Universidad de Alcalá
Escuela Politécnica Superior

MÁSTER UNIVERSITARIO EN INGENIERÍA DE TELECOMUNICACIÓN

Trabajo Fin de Máster

Desarrollo de un sistema para la interpretación y predicción de la situación del tráfico mediante Deep Learning

Autor: Carlos Herranz Perdiguero

Tutor: Roberto Javier López Sastre

2018

Universidad de Alcalá

Escuela Politécnica Superior

Máster Universitario en Ingeniería de Telecomunicación

Trabajo Fin de Máster

Desarrollo de un sistema para la interpretación y predicción de la situación del tráfico mediante Deep Learning

Autor: Carlos Herranz Perdiguero
Tutor: Roberto Javier López Sastre

TRIBUNAL:

Presidente: D. Saturnino Maldonado Bascón

Vocal 1º: D. Miguel Ángel García Garrido

Vocal 2º: D. Roberto Javier López Sastre

FECHA: 27 de junio de 2018

Desarrollo de un sistema para la interpretación y predicción de la situación del tráfico mediante Deep Learning.

Carlos Herranz Perdiguero.

27 de junio de 2018

Never say never, because limits, like fears,
are often just an illusion.

Agradecimientos

Con este Trabajo Fin de Máster acaba una de las etapas más importantes de mi vida. Para llegar hasta aquí, ha sido imprescindible la ayuda y comprensión de muchas personas, y es por ello que se lo quiero agradecer. En primer lugar, a mi tutor de TFM, muchas gracias, Roberto, por tu ayuda y apoyo. He aprendido mucho de ti. Gracias también a los profesores del Máster por la gran labor que realizáis transmitiéndonos vuestros conocimientos. A los compañeros del grupo GRAM, por los buenos ratos pasados. Y a mi familia por lo mucho que me han apoyado y porque siempre están conmigo.

Muchas gracias a todos.

Índice general

Lista de figuras

Lista de tablas

Resumen

La comprensión semántica de una escena es un aspecto clave en múltiples aplicaciones de inteligencia artificial, tanto para los Sistemas Inteligentes de Transporte como para los robots.

En este Trabajo Fin de Máster se diseña, desarrolla y evalúa un sistema que, basado en la segmentación semántica de imágenes, obtenida mediante una red neuronal convolucional, permite realizar las distintas tareas que abarca la comprensión de un escena: clasificación, detección de objetos y la propia segmentación semántica, de una manera sencilla y eficiente. Además, proponemos una solución enfocada a vehículos inteligentes, que permite, utilizando la segmentación semántica, estimar la velocidad a la que debe circular el vehículo. Para ello, hemos construido una nueva base de datos en la que poder evaluar este nuevo problema. Los resultados confirman que es posible y beneficioso confiar en la segmentación semántica para llevar a cabo las distintas tareas.

Palabras clave: Redes Neuronales Convolucionales, comprensión semántica de escenas, estimación de velocidad, Sistemas de Trasporte Inteligentes, *deep learning*.

Abstract

Semantic scene understanding is a key aspect of multiple artificial intelligence applications, from Intelligent Transportation Systems to robotics.

In this Final Project, we design, develop and evaluate a system that, based on the semantic segmentation of images obtained through a convolutional neural network, allows to carry out the different tasks comprising scene understanding: classification, object detection and the aforementioned semantic segmentation, in a simple yet efficient manner. In addition, we propose a solution focused on intelligent vehicles, which allows us, using semantic segmentation, to estimate the speed at which the vehicle must be driven. To this end, we have built a new database in which we can evaluate this challenging new problem. The results confirm that it is possible and beneficial to rely on semantic segmentation to successfully perform the different tasks.

Keywords: Convolutional Neural Networks, semantic scene understanding, speed estimation, Intelligent Transportation Systems, deep learning.

Resumen Extendido

Nosotros, los humanos, somos capaces de reconocer e interpretar con extrema facilidad múltiples situaciones cotidianas. Sabemos rápidamente si nos encontramos en una cocina o un cuarto de baño, si nuestro coche circula por una ciudad o estamos en una autovía, y podemos ajustar la velocidad de nuestro vehículo en función de la situación de la vía. Sin embargo, para una máquina, interpretar lo que le rodea es una tarea extremadamente compleja. La inteligencia artificial pretender asemejar el comportamiento de las máquinas al de los humanos y, en ese proceso de aprendizaje, los sistemas de visión por computador están llamados a jugar un papel clave.

En este sentido, el objetivo que se persigue en este trabajo es doble: por una parte, crear un sistema fiable para el reconocimiento de escenas de tráfico y de interior mediante estrategias de visión artificial. Por otra, se busca aportar soluciones que permitan estimar la velocidad adecuada de los vehículos en función de la situación del tráfico. Para cumplir ambos objetivos, vamos a confiar en la información que aporta la segmentación semántica de las imágenes como mecanismo robusto para llevar a cabo las diferentes tareas.

Descripción del trabajo realizado

El trabajo podemos dividirlo en dos grandes bloques: 1) la comprensión semántica de la escena; y 2) la estimación de la velocidad adecuada de los vehículos.

Comprensión semántica de la escena

En lo que respecta a la **comprensión semántica de la escena**, en este trabajo se propone un modelo que confía en la segmentación semántica de las imágenes para llevar a cabo el resto de tareas, esto es, tanto la clasificación de las mismas como la detección de objetos.

Para ello, comenzamos proponiendo un modelo de segmentación semántica, basado en DeepLab [4], empleando como redes base dos Redes Neuronales Convolucionales (CNNs) muy utilizadas: VGG-16 [45] y ResNet-101 [26]. A esta última le añadimos una supervisión multiescala, que consiste en procesar las imágenes a diferentes factores de escala $\{0.5, 0.75, 1\}$ de la resolución original mediante tres redes que trabajan en paralelo, para finalmente fusionar los resultados. Mediante este procedimiento conseguimos mejorar la información contextual que la red obtiene de la imagen.

Con las segmentaciones semánticas de las escenas, buscamos clasificar con éxito las mismas. Para ello, proponemos dos soluciones simples pero que se demuestran muy efectivas, basadas en el empleo de Máquinas de Vectores Soporte (SVMs) con diferentes núcleos (lineal, intersección de histogramas, χ^2 y Jenson-Shannon). Para la primera de ellas, empleamos como características histogramas normalizados que generamos con las etiquetas de clase de la segmentación semántica. La segunda de nuestras propuestas consiste en emplear vectores *one-hot*, esto es, vectores binarios que indiquen la presencia o ausencia de las clases en la segmentación. Además, al construir los histogramas o vectores, se propone el uso de pirámides espaciales que nos permitan adquirir información de localidad. Ambas estrategias consiguen grandes resultados cuando las evaluamos en distintas bases de datos.

Por último, para la detección de objetos a partir de la segmentación semántica empleamos el procedimiento descrito por [21]. Esta estrategia consiste en trazar una *bounding box* alrededor de cada componente conectada de píxeles pertenecientes a las clases de objeto a detectar.

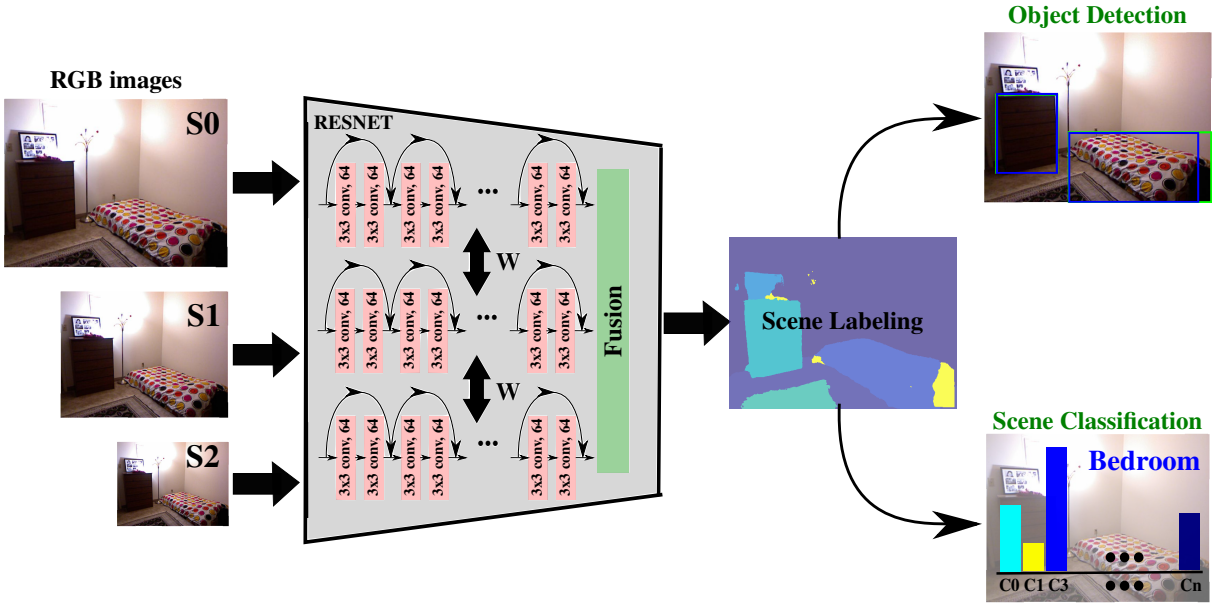


Figura 1: Modelo de comprensión de la escena.

Estimación de la velocidad adecuada

A modo de predicción de la situación del tráfico, en este segundo bloque tratamos la tarea de **estimar la velocidad adecuada** que un determinado vehículo debería llevar en función de dónde se encuentra y las condiciones del tráfico que le rodean, empleando exclusivamente información visual. Para ello, en primer lugar, ha sido necesario generar una base de datos grabando una serie de secuencias de vídeo, y etiquetando cada imagen con la velocidad que lleva el vehículo.

A partir de aquí, proponemos dos soluciones. Una primera se basa en hacer uso de nuestro modelo de segmentación semántica. Del mismo modo que empleábamos histogramas normalizados o vectores *one-hot* con los resultados de la segmentación para clasificar escenas, ahora vamos a emplear esos mismos histogramas y vectores para alimentar diferentes modelos de regresión capaces de estimar la velocidad del vehículo a partir de información visual. Entre los modelos de regresión que se han utilizado en el trabajo están: un regresor lineal, un regresor de lasso, un regresor basado en SVMs y un regresor basado en *boosting trees*. La segunda solución propuesta hace uso de redes neuronales; en concreto, conseguimos adaptar dos CNNs, una VGG y una ResNet, mayoritariamente usadas en clasificación, a la tarea de regresión. De este modo, se comparan diferentes modelos que emplean distinta información; mientras que, en el primer caso, se emplean las segmentaciones semánticas como descriptores, las redes neuronales son capaces de realizar la regresión puramente a partir de las imágenes. La figura 2 muestra nuestro modelo cuando trabajamos con la segmentación semántica.

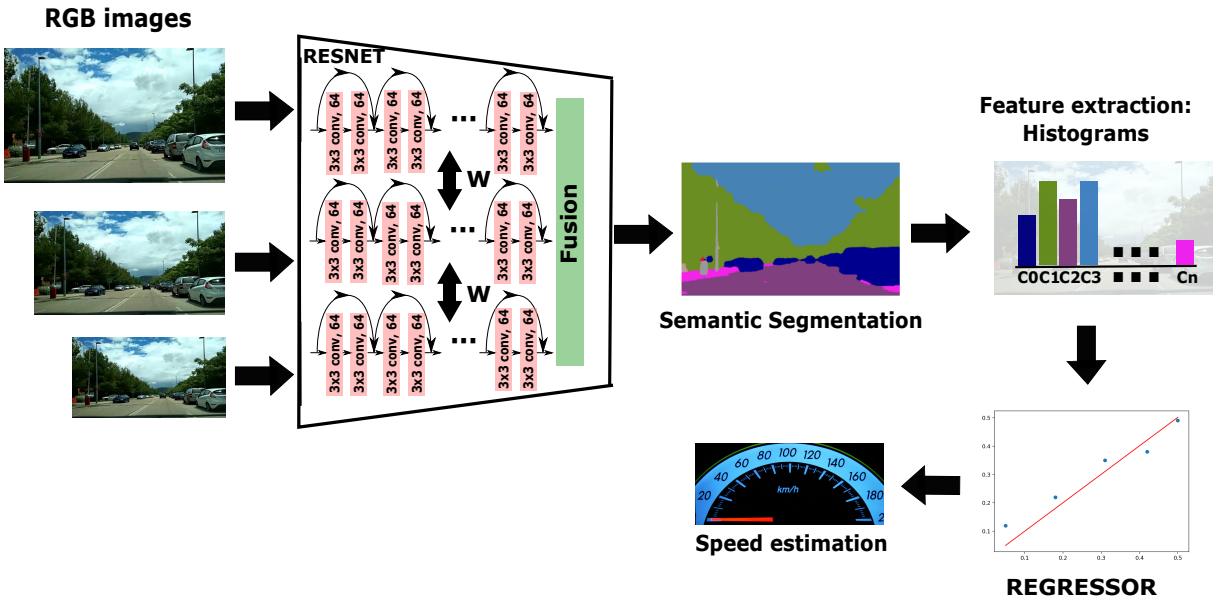


Figura 2: Modelo de estimación de la velocidad.

Resultados

Finalmente, se presenta un resumen de los resultados obtenidos. Para evaluar nuestro método de comprensión de escenas, vamos a emplear bases de datos que abarcan dos de los campos de aplicación en los que este tipo de información es especialmente relevante: los Sistemas de Transporte Inteligente, haciendo uso de escenas de tráfico, y la robótica, con escenas de interior.

Escenas de tráfico

Por una parte, en lo que respecta a escenas de tráfico, vamos a emplear la base de datos Cityscapes [9] para entrenar nuestro modelo de segmentación semántica. Una vez entrenado, la clasificación de escenas va a ser llevada a cabo en una base de datos que hemos generado extrayendo recorridos de Google Street View (GSV) y que hemos dividido en dos conjuntos, en función del país de donde han sido extraídas las mismas. Proponemos clasificar entre tres posibles tipos de vía: autovía, carretera secundaria y entorno urbano.

La tabla 1 compara los resultados de nuestro modelo en materia de segmentación semántica con el estado del arte en Cityscapes. Aunque no consigue obtener la precisión de las propuestas más recientes, obtiene unos resultados bastante aceptables a pesar de emplear menos imágenes de entrenamiento que los mejores. En la figura 3 se muestran algunos resultados cualitativos de la segmentación.

Tabla 1: Comparativa con otros modelos de segmentación semántica en Cityscapes.

Modelo	¿Conjunto fino?	¿Conjunto basto?	**IoU**
DeepLabV3+ [6]	X	X	82.1
PSPNet [59]	X	X	81.2
Dilation10 [58]	X		67.1
FCN 8s [30]	X		65.3
ENet [38]	X		58.3
Nuestro - VGG-16	X		64.6
Nuestro - ResNet-101	X		71.7

La tabla 2 muestra un resumen de los resultados de clasificación de nuestros modelos. Comprobamos que emplear histogramas de segmentación semántica se queda muy cerca de los resultados obtenidos por redes neuronales específicamente entrenadas para clasificación.

Escenas de interior

En lo que respecta a las escenas de interior, se ha utilizado la base de datos NYU-Depth v2 (NYUD2). Esta base de datos, a diferencia de Cityscapes, nos permite realizar y evaluar tanto la segmentación semántica como la clasificación de escenas y la detección de objetos comparándonos con otras propuestas.

La tabla 3 muestra una comparativa entre nuestros modelos y el estado del arte en NYUD2 en materia de segmentación semántica. Nuestro modelo, a diferencia de otros métodos, no emplea información de profundidad y, aún así, consigue mejorar al resto de modelos reportados hasta la fecha. Pueden verse en la figura 4 una serie de resultados cualitativos de la segmentación en esta base de datos.

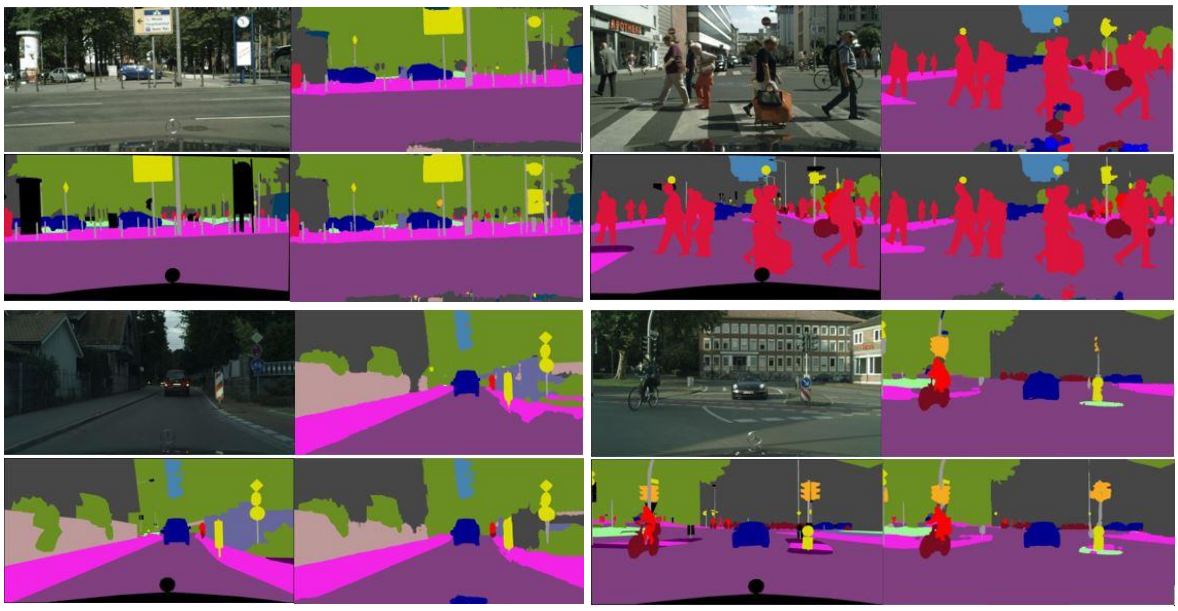


Figura 3: Resultados cualitativos de la segmentación semántica en Cityscapes. Por cada grupo de cuatro imágenes se muestra la imagen original RGB (arriba a la izquierda), el *ground truth* (abajo a la izquierda), el resultado de la segmentación con una VGG (arriba a la derecha) y el resultado de la segmentación con una ResNet como red base (abajo a la derecha).

Del mismo modo, en clasificación de escenas (tabla 4) y detección de objetos (tabla 5), nuestro método consigue desbancar al resto de modelos que han dado resultados en esta base de datos, consiguiendo de este modo el estado del arte en las tres tareas que abarca la comprensión de una escena. Un conjunto de resultados cualitativos de la tarea de detección de objetos pueden verse en la figura 5.

Estimación de la velocidad

En lo que respecta a la estimación de la velocidad, vamos a emplear como métrica de evaluación el error medio cometido, en km/h, en las secuencias de test, entre la velocidad real del coche y la velocidad adecuada predicha por la red. La tabla 6 muestra el error para los distintos métodos de regresión evaluados en una de esas secuencias. En ella, el número de divisiones indicado en los modelos basados en la segmentación semántica, es aquel para el que se obtienen los mejores resultados.

En general, observamos como resulta beneficioso estimar la velocidad a partir de la segmentación. En concreto, para la secuencia evaluada, los *boosting trees* son los que obtienen los mejores resultados, seguidos del regresor de lasso, las SVR y el regresor lineal. Todos ellos obtienen resultados superiores a emplear CNNs entrenadas para la tarea.

Tabla 2: Resultado de la clasificación de escenas en la base de datos Google StreetView con tres posibles clases: entorno urbano, carretera secundaria y autovía.

Método	**Kernel**	**Niveles**	**Precisión España**	**Precisión Suiza**
Histogramas SS L2	Lineal	4	75,76	93,11
	Intersección	4	**77,85**	93,26
	Chi cuadrado	3	76,02	95,98
	Jenson Shannon	3	75,61	**96,41**
Vectores *One-hot*	Lineal	4	**75,65**	94,84
	Intersección	4	74,29	**95,41**
	Chi cuadrado	4	74,32	**95,41**
	Jenson Shannon	4	74,27	**95,41**
Método	**Red**	**Política lr**	**Precisión España**	**Precisión Suiza**
Redes neuronales	VGG-16	Step	**83,91**	99,00
	ResNet-101	Step	81,72	**99,43**

Tabla 3: Comparativa de resultados en segmentación semántica en NYUD2.

Método	**Entrada**	**prec. píxel**	**prec. clase**	**IoU**
FuseNet-SF3 [25]	RGB-D	66.4	44.2	34.0
Context-CRF [28]	RGB	67.4	49.6	37.1
MVCNet [33]	RGB-D	69.1	50.1	38.0
Nuestro - VGG-16	RGB	67.4	48.5	36.3
Nuestro - ResNet-101	RGB	**70.9**	**52.2**	**41.8**

Tabla 4: Comparativa de precisión en clasificación de escenas en NYUD2.

	Método	**Entrada**	**Prec.**
Estado del arte	[51]	RGB-D	63.9
	[46]	RGB-D	65.8
Nuestro modelo	**SVM con Pirámide Espacial y Kernel Aditivo**	RGB	**68.96**

Tabla 5: Resultados de la detección de objetos en la base de datos NYUD2.

Método	**Entrada**	**Precisión media (AP)**
[51]	RGB-D	24.3
[46]	RGB-D	31.1
Nuestro método	RGB	**35.5**

Figura 4: Resultados cualitativos de segmentación semántica en la NYUD2. Por cada grupo de tres imágenes, se muestra de izquierda a derecha, la imagen RGB, el *ground truth* y nuestro resultado cuando empleamos ResNet como red base.

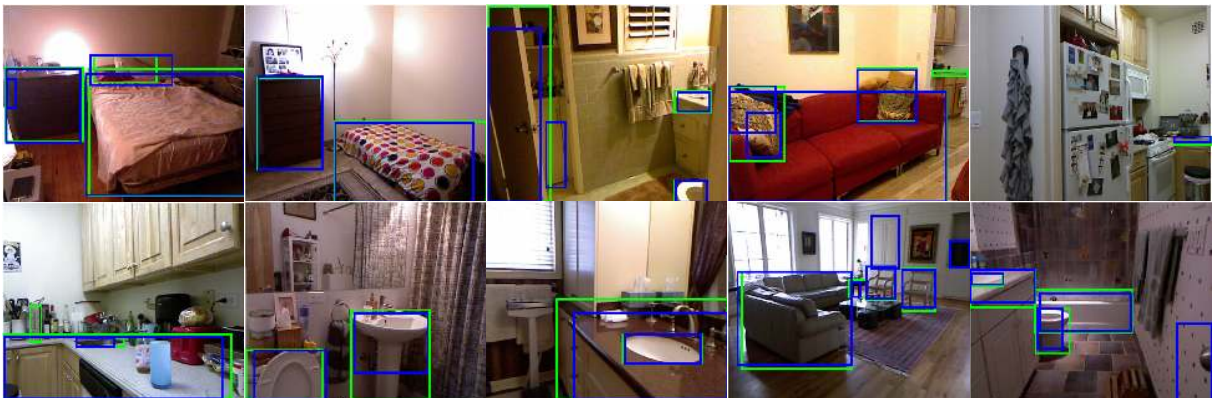


Figura 5: Resultados cualitativos para la tarea de detección de objetos. En verde se muestra las *bounding boxes* del *ground truth* y en azul nuestra predicción.

