Detección de vehículos y flujo de tráfico con redes neuronales convolucionales en visión artificial

Juan Pedro Llerena

MÁSTER EN INGENIERÍA DE SISTEMAS Y CONTROL





TRABAJO FIN DE MÁSTER

Jun. de 2019

DIRECTOR: GONZALO PAJARES MARTINSANZ

MÁSTER EN INGENIERÍA DE SISTEMAS Y CONTROL

PROYECTO ESPECÍFICO PROPUESTO POR EL ALUMNO

Detección de vehículos y flujo de tráfico con redes neuronales convolucionales en visión artificial

---

Autor: Juan Pedro Llerena

Director: Gonzalo Pajares Martinsanz


VNIVERSITAS COMPLVTENSIS
MATRITENSIS


UNED



Autorización de difusión

Autorizamos a la Universidad Complutense y a la Universidad Nacional de Educación a Distancia a difundir y utilizar con fines académicos, no comerciales y mencionando expresamente a sus autores, tanto la memoria de este Trabajo Fin de Máster, como el código, la documentación y/o el prototipo desarrollado.

Firmado el autor:

Juan Pedro Llerena

24 de mayo de 2019

Resumen

El presente trabajo está orientado a la detección de vehículos en movimiento. Para ello se utilizan secuencias de imágenes en el tiempo sobre las que se aplican diferentes técnicas de visión artificial como herramientas para la obtención de variables de tránsito de vehículos en entornos urbanos, bajo el contexto de las ciudades inteligentes.

En lo que se refiere al movimiento, se formula una propuesta computacional desde un punto de vista psicológico. La psicología tiene una rama dedicada a la percepción humana, concretamente la percepción visual. Partiendo de las hipótesis que plantea la psicología, se fundamenta el movimiento en las teorías clásicas de visión artificial. En el presente trabajo se estudia esta evolución clásica y se propone una nueva técnica como herramienta para la identificación del movimiento.

Además, se pone especial énfasis en el estudio de las redes neuronales convolucionales (CNN, *Convolutional Neural Network*) como herramienta específica de clasificación dentro del paradigma general del aprendizaje profundo y en el análisis del movimiento. En el caso de las CNN, por tratase de redes con un elevado número de capas, se lleva a cabo un estudio sobre el número de éstas y se analizan cada una de ellas, incluyendo el diseño de nuevas arquitecturas. Se diseñan hasta un total de veinticuatro arquitecturas, basadas en los conceptos teóricos más relevantes de las CNN. Las arquitecturas mencionadas son entrenadas bajo un conjunto de datos específicos obtenidos manualmente del entorno de estudio, con el objetivo de llevar a cabo una clasificación específica de vehículos.

En este trabajo, se integran el conjunto de técnicas sobre un sistema capaz de proporcionar el flujo de vehículos que transitan por una vía determinada con capacidad para discernir entre distintos tipos de vehículos según su categoría y pertenencia a las diferentes clases establecidas en las CNN.

El conjunto de experimentos realizados permite realizar un análisis crítico de las distintas configuraciones de diseño de las CNN y de los métodos de movimiento en relación a su desempeño.

Palabras clave

Ciudades inteligentes, visión por computador, aprendizaje profundo, red neuronal convolucional, flujo óptico y similitud estructural.

Abstract

This Master Thesis is focused on the detection vehicles in movement. With such purpose we use video sequences of images to apply different artificial vision techniques, obtaining vehicle traffic variables in urban environments, under the context of smart cities.

Concerning the movement, it is analyzed from a psychological view. Psychology has a branch on human perception, namely the visual perception. Those hypotheses, serve as the base of the classic theories of movement in artificial vision. In this work, this classic evolution is studied and a new technique is proposed as a new tool for movement's identification.

In addition, a special emphasis is put on the study of Convolutional Neural Networks (CNN), as a specific classification tool within the general paradigm of deep learning in the analysis of movement. The CNN architecture has a high number of layers, which are being analysed not only on the number but also on the type of layers included in the design of new architectures. Up to a total number of twenty four new architectures are designed, based on the most relevant CNN theoretical concepts. All project architectures are trained from a specific dataset extracted from a specific case of analysis, being the main objective the classification of different classes of road vehicles.

In this work, the set of techniques are integrated into a system capable of providing the flow of vehicles traveling along a determined path, with the capacity to discern between different types of vehicles according to their category and belonging to the different classes established in the CNN.

The set of experiments carried out has allowed to divide the different design configurations of the CNN and the movement methods according to their performances.

Keywords

Smart cities, computer vision, deep learning, convolutional neuronal network, optic flow, structural similarity.

ÍNDICE GENERAL

ÍNDICE GENERAL ..... **VI**

LISTA DE FIGURAS ..... **VIII**

LISTA DE TABLAS ..... **X**

ABREVIATURAS ..... **XI**

AGRADECIMIENTOS ..... **XII**

CAPÍTULO 1: INTRODUCCIÓN **13**

1.1. OBJETIVOS 14
1.2. DESCRIPCIÓN DE LOS ASPECTOS MÁS RELEVANTES DESARROLLADOS 14
1.3. ORGANIZACIÓN DE LA MEMORIA 15

CAPÍTULO 2: IDENTIFICACIÓN DE MOVIMIENTO **17**

2.1 PERCEPCIÓN DEL MOVIMIENTO 17
1.2. FLUJO ÓPTICO 19
2.1.1 Definición teórica *20*
2.1.2 Descripción metodológica *22*
2.2 ÍNDICE DE SIMILITUD ESTRUCTURAL (SSIM) 25
2.2.1 Definición teórica *26*
2.2.2 Descripción metodológica *27*

CAPÍTULO 3: REDES NEURONALES ARTIFICIALES **31**

3.1 INTRODUCCIÓN: LA NEURONA BIOLÓGICA 31
3.2 REDES NEURONALES ARTIFICIALES (RNA) 32
3.2.1 La Neurona artificial *32*
3.2.1.1 Función de activación 33
3.2.2 Clasificación de redes *37*
3.2.2.1 Clasificación topológica 37
3.2.2.2 Clasificación por aprendizaje 39
3.2.3 Entrenamiento de ANN, Algoritmo "Back-Propagation" *40*
3.2.3.1 Optimizadores basados en Gradientes 42
3.2.3.2 Stochastic Gradient Descent (SGD) 43
3.3 REDES NEURONALES CONVOLUCIONALES (CNN) 43
3.3.1 Capas Convolucionales *44*
3.3.2 Capas de pooling *51*
3.3.3 Capas de aplanamiento (Flatten) *52*
3.3.4 Capas totalmente conectadas (full Conect) *53*
3.3.5 Capas de salida o Softmax *53*
3.3.6 Otras capas de interés *54*
3.3.6.1 Convoluciones dilatadas 54
3.3.6.2 Capas Dropout 55
3.3.6.3 Normalización 57

3.4 Reconocimiento de regiones en imágenes. 58
3.4.1 R-CNN *59*
3.4.2 Fast R-CNN. *60*
3.4.3 Faster R-CNN. *61*
3.5 Arquitecturas de redes utilizas 63
3.5.1 AlexNet *63*
3.5.2 ResNet50 ..... *64*
3.5.3 Vehicle *64*

CAPÍTULO 4: ENTRENAMIENTO 67

4.1 Entrenamiento y validación de las CNN 68
4.2 Entrenamiento y validación de R-CNN, fast R-CNN y Faster R-CNN 72

CAPÍTULO 5: TRÁNSITO DE VEHÍCULOS 75

5.1 Variables de tránsito de vehículos 75
5.2 Procedimientos y tecnologías de medida 77
5.3 Tecnología de medida por visión artificial 79

CAPÍTULO 6: RESULTADOS 83

6.1 Consideraciones preliminares 83
6.2 Resultado detección y clasificación de vehículos 85
6.3 Resultados del flujo de vehículos 93
6.4 Casos excepcionales observados 95

CAPÍTULO 7: CONCLUSIONES Y TRABAJO FUTURO 99

7.1 Conclusiones generales 99
7.2 Trabajo futuro 102

REFERENCIAS 105

ANEXO I. RECURSOS UTILIZADOS 109

ANEXO II. IMÁGENES DE CARACTERÍSTICAS 113

LISTA DE FIGURAS

Figura 1: Clasificación del movimiento 18
Figura 2: Movimiento de objetos: (a)Relativo y (b) Absoluto 18
Figura 3: Descripción gráfica del flujo óptico 19
Figura 4: Flujo óptico previsto desde cuatro direcciones diferentes (A,B,C,D) 19
Figura 5: Problema de apertura 21
Figura 6: Metodología de obtención de regiones por flujo óptico 22
Figura 7: (a) Flujo óptico en elemento móvil. (b) Flujo óptico por oscilación de cámara 23
Figura 8: Proceso de extracción de regiones por flujo óptico 24
Figura 9: Identificación de múltiples vehículos en movimiento 24
Figura 10: Solapamiento de objetos por dilatación 25
Figura 11: a) imágenes consecutivas superpuestas (morado, verde). b) SSIMmap 27
Figura 12: Proceso de identificación de regiones en movimiento con SSIM 28
Figura 13: a) Imagen binarizada y segmentada. b) Imagen detección de vehículo por SSIM 28
Figura 14: a) Superposición de 2 imágenes consecutivas. b) SSIM map imágenes anteriores. c) Binarización imagen b. 29
Figura 15: a) Superposición de dos imágenes consecutivas. b) SSIM map. c) Binarización. d) identificación de regiones móviles 30
Figura 16: Modelo general de neurona artificial 33
Figura 17: Modelo de perceptrón para dos clases 34
Figura 18: Función escalón ecuación 15 34
Figura 19: Función definida a trozos ecuación 16 35
Figura 20: a) Función sigmoide con $a = 2$ y $c = 4$; b) Función sigmoide con $a = -2$ y $c = 4$; c) Función tanh 35
Figura 21: Función de activación gaussiana con A=0.5 B=1.5 36
Figura 22: a) Función de activación ReLU; b) Función de activación ReLU con fugas 37
Figura 23: Arquitectura de red *feedforward* multicapa presente en una CNN 37
Figura 24: Red neuronal multicapa de retropropagación 38
Figura 25: Principales clases de arquitecturas de redes acíclicas 38
Figura 26: Ejemplo de red LSTM 39
Figura 27: Arquitectura estándar de una CNN 44
Figura 28: Ejemplo de convolución de tensor I bajo núcleo K con solapamiento total 46
Figura 29: Ejemplo de una convolución discreta 1 48
Figura 30: Ejemplo de una convolución discreta 2 49
Figura 31: Forma alternativa de interpretar los desplazamientos del núcleo 49
Figura 32: Forma alternativa de interpretar los desplazamientos del núcleo 50
Figura 33: Operación de la capa *pooling* 51
Figura 34: Operación de la capa *Pooling* 51
Figura 35: Ejemplo de Invarianza en *max pooling* 52

Figura 36: Ejemplo de invarianzas aprendidas 52
Figura 37: Proceso de aplanamiento o *flattening* 52
Figura 38: Capa *SoftMax* 53
Figura 39: Convolución dilatada para $i = 7$, $k = 3$ y $d = 2$. 54
Figura 40: Tres tipos de ajuste en entrenamiento 55
Figura 41: *Dopout* 56
Figura 42: *Dropout y max pooling* 56
Figura 43: Diferentes clases encontradas en A-6 (Madrid) 58
Figura 44: Proceso de R-CNN 59
Figura 45: Esquema Fast R-CNN 60
Figura 46: Evolución de estructura general CNN a estructura general Fast R-CNN 61
Figura 47: Esquema Faster R-CNNN 61
Figura 48: Evolución de estructura general Fast R-CNN a estructura general de Faster R-CNN 62
Figura 49: Arquitectura de capas AlexNet 63
Figura 50: Bloque de residuo 64
Figura 51: Arquitectura CNN Vehicle 65
Figura 52: Evolución entrenamiento AlexNet 69
Figura 53: Evolución entrenamiento ResNet50 70
Figura 54: Evolución entrenamiento Vehicle 32x32 71
Figura 55: Evolución entrenamiento Vehicle 64x64 71
Figura 56: Evolución entrenamiento Vehicle 128x128 72
Figura 57: Ejemplo de región de conteo acotado por líneas rojas. 80
Figura 58: Conteo y clasificación 80
Figura 59: Arquitectura general del proceso del sistema de flujo vehicular 83
Figura 60: Interfaz general de salida 83
Figura 61: Métodos utilizados para la detección y clasificación 84
Figura 62: Ejemplo de interfaz de clasificación 85
Figura 63: Tres casos excepcionales 95
Figura 64: Discriminación de áreas grandes por superposición de sombras 96
Figura 65: Perdida de contexto en clasificación 96
Figura 66: Ejemplo doble identificación 97
Figura 67: Falsos positivos en regiones y clasificación 97
Figura 69: Mapa de características en primera capa convolucional AlexNet 101
Figura 68: a) Imagen entrada; b) Canal con máxima activación; c) Imagen de activación del canal con máxima respuesta 101
Figura 70: Proceso de entrenamiento AlexNet en HVD 115
Figura 71: Proceso de entrenamiento AlexNet en ACB 116
Figura 72: Proceso de entrenamiento AlexNet en SEG 118
Figura 73: Proceso de entrenamiento AlexNet en RGB 119

