognition Challenge.International Journal of Computer Vision (IJCV). 115(3), 211–252.

[32] Simonyan, K., Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.

[33] GoogleLeNet (2019). BVLC GoogLeNet Model. Disponible on-line:. https://github.com/BVLC/caffe/tree/master/models/bvlc_googlenet. Accessed: 2019-2-11.

[34] ResNet-18 (2019). He, Kaiming, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 770-778. 2016.