Tabla 6: Comparativa entre el error cometido sobre la secuencia de test de autovía.

Método	**Entrada**	**Error (Km/h)**
VGG-16	RGB	12.48
ResNet-101	RGB	12.79
Regresión lineal	Segmentación semántica (1 división)	9.54
SVR	Segmentación semántica (3 divisiones)	9.23
Regresión de Lasso	Segmentación semántica (3 divisiones)	8.41
Boosting Trees	Segmentación semántica (3 divisiones)	**7.76**

Conclusiones

Este trabajo presenta una novedosa y efectiva propuesta para la comprensión de escenas y la estimación de la velocidad adecuada a la situación real del tráfico. A partir de una CNN profunda para la segmentación semántica, presentamos y desarrollamos una gran variedad de estrategias muy robustas para extraer descriptores basados en la segmentación, y poder llevar a cabo tanto la clasificación de escenas, como la detección de objetos, y la propia estimación de la velocidad. En general, demostramos que emplear una segmentación semántica precisa de la escena, como la que nos ofrece nuestro modelo, es una solución extremadamente beneficiosa para adquirir una completa comprensión de la escena, lo que se traduce en reconocer el tipo vía, o incluso predecir la velocidad a la que debería circular un vehículo en función de la situación del tráfico.

Glosario

TFM Trabajo Fin de Máster

ISA Intelligent Speed Adaptation

fps frames por segundo

SVM Máquina de Vectores Soporte

CNN Red Neuronal Convolucional

GSV Google Street View

SVR Máquina de Vectores Soporte para Regresión

GPS Sistema de Posicionamineto Global

NYUD2 NYU-Depth v2

HOG Histograma de Gradientes Orientados

SIFT *Scale-Invariant Feature Transform*

CRF *Conditional Random Field*

DPM Modelo de Partes Deformables

YOLO *You Only Look Once*

SSD *Single Shot Multibox Detector*

FV Vector de Fisher

FCN *Fully Convolutional Networks*

FOV campo de visión

ASPP *Atrous Spatial Pyramid Pooling*

mIOU *mean Intersection Over Union*

Capítulo 1

Introducción

El aprendizaje profundo o *deep learning* se ha convertido en una herramienta fundamental como mecanismo para modelar los diferentes niveles de abstracción en sistemas de aprendizaje automático, aportando un gran impulso a los resultados que estos sistemas pueden conseguir. En concreto, las arquitecturas basadas en Redes Neuronales Convolucionales (CNNs), son hoy en día el estado del arte para múltiples tareas en diversos ámbitos, como la robótica, la medicina, el procesado de voz o los sistemas de transporte inteligente.

A lo largo de este Trabajo Fin de Máster (TFM) se describe, desarrolla y evalúa un completo sistema para el reconocimiento de escenas de tráfico y de interior, junto a un novedoso mecanismo para estimar la velocidad adecuada que los vehículos deberían llevar en cada momento en función de la situación del tráfico. Además, con el objetivo de unificar y simplificar el sistema, se confía todo en la segmentación semántica de las imágenes como técnica robusta para resolver todas estas tareas.

1.1. Motivación

Los sistemas de transporte inteligentes, y especialmente los vehículos autónomos, deben verse beneficiados de los últimos avances en visión e inteligencia artificial. En concreto, con el fin de conseguir vehículos más seguros, creemos que resulta fundamental dotar a los mismos de sistemas para la segmentación semántica de imágenes, que sean precisos, estén especializados en escenas de tráfico, funcionen en tiempo real y que, además, puedan ser embarcados.

Que el vehículo disponga de una segmentación semántica de la escena que le rodea (véase Figura 1.1), permite realizar una interpretación semántica de muy alto nivel, de modo que sea posible, entre otras cosas:

1. Reconocer el tipo de escena en el que nos encontramos (urbana, carretera secundaria, autovía).

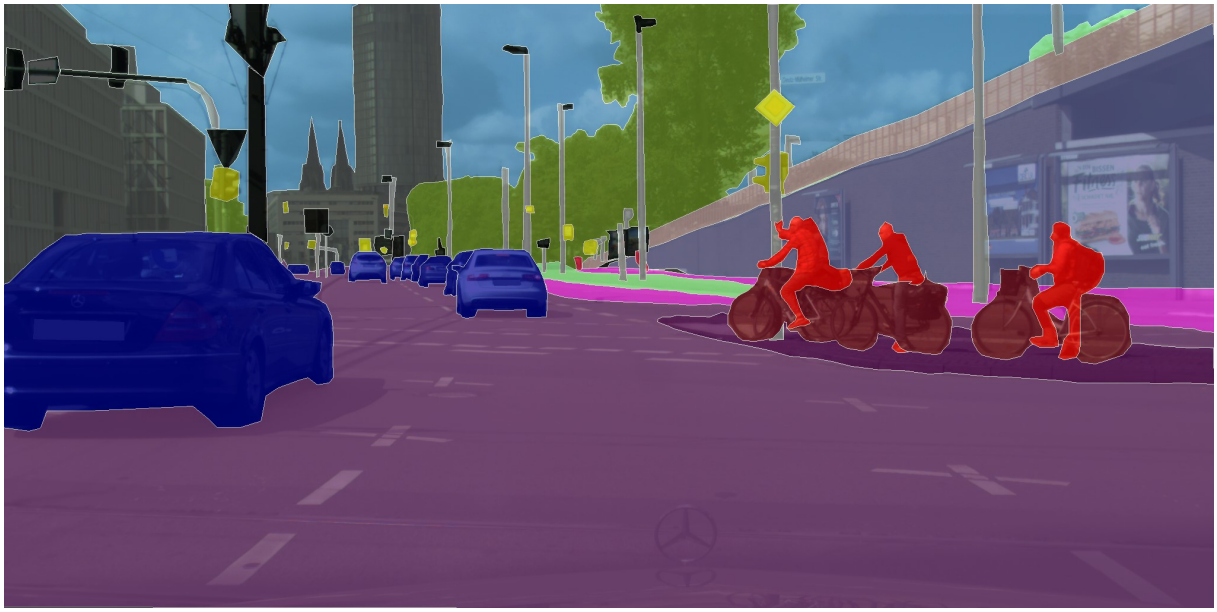


Figura 1.1: Ejemplo de segmentación semántica de una escena de tráfico.

2. Localizar el resto de vehículos y estimar su distancia.

3. Detectar peatones y ciclistas, de modo que podamos evitar colisiones o incluso predecir sus patrones de movimiento.

4. Adaptar la velocidad del vehículo a la situación real de tráfico observada.

Así, en este trabajo se pretende poner de manifiesto la importancia de la segmentación semántica, otorgándole un papel clave en múltiples tareas. ¿Es necesario entrenar una red neuronal por tarea, para clasificación de escenas, detección de objetos o estimación de la velocidad, cuando podemos emplear una única red de segmentación semántica que nos ofrece la información suficiente para llevar a cabo el resto de tareas de una forma más sencilla y eficiente computacionalmente?

Además, el exceso de velocidad es, según el Consejo Europeo de Seguridad en el Transporte, el factor principal de 1/3 de todas las colisiones de vehículos, responsables, directamente, de la muerte de 500 personas por semana en las carreteras europeas. Centrándonos ahora en entornos urbanos, la Fundación MAPFRE apunta que el 72 % de los atropellos en ciudad podría haberse evitado si el conductor hubiese respetado el límite de velocidad. Estas cifras ponen de manifiesto la importancia de investigar en soluciones para la adaptación inteligente de la velocidad, conocidas como sistemas ISA (Intelligence Speed Adaptation).

Nuestra hipótesis de partida es clara. Los avances recientes en visión e inteligencia artificial, posibilitan la implementación de una nueva generación de sistemas ISA capaces de adaptar la velocidad del vehículo a la situación de tráfico real observada por una cámara embarcada en el mismo. De este modo, proponemos solucionar los principales problemas

de las soluciones ISA más avanzadas que se encuentran actualmente en el mercado, como son: a) la imprecisión derivada del proceso de localización GPS a la hora de indicarle al ISA el límite de velocidad de la vía; b) las tasas de fallo de los actuales sistemas de visión artificial centrados en la detección de señales de límites de velocidad así como su no adaptación a la situación de tráfico actual y futura.

1.2. Objetivos

Los objetivos principales de este TFM son tres:

- Desarrollar nuevos modelos de inteligencia artificial para la interpretación de las condiciones de tráfico y el reconocimiento del tipo de escena, mediante modelos de segmentación semántica de imágenes. Proponemos una adaptación de soluciones basadas en *fully convolutional deep networks* [30], entrenando nuestros modelos en la base de datos Cityscapes [9], que contiene secuencias de vídeo anotadas adquiridas desde un vehículo. Con las segmentaciones semánticas, se desarrollarán modelos capaces de distinguir entre las siguientes clases de escenas: 1) autovía; 2) carretera secundaria; 3) escena urbana.

- Adaptar esos modelos de inteligencia artificial a otros dominios. Para ello, sustituiremos las escenas de tráfico por escenas de interior. Realizaremos el reconocimiento de las mismas, y añadiremos una tarea clave en este ámbito: la detección y localización de objetos.

- Aportar soluciones para estimar la velocidad adecuada de los vehículos en el dominio de los sistemas ISA de nueva generación. La evaluación de la situación del tráfico y condiciones de la vía va a ser abordada desde una solución de inteligencia artificial que nos permita realizar estimaciones de la velocidad que debería llevar el vehículo en todo momento. Para ello, se emplearán modelos de regresión para los que la información semántica tendrá un protagonismo crucial.

1.3. Campos de aplicación

Son múltiples los campos en los que este tipo de sistemas tienen utilidad. En este trabajo se ha trabajado sobre dos de ellos:

- **Sistemas de transporte inteligente.** Resulta evidente la importancia que tiene que un vehículo autónomo reconozca el tipo de escena en la que se encuentra. Es cierto que aplicaciones basadas en GPS pueden hoy en día dar esa información con gran precisión localizando su posición, sin embargo, no pueden adaptarse a cambios recientes en las vías, por lo que reforzar y complementar esa información a partir de sistemas de visión artificial es fundamental. Conseguir un sistema ISA que informe

(o actúe directamente) sobre la velocidad adecuada en cada momento supondría una mejora en la seguridad vial de nuestras carreteras.

- **Robótica.** El reconocimiento de escenas de interior, así como la detección y localización de los objetos que las forman es una información fundamental para los robots. Sin embargo, a día de hoy, la comprensión del espacio que les rodea, más allá de la detección obstáculos, es todavía muy limitada. Así, que un robot doméstico tenga conocimiento de si se encuentra en una cuarto de baño o en un cocina, es una habilidad clave y esencial como punto de partida sobre el que desarrollar nuevas aplicaciones más complejas.

Al margen de estos, existen otros campos de aplicación en los que el reconocimiento de escenas tiene cabida, como la indexación y recuperación automática de imágenes en función de su contenido en bases de datos, la capacidad de adaptar y personalizar fotografías en función de la escena (por ejemplo, modificando el brillo o contraste de las mismas), la mejora en la detección de imágenes de contenido sensible, o como método para ofrecer publicidad personalizada a los usuarios (por ejemplo, a partir de las imágenes que suben a una red social).

En los siguientes capítulos se describe detalladamente el sistema implementado. En el capítulo 2 se profundiza sobre segmentación semántica, reconocimiento de escenas y detección de objetos. El capítulo 3 trata la retadora tarea de estimar la velocidad adecuada para un vehículo a partir de lo que *ve* a su alrededor. Concluimos en el capítulo 4.

Capítulo 2

Comprensión semántica de la escena

2.1. Introducción

La comprensión semántica visual de la escena se ha convertido en una capacidad muy importante en muchas aplicaciones, como la conducción autónoma [22], la navegación en interiores [33] o la manipulación robótica [52]. De hecho, es la capacidad que permite a los robots interactuar con el entorno con una intervención o número de sensores mínimo, debido a la valiosa información que ofrece.

Su objetivo consiste en obtener el mayor conocimiento semántico posible de una escena determinada. Esto incluye la categorización (etiquetado de toda la escena), la detección de objetos (predicción de las ubicaciones de los objetos mediante *bounding boxes*) y segmentación semántica (etiquetado de cada píxel).

En los últimos años, se ha avanzado mucho en la resolución de todas estas tareas (por ejemplo, [4, 40, 55, 15]). Sin embargo, la mayoría de los diseños modernos actuales proponen resolver estos problemas por separado. En este capítulo, se demuestra que resulta beneficioso abordar las distintas tareas siguiendo un enfoque ascendente, basado, principalmente, en la segmentación semántica de la escena.

Como se muestra en la Figura 2.1, nuestra solución comienza con una red convolucional profunda para segmentación semántica. Basamos nuestro diseño en una arquitectura que ha sido propuesta recientemente, ResNet [26], mejorando su rendimiento con una supervisión multiescala. A continuación, proponemos dos enfoques sencillos pero precisos para realizar tanto la localización de objetos como las tareas de reconocimiento de escenas, teniendo en cuenta únicamente la segmentación semántica de las escena como entrada.

En general, las principales contribuciones que se presentan en este capítulo son las siguientes:

1. El núcleo de la arquitectura propuesta se basa en una red de segmentación semántica. Proponemos una extensión intuitiva del modelo DeepLab [4], que consiste en fusionar la respuesta de un conjunto de redes basadas en ResNet multiescala.

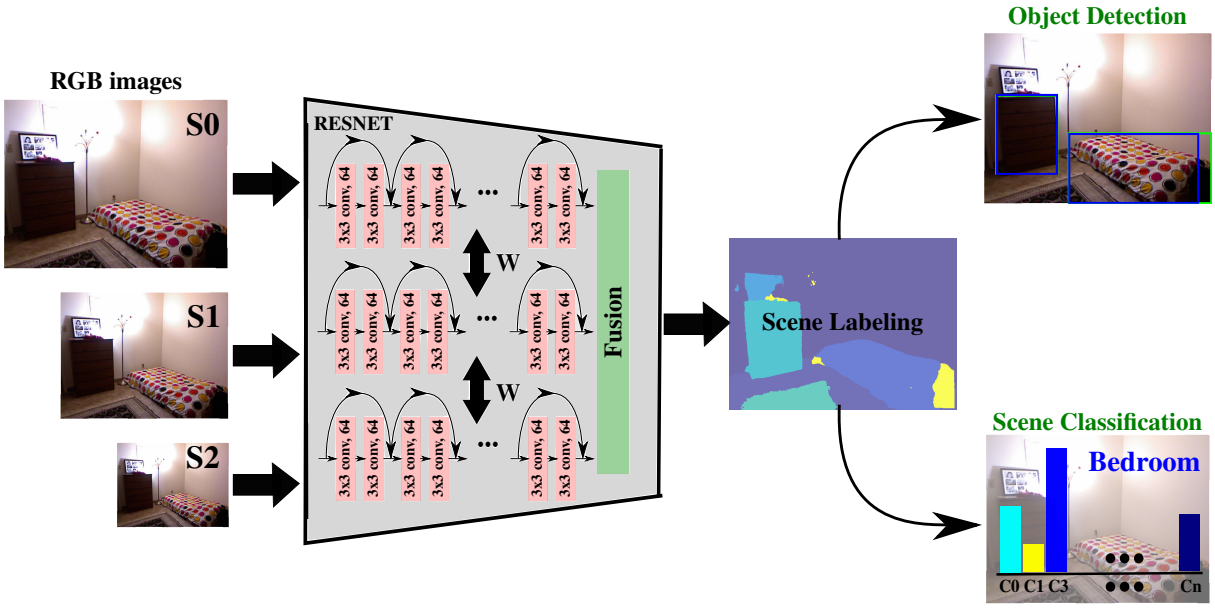


Figura 2.1: Entrenamos una arquitectura multiescala basada en ResNet para realizar un etiquetado preciso de los píxeles de la escena. La innovación clave es aprovechar esta segmentación semántica para realizar tanto la localización de objetos como el reconocimiento de escenas, ofreciendo una solución integral, rápida y precisa, que puede ser integrada en cualquier plataforma móvil.

2. A partir de nuestra segmentación semántica, demostramos que es posible clasificar con éxito las escenas. Técnicamente, presentamos dos soluciones basadas en clasificadores. En primer lugar, proponemos un diseño sencillo que utiliza SVMs con diferentes núcleos, utilizando como características los histogramas de las etiquetas de clase de la segmentación semántica. Segundo, proponemos emplear las SVMs viendo cada segmentación semántica como un vector binario de tipo *one-hot*, indicando la presencia/ausencia de las clases correspondientes. Demostramos que estas dos soluciones sorprendentemente simples, flexibles y rápidas pueden sobrepasar los resultados del estado del arte reportados por complejas redes profundas entrenadas para clasificar directamente las escenas.

3. Finalmente, mostramos el potencial de nuestras segmentaciones semánticas a través de la tarea de localización de objetos. Basta con identificar en la máscara de etiquetado de píxeles para toda la escena, las componentes conectadas que encierran los objetos, permitiéndonos localizarlos.

4. Cuantificando la calidad de nuestras propuestas, nuestros modelos superan todos los resultados anteriores del estado del arte en la desafiante base de datos de interiores NYUD2 para las tres tareas que abarca la comprensión semántica de escenas: categorización de escenas, detección de objetos y segmentación semántica. Además, empleamos una base de datos propia extraída de GSV para evaluar el comporta-

miento de nuestros modelos en escenas de tráfico, entrenando la red de segmentación semántica en Cityscapes [9].

El resto del capítulo está organizado de la siguiente manera. En la Sección 2.2 se revisan los trabajos anteriores. Detallamos nuestras propuestas de forma técnica en la Sección 2.3. La Sección 2.4 incluye la descripción de los experimentos realizados. Los resultados quedan reflejados en la Sección 2.5.

2.2. Estado del Arte

2.2.1. Segmentación semántica

Hasta la década pasada, la mayoría de sistemas empleados en segmentación de imágenes utilizaban un extractor de características (p. ej. descriptores HOG [10] ó SIFT [31]...) al que añadían un clasificador fundamentalmente de tipo *Random Forest* o SVM.

Sin embargo, la irrupción del *deep learning* para clasificación de imágenes ha hecho que la tarea de segmentación se vea también beneficiada. Inicialmente, se desarrollaron modelos en los que se realizaba segmentación por clasificación como en [8]. Por cada píxel de la imagen, se tomaba un parche a su alrededor, y se clasificaba dicho parche asignando al píxel la clase dada al parche. Requería, por tanto, tantas pasadas por una red neuronal de clasificación como píxeles tuviera la imagen, lo que les hacía ser métodos muy lentos.

En 2014, Long *et al.* [30] introdujeron soluciones basadas en *fully convolutional deep networks* consiguiendo un gran avance en los resultados obtenidos. El método propuesto se basaba en sustituir las capas *fully-connected* de las redes de clasificación por capas convolucionales capaces de realizar la predicción píxel a píxel. Este modelo era mucho más rápido que los basados en parche y no requería que las imágenes a segmentar fuesen siempre del mismo tamaño. Desde entonces, una gran parte de los modelos que forman el estado del arte se basan en él, incorporando ligeras pero importantes novedades. Por ejemplo, en [4] se introdujo el concepto de convoluciones dilatadas para incrementar el campo receptivo de las redes neuronales, mientras que Noh *et al.* [37] propusieron una estructura compuesta por una red deconvolucional para aprender máscaras de segmentación.

Otros trabajos [54, 5, 23], incluyen el empleo y composición de características a múltiples escalas. Dado que en las redes neuronales profundas, según se avanza en profundidad se pierde información de localidad, combinar características de diferentes escalas mejora los resultados.

Otra línea tiene que ver con el uso de técnicas de posprocesado para mejorar la segmentación. El primer trabajo en emplear este tipo de estrategias [4] utilizó un *Conditional Random Field* (CRF) para refinar el resultado. Otros métodos [1, 60] buscan afinar la salida de la red mediante modelos end-to-end. Ambas técnicas intentan mejorar el tratamiento dado a los bordes entre clases, para delimitar con mayor precisión los objetos de la imagen.

2.2.2. Clasificación de escenas

El estado del arte en clasificación de escenas está dominado, principalmente, por el uso de CNNs entrenadas con grandes bases de datos. Del mismo modo que las redes neuronales profundas como VGG [45] o ResNet [26] consiguen rendimientos muy buenos en la clasificación de objetos, esas mismas redes pueden ser entrenadas para reconocer escenas. Sin embargo, resulta una tarea mucho más compleja, dado que la varianza y el número de objetos por imagen en clasificación de escenas es mayor y la dificultad de generar un modelo adecuado se incrementa.

Así, algunos trabajos [11, 57] muestran que extraer características locales mediante parches de la imagen puede resultar beneficioso como método de representación de escenas. En ellos, se trabaja a múltiples escalas y se hace uso de codificadores de tipo Fisher Vector. Por otro lado, recientemente, Wu *et al.* [53] han propuesto una arquitectura en la que el muestreo de parches de forma densa se sustituye por propuestas de regiones con el objetivo de tomar únicamente las zonas más representativas de la escena como características.

Por último, trabajos como [61], buscan tener en cuenta los objetos presentes en las escenas, bajo el supuesto de que el conocimiento de los objetos en una imagen debería de ayudar a reconocer el conjunto.

2.2.3. Clasificación y localización de objetos

Los métodos de clasificación y localización de objetos en imágenes han sido tradicionalmente divididos en dos grande grupos: los que confían en ventanas deslizantes y los que se basan en clasificar propuestas de regiones. Antes de la llegada de las CNNs, ambos tipos de métodos, representados por el Modelo de Partes Deformables (DPM) [17] y el *Selective Search* [49] respectivamente, tenían un rendimiento muy similar. Sin embargo, el detector R-CNN [19] trajo consigo una considerable mejora. Este detector, que tiene una primera parte basada en propuesta de regiones mediante el algoritmo de búsqueda selectiva, incorpora una red convolucional para clasificar esas propuestas. Desde entonces los métodos basados en CNNs han prevalecido.

Esa primera aproximación de las CNNs a la tarea de detección de objetos que supuso el R-CNN, ha ido viendo la inclusión de mejoras. Así, el Fast R-CNN, [18] diseñado como una evolución de su predecesor, incorpora una función de pérdidas que aúna tanto la clasificación como la localización (regresión) de las *bounding boxes*, pudiendo ser entrenado *end-to-end*. Finalmente, el Faster R-CNN [41] sustituye la manera de generar propuestas. Se deja de emplear el algoritmo de *Selective Search*, que había demostrado ser el cuello de botella del proceso, y se sustituye por una Red de Propuesta de Regiones (RPN) consiguiendo mejorar los resultados previos.

Más recientemente, un conjunto de métodos que está teniendo gran éxito por sus grandes resultados tanto en tiempo de proceso como en precisión de la detección, buscan eliminar por completo la fase de propuesta de regiones y predecir de manera directa tanto las *bounding boxes* como la clase del objeto que contienen. En este grupo se encuentran

los detectores YOLO [39], que emplea el mapa de características de la última capa de una CNN para dar la predicción, y el *Single Shot Multibox Detector* (SSD) [29], que utiliza como salidas de la red el mapa de características de diferentes capas en función de la escala del objeto a detectar.

2.2.4. Comprensión semántica de la escena: todo en uno

En los últimos años, como se ha descrito, las CNNs se han convertido en el estado del arte para resolver tareas relacionadas con la comprensión semántica de una escena. Cada vez se han ido obteniendo mejores resultados en segmentación semántica [4, 59], clasificación de escenas [26, 45] y detección de objetos [40, 29].