LISTA DE TABLAS

Tabla 1: Rango de áreas de discriminación para cada uno de los métodos de flujo óptico 24
Tabla 2: Tamaño de áreas discriminadas en identificación de movimiento con método SSIM 28
Tabla 3: Base de datos propia 67
Tabla 4: Volumen de BBDD propia adaptado a las diferentes CNN 68
Tabla 5: Resultados entrenamiento CNN's 72
Tabla 6: Entrenamiento y validación R-CNN 73
Tabla 7: Entrenamiento y validación Fast R-CNN 73
Tabla 8: Entrenamiento y validación Faster R-CNN 73
Tabla 9: Ejemplo de tabla de flujo de vehículos 81
Tabla 10: Resumen de validaciones de redes 85
Tabla 11: Ejemplos de clasificación con cada una de las arquitecturas de CNN 87
Tabla 12: Comportamiento de los diferentes detectores ante una misma escena controlada* 89
Tabla 13: Identificación de clases en escena compleja 92
Tabla 14: Flujo de vehículos manual. 93
Tabla 15: Flujo de vehículos y tiempo cómputo para diferentes métodos 93
Tabla 16: Resultados gráficos de conteo de vehículos 95
Tabla 17: Comparación de imágenes en RGB,HVD, ACB y SEG 114
Tabla 18: Seis ejemplos de clasificación con AlexNet en HVD 116
Tabla 19: Seis ejemplos de clasificación con AlexNet en ACB 117
Tabla 20: Seis ejemplos de clasificación con AlexNet en SEG 119
Tabla 21: Seis ejemplos de clasificación con AlexNet en RGB 120

ABREVIATURAS

ACB	Imagen de características Apertura, Cierre y Borde codificada en RGB
ANN	Red Neuronal Artificial (“Artificial Neuronal Network”)
BBDD	Base de datos
BP	Propagación hacia atrás (“Back-Propagation”)
CNN	Red Neuronal Convolucional (“Convolutional Neural Network”)
DAG	Grafo Acíclico Dirigido (“Directed acyclic graph”)
DL	Aprendizaje profundo (“Deep Learning”)
Far	Método de Farneback
FC	Capa totalmente conectada (“Full Connected”)
FPS	Frames por Segundo
GT	Imagen de referencia (“Ground Truth”)
GTI	Grupo de tratamiento de imágenes Universidad Politécnica de Madrid
HS	Método de Horn Schunck
HVD	Imagen en características Horizontal, Vertical y Diagonal codificada en RGB
ILSVRC	ImageNet Large Scale Visual Recognition Challenge
IoT	Internet de las cosas (“Internet of Things”)
IOU	Ratio de unión sobre Intersección (“Intersection over Union”)
LK	Método de Lukas-Kanade
LK DoG	Método de Lucas Kanade derivado del método gaussiano
LSTM	“Long Short-Term Memory”
MSE	Error cuadrático medio (“Means Square Error”)
PPS	Potencial post sináptico
PSNR	Proporción máxima de señal a ruido (“Peak Signal-to-Noise Ratio”)
RBF	Redes con función de base radial (“Recurrent Radial Basis Function”)
R-CNN	Redes neuronales convolucionales basadas en region (“Region-based Convolutional Neural Networks)
ReLu	Unidad lineal rectificada (“Rectified Linear Unit“)
RNA	Red Neuronal Artificial
ROI	Región de interés (“Region of Interest”)
RPN	Red de propuesta de regiones. (“Region Proposal Network”)
SEG	Imagen de características por segmentación de los canales RGB, codificados en RGB
SegNet	Red de segmentación semántica (“Semantic Segmentation Network”)
SGD	Gradiente estocástico descendente (“Stochastic Gradient Descent”)
SSIM	Similitud estructural (“Structural similarity”)
SVM	Máquina de vectores de soporte (“Support Vector Machine”)
TFM	Trabajo Fin de Máster
TTS	Texto a voz (“Text To Speech”)
UAV	Vehículo aéreo no tripulado (“Unmanned Aircraft vehicle”)
UGV	Vehículo terrestre no tripulado (“Unmanned ground vehicle”)

AGRADECIMIENTOS

Tras terminar la Licenciatura en Ciencias Físicas de la mano de Jesús Manuel de la Cruz comencé mi vida laboral siempre con la inquietud por el "mundo del control" y en general la automática. Nunca podré estar lo suficientemente agradecido por haber sembrado esa inquietud en mí, por todo ello, gracias Jesús allá donde estés…

La visión por computador ha resultado una maravillosa novedad durante el estudio en este máster, un campo en el que me he podido adentrar de la mano de Gonzalo Pajares, sin sus notas y tutoriales de clase, figuras, decenas de correos, recomendaciones, correcciones, sabios consejos y sobre todo, su gran voluntad para enseñar o simplemente para tener una charla; este trabajo no hubiera sido posible. Por todo ello y mucho más, gracias Gonzalo.

Agradecer al conjunto de profesores de las materias que he tenido la suerte de cursar, pues ha resultado una aventura apasionante. Especialmente agradecer a Dictino Chaos por introducirme en el campo del control no lineal y no perder nunca la paciencia; a Matilde Santos por abrirme el mundo de la lógica borrosa; A Natividad Duro y Raquel Dormido por atenderme durante el periodo estival para que pudiera comprender los conceptos de minería de datos; a Sebastián Dormido por los comentarios en la materia de tratamiento de señales y las recomendaciones a terceros de mis actitudes. A J. Ignacio Hidalgo, por encaminarme durante el periodo estival en temas del "Simulated Annealing". También agradecer especialmente a Eva Besada, por atenderme siempre que lo he necesitado, ofrecerse a dirigir mi TFM, recomendarme a terceros, encaminarme en el mundo de las heurísticas y quizá en alguna otra aventura académica, Gracias.

A mis queridos amigos Alejandro Martínez, Ignacio Olabarría y Juan de la Torre; gracias por todas vuestras charlas, reflexiones, recomendaciones de estudio, consejos y escaladas que entre estudio y estudio me han llevado, entre otras tantas cosas, a escribir estas líneas. Siempre habéis sido, sois y seréis unas referencias para mí queridos amigos.

Agradecer a mi familia por estar ahí siempre con un apoyo incondicional.

A mi querida Ana; agradecerte toda la paciencia que has tenido y tienes conmigo en ésta y tantas otras aventuras, sin ti hubiera sido imposible llegar aquí y a tantos otros sitios, Gracias.

CAPÍTULO 1: INTRODUCCIÓN

Con el desarrollo tecnológico y los avances en las capacidades de procesado computacional, el aprendizaje profundo ("*deep learning*") se ha constituido en una herramienta útil de gran efectividad, en diversos ámbitos, incluyendo el procesamiento de imágenes.

Las ciudades inteligentes ("*smart cities*"), el internet de las cosas "IoT", (*internet of Things- IIoT*, en la industria), los vehículos autónomos (UAVs, USVs, UGVs, incluyendo los vehículos conectados) son claros ejemplos, donde el reconocimiento de patrones tales como señales de tráfico, control del flujo de vehículos, identificación de texturas, objetos u obstáculos en el entorno resultan de vital importancia.

En el entorno de las *smart cities*, determinar el flujo de vehículos en las arterias principales de las ciudades pude proporcionar una fuente de información clave de cara a la prevención y control de atascos, reducción de la contaminación en las ciudades, reducción de la probabilidad de accidentes, aumentar los beneficios de las empresas en movilidad entre otros. Es en este ámbito donde se centra el presente trabajo. Así se plantea el análisis de vehículos en movimiento circulando en vías de acceso o de salida de una gran ciudad. El enfoque se formula mediante el análisis de secuencias de imágenes. En este caso análisis de vídeo, dada la imposibilidad de disponer de un sistema de captura y transmisión de vídeo in situ. Bajo esta premisa, se aplican técnicas específicas de visión por computador, más concretamente la detección de flujo óptico para la identificación de objetos en movimiento y a partir de ahí clasificar estos en diferentes tipos mediante técnicas basadas en aprendizaje profundo, como son las Redes Neuronales Convolucionales (CNN).

Los sistemas propuestos podrían alojarse en UAV,s con trayectorias planificadas con el objetivo de proporcionar información a los diferentes centros de control de tráfico.

Así pues, la motivación que origina la presente propuesta se enmarca en un contexto de futuro, como son las ciudades inteligentes, donde sin duda se implantarán sistemas basados en capturas de vídeo, que proporcionarán información muy valiosa de cara a dotar en el futuro a las ciudades de esa inteligencia necesaria para mejorar la eficiencia en todos los sentidos, que sin duda redundarán en una mayor calidad de vida. El desarrollo de la aplicación mediante técnicas tecnológicas de vanguardia, actualmente en continua evolución constituyen un reto importante en el desarrollo de la propuesta que se plantea.

1.1. Objetivos

En este trabajo se pretende mostrar diferentes técnicas con las que poder identificar vehículos en tiempo real mediante el análisis de secuencias de imágenes para realizar un control de flujo de vehículos. Los objetivos específicos que se plantean en el proyecto son los siguientes:

I. Identificación de movimiento en escenas con vehículos en movimiento.
II. Identificación de objetos en escenas con técnicas de Deep Learning (DL), basadas en CNN.
III. Generación de base de datos de entrenamiento y validación propia de elementos a clasificar.
IV. Comparación en detección y clasificación de vehículos entre técnicas de detección de cambios y técnicas de identificación de movimiento.
V. Obtención de variables de tráfico con las técnicas indicadas.
VI. Análisis de los resultados entre las diferentes técnicas utilizadas.

1.2. Descripción de los aspectos más relevantes desarrollados

En este trabajo se proponen diferentes técnicas de visión artificial como herramientas para determinar variables de tránsito de vehículos. Para ello se realiza un estudio de todas estas técnicas con el fin de proporcionar una herramienta de software en visión por computador, con la que poder determinar estas variables.

Para alcanzar este objetivo se desglosa el trabajo en 7 capítulos, poniendo espacial énfasis en lo referente al análisis del movimiento como método para determinar regiones donde se ha producido movimiento (capítulo 2) y en las técnicas de aprendizaje profundo con redes neuronales convolucionales (capítulo 3).

Para el estudio del movimiento se acude a técnicas clásicas de la visión por computador y se comparan con una técnica propuesta en este trabajo para la determinación de regiones en movimiento, basada en el índice de similitud estructural entre imágenes consecutivas.

En lo referido a las técnicas de aprendizaje profundo se analizan las estructuras de las CNN aplicando técnicas de transferencia de conocimiento entre estructuras de redes validadas en la literatura, las cuales son reentrenadas para una clasificación y detección concreta del trabajo.

También se implementan arquitecturas de redes neuronales artificiales propias fundamentadas en la teoría expuesta, con el objetivo de acelerar los procesos de cómputo, tanto en detección-clasificación, como en sus fases de entrenamiento.

A esta fase de entrenamiento se dedica un capítulo completo (capítulo 4), dada la gran importancia del mismo, para el correcto funcionamiento de una aplicación concreta sustentada en las redes neuronales artificiales (RNAs). En él se mencionan algunas plataformas que proporcionan datos etiquetados y se presenta una base de datos de elaboración propia creada para el propósito concreto de este trabajo.

Finalmente, se exponen resultados experimentales con los que se obtienen diferentes conclusiones y se proponen algunos posibles trabajos futuros.

1.3. Organización de la memoria

A continuación, se exponen de forma resumida los contenidos que aparecen en los diferentes capítulos de este trabajo.

Capítulo 1. Introducción.