Sin embargo, no muchos trabajos abordan este problema como un todo. Entre los que lo han hecho, los primeros trabajos, utilizaban características específicas (por ejemplo HOG [10] o SIFT [31]) para capturar algunas propiedades consideradas representativas. Probablemente, una de las primeras aproximaciones para conseguir una mayor comprensión semántica de la escena es [27], donde se propone un modelo que es capaz de clasificar la imagen (clasificación de escenas) y localizar los objetos que forman parte de ella. Además, en función de la relación entre el tipo de escena y los objetos, el modelo también puede determinar sus etiquetas (segmentación semántica).

En [56] se presenta un modelo holístico capaz de cumplir las tres tareas. Utilizan un modelo gráfico jerárquico -un CRF- con variables que representan aspectos como la presencia o ausencia de una clase. De esta manera, el modelo puede aprender conjuntamente sobre las diferentes tareas que abarca la comprensión semántica de la escena.

Gupta *et al.* [34, 20] proponen un método para detectar contornos en imágenes de profundidad y realizar la segmentación semántica de las mismas. Posteriormente emplean los resultados de la segmentación como características para la clasificación de escenas.

Más recientemente, el uso de redes neuronales con múltiples capas ha hecho que sea posible aprender características directamente de grandes cantidades de datos. Gupta *et al.* [21] usan el detector R-CNN [19] en imágenes de profundidad para detectar objetos en escenas de interior. En ese trabajo, Gupta *et al.* proponen codificar la información de profundidad con tres canales, llamados HHA. Esto permite aplicar directamente un modelo de CNN, pre-entrenado en imágenes RGB, a los canales HHA para extraer características. Dado que los datos de entrenamiento son limitados, aumentan el conjunto de entrenamiento mediante la renderización de escenas sintéticas. Wang *et al.* [51] calculan las características CNN a partir de una representación aumentada en píxeles que comprende múltiples modalidades de RGB, HHA y normales de superficie. Luego, cada modalidad es codificada usando Vectores de Fisher (FVs). Las características multimodales de FV resultantes se fusionan y, finalmente, se combinan con la imagen completa basada en las características CNN, para obtener el resultado final de la clasificación de la escena. Song *et al.* [46] hacen uso de la transferencia de modelos RGB pre-entrenados basados en CNNs, y los afina con el conjunto de datos RGB-D de destino. Proponen una estrategia alternativa

para aprender rasgos de profundidad y llevar a cabo la tarea de clasificación de escenas, combinando un entrenamiento local débilmente supervisado a partir de parches, seguido de un proceso global de ajuste fino con las imágenes completas.

Nuestro trabajo difiere claramente de las referencias anteriores tanto en el enfoque de segmentación como en el reconocimiento de escenas y localización de objetos. Primero, a diferencia de [34, 21, 20, 51, 46], nuestro modelo no necesita usar imágenes de profundidad. En su lugar, proponemos una red de segmentación semántica multiescala, que sólo funciona con imágenes RGB. A continuación, resolvemos las tareas de localización de objetos y categorización de escenas utilizando simplemente la segmentación semántica como característica. Demostramos que un enfoque tan simple puede superar los resultados más avanzados para las tareas de segmentación semántica, detección de objetos y categorización de escenas.

2.3. Descripción de los modelos

2.3.1. Modelo de segmentación semántica

El uso de *Fully Convolutional Networks* (FCN), introducido por primera vez por Long *et al.* [30], ha logrado grandes resultados al adaptar las redes de clasificación de imágenes a la tarea de segmentación semántica. Así, los diseños VGG [45] o ResNet [26] tienen sus versiones de segmentación semántica, que se basan principalmente en la sustitución de las últimas capas *fully-connected* por capas *fully covolutional*.

En este sentido, proponemos el uso de estas redes añadiendo las aportaciones del conocido modelo de segmentación semántica DeepLab [4]. DeepLab, que está disponible públicamente, presenta tres contribuciones principales:

1. Convolución *atrous* o convoluciones dilatadas (Figura 2.2), como una forma de aumentar el campo de visión (FOV) de los filtros convolucionales.

2. *Atrous Spatial Pyramid Pooling* (ASPP), que hace uso de varias convoluciones *atrous* paralelas con diferentes *factores de dilatación* (ver Figura 2.3) para extraer características a distintas escalas antes de fusionarlas.

3. Un CRF de post-procesamiento para refinar los resultados de segmentación dados por la CNN bajo la premisa de que píxeles cercanos y de apariencia similar deberían pertenecer a la misma clase. El CRF produce una gran mejora a la hora delimitar los bordes entre objetos.

Para nuestro modelo, adoptamos la arquitectura DeepLab, empleando sendas redes basadas en VGG-16 y ResNet-101. Además, exploramos un procedimiento de aprendizaje conjunto a múltiples escalas. Para ello, procesamos la imagen con factores de escala $\{0.5, 0.75, 1\}$ de la resolución original. De esta manera, tenemos tres redes completas trabajando

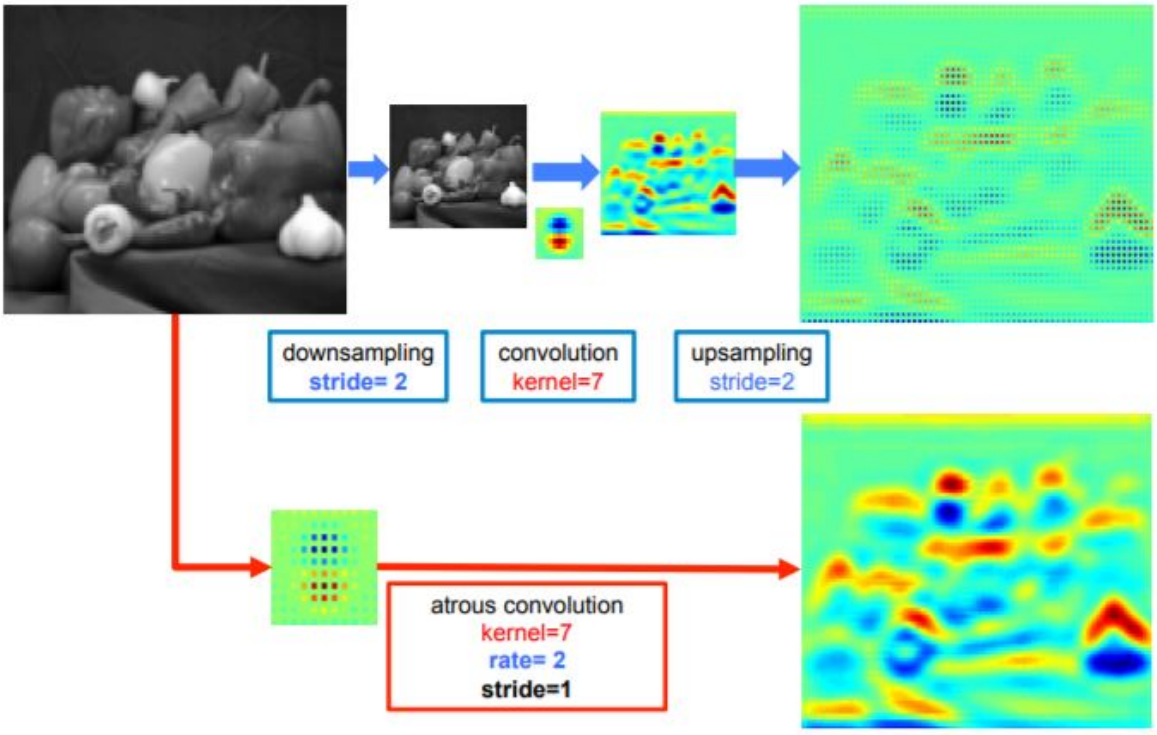


Figura 2.2: La convolución *atrous*, permite controlar la resolución espacial a la salida de una capa convolucional. Imagen extraída de [4].

en paralelo, como ramas de toda la arquitectura (Ver Figura 2.1). Al final, fusionamos los resultados en cada posición de píxel tomando la respuesta máxima dada por la red para cada escala. Este procedimiento puede ser visto como una forma de aumentar y mejorar la información contextual que nuestra red obtiene de la imagen. El efecto al procesar escalas más pequeñas es similar a aumentar el FOV de los filtros ya que cada filtro cubre una parte más grande de la imagen, mientras que las escalas más grandes están más orientadas a los detalles.

Por lo tanto, la supervisión multiescala, junto con la convolución *atrous* y el ASPP, permite que nuestras redes mejoren su rendimiento, al tener diferente información del área que rodea a cada uno de los píxeles de la imagen.

Técnicamente, partimos de un conjunto de imágenes, $\{I_1, I_2, \ldots, I_N\}$ con sus correspondientes anotaciones de segmentación semántica, $\{\phi_1, \phi_2, \ldots, \phi_N\}$. Usamos la red descrita, y representamos con $\hat{\theta}_i$ el mapa de puntuaciones a la salida de nuestra red, y con $\hat{\phi}_i$ nuestra estimación para la segmentación semántica.

Nuestro proceso de aprendizaje multiescala resuelve la siguiente optimización. Para cada imagen de entrenamiento, durante el aprendizaje, aplicamos una función soft max para mapear las puntuaciones dadas por nuestra red a una distribución de probabilidad sobre el conjunto completo de clases, C, por cada posición k en el mapa de salida de la

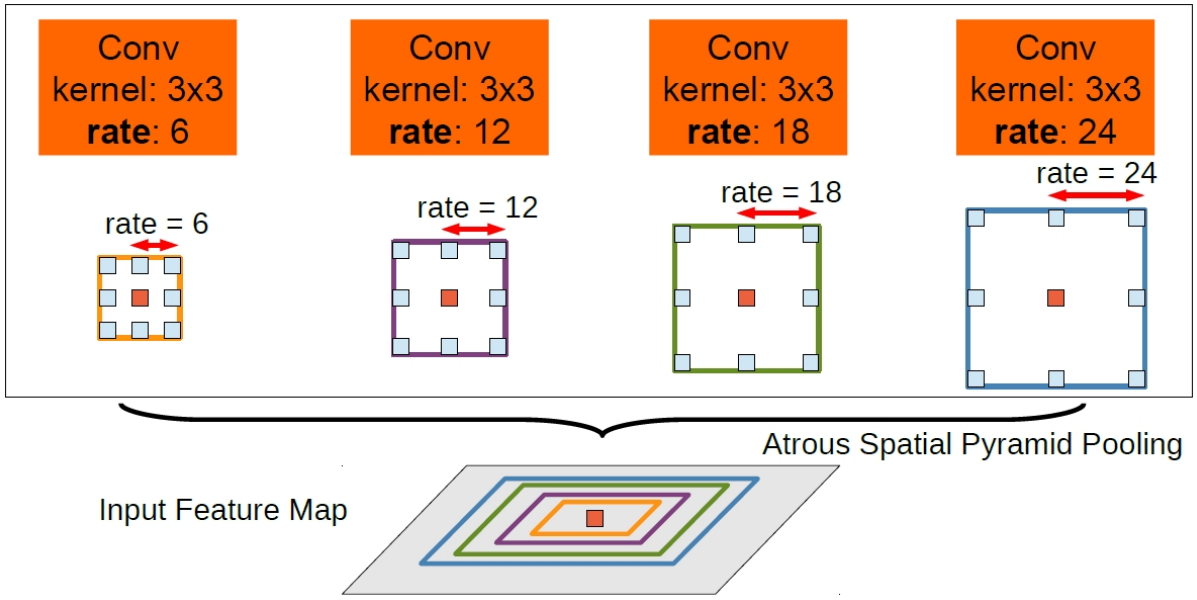


Figura 2.3: Atrous Spatial Pyramid Pooling toma información de diferentes escalas. Imagen extraída de [4].

CNN:

$$\hat{p}_{kc} = \frac{\exp(\hat{\theta}_{kc})}{\sum_{c'} \exp(\hat{\theta}_{kc'})}. \tag{2.1}$$

El resultado, $\hat{p}_{kc}$, se utiliza para obtener la entropía cruzada, ζ, función de pérdida con la que se optimiza nuestro modelo:

$$\zeta = \frac{-1}{K} \sum_{k=1}^{K} \log(\hat{p}_k, \phi), \tag{2.2}$$

de manera que nuestra estimación para cada píxel de la imagen, $\hat{\phi}$, sea la clase para la que la respuesta del mapa de salida de la CNN sea máxima:

$$\hat{\phi} = argmax(\hat{\theta}). \tag{2.3}$$

2.3.2. Método de Clasificación de Escenas

Con la irrupción del aprendizaje profundo y de las CNNs, la clasificación de escenas se ha llevado a cabo utilizando redes profundas diseñadas originalmente para el reconocimiento de objetos, logrando el estado del arte en la materia. En este trabajo, se pretende utilizar los resultados de la segmentación semántica para clasificar las escenas. Esta idea se basa en el hecho de que los objetos que aparecen en una imagen deben definir la escena

en la que se encuentran: las partes forman el todo. Así, demostramos que nuestros diseños sorprendentemente simples, flexibles y rápidos pueden superar los complejos modelos profundos entrenados para la categorización de escenas.

En primer lugar, proponemos utilizar una SVM con kernels aditivos [50]. Probamos varios de ellos como el de intersección de histogramas, χ^2 o Jenson Shannon, ya que han reportado un buen desempeño al trabajar con características de tipo histograma, de modo que, como descriptores para cada imagen, $\bar{F}_i$, tomamos un histograma de C intervalos, $hist$, representando el número de píxeles de cada una de las clases C en la estimación de la segmentación semántica $\hat{\phi}$:

$$F_i = hist(\hat{\phi}) = [f_{i1}, f_{i2}, ..., f_{iC}], \tag{2.4}$$

que normalizamos, según:

$$\bar{F}_i = norm(F_i), \tag{2.5}$$

donde f_{ic} es el número de píxeles de la clase c en la imagen i, y $norm$ corresponde a una normalización L2.

Además, al construir los histogramas, se propone el uso de una pirámide espacial de N niveles. Con el fin de añadir información de localización a nuestros descriptores, para el caso de N=2 niveles dividimos la imagen en cuatro partes y extraemos un histograma, siguiendo el procedimiento descrito en la ecuación 2.5, para cada uno de ellos. Así, los nuevos descriptores serán una concatenación de cinco vectores de tamaño C: el original, que representa la imagen completa, y uno para cada una de las cuatro partes de segundo nivel. Del mismo modo, para el nivel N, dividimos la imagen en 4^{N-1} partes concatenando los histogramas de cada parte con los de anteriores niveles. La figura 2.4 muestra gráficamente la construcción del descriptor para una imagen con una pirámide espacial de N=3 niveles.

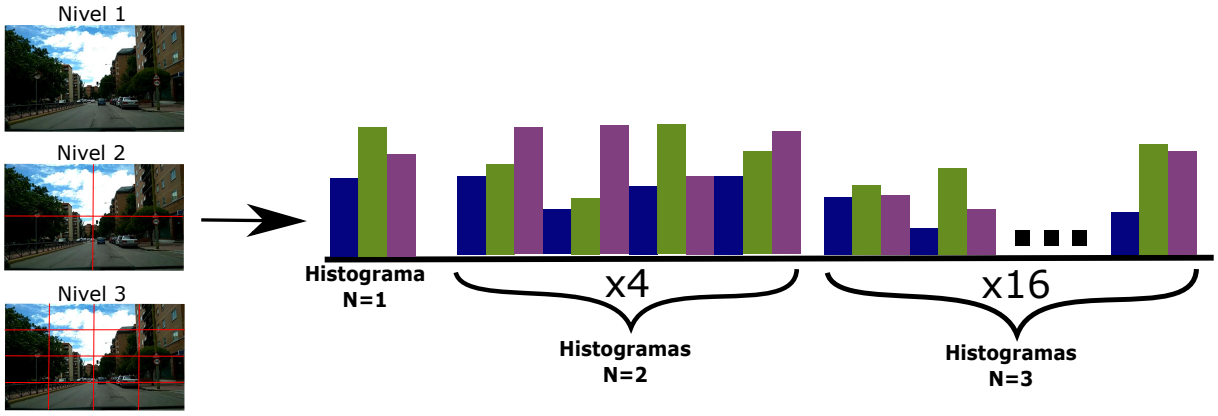


Figura 2.4: Ejemplo de pirámide espacial con N=3 niveles. El descriptor de la imagen es una concatenación de 21 histogramas: 1 del primer nivel, 4 del segundo nivel y 16 del tercer nivel.

Aparte de este método, proponemos una solución alternativa para el problema del reconocimiento de escenas. Introducimos lo que llamamos vectores de tipo *one-hot*, G_i, para alimentar una SVM. Se define un vector *one-hot* para cada imagen como:

$$G_i = [u(f_{i1}), u(f_{i2}), ..., u(f_{iC})] = [g_{i1}, g_{i2}, ..., g_{iC}], \tag{2.6}$$

siendo u la función:

$$u(x) = \begin{cases} 1 & x > \delta \\ 0 & x \leq \delta \end{cases}. \tag{2.7}$$

El parámetro umbral, δ, es el número mínimo de píxeles necesarios para considerar que una clase aparece en la imagen de forma que $g_1, ..., g_C \in \{0, 1\}$, es decir, un vector *one-hot* sólo tiene en cuenta si un objeto o clase está presente o no en la imagen, independientemente del número de píxeles que lo formen.

2.3.3. Método de Detección de Objetos

Dada la segmentación semántica podemos clasificar la escena, pero, ¿cómo podemos localizar los objetos? En esta sección describimos cómo obtener las *bounding boxes* (BB) de una máscara de segmentación para evaluar la precisión de detección de objetos de nuestro modelo.

Para extraer las *bounding boxes* para cada clase de objeto c a partir de las máscaras de segmentación semántica del *ground truth* y los resultados de la segmentación, primero proyectamos los píxeles ignorados (píxeles con etiqueta igual a 255) desde las máscaras del *ground truth* a las máscaras de segmentación generadas por nuestro modelo. Luego, para cada clase de objeto, obtenemos una máscara binaria, y trazamos los límites de la región poniendo una *bounding box* alrededor de cada componente conectada de píxeles pertenecientes a la clase c.

Por último, con las *bounding boxes* extraídas del resultado de la segmentación, asignamos a cada una de las cajas estimadas (BB_i) una puntuación, s_i, calculando la media de las puntuaciones, $\hat{\theta}_{jc}$, de los píxeles dentro de la *bounding box* que pertenecen a la clase c, siguiendo la siguiente ecuación:

$$s_i = \frac{1}{P_i^c} \sum_{j=1}^{P_i^c} \hat{\theta}_{jc}, \tag{2.8}$$

donde P_i^c representa el número total de píxeles de la clase c en la caja i (BB_i).

2.4. Bases de datos y detalles de la implementación

Para evaluar la efectividad de nuestra propuesta, proponemos la realización de experimentos en dos tipos de bases de datos con imágenes de tráfico e interiores. En el ámbito

de escenas de tráfico entrenaremos nuestro modelo de segmentación semántica en Cityscapes [9] y realizaremos clasificación de escenas en una base de datos generada a partir de secuencias extraídas de GSV. Por su parte, en la desafiante base de datos de escenas de interior NYUD2 [44] llevaremos a cabo las tres tareas que abarca la comprensión semántica de escenas: segmentación semántica, clasificación y detección de objetos. Los detalles de las bases de datos, experimentos y resultados obtenidos se describen en las siguientes secciones.

2.4.1. Cityscapes

Cityscapes es una base de datos que contiene imágenes de escenas de tráfico urbano en alta resolución (1024x2048 píxeles), recogidas en calles de 50 ciudades distintas, en diferentes épocas del año, condiciones de iluminación y horas del día. En concreto, la base de datos ofrece dos conjuntos de imágenes: un conjunto fino (*fine*), formado por 5000 imágenes, en el que la anotación de los píxeles se ha realizado teniendo especial cuidado con la transición (bordes) entre objetos; y, otro más basto (*coarse*), formado por 20000 imágenes para el que se ha priorizado el tiempo de anotación quedando sin etiquetar las fronteras entre clases. Para todos los experimentos se ha empleado exclusivamente el conjunto fino, que está dividido en 2975 imágenes de entrenamiento, 500 de validación y 1525 de test. Algunas imágenes de esta base de datos pueden verse en la figura 2.5. Cityscapes no aporta el *ground truth* de las imágenes de test, siendo necesario subir la segmentación obtenida para esas imágenes a un servidor para conocer los resultados. Por ello, todos los resultados aquí recogidos se darán sobre el conjunto de validación.

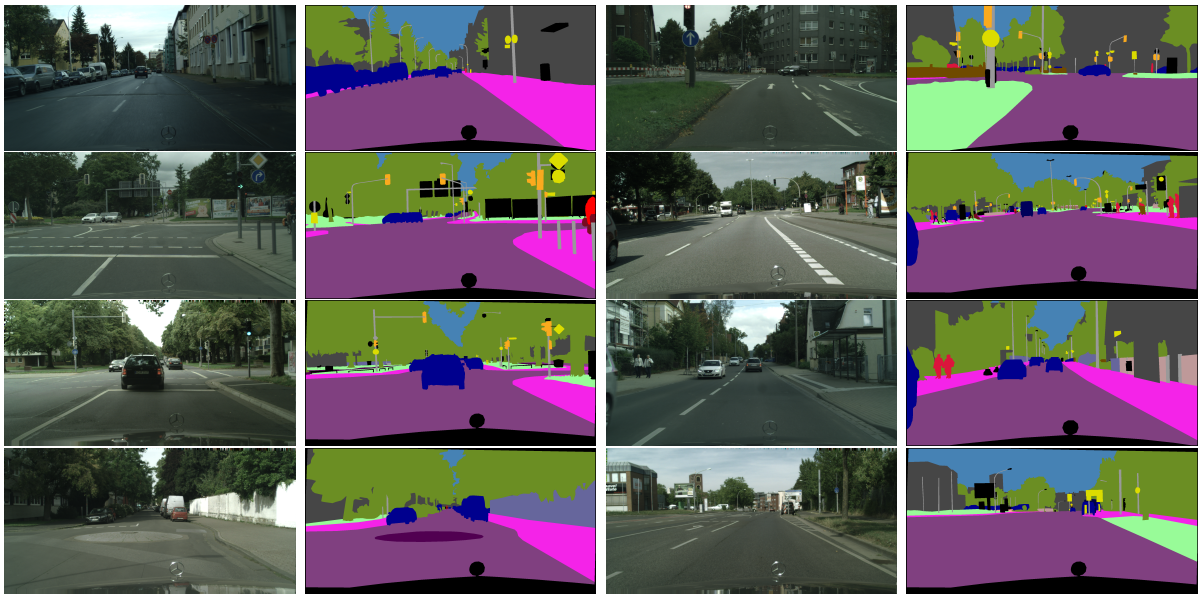


Figura 2.5: Imágenes de la base de datos Cityscapes junto a su segmentación semántica.

Para el **entrenamiento** en esta base de datos, emplearemos un *learning rate* inicial

de 10^{-3} para la VGG-16 y de $2{,}5 \times 10^{-4}$ para la ResNet-101. Además, es conocido que reducir el *learning rate* a lo largo de las iteraciones mejora el entrenamiento. Por ello, se van a evaluar las dos políticas más empleadas en Caffe: *step*, función escalón que disminuye el *learning rate* cada cierto número configurable de iteraciones; y *poly*, para la que la variación del *learning rate* responde a la ecuación $initial_{lr} \times (1 - \frac{iter}{max_{iter}})^{power}$, fijando *power* a 0.9. El *momentum* lo establecemos en 0.9, el *weigth decay* en 5×10^{-4} y el número de iteraciones en 60000.