En este capítulo introductorio se plantea el problema, los objetivos a alcanzar en la resolución del mismo, así como la organización del trabajo, el cual refleja lo indicado previamente con mención expresa al planteamiento general y descripción de objetivos.

Capítulo 2. Identificación del movimiento.

A lo largo de este capítulo se estudia el movimiento desde el punto de vista de la psicología de la percepción visual para extender las reflexiones al ámbito de la visión artificial a través del concepto de flujo óptico. Finalmente se introduce el concepto de índice de similitud estructural como propuesta para la identificación de regiones en movimiento.

Capítulo 3. Redes neuronales artificiales.

Partiendo de la motivación biológica de varios trabajos que se extienden desde las publicaciones del premio nobel en medicina Camillo Golgi y José Antonio Ramón y Cajal en 1906, en este capítulo se comienza introduciendo al lector en la fundamentación biológica con la que justificar la estructura de una neurona artificial y seguidamente las redes neuronales artificiales (RNA).

Se pone especial atención en las CNN, su estructura de capas y tipos de capas más comunes de la literatura, así como el estudio de arquitecturas asociadas para la identificación de regiones y clasificación mediante R-CNN, Fast R-CNN y Faster R-CNN, para concluir el capítulo con la implementación por transferencia de conocimiento y entrenamiento de dos arquitecturas famosas y una red ligera de implementación propia.

Capítulo 4: Datos de entrenamiento.

Dada la importancia de las bases de datos y el proceso de entrenamiento en las ANN, en este capítulo hablaremos de algunas BBDD de fácil acceso para el entrenamiento de CNN. Por otro lado, con el fin de entender el proceso de elaboración de las misma y mejorar el comportamiento local de los clasificadores elaborados, se describe la BBDD creada para el entrenamiento de las CNN propias definidas al final del capítulo 3. En este capítulo podremos encontrar datos de entrenamiento y validación con nuestra BBDD para las más de 10 arquitecturas de CNN trabajadas.

Capítulo 5: Tránsito de vehículos

En este capítulo se realiza una introducción a las variables de tránsito de vehículos utilizadas por la ingeniería civil en su campo de la ingeniería de tráfico. Se describen las metodologías y algunas tecnologías más relevantes para la obtención de las mismas. Finalmente, el capítulo describe la técnica de visión artificial implementada para la obtención del flujo vehicular.

Capítulo 6. Resultados.

En este capítulo se recogen los principales resultados obtenidos en los diferentes experimentos, donde se pueden comparar con las diferentes tablas de resultados.

Capítulo 7. Conclusiones y trabajos futuros.

Este apartado recoge las conclusiones extraídas de las pruebas realizadas. Asimismo, se realiza una reflexión sobre posibles trabajos futuros relacionados con puntos concretos estudiados durante la elaboración de este trabajo.

CAPÍTULO 2: IDENTIFICACIÓN DE MOVIMIENTO

En este capítulo se describen los métodos habituales de la literatura con el objetivo de determinar dónde se está produciendo movimiento para posteriormente identificar y clasificar lo que se ha movido a partir de secuencias de imágenes.

Desde el punto de vista del reconocimiento de imágenes podemos decir que este capítulo hace referencia a un método de identificación de regiones o propuestas de regiones de interés (ROI Region of Interes), que como veremos más adelante, constituyen las entradas a las CNN.

Comenzamos definiendo cómo percibimos los humanos el movimiento en nuestro entorno para de este modo exponer diferentes técnicas que permitan obtener dichas relaciones mediante la visión por computador o visión artificial.

No solo se describen con métodos clásicos de determinación de flujo óptico, sino que se propone un nuevo método fundamentado en el índice de similitud estructural (SSIM) de las imágenes, que acelera los procesos habituales y proporciona gran información.

2.1 Percepción del movimiento

La percepción del movimiento mediante la visión es una característica esencial en el mundo animal, donde identificamos movimientos propios, o el de otros objetos móviles (naturales o artificiales), adquiriendo un valor esencial en la supervivencia, conservación y adaptación de las especies [2] y [3]. [1] y [2].

Una de las cuestiones relevantes planteadas en el campo de la percepción del movimiento trata de diferenciar entre el propio movimiento del observador y de otros movimientos ajenos a él.

También existen planteamientos que estudian el movimiento “aparente”, diferenciándolo del movimiento “real”. Estas clasificaciones aparecen resumidas en el siguiente esquema:

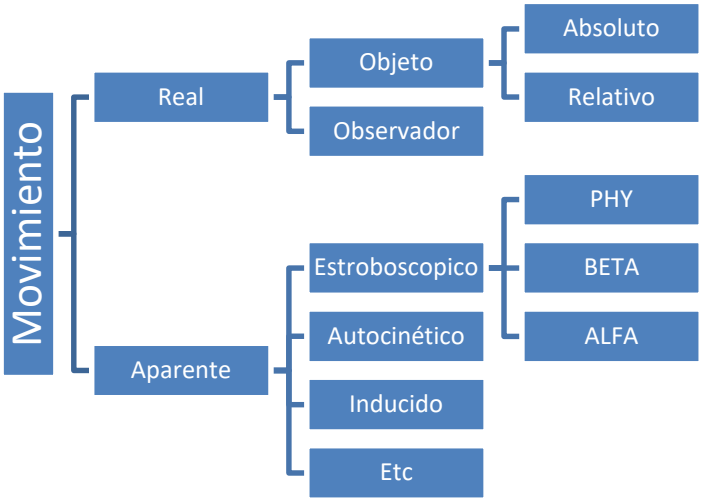


Figura 1: Clasificación del movimiento

En nuestro caso nos centraremos en el campo del movimiento real, desde el cual podremos extender una "sinapsis" entre éste y el aparente.

Para poder diferencial entre el movimiento real de objetos o del observador acudimos a las siguientes teorías de Gibson y Stevens [2], que afirman lo siguiente:

Gibson sugiere que podemos diferenciar el movimiento del observador del movimiento de un estímulo fijándonos en si el objeto se desplaza o no respecto al contexto [1].

La teoría de Stevens [2], afirma que si se mueve toda la imagen sobre el "mosaico retinal", se interpreta como movimiento del observador, pero si se mueve una parte de dicho "mosaico retinal", implica movimiento del objeto integrado en dicho rango.

En lo que se refiere al movimiento de objetos absoluto o relativo diremos que desde el punto de vista perceptivo se denomina movimiento relativo aquel que tiene lugar cuando un móvil se desplaza a través de un fondo estructurado/texturizado, es decir existen referencias con las que posicionar contextualmente el objeto, por ejemplo, un vehículo en una ciudad. Por otro lado, el movimiento de objetos absoluto hace referencia al movimiento de aquellos objetos sobre un "vacío" contextual sin referencias, como puede ser la visualización de una aeronave desde otra a misma velocidad en un cielo limpio y despejado. Los objetos en movimiento que aparecen en sendas imágenes de la figura 2, son buenos ejemplos de ambos tipos de movimientos.

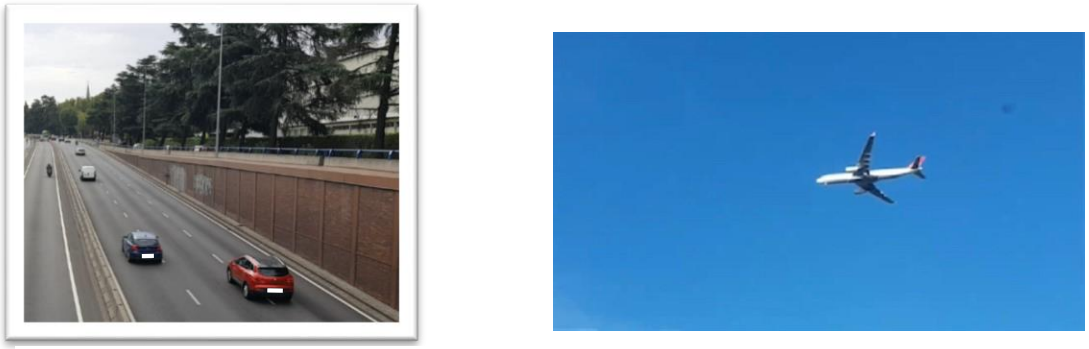


Figura 2: Movimiento de objetos: (a)Relativo y (b) Absoluto

Así pues, el movimiento real se halla inseparablemente ligado a unos determinados patrones de relación entre disposiciones consecutivas. Es decir, la percepción de movimiento está ligada a una cierta correlación entre imágenes consecutivas en el tiempo y captadas a partir de la escena donde se produce el movimiento.

A continuación, se exponen los diferentes métodos con los cuales es posible modelar la percepción del movimiento con vistas a su identificación.

1.2. Flujo óptico

Partiendo del trabajo de Gibson [3], donde fotografió el morro de un avión en diferentes momentos, éste pudo obtener un vector velocidad procedente de la unión de la dimensionalidad espacial de las imágenes más la dimensión temporal procedente de la sucesión de las mismas.

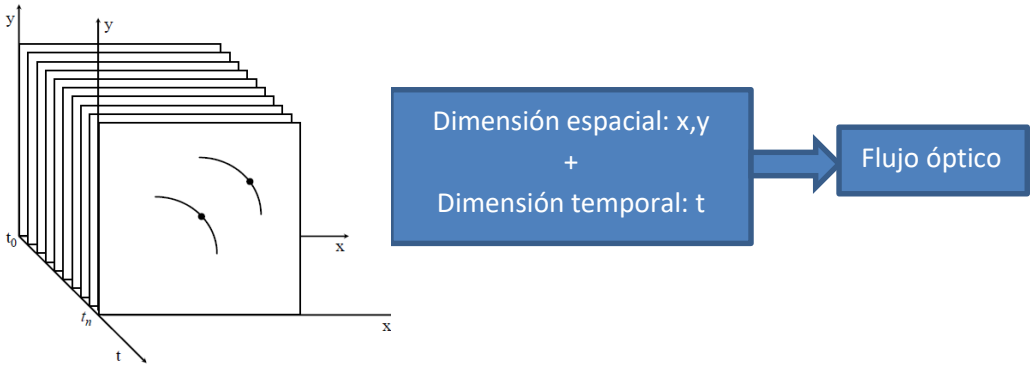


Figura 3: Descripción gráfica del flujo óptico

La figura 3 muestra un esquema relativo al concepto espacio-temporal del flujo óptico en secuencias de imágenes.

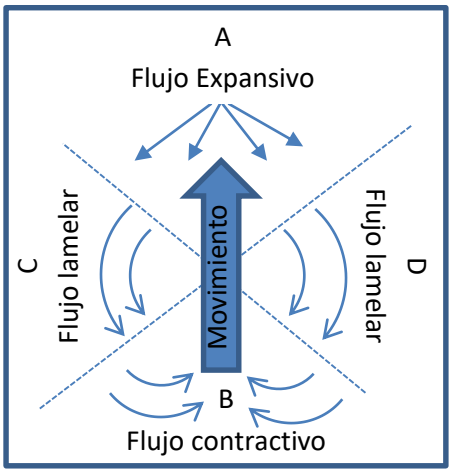


Figura 4: Flujo óptico previsto desde cuatro direcciones diferentes (A,B,C,D)

Es decir, el objetivo es determinar un vector velocidad con las componentes de intensidad espaciales de las imágenes consecutivas.

La interpretación del flujo óptico según movimiento y posición del observador queda gráficamente definida en la figura 4, donde aparece representada la dirección del movimiento de un objeto el cual percibe tres tipos diferentes de flujo óptico dependiendo de la observación desde el móvil. Esto se representa con el nombre correspondiente a la clasificación del flujo óptico y las direcciones de las flechas.

Estos tres tipos diferentes de flujo óptico se describen a continuación:

A. **Flujo frontal, expansivo radial:** Indica aproximación lo que se traduce en un aumento del tamaño de los objetos conforme se acercan
B. **Flujo posterior o contractivo radial**: Indica alejamiento del centro de contracción, lo que se traduce en una disminución en el tamaño de los objetos
C. **Flujo lateral o lamelar**: se aprecia un gradiente en la evolución del flujo óptico en lo que se refiere a la profundidad. Es decir, lo próximo se mueve "más rápido" sobre nuestra retina mientras que los objetos más alejados pueden permanecer casi inmóviles. A este término se le denomina dentro de la psicología que estudia la percepción del movimiento como "paralaje de movimiento".

2.1.1Definición teórica

Dada la función $f(x, y, t)$ de intensidad de píxel (x,y) en el instante t, definimos el flujo óptico como:

$$\frac{\partial f}{\partial t} + \boldsymbol{v}.\boldsymbol{G} = 0 \quad (1)$$

Donde **v** es el vector velocidad y **G** el gradiente espacial.