Además, como técnicas de *data augmentation*, se van a utilizar: (1) *mirroring*, para la que, por cada imagen, la red entrenará, con una probabilidad del 50 %, bien con la original, bien con la imagen reflejada; y (2) *cropping*, en lugar de entrenar con la imagen completa, le pasaremos parches aleatorios de tamaño 513x513, con una doble finalidad, que la red sea capaz de aprender características aún en ausencia de todo el entorno y, especialmente, solventar problemas de limitación de memoria que impiden entrenar con la imagen a resolución completa. Además, con el de fin de asegurarnos de que los parches que coge la red están repartidos, vamos a dividir las imágenes de entrenamiento en dos mitades solapadas, de tamaño 1024x1536. De este modo, incrementamos la probabilidad de que la red pase por toda la imagen a lo largo de las épocas. Nótese que sin solapamiento sería imposible que la red tomase parches del centro de la imagen.

En la fase de **inferencia**, la profundidad de estos modelos (especialmente ResNet-101) junto al gran tamaño de las imágenes de Cityscapes, hacen que no sea posible realizar el test en un solo paso a resolución original. Para solventar estos problemas de limitación de memoria de las tarjetas GPUs empleadas, se han evaluado dos posibilidades: (1) reescalar (*downsample*) las imágenes de test por un factor de 4 (de 1024x2048 a 512x1024), obtener la segmentación semántica y reescalar (*upsample*) el resultado devolviendo la imagen a la resolución original; y (2) dividir las imágenes de test en diferentes partes, 5 en este trabajo, con solapamiento, tal y como se muestra en la Figura 2.6.

Al emplear la opción (2), es importante notar que a la salida es necesario recomponer la imagen original y, debido al solapamiento, es posible que existan discrepancias en las clases asignadas a esos píxeles comunes. El procedimiento seguido para resolver estas divergencias, consiste en sumar, para cada píxel, los *scores* de salida para cada clase y optar por el mayor.

Los resultados muestran que procesar las imágenes a resolución original tiene una mejora significativa en los resultados.

2.4.2. Google Street View

Debido a que en Cityscapes todas las imágenes forman parte de recorridos urbanos, para llevar a cabo clasificación de escenas de tráfico se ha generado una base de datos a partir de secuencias extraídas mediante GSV. Dado que uno de nuestros objetivos es evaluar la posibilidad de realizar reconocimiento de escenas a partir de la segmentación semántica, se han establecido dos divisiones: (1) una primera parte con 62 secuencias

Figura 2.6: División de las imágenes de test para solventar problemas de limitación de memoria. Nótese el solapamiento existente entre ellas para el evitar efecto borde.

extraídas de carreteras españolas, y (2) una segunda mitad con recorridos tomados de Suiza. De este modo, al ser Suiza uno de los países que forma parte de Cityscapes, podemos analizar la influencia que tiene el tener nuestra red de segmentación entrenada en un entorno similar al que vamos a realizar la clasificación.

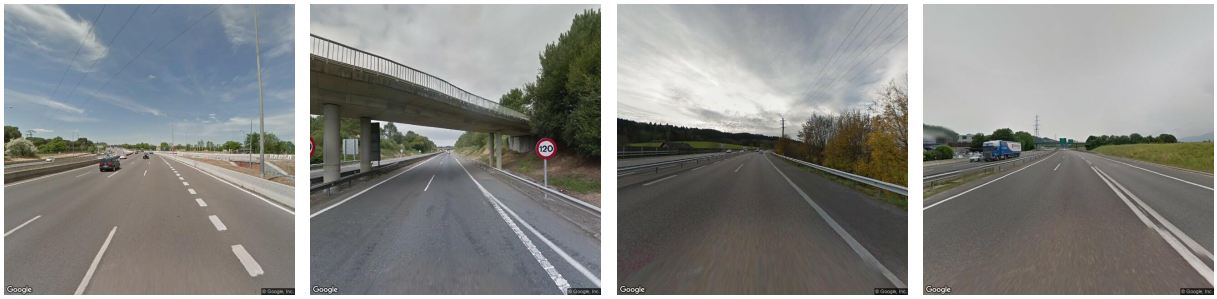
Las 62 secuencias de la base de datos de España conforman un total de 13981 imágenes de las cuales 9052 las emplearemos como imágenes de entrenamiento y el resto, 4929, como imágenes de test. La base de datos de Suiza, está formada por 2581 imágenes de entrenamiento y 697 de test repartidas en 18 secuencias. Todas las imágenes han sido manualmente etiquetadas dividiéndolas en tres clases: autovía, carretera secundaria y entorno urbano. En la Figura 2.7 se muestran algunas de estas imágenes.

2.4.3. NYU Depth v2

La base de datos NYU Depth v2, muy empleada en robótica, está formada por secuencias de vídeo de escenas de interior grabadas mediante la cámara Kinect, que aporta tanto información RGB como de profundidad. De todas esas secuencias, existen 1449 imágenes etiquetadas tanto a nivel de píxel (segmentación semántica), con más de 800 clases diferentes, como de la escena que representan (p. ej. *kitchen, office o living room*). La resolución de las imágenes es de 480x640 píxeles. La figura 2.8 muestra alguna de ellas.

Esas 1449 imágenes, tradicionalmente [35], se han dividido en dos conjuntos, formados por 795 imágenes de entrenamiento y 654 imágenes de test.

Igualmente, de las más de 800 clases distintas para segmentación semántica, algunos



(a) Autovía

(b) Secundaria

(c) Urbano

Figura 2.7: Imágenes de la base de datos Google Street View con sus etiquetas: (a) autovía, (b) carretera secundaria y (c) entorno urbano. Las dos primeras imágenes de cada tipo de vía pertenecen a España, mientras que las dos últimas están extraídas de Suiza.

trabajos han reducido este número agrupando los diferentes objetos que representan. Así, en el trabajo de Silberman *et al.* [35] se propuso realizar una segmentación entre cuatro posibles grupos de objetos: *ground, furniture, structure* y *props*. Mientras, [21] elevó ese número hasta 40 clases, entre las que se incluyen las 37 clases más representadas de las 800 originales, quedando el resto repartidas en otras 3.

Para el **entrenamiento** del modelo de **segmentación semántica**, el número de iteraciones será de 20000 con tamaño de *batch* de 20 para VGG-16 y 1 para la ResNet-101. Además, la política de *learning rate* empleada será de tipo polinómica con un valor inicial de 10^{-3} para ambas redes. Igualmente, como formas de *data augmentation*, los parches que tomaremos serán de 385x385, manteniendo el uso del *mirroring*.

En cuanto a la **inferencia**, será realizada dividiendo las imágenes en dos partes con

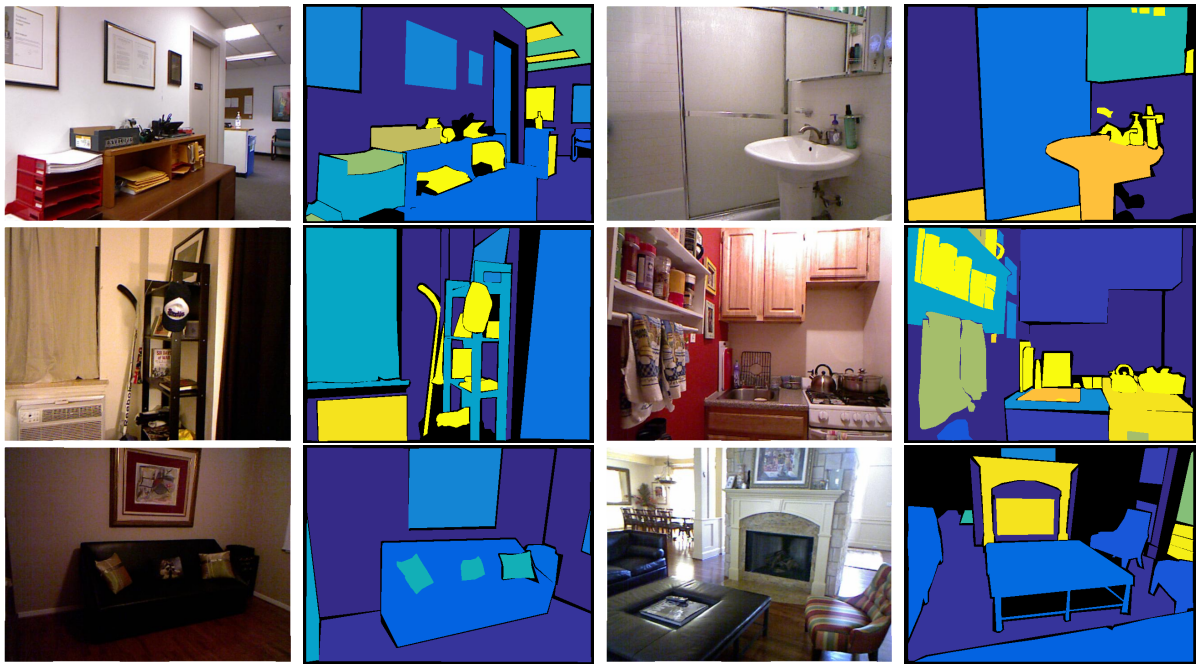


Figura 2.8: Imágenes de la base de datos NYUD2 junto a su segmentación semántica.

solapamiento, por las cinco de Cityscapes, y procediendo del mismo modo.

2.5. Resultados

2.5.1. Escenas de Tráfico

2.5.1.1. Segmentación semántica

Para cuantificar la validez de nuestros modelos, la métrica de evaluación empleada será la *mean Intersection Over Union* (mIOU) que está muy extendida en la comunidad y que se define, para cada clase, como:

$$\frac{TP}{TP + FP + FN}, \tag{2.9}$$

siendo TP los *true positives*, píxeles clasificados correctamente; FP los *false positives*, píxeles que en la anotación aparecen como una clase distinta, pero se predicen como la clase de interés; y FN, los *false negatives*, píxeles anotados como la clase de interés pero a los que la red les ha asignado una clase distinta. La tabla 2.1 muestra los resultados, al emplear VGG o ResNet multiescala como red base, para las 19 clases en el conjunto de validación de Cityscapes. ResNet-101 mejora los resultados de VGG-16 para todas las clases (7.1 % de mIOU), estando su principal aporte en las clases que aparecen menos representadas donde la mejora es muy significativa (p. ej. *train* cuya mIOU aumenta un 24.4 %, *traffic light*, con un incremento del 16.8 % o *pole* donde la mejora es de un 16.1 %).

La tabla 2.2 compara nuestros resultados con otros modelos del estado del arte, indicando el conjunto de entrenamiento con el que están entrenados.

Tabla 2.1: Resultados cuantitativos: IoU para cada una de las clases por cada modelo.

	Road	**Sidewalk**	**Build.**	**Wall**	**Fence**	**Pole**	**T.Light**	**T.Sign**	**Veget.**	**Terrain**
VGG-16	96.8	77.8	88.0	43.8	49.4	31.9	40.8	62.8	89.1	59.3
ResNet-101	97.4	78.4	90.8	47.9	50.7	48.0	57.6	70.5	91.1	58.1
	Sky	**Person**	**Rider**	**Car**	**Truck**	**Bus**	**Train**	**Motorc.**	**Bicyc.**	**mIoU**
VGG-16	91.8	71.3	45.7	90.9	51.3	71.6	46.1	52.1	67.5	**64.6**
ResNet-101	93.4	77.5	54.7	93.6	68.4	81.0	69.5	60.6	72.9	**71.7**

Tabla 2.2: Comparativa de resultados con otros modelos en segmentación semántica en Cityscapes.

Modelo	¿Conjunto fino?	¿Conjunto basto?	**IoU**
DeepLabV3+ [6]	X	X	82.1
PSPNet [59]	X	X	81.2
Dilation10 [58]	X		67.1
FCN 8s [30]	X		65.3
ENet [38]	X		58.3
Nuestro - VGG-16	X		64.6
Nuestro - ResNet-101	X		71.7

La Figura 2.9 muestra los resultados cualitativos de la segmentación. Dicha visualización se corresponde con el mejor modelo conseguido, esto es, con *poly* como política de *learning rate*, procesando las imágenes en test a resolución completa con división en cinco partes con solapamiento y empleando el CRF como posprocesado.

2.5.1.2. Clasificación de escenas

La tabla 2.3 muestra los resultados obtenidos con los distintos métodos evaluados (histogramas de la segmentación semántica con normalización L2, vectores *one-hot* obtenidos a partir de la segmentación semántica y CNNs) sobre la base de datos de Google Street View, tanto para el conjunto de secuencias grabadas en España como para el de Suiza. La métrica de evaluación es la precisión de clasificación, esto es, el porcentaje de imágenes del

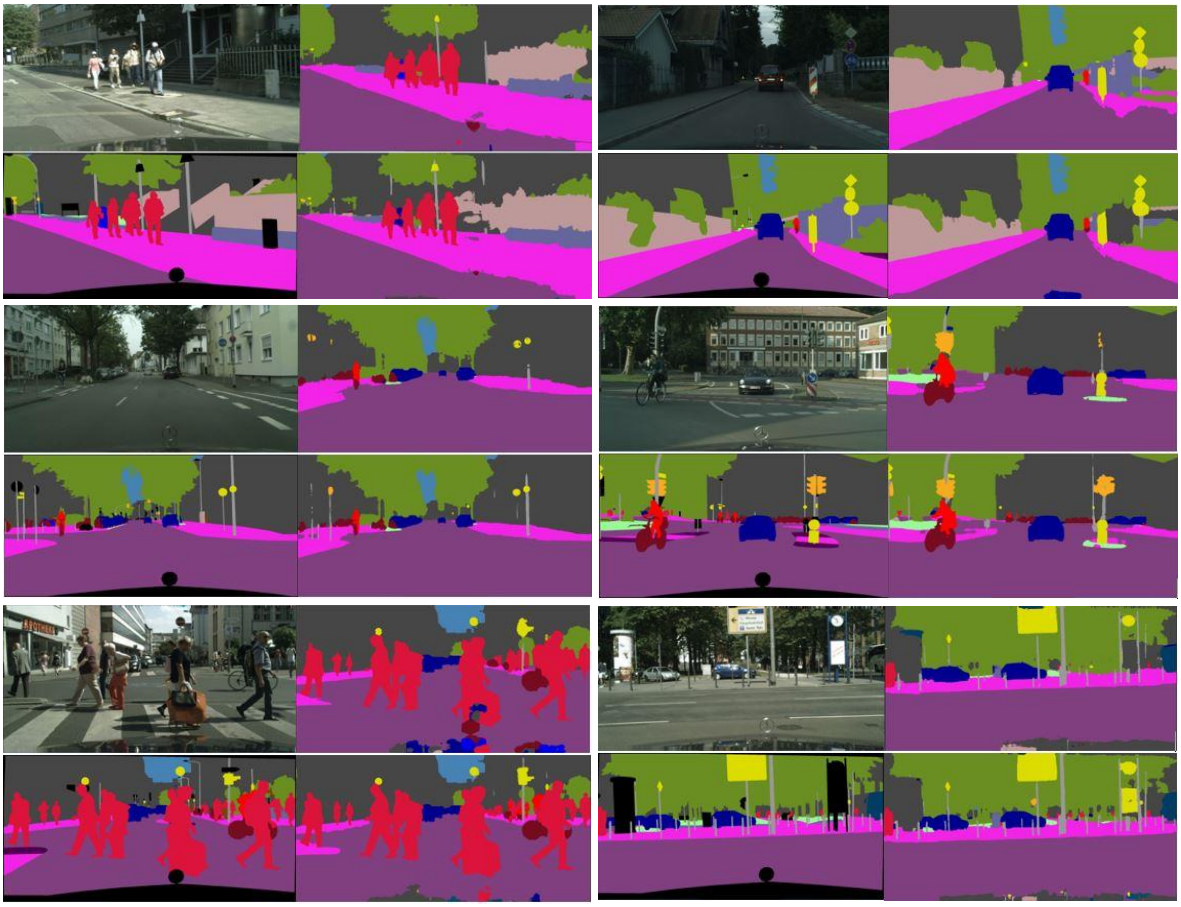


Figura 2.9: Resultados cualitativos de la segmentación semántica en Cityscapes. Por cada grupo de cuatro imágenes se muestra la imagen original RGB (arriba a la izquierda), el *ground truth* (abajo a la izquierda), el resultado de la segmentación con una VGG (arriba a la derecha) y el resultado de la segmentación con una ResNet como red base (abajo a la derecha).

conjunto de test correctamente clasificadas. A la vista de los resultados, podemos extraer varias conclusiones.

- En primer lugar, en esta base de datos, las redes neuronales profundas obtienen mejores resultados de clasificación que las SVMs basadas en la segmentación semántica. Así, en España, el mejor modelo de CNNs (VGG-16 con política de *learning rate* de tipo *step*) clasifica correctamente un 6 % más de imágenes que el mejor modelo basado en SVMs (kernel de intersección, con pirámide espacial de 4 niveles empleando los histogramas normalizados de la segmentación semántica). En Suiza, las redes neuronales clasifican correctamente casi el 100 % de las imágenes, mientras que una SVM con kernel de tipo Jenson Shannon y tres niveles espaciales se queda en un 96,41 %.

- Introducir información de localidad espacial mejora considerablemente los resultados. Todos nuestros modelos basados en SVMs con diferentes kernels, mejoran al aumentar los niveles de la pirámide espacial. En la mayoría de los casos, el mejor resultado se alcanza para cuatro niveles; sólo en algunos casos satura al emplear ese cuarto nivel, haciendo que el resultado con tres niveles sea ligeramente mejor.

- Por último, que el modelo de segmentación semántica esté entrenado sobre imágenes similares a las que luego vamos a clasificar (entendiendo como tal, el país o territorio en el que están tomadas) influye en los resultados. Así, en general, la precisión de clasificación en Suiza es bastante superior a la obtenida sobre imágenes de España.

Para comprobar dónde se producen los errores, la figura 2.10 muestra las matrices de confusión de algunos de nuestros modelos. En **España** (2.10a y 2.10b), una parte importante del error viene al confundir la clase 3 (carretera secundaria), con la clase 2 (autovía). Asimismo, los errores en la clase 1 (entorno urbano) se reparten indistintamente entre las clases 2 y 3. En **Suiza** (2.10c y 2.10d), mientras que la clase 1 es clasificada perfectamente, entre las clases 2 (autovía) y 3 (carretera secundaria) es donde de nuevo se produce un mayor número de confusiones. Esto es algo que entendemos como lógico, ya que, mientras que en las escenas urbanas aparecen elementos muy característicos (fundamentalmente una mayor presencia de edificios y la existencia de peatones) que las hacen muy distinguibles, la diferencia entre una vía secundaria y una autovía (en las que dos clases -cielo y carretera- dominan la mayor parte de las imágenes), teniendo en cuenta que nuestras características son los histogramas de la segmentación semántica, no es tan evidente.

Finalmente, la tabla 2.4 muestra los resultados al convertir el problema de clasificación en biclase. Para ello, las clases autovía y carretera secundaria han sido fusionadas en una sola: entorno no urbano. Para este nuevo escenario se repiten algunos resultados:

- La precisión de clasificación en Suiza es superior a la de España

- Emplear información de localidad mejora considerablemente los resultados.

- Utilizar histogramas normalizados es mejor que vectores *one-hot*.

Son resultados coherentes con lo que veíamos anteriormente, en Suiza, donde el error se concentraba por completo entre las clases 2 y 3, al unificarlas, obtenemos una tasa de acierto muy cercana al 100 %. En España, el error entre las clases 2 y 3 suponía también una parte importante del error total, por lo que al simplificar el problema, conseguimos reducir a la mitad el error de clasificación, que se queda en entorno al 12 % para los mejores modelos.

Tabla 2.3: Resultado de la clasificación de escenas en la base de datos Google Street View con tres posibles clases: entorno urbano, carretera secundaria y autovía.

Método	**Kernel**	**Niveles**	**Precisión España**	**Precisión Suiza**
Histogramas SS L2	Lineal	1	69,36	90,10
		2	73,18	90,39
		3	75,49	91,68
		4	75,76	93,11
	Intersección	1	70,28	86,23
		2	74,29	87,09
		3	76,04	91,39
		4	77,85	93,26
	Chi cuadrado	1	70,38	88,09
		2	73,85	90,10
		3	76,02	95,98
		4	77,36	94,84
	Jenson Shannon	1	70,36	88,67
		2	73,77	92,25
		3	75,61	96,41
		4	76,95	95,84
Vectores *One-hot*	Lineal	1	63,01	66,43
		2	71,13	89,24
		3	73,79	91,39
		4	75,65	94,84
	Intersección	1	63,06	66,43
		2	71,03	87,66
		3	73,85	91,68
		4	74,29	95,41
	Chi cuadrado	1	63,06	66,43
		2	71,03	87,66
		3	73,85	91,68
		4	74,32	95,41
	Jenson Shannon	1	63,42	66,28
		2	70,91	88,38
		3	73,91	91,82
		4	74,27	95,41
Método	**Red**	**Política lr**	**Precisión España**	**Precisión Suiza**
Redes neuronales	VGG-16	Step	83,91	99,00
		Poly	83,63	99,14
	ResNet-101	Step	81,72	99,43
		Poly	81,45	99,71

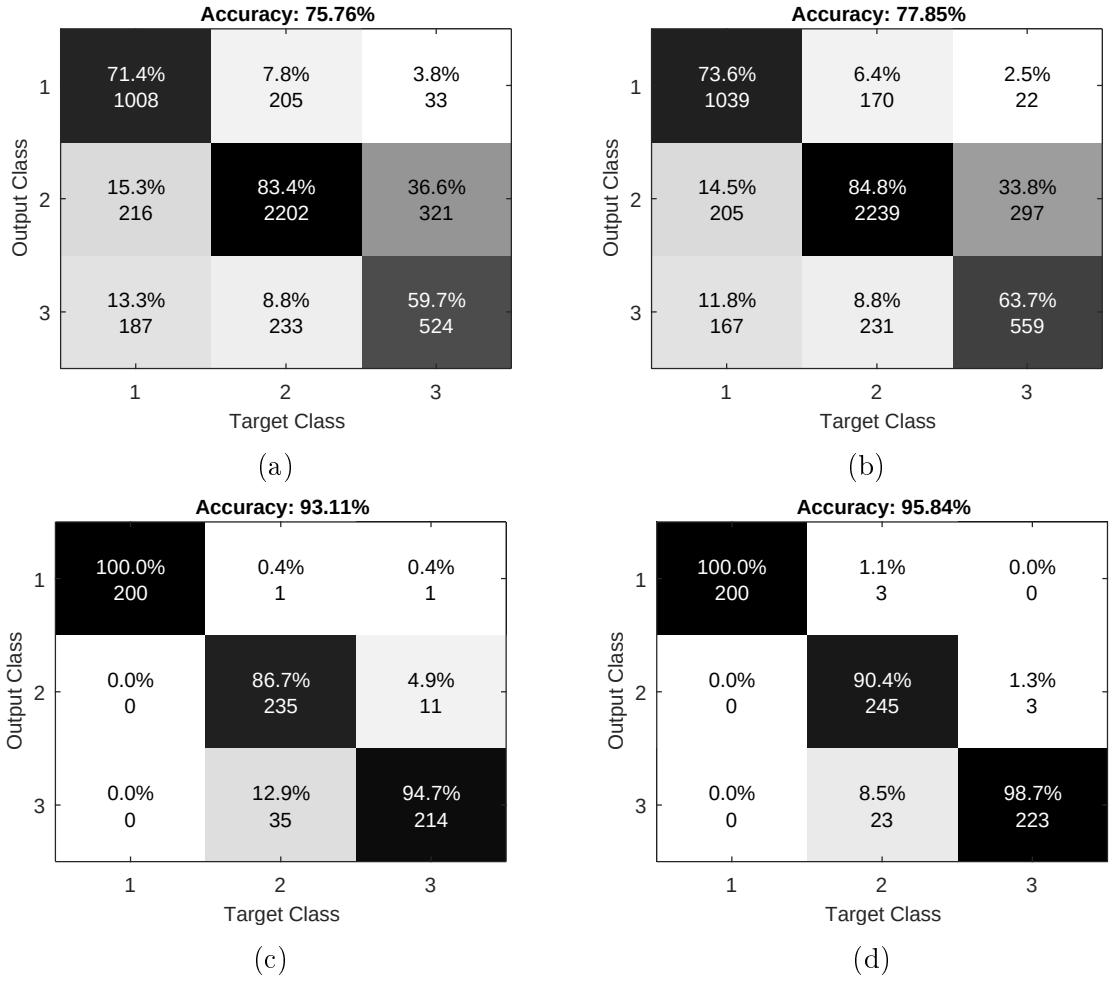


Figura 2.10: Se muestran matrices de confusión de: (a) kernel lineal con 4 divisiones e histogramas con normalización L2 sobre el conjunto de test de España, (b) kernel intersección con 4 divisiones e histogramas con normalización L2 sobre el conjunto de test de España, (c) kernel lineal con 4 divisiones e histogramas con normalización L2 sobre el conjunto de test de Suiza y (d) kernel Jenson Shannon con 4 divisiones e histogramas con normalización L2 sobre el conjunto de test de Suiza.