Desarrollando f en serie de taylor obtenemos:

$$f(x + dx, y + dy, t + dt) = f(x, y, t) + f_x dx + f_y dy + f_t dt + O(\partial^2) \quad (2)$$

De este modo la ecuación de flujo óptico queda:

$$-f_t = f_x \frac{dx}{dt} + f_y \frac{dy}{dt} \quad (3)$$

Es decir podemos determinar el vector velocidad, $\boldsymbol{v} = (u, v) = (\frac{dx}{dt}, \frac{dy}{dt})$, dado que las cantidades $f_x, f_y \; y \; f_t$, son cantidades medibles.

Esta ecuación asume que la intensidad de la imagen durante el movimiento permanece constante a lo largo de la trayectoria s. Esto implica, entre otras consecuencias, las siguientes: cualquier cambio en la intensidad de la imagen se debe al movimiento, la iluminación de la escena permanece constante o las superficies de los objetos son opacas (**superficies Lambertianas**). En ocasiones la suposición de que la intensidad permanece constante no es cierta, particularmente si la iluminación cambia. Para evitar esta posibilidad se aplica la restricción de gradiente constante a lo largo de la trayectoria del movimiento, por tanto, se recurre a la consideración de que a pesar de que pueda violarse la restricción de intensidad constante lo que si va a permanecer constante a lo largo de dicha trayectoria es el gradiente:

$$\frac{d\nabla f}{ds} = 0 \quad (4)$$

$$\begin{bmatrix} \partial^2 f/\partial x^2 & \partial^2 f/\partial x \partial y \\ \partial^2 f/\partial x \partial y & \partial^2 f/\partial y^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = -\frac{\partial \nabla f}{\partial t} \quad (5)$$

Considerando un entorno de vecindad Ω alrededor de cada punto sobre el que se extiende el sumatorio se obtiene:

$$\begin{bmatrix} \sum_\Omega f_x^2 & \sum_\Omega f_x f_y \\ \sum_\Omega f_x f_y & \sum_\Omega f_y^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = -\begin{bmatrix} \sum_\Omega f_t f_x \\ \sum_\Omega f_t f_y \end{bmatrix} \equiv Av = b \quad (6)$$

Despejando la velocidad **v** de la ecuación anterior obtenemos de forma matricial:

$$v = (A^T A)^{-1} A^T b \quad (7)$$

A este método que considera que el flujo óptico es constante en la vecindad de un pixel se le conoce como el **método de Lucas-Kanade** [2] y [4].

Otro método altamente utilizado en la determinación del flujo óptico es el **método de Horn-Schunck** [4]**.**

Este método resuelve el problema de apertura. El problema de apertura consiste en que dada la ecuación de flujo óptico (3), tenemos una ecuación con dos incógnitas (u,v), esto quiere decir que tenemos infinitas soluciones.

Podemos formular (u,v) en términos de (f_x, f_y) pero no en su dirección perpendicular:

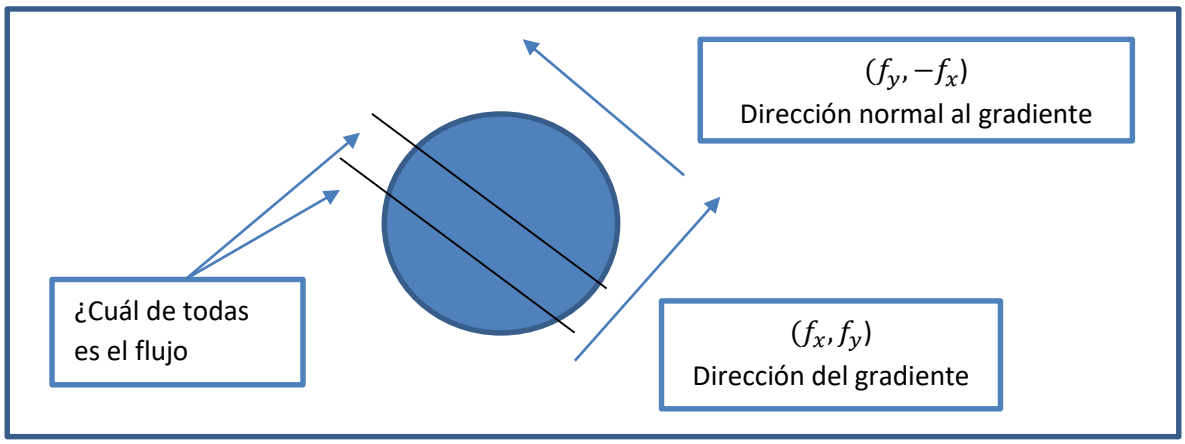


Figura 5: Problema de apertura

El método de **Horn-Schunck** es considerado como un método global porque determina el vector de flujo óptico en cada píxel. Este método introduce el término de suavidad, que viene a decir que la variación de intensidad entre una zona oscura y otra iluminada se realiza de forma suave pasando por diferentes niveles entre imágenes consecutivas.

Para ello el método introduce una función de energía a minimizar:

$$E = \iint \left[(f_x u + f_y v + f_t)^2 + \alpha(\|\nabla u\|^2 + \|\nabla v\|^2) \right] dxdy \quad (8)$$

Donde α es una constante de regularización. Valores mayores de α corresponden a flujos más suaves [6].

2.1.2 Descripción metodológica

Tras la formulación matemática del flujo óptico, en este apartado se describe la metodología implementada para la obtención de regiones utilizando el flujo óptico como elemento principal de dicha metodología. En nuestro problema de identificación de movimiento se realiza el proceso a alto nivel que aparece en la figura6.

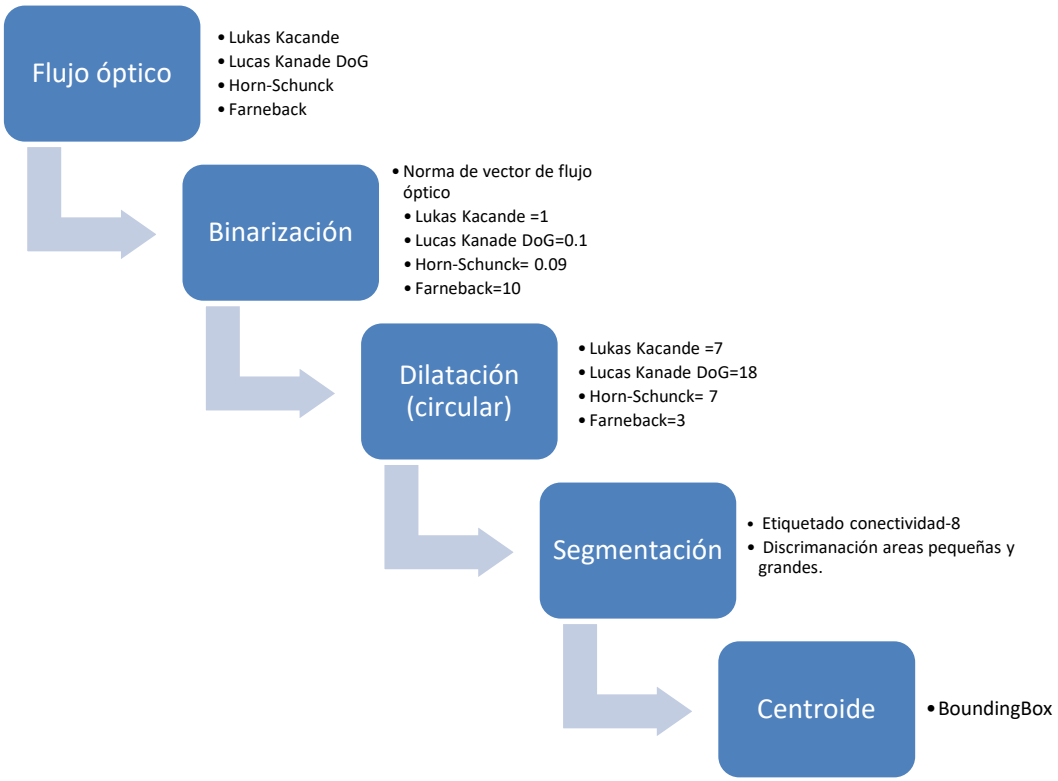
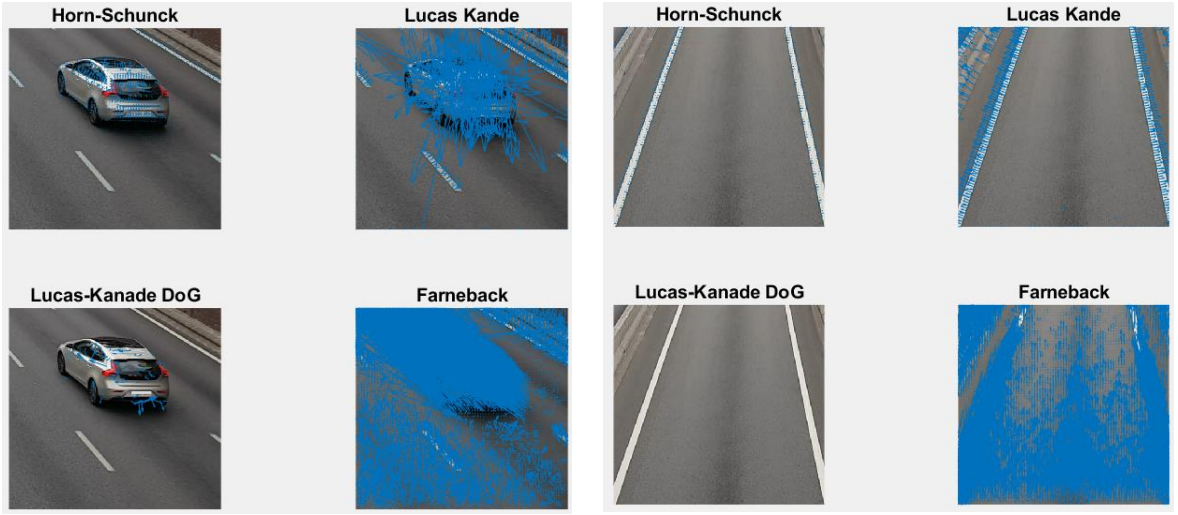


Figura 6: Metodología de obtención de regiones por flujo óptico

En la secuencia de imágenes de la figura 7, se muestra el comportamiento de cada uno de los métodos de determinación de flujo óptico de un elemento en movimiento, observándose distintos resultados según el método.



(a) (b)

Figura 7: (a) Flujo óptico en elemento móvil. (b) Flujo óptico por oscilación de cámara

Una vez identificado sobre la escena los objetos o regiones en movimiento, se aplica el proceso de Binarización bajo umbral "l" de magnitud del módulo del vector de flujo óptico asociado a cada píxel, la motivación de esto reside en eliminar el ruido que puede aparecer por perturbación de la imagen, así como de la definición de regiones compactas.

Puesto que con el proceso de Binarización solo obtenemos resultado a nivel de píxel, es fácil que una geometría irregular en movimiento pueda tener componentes de velocidad dentro y fuera del margen del módulo definido, se aplica un proceso de dilatación morfológica que nos permite unir estas pequeñas regiones. Esto se realiza mediante una dilatación estructural basada en un círculo de radio "r" [4].

El siguiente proceso consiste en una segmentación de las regiones, es decir etiquetar cada una de las regiones diferentes dentro de la imagen. Esta operación se realiza mediante el algoritmo de etiquetado de componentes conexas [7]. Para este propósito se utiliza la segmentación por conectividad-8, y puesto que podemos encontrarnos con áreas muy grandes y muy pequeñas que pueden no pertenecer a un objeto móvil, se discriminan dichas áreas bajo los criterios de áreas máximas y mínimas mostrados en la (tabla 1).

Para finalizar el proceso, se determina el centroide de las regiones conexas resultantes y se genera un rectángulo con área mínima capaz de abarcar las regiones aglutinadas por cada uno de sus centroides, generando una caja que como veremos en próximos capítulos utilizaremos para recortar dicha sección de una imagen y clasificarla, que es lo que se conoce comúnmente como *"Bounding Box".*

En la figura 8 podemos ver gráficamente cada fase anteriormente descrita, donde se han separado los cuatros métodos implementados para poder observar el comportamiento con cada uno de ellos.