Tabla 2.4: Resultado de la clasificación de escenas en la base de datos Google Street View con dos posibles clases: entorno urbano y entorno no urbano.

Método	**Kernel**	**Niveles**	**Precisión España**	**Precisión Suiza**
Histogramas SS L2	Lineal	1	82,33	99,57
		2	85,72	99,57
		3	86,69	99,86
		4	87,58	100
	Intersección	1	84,36	97,70
		2	87,46	97,70
		3	87,99	97,85
		4	88,54	97,99
	Chi cuadrado	1	84,36	97,56
		2	86,93	97,70
		3	87,97	98,57
		4	88,33	98,71
	Jenson Shannon	1	84,26	97,42
		2	86,91	97,85
		3	87,73	99,00
		4	88,05	99,14
Vectores *One-hot*	Lineal	1	76,95	86,51
		2	83,85	98,57
		3	86,31	99,86
		4	86,85	100
	Intersección	1	77,64	86,51
		2	83,77	98,57
		3	86,12	99,86
		4	84,70	100
	Chi cuadrado	1	77,64	86,51
		2	83,77	98,57
		3	86,12	99,86
		4	84,70	100
	Jenson Shannon	1	77,64	86,66
		2	83,87	98,57
		3	86,10	99,86
		4	84,72	100
Método	**Red**	**Política lr**	**Precisión España**	**Precisión Suiza**
Redes neuronales	VGG-16	Step	89,42	100
		Poly	89,57	100
	ResNet-101	Step	89,69	100
		Poly	89,49	100

2.5.2. Escenas de interior

2.5.2.1. Segmentación semántica

En este apartado, vamos a comparar nuestro modelo de segmentación semántica multiescala con otros métodos que forman el estado del arte en la base de datos NYUD2. Para ello, de las 894 clases de objetos diferentes existentes en NYUD2, vamos a considerar las 37 más representativas, agrupando el resto en otras tres clases: *otras estructuras*, *otros muebles* y *otros accesorios* tal y como se describe en [34], quedando la tarea de segmentación semántica reducida a 40 clases.

En los últimos años se han publicado varios métodos, algunos de los cuales utilizan información de profundidad para mejorar sus resultados. La tabla 2.5 muestra que nuestro modelo, al utilizar ResNet como red base con el procedimiento multiescala descrito anteriormente, consigue superar a todos los demás, en las tres métricas (precisión a nivel de píxel, precisión media por clase y IOU) evaluadas. En la figura 2.11 se presentan una serie de ejemplos cualitativos.

Tabla 2.5: Resultados de la segmentación semántica para 40 clases. Nuestro modelo, sin emplear información de profundidad, consigue resultados del estado del arte.

Método	**Entrada**	**prec. píxel**	**prec. clase**	**IoU**
RCNN [21]	RGB-HHA	60.3	35.1	28.6
FCN-16 [30]	RGB-HHA	65.4	46.1	34.0
Eigen *et al.* [14]	RGB-D-N	65.6	45.1	34.1
FuseNet-SF3 [25]	RGB-D	66.4	44.2	34.0
Context-CRF [28]	RGB	67.4	49.6	37.1
MVCNet [33]	RGB-D	69.1	50.1	38.0
Nuestro - VGG-16	RGB	67.4	48.5	36.3
Nuestro - ResNet-101	RGB	**70.9**	**52.2**	**41.8**

Finalmente, los resultados de la IOU para cada una de las 40 clases se dan en la tabla 2.6. Nuestro modelo supera a todos los demás métodos que han reportado estos resultados detallados, obteniendo una mejora particularmente considerable en las clases más difíciles. Por lo tanto, nuestra solución de segmentación semántica, define el nuevo estado del arte en esta base de datos.

2.5.2.2. Clasificación de escenas

Las imágenes de la base de datos NYUD2 están divididas 27 clases de escenas de interior. Sin embargo, sólo unas pocas de esas clases están bien representadas en todo el

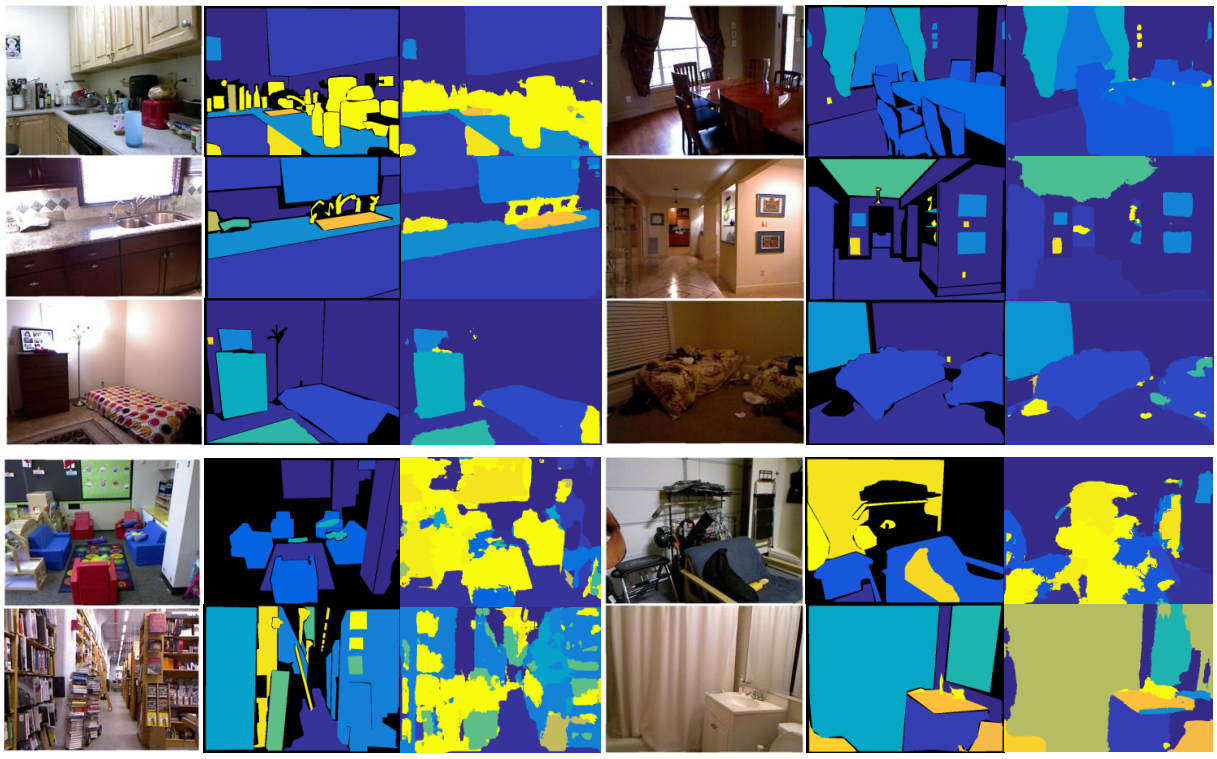


Figura 2.11: Resultados cualitativos de segmentación semántica en la NYUD2. Por cada grupo de tres imágenes, se muestra de izquierda a derecha, la imagen RGB, el *ground truth* y nuestro resultado cuando empleamos ResNet como red base. Las tres primeras filas se corresponden con las mejores imágenes de test en precisión de píxel, mientras que las dos últimas filas, son las peores.

conjunto. Por ello, seguimos el procedimiento descrito en [34] tomando las 9 clases más populares y combinando el resto en una décima clase.

La tabla 2.7 muestra los resultados de nuestros métodos cuando los comparamos con otros modelos del estado del arte. Para la evaluación, establecemos como referencias dos CNNs para clasificación, una VGG-16 [45] y una ResNet-101 [26], que hemos entrenado para reconocer directamente las escenas.

Tomar los histogramas normalizados de la segmentación semántica como características para alimentar una SVM lineal ya mejora todos los métodos previamente reportados y se mantiene muy cerca de la ResNet-101 entrenada para clasificación. Si en su lugar usamos vectores *one-hot* como se describe en la Sección 2.3.2, aumentamos la precisión en un 1,3 %. Usar un kernel de tipo Jenson-Shannon, con histogramas normalizados, obtiene otro 0,3 % de mejora y, finalmente, crear una pirámide espacial de dos niveles mejora el rendimiento en un 0,76 % adicional, alcanzando una precisión de clasificación del 68,96 %.

Figura 2.12: Resultados cualitativos para la tarea de detección de objetos. En verde se muestra las *bounding boxes* del *ground truth* y en azul nuestra predicción.

2.5.2.3. Detección de objetos

Aquí estamos interesados en investigar la tarea de detectar clases de objetos en interiores. La base de datos NYUD2 [44] se propuso originalmente para estudiar la segmentación semántica e inferir superficies de apoyo como mesas o sillas. Sin embargo, podemos derivar *bounding boxes* siguiendo el método descrito en la sección 2.3.3 y evaluar así las posibilidades de detección de objetos que tiene nuestro modelo.

Seguimos la propuesta de [34], y de las 40 categorías de objetos que hemos segmentado, detectamos las 17 clases más representativas. La tabla 2.8 resume los resultados obtenidos por nuestro modelo, comparándolos con los distintos trabajos de Gupta *et al.* [34, 20, 21]. Observamos que somos capaces de superar a todos esos métodos, los cuales explotan conjuntamente la información de profundidad y apariencia. Además, logramos una mejora de más del 5 % en comparación con los mejores resultados presentados por Gupta *et al.* [21]. En la figura 2.12 proporcionamos algunos resultados cualitativos de detección.

Tabla 2.6: IoU para cada una de las 40 clases e la segmentación semántica en NYUD2.

	wall	**floor**	**cabinet**	**bed**	**chair**	**sofa**	**table**	**door**	**window**	**book sh.**
[44]-SC	60.7	77.8	33.0	40.3	32.4	25.3	21.0	5.9	29.7	22.7
[42]	32.5	28.2	16.6	12.9	27.7	17.3	32.4	38.6	26.5	10.1
[34]	68	44	30	8.3	33	20	40	47	44	10
[21]	47.9	47.9	29.9	20.3	32.6	18.1	40.3	51.3	42.0	11.3
[20]	41.3	47.6	29.5	12.9	34.8	18.1	40.7	51.7	41.2	6.7
Our-ResNet	55.0	61.0	40.9	33.1	45.0	39.9	54.5	53.0	57.1	14.5
	picture	**counter**	**blinds**	**desk**	**shelves**	**curtain**	**dresser**	**pillow**	**mirror**	**floor mat**
[44]-SC	35.7	33.1	40.6	4.7	3.3	27.4	13.3	18.9	4.4	7.1
[42]	60.0	74.4	37.1	42.2	6.1	27.6	7.0	19.7	17.9	20.1
[34]	81	48	55	40	5.1	34	22	28	19	22
[21]	68.0	81.3	44.9	65.0	3.5	29.1	34.8	34.4	16.4	28.0
[20]	67.9	81.5	45.0	60.1	5.2	26.9	25.0	32.8	21.2	30.7
Our-ResNet	74.0	80.6	55.9	64.6	10.9	46.6	40.2	37.0	34.0	28.4
	clothes	**ceiling**	**books**	**fridge**	**tv**	**paper**	**towel**	**sh. curt.**	**box**	**wh. board**
[44]-SC	6.5	73.2	5.5	1.4	5.7	12.7	0.1	3.6	0.1	0
[42]	9.5	53.9	14.8	1.9	18.6	11.7	12.6	5.4	3.3	0.2
[34]	6.9	59	4.4	15	9.3	1.9	14	18	4.8	37
[21]	4.7	60.5	6.4	14.5	31.0	14.3	16.3	4.2	2.1	14.2
[20]	7.7	61.2	7.5	11.8	15.8	14.7	20.0	4.2	1.1	10.9
Our-ResNet	17.5	63.5	29.2	57.5	57.3	27.3	33.9	17.8	9.6	32.1
	person	**night st.**	**toilet**	**sink**	**lamp**	**bath tub**	**bag**	**other str.**	**other fur.**	**other pr.**
[44]-SC	6.6	6.3	26.7	25.1	15.9	0	0	6.4	3.8	22.4
[42]	13.6	9.2	35.2	28.9	14.2	7.8	1.2	5.7	5.5	9.7
[34]	16	20	50	26	6.8	33	0.65	6.9	2	22
[21]	0.2	27.2	55.1	37.5	34.8	38.2	0.2	7.1	6.1	23.1
[20]	1.4	17.9	48.1	45.1	31.1	19.1	0.0	7.6	3.8	22.6
Our-ResNet	76.9	41.8	73.7	50.2	44.6	31.5	7.3	25.6	15.5	32.5

Tabla 2.7: Precisión en clasificación de escenas en NYUD2. Establecemos dos referencias: VGG-16 and ResNet-101, y comparamos nuestros métodos con los modelos que conforman el estado del arte.

	Método	**Entrada**	**Precisión.**
Estado del arte	[21]	RGB-D	45.4
	[51]	RGB-D	63.9
	[46]	RGB-D	65.8
Referencias	VGG-16 clasificación	RGB	64.37
	ResNet-101 clasificación	RGB	67.73
Nuestros modelos	**SVM Lineal con histogramas**	RGB	66.51
	SVM Lineal con vectores one-hot	RGB	67.89
	SVM con Kernel Aditivo	RGB	68.20
	SVM con Pirámide Espacial y Kernel Aditivo	RGB	**68.96**

Tabla 2.8: Resultados de la detección de objetos en la base de datos NYUD2 [44]. Empleamos como métrica la precisión media (AP). Comparamos nuestro método, que pone *bounding boxes* a la segmentación semántica obtenida por nuestro modelo, con los resultados reportados por Gupta *et al.* [34, 20, 21].

Input	[34] **RGBD**	[20] **RGBD**	[21] **RGBD**	**Nuestro método RGB**
Bed	52.1	56.0	**66.5**	58.7
Chair	6.4	23.5	40.8	**43.9**
Sofa	17.5	34.2	42.8	**52.7**
Counter	32.7	24.0	37.6	**51.3**
Lamp	1.4	26.7	**29.3**	27.7
Pillow	3.3	20.7	**37.4**	19.2
Sink	14.0	22.8	24.2	**46.1**
Table	9.3	17.2	**24.3**	23.4
Bathtub	28.4	19.3	22.9	**33.7**
Television	3.1	19.5	37.2	**42.7**
Bookshelf	6.7	17.5	21.8	**35.1**
Toilet	13.3	41.5	53.0	**74.0**
Box	0.7	0.6	**3.0**	2.5
Desk	0.8	6.2	**10.2**	7.0
Door	5.0	9.5	20.5	**23.3**
Dresser	13.3	16.4	26.2	**26.5**
Night-stand	–	32.6	39.5	**43.5**
Media sobre 16 classes	13	24.3	31.1	**35.5**
Media sobre 17 classes	–	23.0	31.6	**36.0**

Capítulo 3

Estimación de la velocidad para un vehículo en función de la situación del tráfico

3.1. Introducción

Estimar y conocer la velocidad adecuada para un vehículo en una determinada situación vial resulta fundamental tanto como sistema de información, alertando al conductor de posibles discrepancias entre la velocidad recomendada estimada y la real del vehículo, como entrada en los sistemas de control que permita adecuar la velocidad del automóvil a la recomendada, en el caso de vehículos con mayor nivel de autonomía. El exceso de velocidad continúa siendo uno de los principales factores de accidente en carreteras. De hecho, el 72 % de atropellos en ciudad podrían evitarse con una velocidad adecuada del vehículo según la Fundación MAPFRE y es la causa responsable de la muerte de 500 personas cada semana en carreteras europeas según el Consejo Europeo de Seguridad en el Tranporte.

Los sistemas ISA tradicionales se basan en el empleo de un GPS que, junto a un dispositivo de almacenamiento con información de la red de carreteras, permiten determinar la posición del vehículo y conocer la velocidad límite en cada sección de la vía, alertando al conductor cuando excede esa velocidad, entra en un zona con una nueva velocidad máxima permitida, o se aproxima a zonas peligrosas como colegios. Otras variantes de sistemas ISA, basados en visión artificial, se fundamentan en la interpretación de determinados elementos viales, como las señales de tráfico, para ayudar al conductor a mantener una velocidad segura.

Así, aunque la velocidad límite de la vía proporcionada por sistemas ISA basados en GPS es útil, no debemos confundirla con la velocidad adecuada, que es la que se adapta a la situación del tráfico en cada momento. En efecto, para una misma vía, la fluidez del tráfico, la visibilidad o la presencia de obstáculos en la carretera, han de influir en

la velocidad recomendada en cada instante. Por tanto, la velocidad adecuada a una vía depende de factores que un GPS no puede cuantificar (ver figura 3.1).

Figura 3.1: Aunque la velocidad límite en ambos sentidos de esta autovía sea la misma, resulta evidente que la velocidad recomendada debe ser muy diferente. ¿Puede una cámara adquirir dicha información?

Partiendo de este punto, ¿es posible mejorar o complementar dicha información, procesando las imágenes recogidas por una cámara ubicada en la parte frontal del vehículo? ¿Es posible conseguir un modelo que, a partir de una imagen, sea capaz de predecir la velocidad adecuada que éste debería llevar? Nótese la dificultad de la tarea propuesta, ¿puede un ser humano, a partir de una simple imagen, discernir entre si un vehículo debe ir a 90 o 110 km/h en una autovía?

Así, en este capítulo, se propone el uso de diferentes técnicas de regresión que permitan generar modelos para estimar esa velocidad a partir, únicamente, de información visual. De forma sencilla, para el entrenamiento de esos modelos vamos a requerir exclusivamente de una serie de imágenes RGB de la vía etiquetadas con la velocidad del vehículo en cada una de ellas. Para ello, es necesario que la velocidad que lleve el vehículo se ajuste a los límites de velocidad de la vía y a la situación del tráfico, pues nuestros modelos aprenden del comportamiento visto en esas imágenes. A partir de ahí, algunos modelos que proponemos trabajan directamente con la imagen, mientras que otros requieren realizar previamente una segmentación semántica de la misma (ver figura 3.2). Intuitivamente, trabajar con la información semántica de las imágenes parece muy útil: si un coche *ve* muchos píxeles de otros vehículos, parece lógico que la velocidad adecuada deberá ser menor que si todo lo que *observa* es carretera. En el test, nuestros modelos estimarán la velocidad adecuada de acuerdo a lo aprendido en el entrenamiento.

Ya adelantamos que, a la vista de los resultados obtenidos, emplear la segmentación

semántica, permite a los modelos trabajar con una información mucho más relevante para la predicción de la velocidad consiguiendo resultados bastante aceptables.

En las siguientes secciones se trata el estado del arte en materia de estimación de la velocidad (Sección 3.2), se describe la base de datos empleada para entrenar y evaluar los diferentes modelos que proponemos (Sección 3.3), se describen brevemente dichos modelos (Sección 3.4) y, por último, se presentan los resultados cuantitativos y cualitativos obtenidos (Sección 3.5).

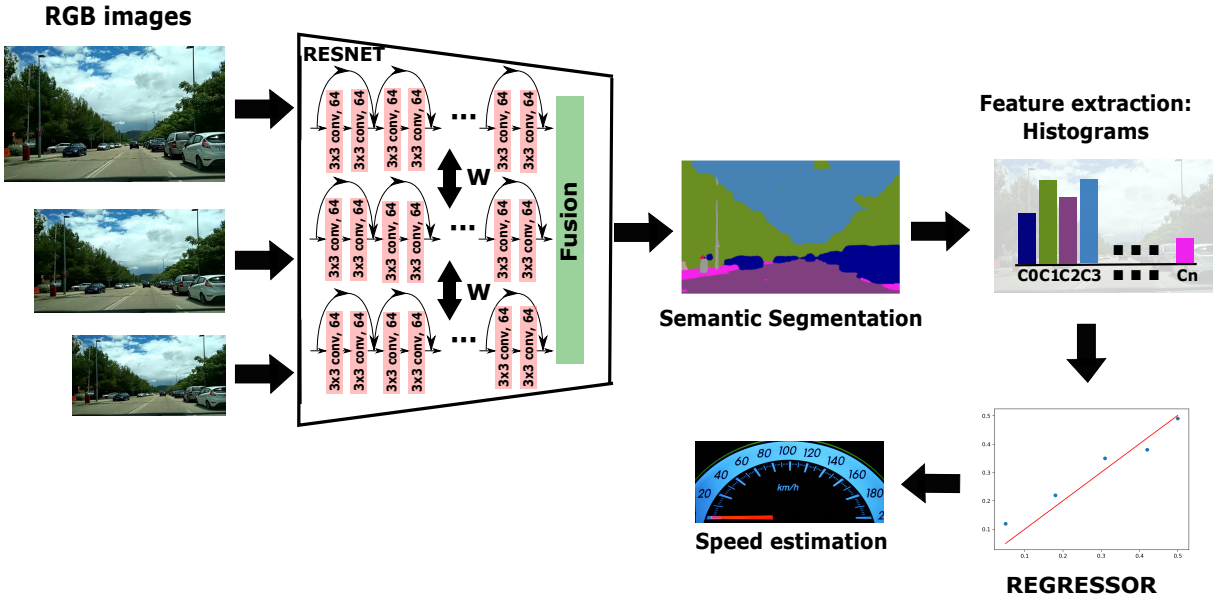


Figura 3.2: Hacemos uso de nuestra arquitectura multiescala basada en ResNet para segmentar semánticamente las imágenes. Con los resultados de la segmentación construimos histogramas con el número de píxeles de cada una de las clases representadas, que usaremos como descriptores para entrenar diferentes regresores con los que estimar la velocidad.

3.2. Estado del Arte

Aunque ser capaz de estimar la velocidad adecuada para un vehículo es una tarea clave para la industria automovilística, que año tras año está incrementando el presupuesto destinado a proyectos de i+D en su lucha por conseguir un vehículo completamente autónomo, no existen trabajos que busquen predecir esta velocidad a partir de las imágenes directamente.

Probablemente, lo más cercano que existe en la literatura al problema que intentamos resolver consiste en estimar la velocidad real que lleva un vehículo. Se han propuesto diferentes técnicas para ello, desde el diseño de métodos de procesamiento de imágenes y algoritmos de reconocimiento de patrones [7] a propuestas de estimación del movimiento basadas en la sustracción del fondo [13].

Sin embargo, lo que probablemente haya tenido un mayor éxito, es el empleo de técnicas de flujo óptico [43, 47]. Estas técnicas calculan la velocidad estudiando el cambio que se produce entre un par de imágenes consecutivas aplicando, con buenos resultados, diferentes métodos que se han ido desarrollando durante las últimas décadas, como el de Lucas-Kanade [32], método disperso que únicamente tiene en cuenta ciertos puntos clave del par de imágenes, o el de Farneback [16], que sigue un procedimiento denso evaluando todos los píxeles de las imágenes.

Nótese la diferencia con nuestra propuesta, mientras que los trabajos mencionados tienen como objetivo estimar la velocidad real a la que circula el vehículo, nuestra propuesta consiste en estimar la velocidad adecuada a la que debería ir, partiendo de un entrenamiento en el que la velocidad real que lleva el conductor es la adecuada. Nuestro objetivo no es conocer cómo de rápido va un coche, sino cómo de rápido debería ir

3.3. Base de Datos

Para confeccionar la base de datos sobre la que realizar experimentos y evaluar nuestros modelos, ha sido utilizada la aplicación para móviles Road Recorder[1]. Esta aplicación permite, de forma simultánea, llevar a cabo la grabación de la vía y almacenar la velocidad del vehículo calculada mediante GPS, permitiendo etiquetar cada imagen de la secuencia con su respectiva velocidad. La figura 3.3 muestra un conjunto de capturas de la interfaz gráfica de esta aplicación.

En concreto, además del vídeo en formato mp4, cada segundo se almacenan los siguientes campos de una estructura, en formato xml:

- date: fecha, hora, minuto y segundo del instante en el que se ha obtenido el registro.
- x_loc: latitud del punto por el que circula el vehículo.
- time: marca de tiempo, en segundos, relativa para cada secuencia. Vale 0 para el primer registro, 1 para el obtenido un segundo después, etc.
- y_loc: longitud del punto por el que circula el vehículo.
- speed: velocidad a la que circula el vehículo.

Se han grabado dos recorridos, uno en entorno urbano y otro en autovía, por carreteras de la Comunidad de Madrid a 30 frames por segundo (fps), para los que la conducción ha procurado ajustarse a lo que consideramos una velocidad adecuada. Las figuras 3.5 y 3.6 muestran algunas imágenes de ambos recorridos.

Dado que a 30 fps apenas hay variación entre una imagen y la siguiente, por conveniencia, se ha decidido reducir el número de imágenes por un factor de 15, esto es, trabajaremos a 2 fps. Esto permite un mejor manejo del volumen de datos a procesar sin

[1] http://www.roadrecorder.eu/

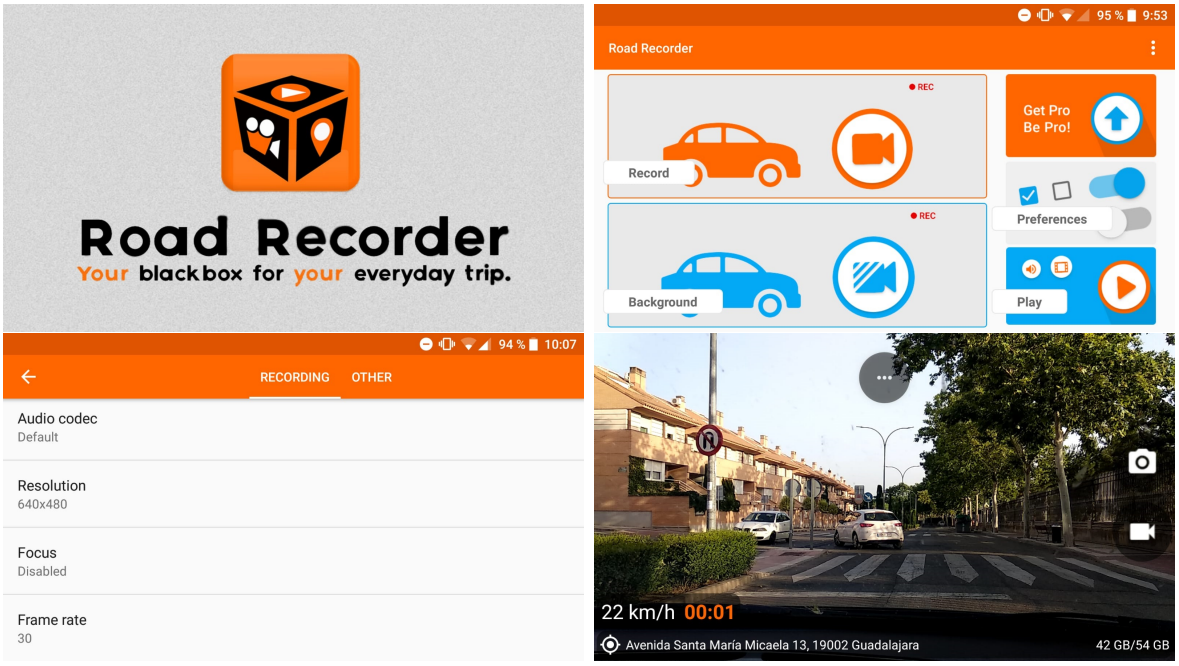


Figura 3.3: Aplicación Road Recorder: interfaz gráfica.

perjuicio de perder información relevante, puesto que, además, como se ha comentado, aunque la grabación se realice a 30 fps únicamente vamos a obtener un registro con la velocidad que lleva el vehículo cada segundo. Dado que, aún así, tenemos más imágenes que registros xml, se ha optado por asignar la misma velocidad a todos los *frames* extraídos en el mismo segundo.

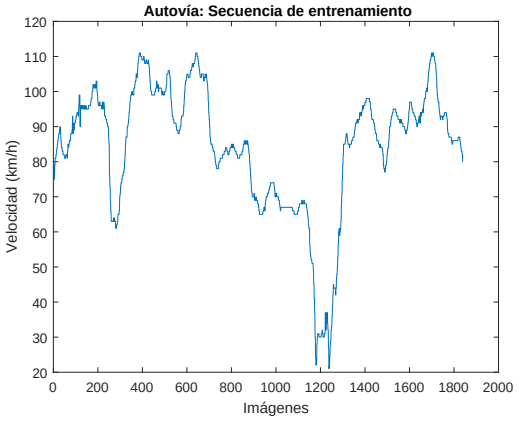
Para estructurar la base de datos, ambos recorridos han sido divididos en secuencias de entrenamiento y test. El recorrido urbano está dividido en tres secuencias, de las cuales dos se van a emplear para el entrenamiento, lo que supone un total de 3775 imágenes, y la restante, de 1200 imágenes, para el test. El recorrido grabado en autovía está formado por dos secuencias de 1840 y 3122 imágenes para entrenamiento y test respectivamente. Todas las imágenes tienen un tamaño de 640x384 píxeles.

Las gráficas de la figura 3.4 muestran la velocidad que mantiene el vehículo a lo largo de las secuencias. Finalmente, la tabla 3.1 da cuenta de la media y desviación típica de la velocidad que lleva el vehículo en cada conjunto de imágenes. En esa tabla se aprecia una dificultad añadida a la que tendrán que hacer frente nuestros modelos: la velocidad media de las secuencias de entrenamiento y test en autovía es diferente.

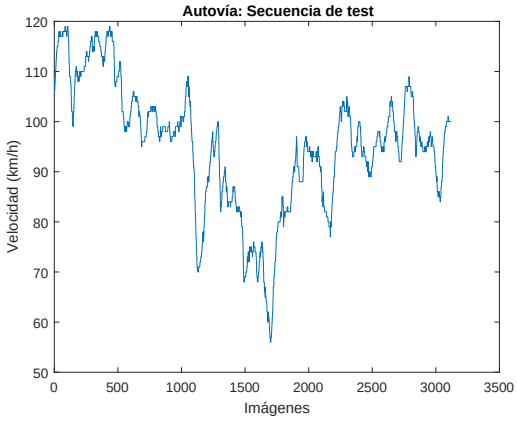
3.4. Modelos de regresión

En esta sección se van a describir brevemente los modelos de regresión empleados para estimar la velocidad adecuada del vehículo. Han sido divididos en dos grupos:

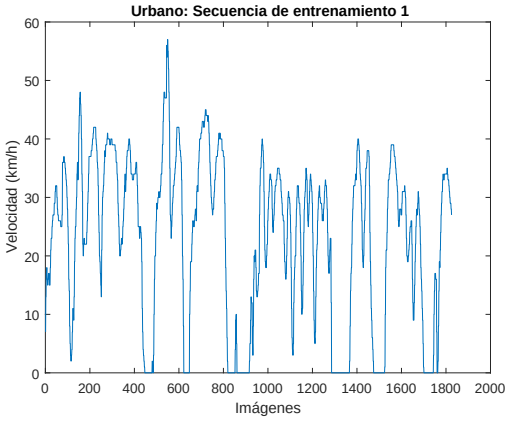
- Regresores tradicionales que emplearán como descriptores de cada imagen los his-



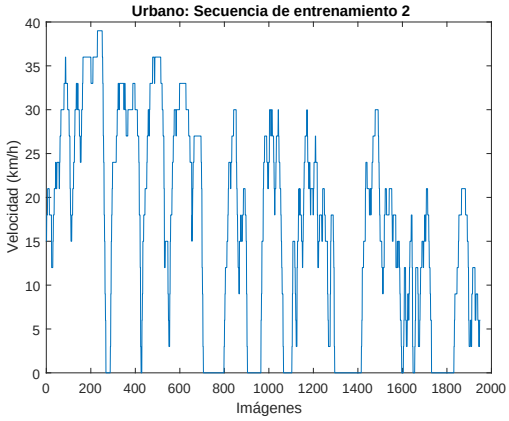
(a)



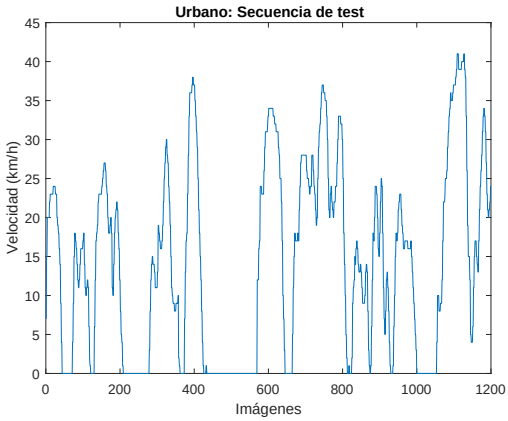
(b)



(c)



(d)



(e)

Figura 3.4: Velocidad del vehículo en las distintas secuencias de la base de datos: (a) secuencia de entrenamiento en autovía, (b) secuencia de test en autovía, (c) y (d) secuencias de entrenamiento en entorno urbano y (e) secuencia de test en entorno urbano.

Figura 3.5: Algunas imágenes de la base de datos del recorrido grabado en autovía.

togramas de la segmentación semántica

- Redes neuronales convolucionales profundas para los que la entrada será la imagen RGB.

3.4.1. Regresores tradicionales basados en los histogramas de la segmentación semántica

En esta primera parte, se va a hacer uso de los histogramas de la segmentación semántica para estimar la velocidad del vehículo. Para ello, en primer lugar, pasamos las

Tabla 3.1: Velocidad media y desviación típica de las distintas secuencias de la base de datos.

Secuencia	Conjunto	Vel. media (km/h)	Des. típica (km/h)
Autovía	Entrenamiento	84.31	18.15
Autovía	Test	95.08	12.81
Urbana	Entrenamiento	19.55	13.60
Urbana	Test	19.59	14.78

Figura 3.6: Algunas imágenes de la base de datos del recorrido grabado en entorno urbano.

imágenes de las secuencias por nuestro modelo de segmentación entrenado en Cityscapes descrito en la Sección 2.3.1. Con los resultados de la segmentación, generamos histogramas con el número de píxeles pertenecientes a cada clase siguiendo el procedimiento descrito para el problema de clasificación de escenas de la Sección 2.3.2, a los que aplicaremos normalización L2. Dichos histogramas serán los vectores de observación con los que alimentaremos a los diferentes regresores. Los regresores que evaluaremos serán: regresor lineal, regresor de lasso, Máquinas de Vectores Soporte para Regresión (SVRs) y *boosting trees*. Para todos ellos, evaluaremos el impacto que tiene añadir información espacial mediante pirámides de hasta 3 niveles.

3.4.1.1. Regresor lineal

Es el método más sencillo de regresión que busca estimar los coeficientes β_n para los que el error cuadrático medio entre la velocidad real y la estimada, y_i, es mínimo.

$$y_i = \beta_0 + \beta_1 x_{i1} + \beta_2 x_{i2} + ... + \beta_n x_{in}, \tag{3.1}$$

donde $x_i = [x_{i1}, x_{i2}, ..., x_{in}]$ es el histograma de n elementos, extraído de la segmentación semántica.

3.4.1.2. Regresor de Lasso

El regresor de lasso [48] es una variante del regresor lineal que introduce un término de regularización en la función a minimizar, buscando penalizar aquellas soluciones con coeficientes muy elevados. Esto se traduce en una simplificación del modelo pues, en función del peso asignado al término de penalización, lasso pone a cero más o coeficientes. A pesar de esta restricción, que podría parecer contraproducente, se ha comprobado como este estimador es capaz de tener un error cuadrático medio menor que el regresor lineal en la fase de test, por una mejor generalización.

Técnicamente, el regresor de lasso busca encontrar los coeficientes β_n que optimizan la siguiente función:

$$\min_{\beta_0,\beta} \left(\frac{1}{2N} \sum_{i=1}^{N} (y_i - \beta_0 - x_i^T \beta)^2 + \lambda \sum_{j=1}^{p} |\beta_j| \right), \tag{3.2}$$

donde λ es un valor positivo ajustable que indica la mayor o menor importancia que tendrá el término de regularización β_j. A mayor λ, más restrictivo es el modelo en la búsqueda de coeficientes y, por lo tanto, aumenta el número de coeficientes β_n iguales a cero.

3.4.1.3. SVR: Máquinas de Vectores Soporte para Regresión

Las SVMs son un conjunto de algoritmos de aprendizaje supervisado muy conocido desde hace más de dos décadas. Aunque originalmente pensados para clasificación [2], es posible adaptar el problema para trabajar en regresión [12]. Es lo que se conoce como SVRs.

Partiendo de un conjunto de N vectores de entrenamiento, x_i, con su correspondiente repuesta objetivo y_i, para su versión lineal, se busca encontrar una función:

$$f(x) = x^T \beta + b, \tag{3.3}$$

siendo $f(x)$ tan suave como sea posible. Para ello, se busca minimizar la norma de los parámetros β. En forma de ecuación, se optimiza la expresión:

$$\min \left(\frac{1}{2} ||\beta||^2 \right), \tag{3.4}$$

con una condición: que todos los vectores de entrenamiento se encuentren dentro de un umbral ϵ :

$$|y_i - (x_i' \beta + b)| \leq \epsilon. \tag{3.5}$$

Es decir, nuestro objetivo es encontrar una función $f(x)$ de tal forma que exista, como mucho, una desviación de ϵ con respecto a los valores y_i reales para todos los vectores de entrenamiento y que, al mismo tiempo, sea tan suave como sea posible con el objetivo de conseguir una mejor generalización del modelo.

Nótese que a diferencia de su uso para clasificación, donde cada vector de observación puede pertenecer a una clase u otra, aquí es necesario tolerar cierto error ϵ que permita encontrar el hiperplano que maximiza el margen. Como aún así dicha condición no tiene por qué existir para todos los puntos, pues estamos asumiendo que existe una función $f(x)$ que aproxima a todos con precisión ϵ, se crea el concepto de margen blando (*soft margin*), permitiendo cierto error, ξ y ξ^*, en la regresión, quedando la optimización formulada como:

$$\text{mín}\left(\frac{1}{2}||\beta||^2 + C\sum_{l=1}^{N}(\xi + \xi^*)\right), \tag{3.6}$$

sujeto a:

$$\begin{cases} y_i - (x_i'\beta + b) \leq \epsilon + \xi \\ (x_i'\beta + b) - y_i \leq \epsilon + \xi^* \\ \xi, \xi^* \geq 0 \end{cases}, \tag{3.7}$$

donde la constante $C > 0$ determina el compromiso entre la suavidad de la función y la cantidad de errores superiores a ϵ permitidos.

3.4.1.4. Boosting Trees

Los conjuntos de árboles son una de las técnicas más robustas para la predicción de variables [3]. Con respecto a otros modelos de regresión, reducen el ruido o varianza que lleva a que exista diferencia entre el valor real y el estimado. Intuitivamente, múltiples árboles trabajando para predecir una variable ofrecerán un rendimiento mucho mejor que un único estimador. Estos algoritmos se clasifican generalmente en dos grupos:

1. Boosting: los árboles trabajan de forma secuencial, la salida de uno es tomada como entrada del siguiente para mejorar la predicción.

2. Bagging: varios árboles trabajan cada uno de forma independiente y, al final, se hace uso de técnicas de promediado para dar un valor conjunto.

En este trabajo se ha empleado un regresor basado en *boosting trees*, en el que cada árbol va a aprender de los errores (residuos) cometidos por el árbol precedente. En cada iteración se entrena un nuevo árbol con la diferencia entre la respuesta real observada y la predicción realizada por los árboles previos. En concreto, el objetivo que se persigue es minimzar el error cuadrático medio según el procedimiento descrito en [24].

3.4.2. Redes Neuronales Convolucionales: VGG y ResNet

Aunque mayoritariamente empleadas para clasificación, se propone adaptar dos CNNs profundas, VGG [45] y ResNet [26], adecuándolas a la tarea de regresión. Para ello, con

respecto a las variantes de clasificación empleadas en reconocimiento de escenas, se realizarán dos variaciones:

1. **Un única salida.** Modificamos la salida de la última capa *fully-connected*, que en clasificación tiene tantas salidas como clases posibles, a una sola, destinada a realizar la estimación de la velocidad.

2. **Función de pérdidas: distancia euclídea.** Sustituimos la función de pérdidas que deja de ser un softmax, para ser una pérdida basada en la distancia euclídea. De este modo, en cada iteración la salida o predicción se compara con la velocidad etiquetada a cada imagen.

Dos cambios muy simples, pero suficientes para permitir a estas redes llevar a cabo regresión. Además, y a diferencia del resto de modelos que descritos, trabajarán con información de apariencia, es decir, serán entrenadas directamente con las imágenes RGB. La estructura nuestras redes queda reflejada en la figura 3.7.

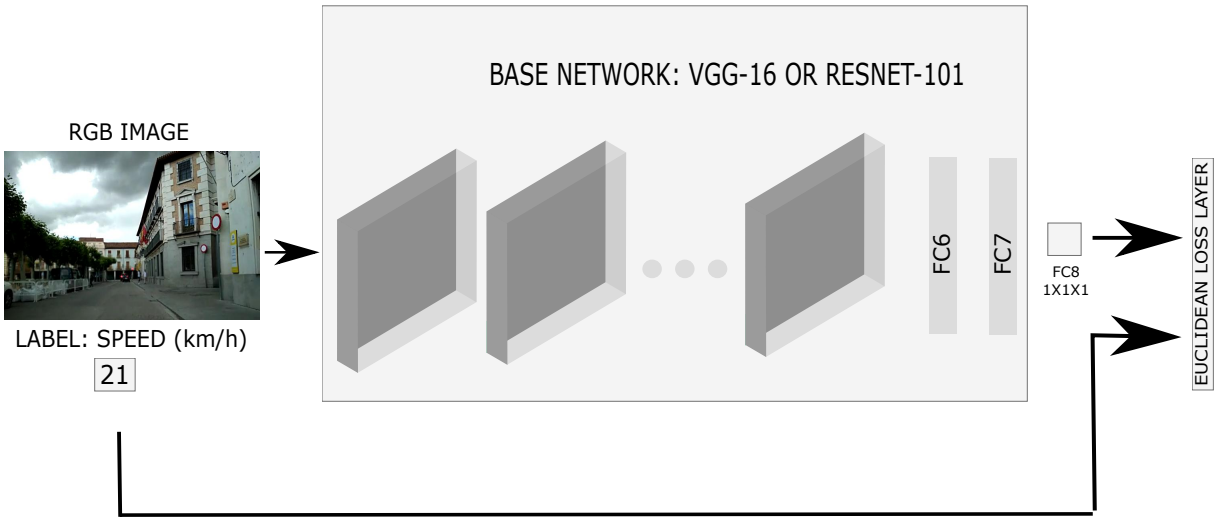


Figura 3.7: Para adaptar las CNNs, inicialmente concebidas para clasificación, a la tarea regresión, hacemos que las redes tengan una única salida modificando la última *fully-connected* (fc8), y entrenamos con una pérdida basada en la distancia euclídea.

3.5. Resultados

En esta sección se presentan los resultados obtenidos al emplear los distintos algoritmos propuestos para estimar la velocidad adecuada del vehículo.

3.5.1. Optimimzación de parámetros

La mayoría de los modelos de regresión poseen algún parámetro que es necesario optimizar. Así, el parámetro λ en el regresor de lasso indica la mayor o menor importancia que otorgamos al término que penaliza la magnitud de los coeficientes β. En los *boosting trees*, la profundidad de los árboles define el máximo número de nodos de decisión que pueden ser atravesados desde la raíz a la hoja más lejana. Todos tienen una gran influencia en el resultado final.

Para seleccionar los valores óptimos de los modelos basados en la segmentación semántica, se ha empleado la técnica de validación cruzada de tipo K-Fold (Figura 3.8). Este mecanismo consiste en dividir el conjunto de imágenes de entrenamiento en K grupos, entrenando con K-1 de ellos y evaluando en el restante. Este procedimiento se repite K veces dejando cada vez un grupo diferente para evaluar. Finalmente, se promedian los valores óptimos obtenidos de cada uno de los parámetros evaluados. En todos los experimentos se ha empleado un valor de K igual a 5.

Comenzando con el **regresor de lasso**, aunque podría parecer que lo ideal es tomar siempre el valor de λ con el que el error cuadrático medio en entrenamiento es menor, se ha demostrado que esto no siempre es así. Con el fin de simplificar el modelo (más coeficientes a cero) y permitir una mejor generalización del mismo, es común seleccionar un valor de λ lo más grande posible, siempre que el error cuadrático medio no exceda cierto límite. Uno de los criterios más habituales para establecer este límite, y que ha sido empleado en este trabajo, es tomar el mayor valor de λ posible para el que el error cuadrático medio asociado esté dentro de un margen de una unidad respecto al mínimo.

Para las **SVRs**, se han optimizado mediante validación cruzada los parámetros C y ϵ. El parámetro C es un valor positivo que reduce el sobreentrenamiento controlando la penalización que se da a aquellos vectores de observación que se quedan fuera del límite ϵ, esto es, es una medida de cuántos vectores de entrenamiento pueden tener una desviación superior a ϵ. Por su parte, ϵ es el umbral tolerado en el entrenamiento dentro del cual se considera que un vector de observación da la salida correcta.