También se puede ver bajo el nombre del método utilizado por columna, los valores de las constantes utilizados para dilatación "r" y el umbral "l" para la binarización de las imágenes por flujo óptico.

En la última fila de la figura 8 se muestra el *bounding box*, con el centroide "+" sobre la misma.

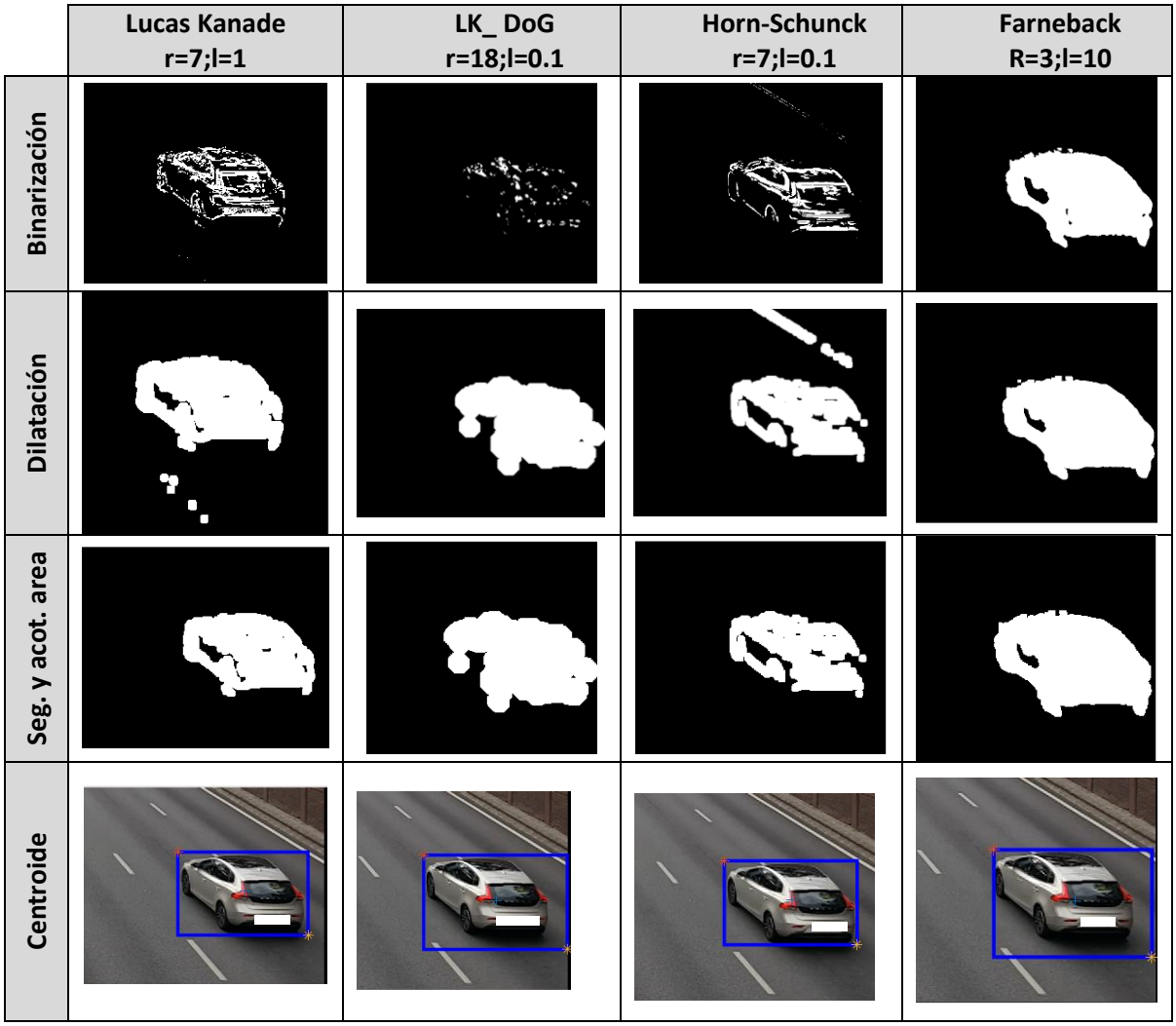


Figura 8: Proceso de extracción de regiones por flujo óptico

Áreas [pixel^2]	LK	LK-DoG	HS	Farne
Máxima	50000	50000	50000	50000
Mínima	5000	1457	5000	5000

Tabla 1: Rango de áreas de discriminación para cada uno de los métodos de flujo óptico



Figura 9: Identificación de múltiples vehículos en movimiento

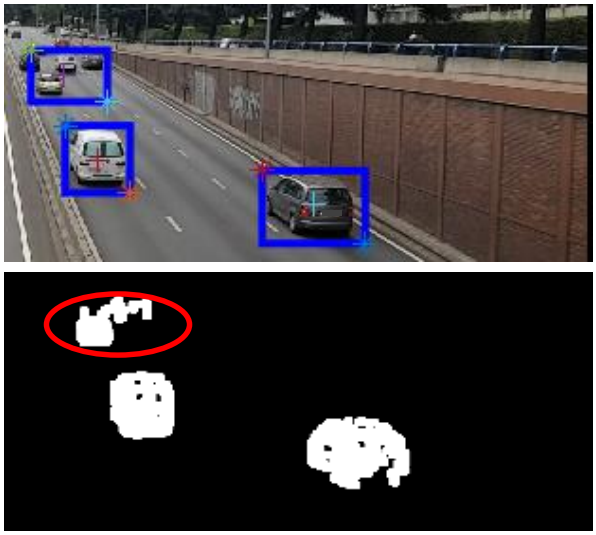


Figura 10: Solapamiento de objetos por dilatación

Uno de los problemas que aparecen en el caso de múltiples vehículos es la superposición de los mismos debido al proceso de dilatación, esto puede evitarse acotando la región del campo visual, así como con el posicionamiento de la cámara y un buen estudio de los valores medios máximos del módulo de los vectores de flujo óptico.

Este mismo problema también aparece con las sombras de vehículos, ya que partimos del vector de velocidades de flujo óptico y las sombras se desplazan del mismo modo que los vehículos, por lo que resulta esencial el estudio de la región donde se desea emplazar dichas cámaras.

Por otro lado, y aunque en este caso, ha quedado fuera del ámbito de aplicación, para este último caso cabe la posibilidad de aplicar técnicas de identificación de sombras con vistas a su identificación y supresión [8].

2.2 Índice de similitud estructural (SSIM)

En la identificación del movimiento resulta esencial una variable temporal, este aspecto se consigue con la captura de imágenes (frames) consecutivas en el tiempo. De este modo es posible detectar movimiento en una secuencia de imágenes con una simple operación de diferencia entre imágenes consecutivas píxel a píxel entre fotogramas.

Una de tales aproximaciones es el índice de similitud estructural (SSIM, Structural Similarity Index for Mesuring) que se utiliza para medir la similitud entre dos imágenes, cuando han sido sometidas por ejemplo a un proceso de compresión y descompresión. El índice SSIM es una métrica de referencia para tratar de mejorar la calidad de las imágenes, medida por relaciones tales como la relación señal / ruido pico (PSNR) y el error cuadrático medio (MSE).

Esta métrica evalúa el impacto visual de tres características de una imagen: luminancia, contraste y estructura [10] y [11].

El objetivo es determinar cuándo se ha producido movimiento y dónde dentro del campo visual del sistema de percepción. Para ello se parte de la hipótesis de que una secuencia de imágenes de una escena sin movimiento, en un corto periodo de tiempo, no presentará variación, en caso contrario aparece algún elemento móvil dentro de la misma.

Es decir, se trata de obtener la similitud entre la imagen actual y la inmediatamente anterior en el tiempo de la secuencia y de este modo se puede identificar dónde se ha producido cambio, lo cual en este caso se manifiesta por un aumento de la entropía de la zona en cuestión.

2.2.1 Definición teórica

Como se ha comentado previamente, el SSIM se realiza sobre dos imágenes X e Y de la misma resolución espacial. Con tal propósito se definen diferentes variables estadísticas a nivel de píxel y sobre la imagen de grises, como son: μ_x, μ_y corresponden a los valores de las medias en un entorno de vecindad de un píxel dado en las respectivas imágenes X e Y. Los términos σ_x, σ_y y σ_{xy} son las desviaciones típicas de las imágenes X e Y, mientras que σ_{xy} hace referencia a la covarianza de X e Y. Mientras que los valores medios de píxel están relacionados con la luminosidad media de los píxeles de X e Y respectivamente, las desviaciones típicas nos indican una estimación del contraste entre los píxeles de cada imagen y finalmente la covarianza proporciona una estimación de la tendencia de variar conjuntamente X e Y [11].

Los índices de comparación de luminosidad, contraste y estructura entre las imágenes X e Y vienen dados respectivamente por [10]:

$$\begin{aligned} l(x,y) &= \frac{2\mu_x\mu_y + C1}{\mu_x^2 + \mu_y^2 + C1} \\ c(x,y) &= \frac{2\sigma_x\sigma_y + C2}{\sigma_x^2 + \sigma_y^2 + C2} \\ s(x,y) &= \frac{\sigma_{xy} + C3}{\sigma_x\sigma_y + C3} \end{aligned} \qquad (9)$$

C1, C2 y C3 son constantes que se utilizan para evitar que los denominadores de los índices anteriores presenten singularidades, es decir tiendan a cero. Así pues, se define SSIM como el producto ponderado de los tres índices de similitud:

$$SSIM(x,y) = [l(x,y)^{\alpha}.c(x,y)^{\beta}.s(x,y)^{\gamma}] \qquad (10)$$

Donde α,β y γ, son parámetros que indican la relevancia de cada uno de los índices, en nuestro caso al igual que en [10] tomaremos $\alpha = \beta = \gamma = 1$. También tomamos que $C3 = C2/2$, de este modo obtenemos:

$$SSIM(x,y) = \frac{(2\mu_x\mu_y + C1)(2\sigma_{xy} + C2)}{(\mu_x^2 + \mu_y^2 + C1)(\sigma_x^2 + \sigma_y^2 + C2)} \qquad (11)$$

Para obtener C1 y C2 [10] propone:

$$\begin{aligned} C1 &= (K_1 L)^2 \\ C2 &= (K_2 L)^2 \end{aligned} \qquad (12)$$

Donde L es el rango dinámico de intensidad el cual depende de la precisión de la imagen (en nuestro caso es de 8 bits lo que implica L=255). Finalmente $K_1 \ll 1$ y $K_2 \ll 1$, son constantes a determinar empíricamente de forma experimental pero con valores mucho más pequeños que 1, en nuestro caso [6] los valores son los siguiente:

$$K_1 = 0.01 \; y \; K_2 = 0.03 \qquad (13)$$

Finalmente, SSIM asignará el valor de 1 para imágenes idénticas y cero para imágenes totalmente diferentes.

Para realizar las estadísticas a nivel local, [10] propone la utilización de ventanas de ponderación gaussianas, normalizadas a 1, con las que poder determinar el SSIM localmente y poder generar un mapa de SSIM. Estas gaussinas tienen un radio o desviación estándar de 1.5 [6] y [10].

En la figura 11 se puede observar en primer lugar la variación que han sufrido dos imágenes consecutivas donde se representa la imagen actual en morado y en verde la inmediatamente anterior, el contexto de la imagen no ha sufrido una gran variación por ello se mantiene la escena en grises. Por otro lado, en la imagen b) Se muestra, a nivel de píxel, la variación que ha sufrido la consecución de imágenes en función del SSIM, donde se observa en colores oscuros las regiones donde se ha producido un cambio. No solo se observa el vehículo en movimiento procedente de la escena, sino que se observa también variaciones en el contexto (líneas de carretera, muro, arcén y asfalto). Esto se debe a la vibración de la cámara durante el proceso de captura de los vídeos, ya que éstos se obtuvieron con la cámara en la mano del observador.



Figura 11: a) imágenes consecutivas superpuestas (morado, verde). b) SSIMmap

2.2.2 Descripción metodológica

Este método resulta muy sensible identificando variaciones procedentes de vibraciones por el viento por lo que en este trabajo se ha decidido combinarlo con un proceso de binarización y segmentación para eliminar ciertas regiones pequeñas procedentes de dichas oscilaciones, así como regiones grandes que puedan proceder de fondos como el cielo o sombras. En la figura 12 se muestra el proceso completo asociado a la metodología completa del método.