Los ***boosting trees*** poseen, fundamentalmente, tres parámetros a optimizar: el número de árboles, la profundidad de los mismos, que controla el máximo número de nodos que puede haber entre la raíz y la hoja más lejana, y el *learning rate* o tasa de aprendizaje, parámetro que determina el impacto de cada árbol que se añade al modelo. Los *boosting trees* empiezan realizando una determinada estimación que se va refinando tras cada árbol. El *learning rate* regula la magnitud del cambio en las estimaciones. Igualmente, se ha empleado validación cruzada para la obtención de los valores óptimos en los diferentes experimentos realizados.

Por último, en lo que se refiere a las **CNNs**, se han entrenado durante 4000 iteraciones con un *learning rate* inicial de 10^{-4}, reduciéndolo a 10^{-5} al cabo de 2000 iteraciones, tanto para VGG como para ResNet. El resto de parámetros han sido fijados como sigue: $momentum = 0.9$, $weigth\ decay = 10^{-4}$ y tamaño del *batch* de 20 para ambas arquitecturas.

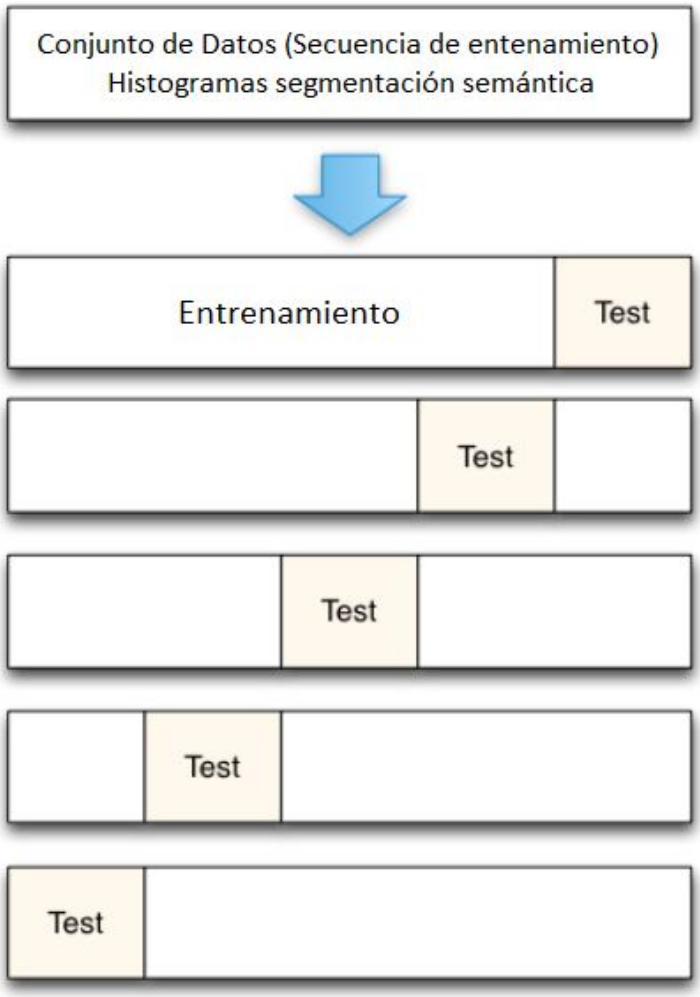


Figura 3.8: Validación cruzada de tipo K-Fold. El conjunto de entrenamiento de divide en K conjuntos dejando, en cada iteración, uno de ellos para evaluar los parámetros a optimizar.

3.5.2. Resultados cuantitativos

Como métrica de evaluación cuantitativa de los distintos modelos, se va a medir el error de predicción, que definimos como la diferencia en valor absoluto entre la velocidad real, V_r, y la estimada por nuestro modelo, V_p, promediado para las N imágenes de la secuencia de test:

$$\frac{1}{N}\sum_{i=1}^{N}|V_{ri} - V_{pi}|. \tag{3.8}$$

Es importante hacer notar que con esta medida del error, estamos asumiendo que durante todo el recorrido de entrenamiento y test, el vehículo ha llevado la misma velocidad ante las mismas circunstancias de la vía. Aunque esto se ha procurado conseguir evitando

grandes diferencias en la conducción durante la grabación de las secuencias, entendemos que va a existir siempre un error inherente a la obtención de la base de datos.

En cuanto a los experimentos, en primer lugar, entrenamos nuestros modelos con todas las imágenes de entrenamiento de la base de datos, independientemente del tipo de vía, esto es, con las 3775 urbanas y las 1840 de autovía. La tabla 3.2 recoge el error cometido por los distintos modelos empleados en las secuencias de test. En general, se observa cómo los modelos de regresión tradiciones, trabajando con la segmentación semántica de las imágenes, obtienen mejores resultados que las redes neuronales profundas para el recorrido urbano. Sin embargo, son las redes neuronales las que mejor se comportan en la secuencia de autovía. Probablemente, el hecho de tener más parámetros, permite a las redes neuronales ajustarse más a ambos tipos de recorrido, sin que quede un tipo de secuencia muy penalizada respecto a la otra.

Tabla 3.2: Comparativa entre el error cometido sobre las secuencia de test de autovía y entorno urbano al entrenar un regresor conjunto para los dos tipos de vía.

Método	**Divisiones**	**Er. urbano** (Km/h)	**Er. autovía** (Km/h)
VGG-16	-	12.58	**11.57**
ResNet-101	-	11.49	11.87
Regresión lineal	1	9.63	21.82
	2	9.15	18.35
	3	11.97	15.78
SVR	1	11.01	18.97
	2	10.69	16.76
	3	11.96	16.97
Regresión de Lasso	1	9.66	24.60
	2	**8.74**	19.80
	3	10.66	18.13
Boosting Trees	1	13.41	15.28
	2	9.78	15.75
	3	11.69	13.86

Además, para esta tarea no se aprecia una mejora sustancial por incluir resolución espacial a múltiples escalas y, mientras que en la secuencia urbana los regresores lineal y de lasso son los que menos error cometen, en la grabada en autovía, el error medio es

menor cuando empleamos SVRs y los *boosting trees*. En cualquier caso, el error cometido por todos los modelos es elevado; la precisión no es suficientemente buena.

¿Podemos mejorar la estimación de algún modo? Hasta ahora, hemos entrenado un mismo regresor para todas las secuencias de la base de datos, pero, ¿qué ocurre si entrenamos un modelo independiente para cada tipo de vía? Intuitivamente, dado que los datos serán más homogéneos, los modelos resultantes podrán ajustar sus parámetros mucho mejor al tipo de vía en cuestión.

La tabla 3.3 recoge el error obtenido al entrenar un modelo de regresión por cada tipo de vía. A la vista de los resultados, observamos como, nuevamente, los modelos basados en la segmentación continúan dando mejores resultados que las redes neuronales en la secuencia urbana. Además, con esta configuración, los métodos de regresión tradicionales también consiguen mejorar a las redes neuronales en autovía. Esto corrobora un hecho que venimos apreciando a lo largo de este trabajo: el gran valor de la información que ofrece la segmentación semántica y el conocer el tipo de vía en el que nos encontramos.

Tabla 3.3: Comparativa entre el error cometido sobre las secuencias de test de autovía y entorno urbano al entrenar un regresor independiente para cada tipo de vía.

Método	**Divisiones**	**Er. urbano** (Km/h)	**Er. autovía** (Km/h)
VGG-16	-	11.86	12.48
ResNet-101	-	9.59	12.79
Regresión lineal	1	**6.02**	9.54
	2	7.49	10.62
	3	12.67	18.94
SVR	1	8.25	10.21
	2	8.14	9.50
	3	9.26	9.23
Regresión de Lasso	1	6.67	8.72
	2	7.12	8.54
	3	8.41	8.41
Boosting Trees	1	8.81	7.80
	2	10.25	8.08
	3	10.28	**7.76**

Por otra parte, de nuevo, existen diferencias entre los mejores modelos para cada tipo

de vía. Para la secuencia en autovía, de entre los algoritmos basados en la segmentación semántica, los *boosting trees* son los que mejor resultado ofrecen, seguidos del regresor de lasso y las SVRs. Además, el regresor lineal no es capaz de mejorar los resultados al aumentar los niveles de división. Por contra, en la secuencia urbana, el regresor lineal es el que mejor se comporta, seguido del regresor de Lasso y las SVRs. Los *boosting trees* pasan de ser el mejor modelo en la secuencia de autovía al peor. Ninguno de los regresores mejora al aumentar el número de divisiones en esta secuencia.

En cualquier caso, y con independencia de qué modelo es el mejor en cada escenario, queda claro que, para los modelos basados en la segmentación, resulta beneficioso entrenar un regresor para cada tipo de vía por separado. Sin embargo, en una teórica implementación de este tipo de aplicación, habría una evidente dificultad: sería necesario conocer en qué tipo de vía nos encontramos para emplear un modelo de regresión u otro. Para ello, podríamos hacer uso de nuestro clasificador de escenas descrito en el capítulo 2. Las redes neuronales son mucho más consistentes en este sentido; con ellas conseguimos obtener resultados muy similares tanto si entrenamos un modelo independiente para cada tipo de vía como si el entrenamiento se realiza en conjunto. Las gráficas de la figura 3.9 ofrecen una comparativa visual del error cometido por los diferentes métodos para las distintas secuencias y métodos de entrenamiento.

Finalmente, las imágenes de las figuras 3.10 y 3.11 recogen de forma gráfica una comparativa entre la velocidad real del vehículo y la estimada por diferentes modelos de regresión en los recorridos de test de autovía y entorno urbano, respectivamente. Para cada tipo de vía, se muestran las dos CNNs empleadas y los dos mejores modelos basados en la segmentación semántica.

En la secuencia de test de autovía (3.10), todos los modelos detectan que es necesario reducir la velocidad a mitad del recorrido, en un momento en el que el conductor sale de la autovía por una vía de servicio para, posteriormente, incorporarse a otra autovía. En general, se observa que a las redes neuronales les cuesta más ajustarse a la velocidad real del vehículo que a los modelos basados en la segmentación. Esto es especialmente evidente en la parte inicial del recorrido, donde el error cometidos por las CNNs supera los 30 km/h.

En el caso urbano (3.11), llama la atención que las redes neuronales, que reciben información de apariencia, no son capaces de reducir y aproximar a cero la velocidad que estiman cuando el vehículo está parado, principalmente ante semáforos en rojo. Por contra, los regresores basados en la segmentación, sí ajustan mucho mejor esos momentos aún cuando no tienen información de apariencia. Cabe pensar que estos regresores reconocen dichas situaciones ante una elevada presencia de píxeles de la clase semáforo.

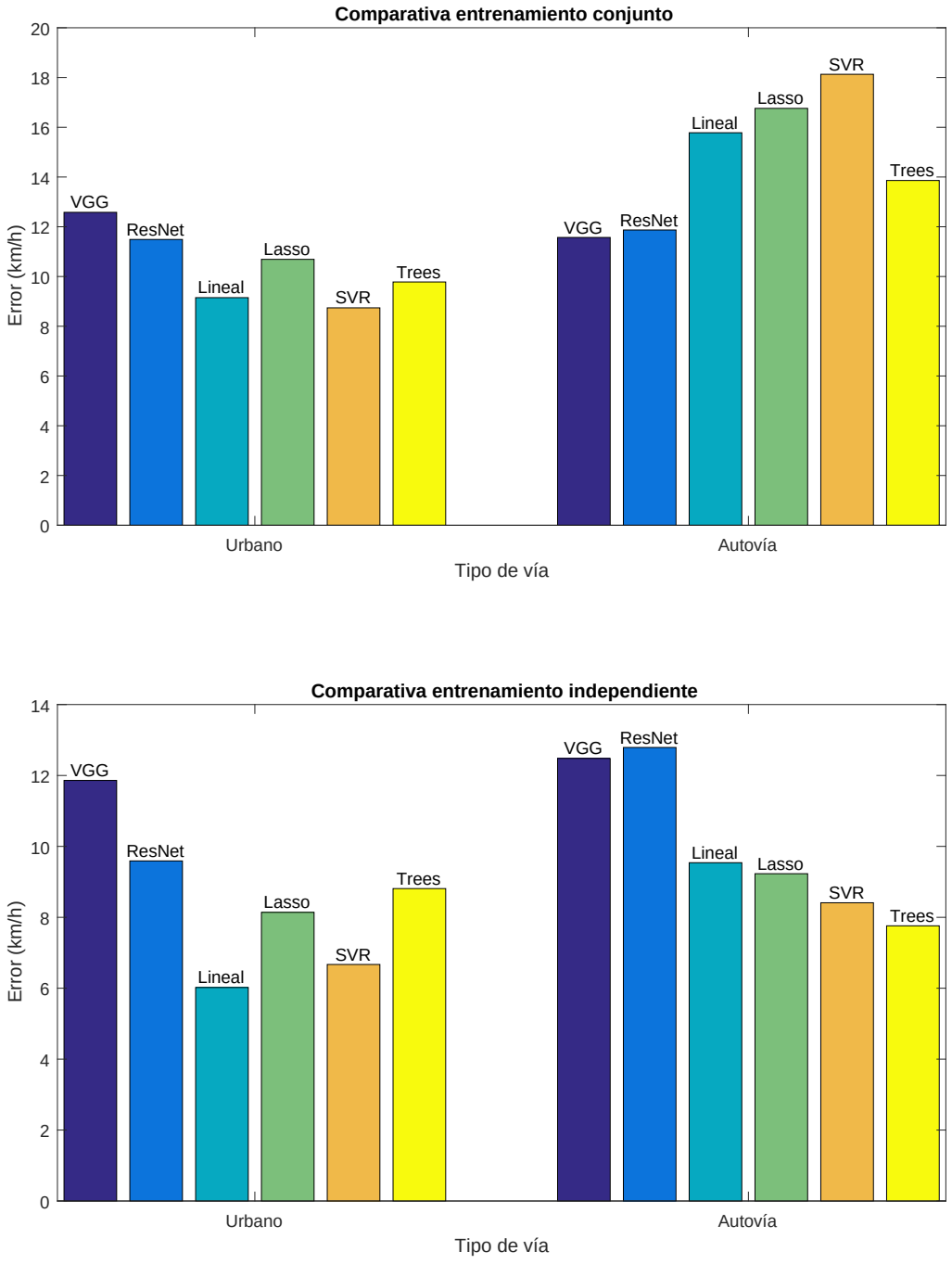


Figura 3.9: Comparación gráfica del error cometido en ambas secuencias de test para los dos tipos de entrenamiento planteados. Para los métodos basados en la segmentación semántica se representa la configuración con la que cada modelo obtiene un menor error.

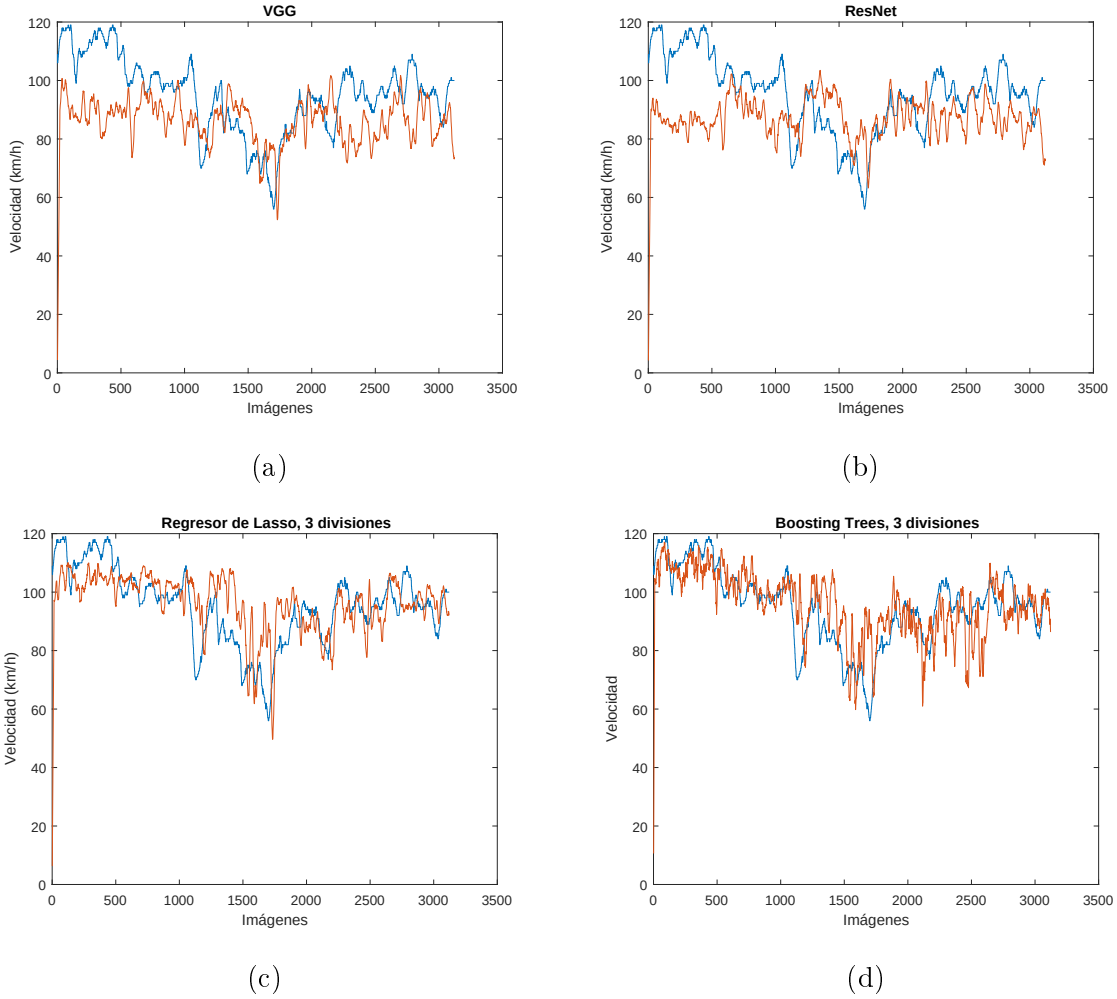


Figura 3.10: Velocidad real (azul) vs. velocidad estimada (rojo) por distintos modelos de regresión en autovía.

3.5.3. Resultados cualitativos

En esta sección se pretende realizar un estudio más detallado de qué es lo que están aprendiendo estos modelos, evaluando la validez de los mismos. Mostraremos algunos de los momentos más relevantes de ambas secuencias de test, justificando, en lo posible, los resultados que nos ofrecen los modelos.

3.5.3.1. Autovía

Como se ha visto en la figura 3.10, a grandes rasgos, el comportamiento de los distintos algoritmos es muy similar en cuanto a la velocidad adecuada estimada en cada momento. Así, se va a utilizar el mejor de nuestros modelos como referencia para el análisis cualitativo, es decir, para la secuencia grabada en autovía emplearemos los resultados obtenidos

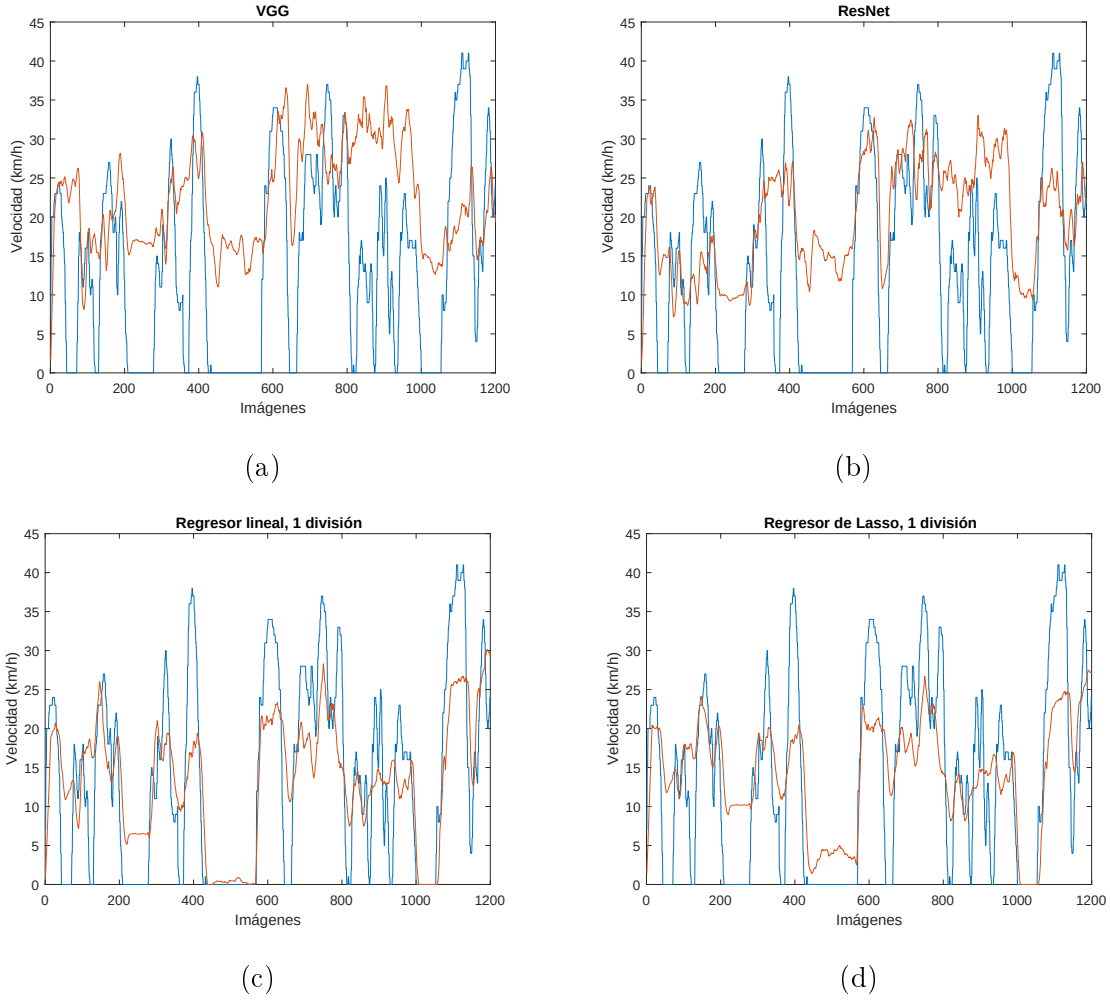


Figura 3.11: Velocidad real (azul) vs. velocidad estimada (rojo) por distintos modelos de regresión en entorno urbano.

por los *boosting trees* en el que los histogramas han sido generados con tres niveles de división espacial. El comportamiento de este modelo está reflejado en la figura 3.10d.

Probablemente, el tramo más crítico y representativo de esta secuencia es la disminución de velocidad que se produce a mitad de recorrido, entorno a la imagen 1450. Es por ello, que vamos a centrar el análisis cualitativo en esa parte de la secuencia. En él, nuestro modelo estima subidas y bajadas en la velocidad adecuada mucho más drásticas que la velocidad real a la que circula el vehículo.

Para analizar qué está ocurriendo, vamos a observar la figura 3.12. Se trata de 24 imágenes correspondientes al segmento bajo estudio. En la parte superior de cada imagen puede leerse tanto la velocidad real como la recomendada por el modelo.

El vehículo va circulando por una autovía (3.12a, 3.12b), en la que la velocidad real y recomendada son similares, cuando decide tomar una salida (3.12c) disminuyendo, en consecuencia, la velocidad. Evidentemente, nuestro modelo no es capaz de ajustar su

recomendación cuando todavía está en la autovía pues desconoce las intenciones del conductor, por lo que se produce una discrepancia entre la velocidad recomendada, que es de 94 km/h, y la real del vehículo, que se ha reducido a 69 km/h.