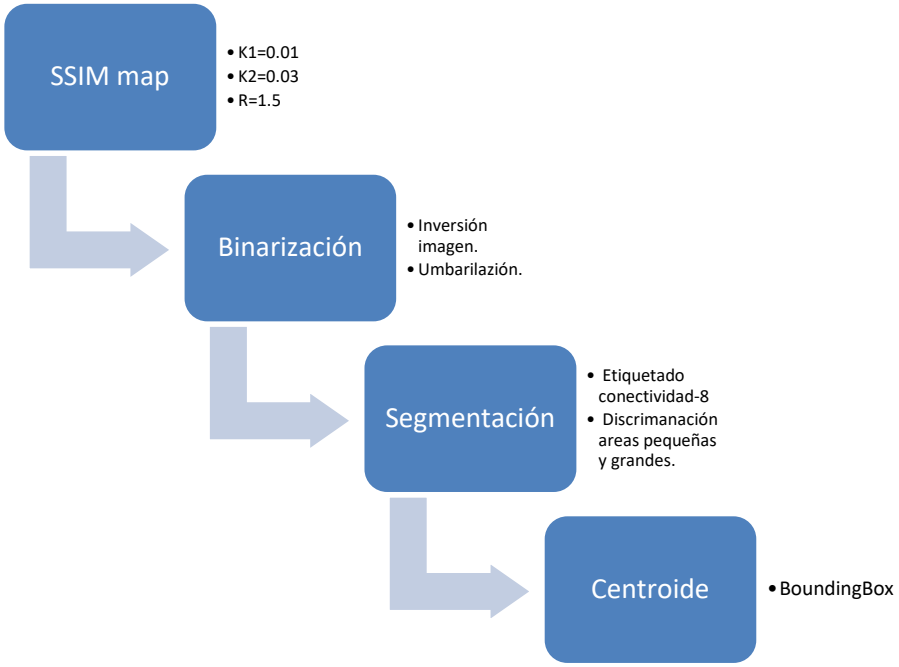


Figura 12: Proceso de identificación de regiones en movimiento con SSIM

Los valores tomados para las áreas de discriminación son los que se muestran en la tabla 2.

Tipo de áreas a discriminar	**Valores asignados [pixel^2]**
Áreas más pequeñas que:	9000
Áreas más grandes que:	50000

Tabla 2: Tamaño de áreas discriminadas en identificación de movimiento con método SSIM

Las regiones identificadas se rodean con el correspondiente "*bounding box*" en torno a su centroide. Dicho recuadro posee el perímetro mínimo posible para inscribir la región identificada [6]. De este modo se obtienen los resultados mostrados en la figura 13, donde aparece la imagen binarizada en (a) y el correspondiente recuadro en (b) junto su centroide representado por el símbolo "+".



Figura 13: a) Imagen binarizada y segmentada. b) Imagen detección de vehículo por SSIM

A continuación, se muestra un conjunto de ejemplos sobre los que se ha aplicado SSIM. En la figura 14 se muestra el proceso completo de la metodología indicada anteriormente para una escena sin vehículos en movimiento, donde se puede apreciar el ruido procedente de la vibración de la cámara. La primera de las imágenes de la figura 14 muestra la superposición de dos imágenes consecutivas de una escena en la que no hay movimiento (deseado), de este modo a nivel "ocular" no se puede

apreciar variación. En la imagen (b) se muestra el mapa de SSIM procedente de las imágenes de a), donde podemos apreciar esas variaciones producidas por perturbaciones debidas a movimiento subyacente durante la captura. Finalmente, en la imagen (c) se muestra el mapa generado por el método SSIM de la imagen (b) tras el proceso de binarización definido anteriormente.

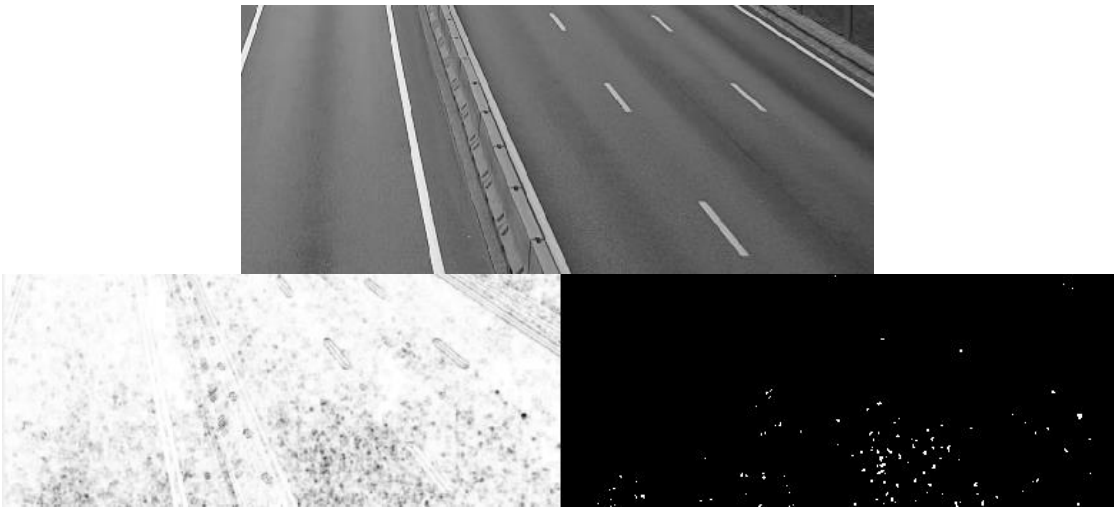


Figura 14: a) Superposición de 2 imágenes consecutivas. b) SSIM map imágenes anteriores. c) Binarización imagen b.

En la figura 15 se muestra el comportamiento del método ante una escena en la que hay movimiento de varios objetos (vehículos). En esta figura aparece en primer lugar la superposición de dos imágenes consecutivas, representando en color verde la primera imagen y en morado la inmediatamente posterior en el tiempo. En la imagen (b) se observa el mapa de SSIM de la escena de (a), representando el movimiento con colores grises y la similitud con blanco, siempre a nivel local (píxel). Para la imagen (c) se ha sometido a (b) al proceso de binarización y segmentación indicado en la metodología, observando claramente el movimiento más relevante de los vehículos y discriminando el ruido procedente de las perturbaciones. Finalmente, en la imagen (d) se muestra el *bounding box* de la escena con los centroides correspondientes a las regiones en movimiento.

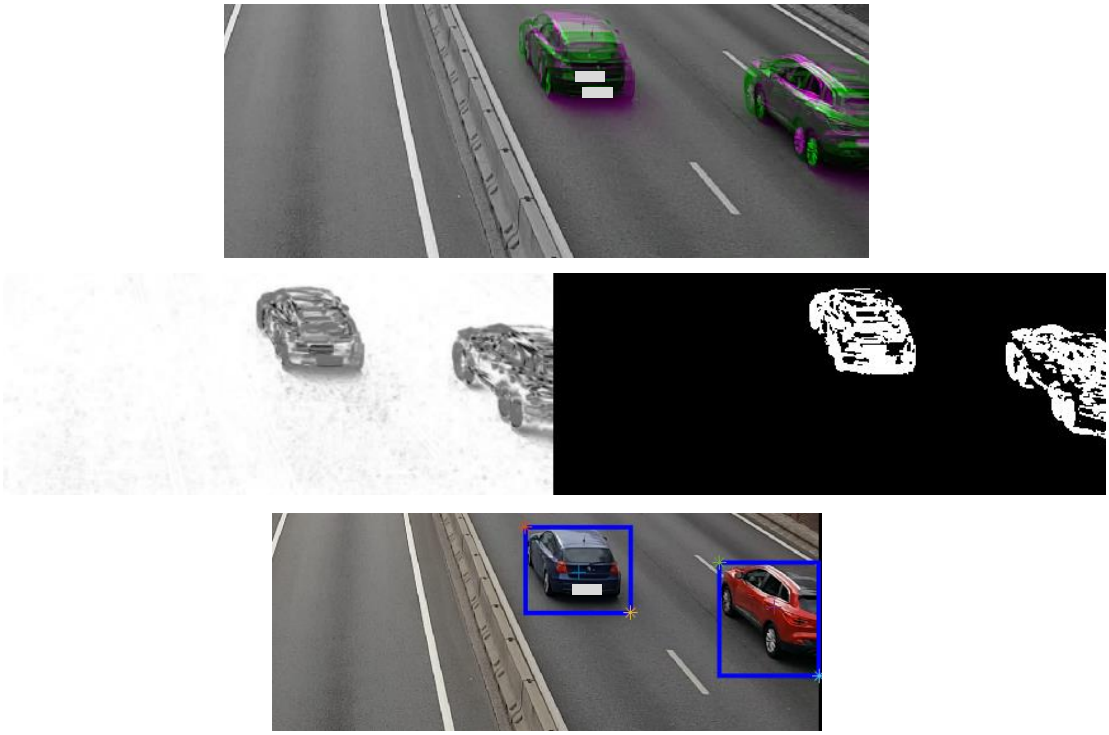


Figura 15: a) Superposición de dos imágenes consecutivas. b) SSIM map. c) Binarización. d) identificación de regiones móviles

A partir de los resultados mostrados en la imagen 15, se deduce cómo la entrada en escena de objetos móviles disminuye el ruido. Esto es consecuencia de la reducción de "sensibilidad" del método al disminuir localmente los índices en función del cómputo global, es decir, el sistema es más "fiable" cuanto mayor sea el movimiento o la disimilitud entre imágenes.

CAPÍTULO 3: REDES NEURONALES ARTIFICIALES

Las redes neuronales artificiales (RNA) cada vez están más presentes en las aplicaciones tecnológicas de la sociedad, donde podemos encontrarlas en simples cámaras fotográficas, aplicaciones de realidad aumentada y virtual, controladores de sistemas, así como en herramientas de predicción de comportamientos económicos o incluso en generación de gráficos en función del contexto que rodea la escena [12].

En este capítulo nos adentraremos en el mundo de las RNA partiendo de su motivación biológica hasta llegar a la implementación y validación de las arquitecturas utilizadas en este trabajo, con especial énfasis en las arquitecturas de las redes neuronales convolucionales (CNN).

3.1 Introducción: La neurona biológica

La neurona biológica, fue descubierta en 1836 y su estructura con varias entradas y una salida les supuso a sus descubridores, Camillo Golgi y Santiago Ramón y Cajal, el Premio Nobel de Medicina en 1906. En el cerebro humano existen aproximadamente diez mil millones de neuronas (1010 células nerviosas). Cada neurona tiene un cuerpo celular (soma), que tiene un núcleo celular. A partir del cuerpo de la neurona se ramifican una cantidad de fibras llamadas dendritas y una única fibra larga llamada axón. El axón mide en general 100 veces más que el diámetro del cuerpo de la célula (aproximadamente 1 cm). Los axones y dendritas transmiten las señales entre neuronas: el axón permite enviar señales a otras neuronas mientras que las dendritas permiten la recepción de dichas señales. Una neurona se conecta con un número variable de neuronas (entre 100 y 100000), formando un conjunto de conexiones sinápticas. Se estima que existen unas 1014 sinapsis en el cerebro de un adulto. Las señales se propagan entre neuronas mediante una reacción electroquímica: A nivel de la sinapsis, la neurona que genera la señal (que puede ser excitadora o inhibitoria) emite unos neurotransmisores que activan o desactivan los receptores de la neurona que recibe la señal, cuando se supera un umbral. La excitación se propaga rápidamente entre neuronas gracias a su gran conectividad. Esta fisiología es bien conocida y se puede encontrar en multitud de textos como por ejemplo en [13] y [14].

La regla de aprendizaje de Hebb indica que ocurre un cambio metabólico en la sinapsis cuando la entrada de una neurona está siendo activada repetidamente de manera persistente, reduciendo la resistencia de la sinapsis [15]. Los huecos presentes en las sinapsis y que se encargan de atenuar las señales de entrada de una neurona, cambian de tamaño en respuesta al “aprendizaje”. En términos informáticos la red neuronal del cerebro forma un sistema masivo de procesamiento paralelo de

información, en contraste con las computadoras en las cuales un único procesador ejecuta linealmente una serie de instrucciones [15]. Por otra parte, la velocidad de procesamiento representa una diferencia importante. Mientras que el cerebro opera a unos 100 Hz (100 operaciones por segundo), una computadora convencional procesa varias centenas de millones de operaciones por segundo. A pesar del "hardware" lento con el que está construido, el cerebro posee una serie de capacidades notables:

- Su rendimiento tiende a degradarse lentamente bajo daño parcial. En contraste, la mayoría de los programas y sistemas son frágiles: si se daña alguna parte arbitrariamente, es muy probable que el sistema completo deje de funcionar.
- Puede aprender (reorganizarse) a partir de la experiencia.
- Es tolerante a fallos. La recuperación de los daños es posible si unidades sanas logran reemplazar a las unidades dañadas, aprendiendo y asumiendo funciones de éstas.
- Ejecuta una cantidad masiva de cálculos en paralelo de forma extremadamente eficiente. Por ejemplo, la percepción visual compleja, que equivale a 10 etapas de procesamiento, ocurre en menos de 100 milisegundos.
- Sustenta a la inteligencia y la conciencia. La ciencia todavía no ha logrado determinar cómo ocurre esto exactamente.

3.2 Redes neuronales artificiales (RNA)

A continuación, se realiza una revisión a nivel computacional de los conceptos de las RNA partiendo de la unidad básica de la neurona artificial, sus clasificaciones y la forma de aprendizaje (entrenamiento) de las RNA que utilizaremos a lo largo del trabajo.

3.2.1 La Neurona artificial

La neurona artificial trata de capturar la esencia de los sistemas neuronales biológicos e imitar su comportamiento.

Los primeros avances datan de los años cincuenta y sesenta, cuando el científico Frank Rosenblatt, inspirado por los trabajos previos de Warren MCCulloch y Walter Pitts, desarrolló los perceptrones [4]. Estos primeros avances sobre neuronas artificiales se fundamentan en que una neurona artificial puede describirse de la siguiente manera:

- Recibe una serie de "entradas" (sean datos originales o "salidas" de otras neuronas). Cada entrada llega a través de una conexión que tiene una cierta "fuerza", o "peso", que equivale a la eficiencia sináptica de una neurona biológica. Cada neurona tiene también un determinado valor de umbral. La suma ponderada de las entradas menos el valor de umbral compone la "activación" de la neurona (también conocida como Potencial Post-Sináptico (PPS) de la neurona).

- La señal de activación pasa a través de una función de activación, o función de transferencia, para producir la "salida" de la neurona. Esta función limita el rango de valores que puede tomar la variable de salida de la neurona.

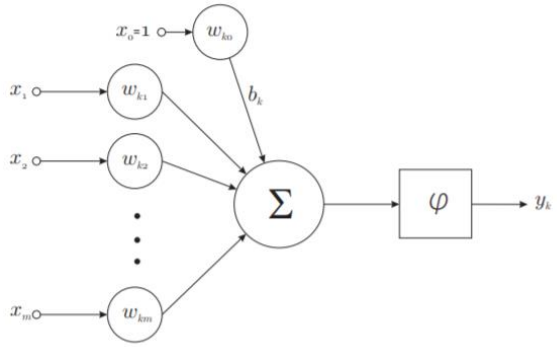


Figura 16: Modelo general de neurona artificial

Para formular lo anteriormente definido, consideremos que existe una señal x_j a la entrada de la sinapsis j de la neurona k, al atravesar la j-ésima conexión sináptica, la variable x_j se multiplica por el peso w_{kj} y un umbral b_k ("Bias") es agregado al resultado. Este resultado pasa a ser mapeado por una función de activación φ. En la literatura resulta habitual considerar b_k como un peso extra w_{k0}.

$$y_k = \varphi\left(\sum_{j=1}^{m} w_{kj}x_j + b_k\right) \quad (14)$$

3.2.1.1 Función de activación

La primera neurona artificial funcional, llamada perceptrón, estaba compuesta de una función de activación (φ) escalón la cual sólo puede tomar valores binarios.

Actualmente no es común usar redes neuronales basadas en perceptrones, el principal modelo que se usa es la neurona sigmoidal [16]. Esta neurona puede tomar valores de entrada entre 0 y 1, a diferencia del perceptrón que sólo puede tomar valores de entrada binarios. Esta función de activación satura la salida y como comenta [15] su derivabilidad posee una vinculación directa con el proceso de aprendizaje. La figura 17 muestra el elemento de activación de una neurona, en este caso, un perceptrón. [4].

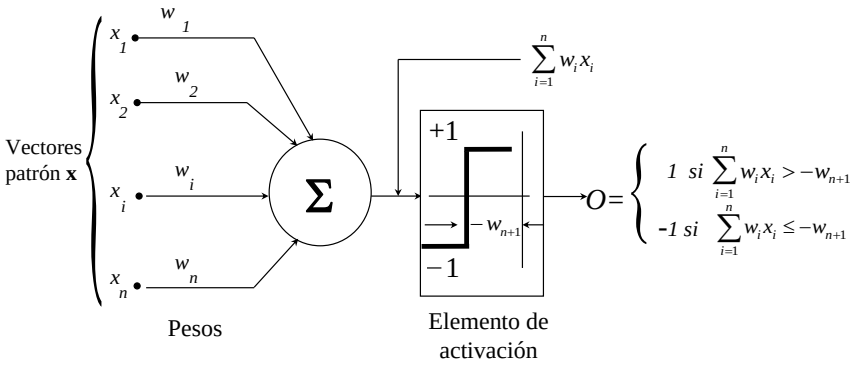


Figura 17: Modelo de perceptrón para dos clases

A continuación, definiremos las principales funciones de activación que podemos encontrar en la literatura.

Función escalón

Esta función también recibe el nombre de neurona de McCulloch-Pitts [4] y queda definida por la ecuación (15). Se utiliza principalmente en problemas de clasificación ya que la salida binaria es apropiada para la selección entre dos clases (Ej: fondo, objeto concreto).

$$\varphi(v) = \begin{cases} 1, & v \geq 0 \\ 0, & v < 0 \end{cases} \quad (15)$$

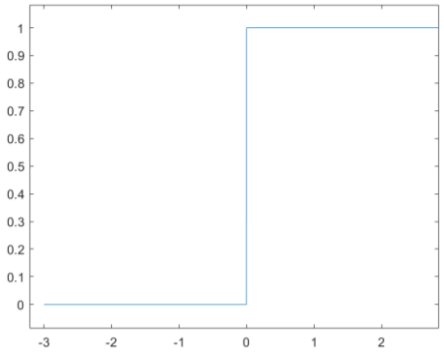


Figura 18: Función escalón ecuación 15

Función lineal a trozos

Esta función se define mediante la ecuación (16) y tiene la siguiente forma:

$$\varphi(v) = \begin{cases} 1, & v \leq -1/2 \\ v + 0.5, & 1/2 > v > -1/2 \\ 0, & v \geq 1/2 \end{cases} \quad (16)$$

El factor de amplificación (pendiente) en la región lineal de operación se asume que es la unidad. La forma de esta función de activación se aproxima a la respuesta de un amplificador no lineal. Se pueden dar los siguientes dos importantes casos particulares de la función lineal a trozos [17]:

- La función lineal a trozos se convierte en un combinador lineal si la región lineal de operación se mantiene sin entrar en saturación.

- La función lineal a trozos se convierte en un escalón si el factor de amplificación en la región lineal tiende a infinito.

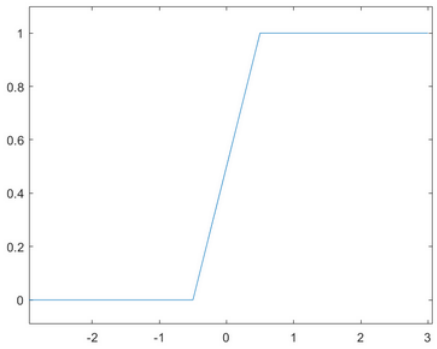


Figura 19: Función definida a trozos ecuación 16

Función sigmoide

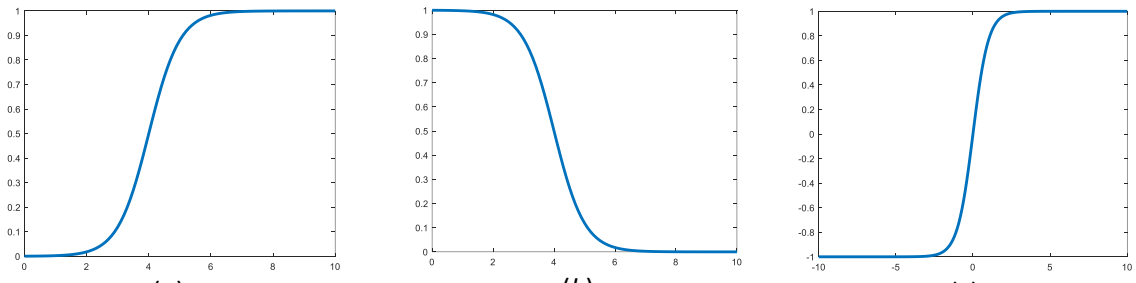
EL sigmoide es una función muy popular con forma de "s", estrictamente creciente y suave. Se describe mediante la ecuación (17):

$$\varphi(a, v, c) = \frac{1}{1 + e^{-a(v-c)}} \qquad (17)$$

Dependiendo del signo de la variable "a" la función sigmoide se abre hacia la izquierda o hacia la derecha, siendo apropiada para representar conceptos tales como "muy grande" o "muy negativo". La importancia de esta función es que su derivada es siempre positiva y cercana a cero para valores grandes en valor absoluto. Esta característica resulta fundamental a la hora de aplicar las reglas de aprendizaje.

Haciendo que la pendiente "a" sea muy elevada se logra que la sigmoide se aproxime a la función escalón. Además, si se desea trasladar el intervalo de [0 1] a [-1 1] esta función sigmoide se suele definir como la tangente hiperbólica:

$$\varphi(v) = \tanh v = 2sigm(2v) - 1 \qquad (18)$$



(*a*) (*b*) (c)

Figura 20: a) Función sigmoide con $a = 2$ y $c = 4$; b) Función sigmoide con $a = -2$ y $c = 4$; c) Función tanh

Función gausiana

Finalmente, otra de las funciones más utilizadas como activación en las RNA son las gaussianas, manteniendo las propiedades de derivabilidad anteriormente mencionada en la sigmoide. En este caso se puede ajustar la anchura y la amplitud de la misma mediante los parámetros A y B:

$$\varphi(v) = \mathrm{A}e^{-Bv} \qquad (19)$$

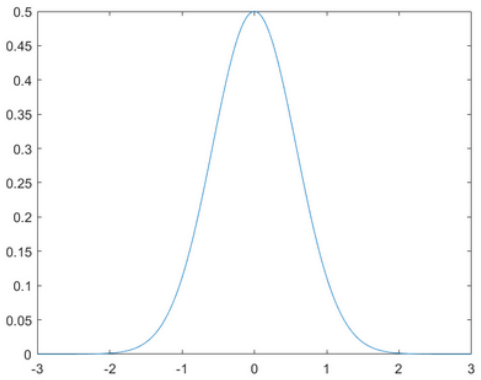


Figura 21: Función de activación gaussiana con A=0.5 B=1.5

Función ReLU

En la actualidad la mayoría de las ANN utilizan las funciones de activación Rectified Linear Unit (ReLU), que se define como:

$$\varphi(v) = \max(0, \mathrm{v}) \qquad (20)$$

Esto permite el paso de todos los valores positivos sin cambiarlos, mientras que pone a cero todos los valores negativos. Se podría entender como una función definida a trozos con pendiente 1 y ordenada en el origen a cero. Por su gran relevancia en las CNN, a continuación se indican su características más importantes:

a) ***Gradiente no saturado***, por el hecho de que *x* > 0; por lo que el problema de la dispersión del gradiente en proceso de propagación inversa se ve aliviado y los parámetros en la primera capa de la red neuronal pueden actualizarse de forma rápida;

b) ***Baja complejidad computacional***, dada su propia definición.