Sin embargo, en cuanto toma la salida y comienza a circular por la vía de servicio (3.12d, 3.12e, 3.12f), en la que carretera es más estrecha, enseguida el modelo reduce la velocidad recomendada hasta llegar a los 66 km/h, muy parecida a la que lleva el vehículo.

En las imágenes siguientes, el vehículo sigue por una vía de incorporación a otra autovía y las velocidades real y estimada son muy similares (3.12g, 3.12h, 3.12i, 3.12j, 3.12k).

Al entrar completamente en la nueva autovía (3.12l), nuestro modelo ya comienza a aumentar la velocidad recomendada (3.12m, 3.12n) hasta los 87 km/h. A pesar de ello, la velocidad real del vehículo se mantiene por debajo de los 70 km/h. Esto se debe a que, de nuevo, el coche va a tomar una salida en unos pocos metros, algo que nuestro modelo, no es capaz de percibir. En cuanto el vehículo toma la salida (3.12ñ), la velocidad recomendada empieza nuevamente a descender en ese tramo de calzada estrecha (3.12o, 3.12p, 3.12q, 3.12r), ajustándose a la velocidad real que lleva el vehículo. Una vez incorporados a la nueva vía, tanto la velocidad real como la estimada vuelven a aumentar (3.12s, 3.12t, 3.12u, 3.12v, 3.12w).

Comprobamos, de este modo, que nuestro modelo sí está funcionando adecuadamente en esos tramos. Las diferencias que existen entre la velocidad real y la estimada son debido a que nuestro regresor no puede anticipar la intención del conductor. En el resto de la secuencia, antes y después del tramo analizado, las velocidades real y estimada por el modelo son muy similares. En ellas, el vehículo circula por una autovía sin grandes complicaciones de tráfico, cambios de carretera u otro tipo de circunstancias reseñables.

3.5.3.2. Urbano

Para el recorrido urbano, se muestran resultados cualitativos de diferentes puntos del recorrido. De nuevo, los valores de velocidad recomendada que aparecen en las imágenes se corresponden con la estimación realizada por el modelo que menor error obtiene para esta secuencia: un regresor lineal sin divisiones espaciales. En la figura 3.11c veíamos el resultado de dicho modelo a lo largo de toda la secuencia.

El vehículo comienza su recorrido por una vía en la que la velocidad estimada y la real coinciden (3.13a). Al final de esa calle, alcanzamos a un vehículo que está detenido y debemos frenar por completo. En ese momento, nuestro modelo, aunque reduce la velocidad, no consigue predecir que, ante esa situación, la velocidad adecuada es cero, sino que se queda en los 12 km/h (3.13b). En las siguientes imágenes (3.13c, 3.13d, 3.13e) se repite este comportamiento: al llegar a una fila de coches que esperan a que se abra el semáforo, nuestro modelo sí reduce la velocidad estimada, pero no llega a aconsejarnos frenar por completo. Seguidamente, se encara una recta 3.13f con buena estimación de la velocidad.

Frame #1451: velocidad recom. 86 km/h; velocidad real 83 km/h

(a)

Frame #1481: velocidad recom. 88 km/h; velocidad real 77 km/h

(b)

Frame #1496: velocidad recom. 94 km/h; velocidad real 69 km/h

(c)

Frame #1513: velocidad recom. 90 km/h; velocidad real 70 km/h

(d)

Frame #1525: velocidad recom. 78 km/h; velocidad real 74 km/h

(e)

Frame #1539: velocidad recom. 66 km/h; velocidad real 75 km/h

(f)

Frame #1561: velocidad recom. 80 km/h; velocidad real 74 km/h

(g)

Frame #1577: velocidad recom. 80 km/h; velocidad real 75 km/h

(h)


Frame #1597: velocidad recom. 66 km/h; velocidad real 68 km/h

(i)


Frame #1620: velocidad recom. 80 km/h; velocidad real 73 km/h

(j)


Frame #1628: velocidad recom. 79 km/h; velocidad real 75 km/h

(k)


Frame #1643: velocidad recom. 85 km/h; velocidad real 72 km/h

(l)


Frame #1663: velocidad recom. 83 km/h; velocidad real 65 km/h

(m)


Frame #1679: velocidad recom. 87 km/h; velocidad real 60 km/h
THC
COPERO

(n)


Frame #1701: velocidad recom. 86 km/h; velocidad real 56 km/h

(ñ)


Frame #1712: velocidad recom. 86 km/h; velocidad real 60 km/h

(o)

(p)

(q)

(r)

(s)

(t)

(u)

(v)

(w)

Figura 3.12: Resultados cualitativos de la velocidad estimada por nuestro mejor modelo en diferentes imágenes de la secuencia de test en autovía.

A continuación, ante la presencia de un semáforo en primer plano (3.13g), comprobamos como el modelo sí es capaz de predecir que en ese punto el coche debe detenerse. El vehículo continúa su recorrido (3.13h, 3.13i) por distintas calles, hasta que, de nuevo, debemos detenernos ante la presencia de un coche parado y, aunque el modelo reduce la velocidad aconsejada hasta los 11km/h, vuelve a no llegar a reconocer que hemos de pararnos por completo 3.13j. En las siguientes imágenes, comienza a llover (3.13k, 3.13l, 3.13m, 3.13n) y, en todas ellas, vemos como, aunque el cristal esté mojado y haya que hacer uso del limpiaparabrisas, el modelo funciona bastante bien ante distintas velocidades. Nuevamente, ante un semáforo en primer plano (3.13ñ), el modelo es capaz de reconocer la situación y predecir que es necesario que nos detengamos. Finalmente, se observa el paso por una rotonda (3.13o) en la que el modelo ajusta correctamente la velocidad.

Por lo tanto, en secuencias urbanas podemos reconocer varias tendencias de nuestro modelo. En primer lugar, la velocidad estimada en diferentes puntos de la secuencia se ajusta relativamente bien a lo que sería una conducción adecuada. Además, el regresor ha aprendido que, ante la presencia de un semáforo, es necesario detenerse. El principal problema observado radica en que no somos capaces de predecir correctamente situaciones en las que es necesario detenerse cuando nos aproximamos a otro vehículo que se encuentra parado en medio de la carretera. En esas situaciones el modelo sí es capaz de reconocer que es necesario aminorar la velocidad, pero no hasta el punto de tener que parar por completo. Finalmente, a tenor de los resultados, no parece verse afectado ante condiciones climatológicas desfavorables.


Frame #30: velocidad recom. 20 km/h; velocidad real 20 km/h

(a)


Frame #51: velocidad recom. 12 km/h; velocidad real 0 km/h

(b)


Frame #153: velocidad recom. 24 km/h; velocidad real 25 km/h

(c)


Frame #190: velocidad recom. 17 km/h; velocidad real 21 km/h

(d)


Frame #214: velocidad recom. 6 km/h; velocidad real 0 km/h

(e)


Frame #335: velocidad recom. 19 km/h; velocidad real 19 km/h

(f)


Frame #437: velocidad recom. 0 km/h; velocidad real 0 km/h

(g)


Frame #579: velocidad recom. 17 km/h; velocidad real 24 km/h

(h)

(i)

(j)

(k)

(l)

(m)

(n)

(ñ)

(o)

Figura 3.13: Resultados cualitativos de la velocidad estimada por nuestro mejor modelo en diferentes imágenes de la secuencia de test en entorno urbano.

Capítulo 4

Conclusiones

A lo largo de este Trabajo Fin de Máster, se han propuesto soluciones en materia de visión artificial para llevar a cabo tanto la comprensión de una escena como la novedosa tarea de estimar la velocidad adecuada de los vehículos. Dichos métodos abarcan una gran cantidad de variantes que van desde la segmentación semántica, a la clasificación o la regresión. Se han evaluado todas las propuestas en diferentes bases de datos, comparándonos con el estado del arte allí donde ha sido posible.

Hemos podido corroborar que **la segmentación semántica proporciona una información especialmente útil en el contexto de la visión artificial y la interpretación de escenas**. En este trabajo, hemos demostrado que, frente a utilizar una red específica para cada tarea como viene siendo habitual en la literatura, **es posible realizar segmentación semántica, clasificación, y detección de objetos, basándonos en una única CNN de segmentación.** Los resultados de la misma pueden ser utilizados para clasificar, con estrategias tan sencillas como construir histogramas con el número de píxeles por clase, o detectar objetos, trazando directamente *bounding boxes* sobre las componentes conectadas de la segmentación. Este método consigue buenos resultados en clasificación de escenas de tráfico, y supera al resto de métodos del estado del arte en la base de datos NYUD2.

Con respecto a la estimación de la velocidad adecuada, se trata, sin duda, de una tarea extremadamente compleja por diferentes motivos. Por un parte, por la dificultad de generar una base de datos en la que el vehículo, ante situaciones de tráfico idénticas, se comporte siempre de la misma forma. Es decir, si volviésemos a realizar los recorridos grabados, con exactamente las mismas condiciones de tráfico, ¿iríamos a la misma velocidad? Somos nosotros quienes, subjetivamente, estamos estableciendo la velocidad óptima, entrenando una serie de modelos asumiendo que la velocidad que lleva el vehículo en la secuencia de entrenamiento es la ideal. Y estamos evaluando el comportamiento del modelo entrenado bajo esa misma premisa, que la velocidad que lleva el vehículo en la secuencia de test es la idónea para las circunstancias de la vía. Por otra, ¿aporta una imagen la suficiente información para conocer la velocidad adecuada? ¿Es una imagen tomada en una autovía en la que la velocidad adecuada sean 100 km/h suficientemente

distinta a una imagen a 120 km/h? Para un ser humano no es sencillo. Para verificarlo, se han evaluado diferentes métodos de regresión, desde redes neuronales con las imágenes RGB como entrada, a técnicas tradiciones para las que hemos empleado la segmentación semántica de las imágenes como descriptores.

A pesar de las dificultades que presupone esta tarea, **la propuesta de estimación de velocidad que hacemos es capaz de seguir los cambios de velocidad del vehículo con bastante precisión** en distintos tipos de vía (autovía y urbano), y somos capaces de justificar aquellas situaciones en las que se produce una mayor discrepancia entre la velocidad que nuestro sistema predice y la velocidad real que lleva el vehículo, por las limitaciones que tiene la red en tanto que no conoce las intenciones del conductor. Es especialmente relevante que aquellos modelos que emplean información semántica superan en precisión a los que toman directamente la imagen en color.

Con todo, se ha diseñado, desarrollado y evaluado un completo sistema de visión e inteligencia artificial, basado en *deep learning*, para el que el trabajo realizado ha ido desde generar bases de datos, a estudiar, seleccionar, desarrollar y optimizar diferentes modelos de clasificación, regresión y segmentación semántica, para adecuarlos a las diferentes tareas. A partir de este punto, podría resultar interesante el empleo de redes de predicción futuras sobre segmentación semántica de imágenes [36], que permitan adecuar la velocidad del vehículo con antelación. Por el momento, dichas propuestas de red únicamente predicen de forma aceptable hasta un máximo de 0.5 segundos en el futuro y en escenas urbanas, es decir, cuando la variación de la imagen es muy pequeña. Finalmente, incorporar información RGB a la obtenida de la segmentación podría ser otro paso a tener en cuenta a la hora de continuar la investigación en esta línea, puesto que, especialmente en contexto urbano, conocer el color de los semáforos o de las señales de tráfico, debería ayudar a mejorar la estimación de la velocidad.

Bibliografía

[1] Anurag Arnab, Sadeep Jayasumana, Shuai Zheng, and Philip H. S. Torr. Higher order potentials in end-to-end trainable conditional random fields. *CoRR*, abs/1511.08119, 2015. 7

[2] Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, COLT '92, pages 144–152, 1992. 39

[3] L. Breiman. Bias, variance, and arcing classifiers. Technical Report 460, Statistics Department, University of California at Berkeley, 1996. 40

[4] Liang-Chieh Chen, George Papandreou, Iasonas Kokkinos, Kevin Murphy, and Alan L Yuille. Deeplab: Semantic image segmentation with deep convolutional nets, atrous convolution, and fully connected crfs. *arXiv:1606.00915*, 2016. xv, 5, 7, 9, 10, 11, 12

[5] Liang-Chieh Chen, Yi Yang, Jiang Wang, Wei Xu, and Alan L. Yuille. Attention to scale: Scale-aware semantic image segmentation. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 3640–3649, 2016. 7

[6] Liang-Chieh Chen, Yukun Zhu, George Papandreou, Florian Schroff, and Hartwig Adam. Encoder-decoder with atrous separable convolution for semantic image segmentation. *CoRR*, abs/1802.02611, 2018. xviii, 20

[7] Savan Chhaniyara, Pished Bunnun, Lakmal D Seneviratne, and Kaspar Althoefer. Optical flow algorithm for velocity estimation of ground vehicles: A feasibility study. 1, 01 2008. 33

[8] Dan Ciresan, Alessandro Giusti, Luca M. Gambardella, and Jürgen Schmidhuber. Deep neural networks segment neuronal membranes in electron microscopy images. In F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, editors, *Advances in Neural Information Processing Systems 25*, pages 2843–2851. Curran Associates, Inc., 2012. 7

[9] Marius Cordts, Mohamed Omran, Sebastian Ramos, Timo Rehfeld, Markus Enzweiler, Rodrigo Benenson, Uwe Franke, Stefan Roth, and Bernt Schiele. The cityscapes dataset for semantic urban scene understanding. In *Proc. of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016. xviii, 3, 7, 15

[10] Navneet Dalal and Bill Triggs. Histograms of oriented gradients for human detection. In *Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05) - Volume 1 - Volume 01*, CVPR '05, pages 886–893, Washington, DC, USA, 2005. IEEE Computer Society. 7, 9

[11] Mandar Dixit, Si Chen, Dashan Gao, Nikhil Rasiwasia, and Nuno Vasconcelos. Scene classification with semantic fisher vectors. In *CVPR*, pages 2974–2983. IEEE Computer Society, 2015. 8

[12] Harris Drucker, Christopher J. C. Burges, Linda Kaufman, Alexander J. Smola, and Vladimir Vapnik. Support vector regression machines. In *NIPS*, pages 155–161. MIT Press, 1996. 39

[13] A Juozapavicius E. Atkociunas, R. Blake and M. Kazimianec. Image processing in road traffic analysis. In *Image Processing in Road Traffic Analysis*, volume 10, pages 315–332, 2005. 33

[14] David Eigen and Rob Fergus. Predicting depth, surface normals and semantic labels with a common multi-scale convolutional architecture. *ICCV*, 2014. 26

[15] P. Espinace, T. Kollar, A. Soto, and N. Roy. Indoor scene recognition through object detection. In *ICRA*, 2010. 5

[16] Gunnar Farnebäck. Two-frame motion estimation based on polynomial expansion. In *Proceedings of the 13th Scandinavian Conference on Image Analysis*, SCIA'03, pages 363–370, 2003. 34

[17] Pedro F. Felzenszwalb, Ross B. Girshick, David McAllester, and Deva Ramanan. Object detection with discriminatively trained part-based models. *IEEE Trans. Pattern Anal. Mach. Intell.*, 32(9):1627–1645, September 2010. 8

[18] Ross Girshick. Fast r-cnn. In *Proceedings of the 2015 IEEE International Conference on Computer Vision (ICCV)*, ICCV '15, pages 1440–1448, Washington, DC, USA, 2015. IEEE Computer Society. 8

[19] Ross Girshick, Jeff Donahue, Trevor Darrell, and Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *CVPR*, 2014. 8, 9

[20] Saurabh Gupta, Pablo Arbeláez, Ross Girshick, and Jitendra Malik. Indoor scene understanding with rgb-d images: Bottom-up segmentation, object detection and semantic segmentation. *IJCV*, 2015. 9, 10, 28, 29, 30

[21] Saurabh Gupta, Ross Girshick, Pablo Arbeláez, and Jitendra Malik. Learning rich features from RGB-D images for object detection and segmentation. In *ECCV*, 2014. XVI, 9, 10, 18, 26, 28, 29, 30

[22] Q. Ha, K. Watanabe, T. Karasawa, Y. Ushiku, and T. Harada. MFNet: Towards real-time semantic segmentation for autonomous vehicles with multi-spectral scenes. In *IROS*, 2017. 5

[23] Bharath Hariharan, Pablo Arbeláez, Ross Girshick, and Jitendra Malik. Hypercolumns for object segmentation and fine-grained localization. In *Computer Vision and Pattern Recognition (CVPR)*, 2015. 7

[24] Trevor Hastie, Robert Tibshirani, and Jerome Friedman. *The Elements of Statistical Learning*. Springer Series in Statistics. Springer New York Inc., 2001. 40

[25] C. Hazirbas, L. Ma, C. Domokos, and D. Cremers. Fusenet: Incorporating depth into semantic segmentation via fusion-based cnn architecture. In *Asian Conference on Computer Vision (ACCV)*, November 2016. XX, 26

[26] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. XV, 5, 8, 9, 10, 27, 40

[27] Li-Jia Li, Richard Socher, and Fei-Fei Li. Towards total scene understanding: Classification, annotation and segmentation in an automatic framework. In *CVPR*, 2009. 9

[28] Guosheng Lin, Chunhua Shen, Anton van den Hengel, and Ian D. Reid. Exploring context with deep structured models for semantic segmentation. *CoRR*, abs/1603.03183, 2016. XX, 26

[29] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, and Alexander C. Berg. Ssd: Single shot multibox detector. In *ECCV*, 2016. 9

[30] J. Long, E. Shelhamer, and T. Darrell. Fully convolutional networks for semantic segmentation. In *CVPR*, 2015. XVIII, 3, 7, 10, 20, 26

[31] David G. Lowe. Distinctive image features from scale-invariant keypoints. *Int. J. Comput. Vision*, 60(2):91–110, November 2004. 7, 9

[32] Bruce D. Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the 7th International Joint Conference on Artificial Intelligence - Volume 2*, IJCAI'81, pages 674–679, 1981. 34

[33] L. Ma, J. Stückler, C. Kerl, and D. Cremers. Multi-view deep learning for consistent semantic mapping with rgb-d cameras. In *IROS*, 2017. XX, 5, 26

[34] J. Malik, P. Arbelaez, and S. Gupta. Perceptual organization and recognition of indoor scenes from rgb-d images. In *CVPR*, 2013. 9, 10, 26, 27, 28, 29, 30

[35] Pushmeet Kohli Nathan Silberman, Derek Hoiem and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*, 2012. 17, 18

[36] Natalia Neverova, Pauline Luc, Camille Couprie, Jakob J. Verbeek, and Yann LeCun. Predicting deeper into the future of semantic segmentation. *CoRR*, abs/1703.07684, 2017. 58

[37] Hyeonwoo Noh, Seunghoon Hong, and Bohyung Han. Learning deconvolution network for semantic segmentation. *arXiv preprint arXiv:1505.04366*, 2015. 7

[38] Adam Paszke, Abhishek Chaurasia, Sangpil Kim, and Eugenio Culurciello. Enet: A deep neural network architecture for real-time semantic segmentation. *CoRR*, abs/1606.02147, 2016. XVIII, 20

[39] Joseph Redmon, Santosh Kumar Divvala, Ross B. Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 779–788, 2016. 9

[40] Joseph Redmon and Ali Farhadi. YOLO9000: Better, faster, stronger. In *CVPR*, 2017. 5, 9

[41] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'15, pages 91–99, Cambridge, MA, USA, 2015. MIT Press. 8

[42] X. Ren, L. Bo, and D. Fox. Rgb-(d) scene labeling: Features and algorithms. In *CVPR*, 2012. 29

[43] Dolley Shukla and Ekta Patel. Speed determination of moving vehicles using lucas-kanade algorithm. 2:32–36, 01 2012. 34

[44] Nathan Silberman, Derek Hoiem, Pushmeet Kohli, and Rob Fergus. Indoor segmentation and support inference from rgbd images. In *ECCV*, 2012. 15, 28, 29, 30

[45] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. In *International Conference on Learning Representations*, 2015. xv, 8, 9, 10, 27, 40

[46] X. Song, L. Herranz, and S. Jiang. Depth cnns for rgb-d scene recognition: Learning from scratch better than transferring from rgb-cnns. In *AAAI*, 2017. xx, xx, 9, 10, 30

[47] Indu Sreedevi, Manjari Gupta, and Prof Asok Bhattacharyya. Vehicle tracking and speed estimation using optical flow method. 3, 01 2011. 34

[48] Robert Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society, Series B*, 58:267–288, 1994. 39

[49] J. R. Uijlings, K. E. Sande, T. Gevers, and A. W. Smeulders. Selective search for object recognition. *Int. J. Comput. Vision*, 104(2):154–171, September 2013. 8

[50] Andrea Vedaldi and Andrew Zisserman. Efficient additive kernels via explicit feature maps. *IEEE Trans. Pattern Anal. Mach. Intell.*, 34(3):480–492, 2012. 13

[51] A. Wang, J. Cai, J. Liu, and T. Cham. Modality and component aware feature fusion for rgb-d scene classification. In *CVPR*, 2016. xx, xx, 9, 10, 30

[52] J. M. Wong, V. Kee, T. Le, S. Wagner, G. L. Mariottini, A. Schneider, L. Hamilton, R. Chipalkatty, M. Hebert, D. M. S. Johnson, J. Wu, B. Zhou, and A. Torralba. SegICP: Integrated deep semantic segmentation and pose estimation. In *IROS*, 2017. 5

[53] Ruobing Wu, Baoyuan Wang, Wenping Wang, and Yizhou Yu. Harvesting discriminative meta objects with deep cnn features for scene classification. In *ICCV*, pages 1287–1295. IEEE Computer Society, 2015. 8

[54] Fangting Xia, Peng Wang, Liang-Chieh Chen, and Alan L. Yuille. Zoom better to see clearer: Human and object parsing with hierarchical auto-zoom net. *ECCV*, 2016. 7

[55] J. Xiao, J. Hays, K. A. Ehinger, A. Oliva, and A. Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *CVPR*, 2010. 5

[56] Jian Yao, Sanja Fidler, and Raquel Urtasun. Describing the scene as a whole: Joint object detection, scene classification and semantic segmentation. In *CVPR*, 2012. 9

[57] D. Yoo, S. Park, J. Y. Lee, and In So Kweon. Multi-scale pyramid pooling for deep convolutional representation. In *2015 IEEE Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*, pages 71–80, June 2015. 8

[58] F. Yu and V. Koltun. Multi-scale context aggregation by dilated convolutions. In *ICLR*, 2016. xviii, 20

[59] Hengshuang Zhao, Jianping Shi, Xiaojuan Qi, Xiaogang Wang, and Jiaya Jia. Pyramid scene parsing network. In *CVPR*, 2017. xviii, 9, 20

[60] Shuai Zheng, Sadeep Jayasumana, Bernardino Romera-Paredes, Vibhav Vineet, Zhizhong Su, Dalong Du, Chang Huang, and Philip H. S. Torr. Conditional random fields as recurrent neural networks. *CoRR*, abs/1502.03240, 2015. 7

[61] Bolei Zhou, Aditya Khosla, Àgata Lapedriza, Aude Oliva, and Antonio Torralba. Object detectors emerge in deep scene cnns. In *ICLR*, 2015. 8

Universidad de Alcalá
Escuela Politécnica Superior

ESCUELA POLITECNICA
SUPERIOR

Universidad
de Alcalá