No obstante posee la desventaja de que **la neurona ReLU puede morir cuando recibe un gradiente negativo alto durante la retropropagación** que le permite aprender más porque su derivada es cero cuando su entrada es menor que cero, por lo que el gradiente será cero. Esto se puede evitar al inicializar cuidadosamente los pesos o utilizar ReLU con "fugas", que es como ReLU, pero su salida es lineal multiplicada por un valor pequeño (0,001 aproximadamente) cuando la entrada es negativa.

$$\varphi(v) = \max(0.01\mathrm{v}, \mathrm{v}) \qquad (21)$$

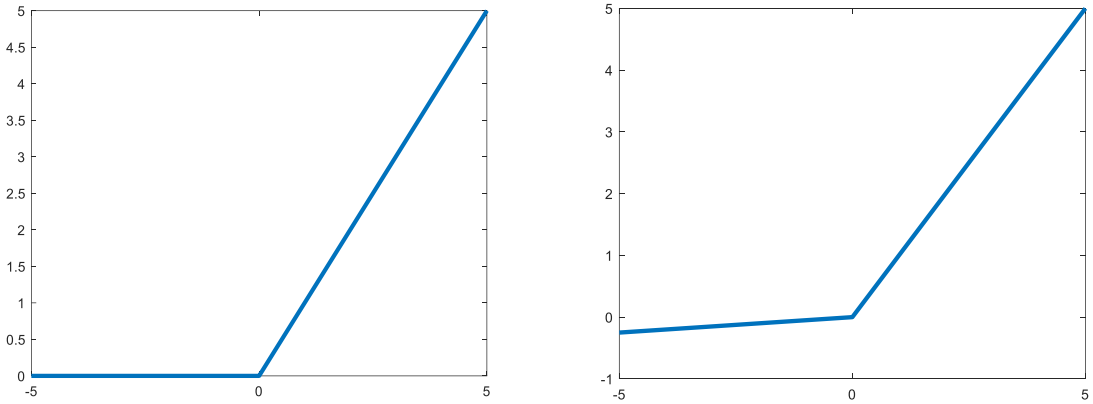


Figura 22: a) Función de activación ReLU; b) Función de activación ReLU con fugas

3.2.2 Clasificación de redes

Según el criterio que escojamos para clasificar las RNA tendremos una clasificación u otra. Lo más común es usar su estructura (topología) y el tipo de aprendizaje como criterios de clasificación.

3.2.2.1 Clasificación topológica

Si nos fijamos en la arquitectura podemos tener dos posibilidades diferentes. Si la arquitectura de la red no presenta ciclos, es decir, no se puede trazar un camino de una neurona a sí misma, la red se llama unidireccional (***feedforward***).

Dentro de este tipo podemos encontrar redes monocapa y multicapa [18].

Una arquitectura típica de este tipo de red son las CNN que son en las que se enfoca el presente trabajo, exponiéndolas en próximos apartados por sus grandes ventajas ante problemas de clasificación con imágenes.

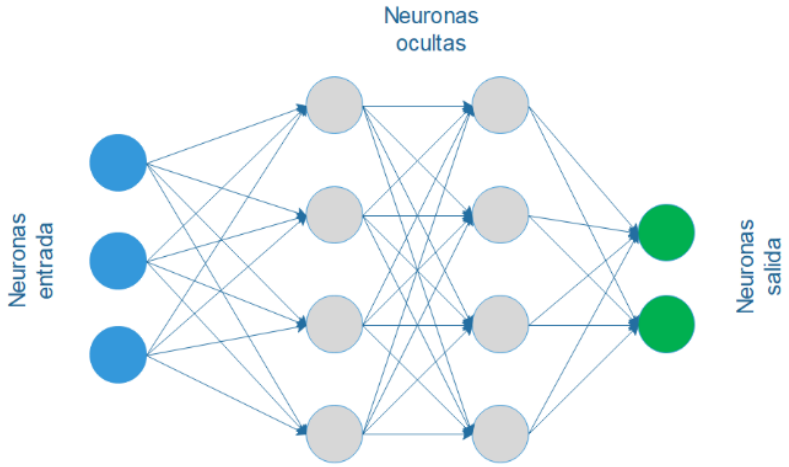


Figura 23: Arquitectura de red *feedforward* multicapa presente en una CNN

La arquitectura de una red neuronal suele constar de una capa de entrada, donde se ubican las neuronas de entrada, una zona intermedia en la que se ubican las capas ocultas y una capa de salida. [16].

Las redes neuronales en las que la salida de una capa se usa como la entrada de la siguiente se denominan *redes de propagación hacia delante*. Las redes en que las neuronas pueden estar conectadas indistintamente con neuronas de niveles previos o posteriores de su mismo nivel se denominan *redes de propagación hacia atrás o recurrentes* [16].

La figura 24 muestra la arquitectura básica de una red neuronal de retropropagación.

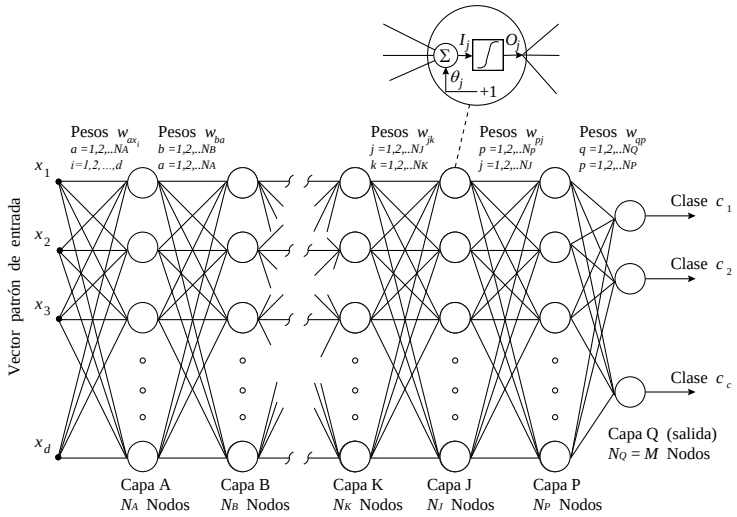


Figura 24: Red neuronal multicapa de retropropagación

La red de la figura 24 consta de capas de neuronas de cómputo estructuradas de forma idéntica y colocadas de manera que la salida de cada neurona en una capa proporciona la entrada de cada neurona en la siguiente capa [4].

Dentro de este tipo de redes podemos diferenciar dos clases principales de redes figura 25.

- Series
- Grafo Acíclico Dirigido (DAG Network)

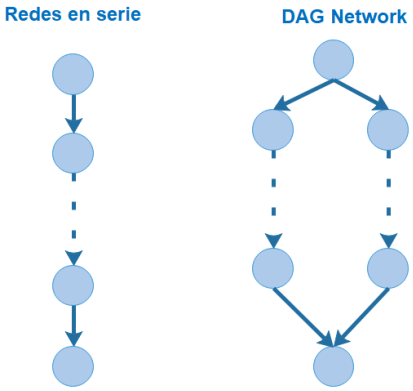


Figura 25: Principales clases de arquitecturas de redes acíclicas

Como ejemplo de estos dos tipos tenemos AlexNet [19] y Resnet50 [20] donde AlexNet corresponde a una rede del tipo Series mientras que ResNet50 es una DAG. Esto se definirá rigurosamente en los siguientes apartados.

Por el contrario, si se puede trazar un camino de una neurona a sí misma la arquitectura presenta ciclos. Este tipo de redes se denominan recurrentes o realimentados (**recurrent**).

En este grupo podríamos incluir las redes "*Long Short-Term Memory*" (LSTM), ampliamente utilizadas en problemas de predicción de modelos temporales por sus grandes ventajas asociadas a las células de memoria [21].

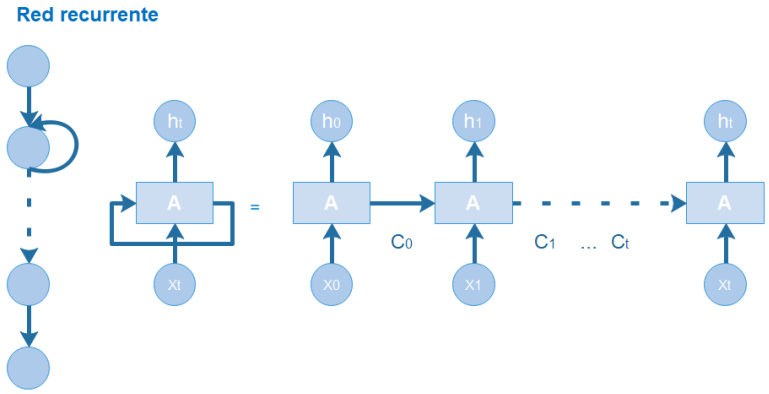


Figura 26: Ejemplo de red LSTM

3.2.2.2 Clasificación por aprendizaje

El otro criterio más habitual para clasificar las redes neuronales es mediante el tipo de aprendizaje que se utilice. Según [22] y [23] hay cuatro clases de aprendizaje distintos:

- **Aprendizaje supervisado**: En este tipo de aprendizaje se le proporciona a la RNA una serie de ejemplos consistentes en unos patrones de entrada, junto con la salida que debería proporcionar la red. El proceso de entrenamiento consiste en el ajuste de los pesos para que la salida de la red sea lo más parecida posible a la salida deseada. Es por ello que en cada iteración se use alguna función que obtenga el error o el grado de acierto que está cometiendo la red.

  Dentro de este tipo de aprendizaje el modelo más utilizado y en el que se centra este trabajo es el denominado "***backpropagation***" [4] y [17], donde por lo general la función de coste utilizada es el error cuadrático medio.

  Realizando un símil con la educación de las personas, al igual que un maestro tiene su metodología y herramientas para el aprendizaje de sus alumnos, una red será mejor o peor no solo dependiendo de su arquitectura, sino de la información para el aprendizaje que se le proporcione.

- **Aprendizaje no supervisado o autoorganizado**: En este tipo de aprendizaje se presenta a la red una serie de ejemplos pero no se presenta la respuesta deseada. Lo que hace la RNA es reconocer regularidades en el conjunto de entradas, es decir, estimar una función densidad de probabilidad p(**x**) que describe la distribución de patrones x en el espacio de entrada R^n .

- **Aprendizaje Híbrido**: Es una mezcla de los anteriores. Unas capas de la red realizan un aprendizaje supervisado y otras capas un aprendizaje no supervisado. Este tipo de entrenamiento es el que tienen redes como las RBF (*Recurrent Radial Basis Function*).

- **Aprendizaje reforzado (*reinforcement learning*)**: Es un aprendizaje con características del supervisado y del autoorganizado. No se proporciona una salida deseada, pero sí que se indica a la red en cierta medida el error que comete, aunque es un error global.

3.2.3 Entrenamiento de ANN, Algoritmo "Back-Propagation"

El entrenamiento de una RNA, consiste en la determinación de los pesos sinápticos y umbrales o "bias" del conjunto de neuronas de la red. Para ello existen multitud de métodos, pero para el caso de redes que tienen funciones de activación diferenciable, existe un método potente y computacionalmente eficiente denominado "***back-propagation***" (BP), o de retropropagación [4] y [15].

En el método BP existe una función de error a minimizar, como puede ser el *error cuadrático medio* (MSE) entre las salidas obtenidas y las deseadas, y el algoritmo actúa modificando los pesos y los umbrales en cada bloque iterativo "*epoch*". Dicho algoritmo fue formalizado en el trabajo presentado por Rumelhart [9] en 1985.

Se asume que la propagación de la señal tiene dos fases, la fase *forward*, que es la fase de funcionamiento normal, en la que las señales se propagan desde las entradas a las salidas, y la fase *backward*, en la cual una vez obtenidas las estimaciones de salida se realiza la corrección de los pesos y umbrales en sentido contrario, es decir, desde la salida hacia la entrada.

El algoritmo BP se puede resumir en los siguientes pasos:

1. **Inicialización:** Partiendo de la hipótesis de que no se dispone de información inicial, se toman pesos y umbrales de una distribución uniforme de media cero y cuya varianza hace que los productos entre pesos y entradas de la neurona estén situados ente la zona lineal y la de saturación de las funciones de activación sigmoidales.

2. **Presentación de las muestras de entrenamiento:** Se presentan a la entrada de la red las muestras de entrenamiento correspondientes a un *epoch*.

3. **Fase forward:** Dada una muestra correspondiente al *epoch* n como (x(n), d(n)), siendo el vector x(n) presentado a la entrada de la RNA. Se realiza la fase forward, capa por capa, según

la ecuación que calcula el denominado "campo local inducido", $v_j^{(l)}(n)$, de la neurona j de la capa l:

$$v_j^{(l)}(n) = \sum_{i=0}^{m_0} w_{ji}^{(l)}(n)\, y_i^{(l-1)}(n) \qquad (22)$$

Donde m_0 es el número de neuronas en la capa anterior, $y_i^{(l-1)}$ es la salida de la neurona i de la capa l-1 en el *epoch* n y $w_{ji}^{(l)}$ es el peso de la conexión entre la neurona j de la capa l procedente de neurona i de la capa l-1.

De este modo se puede determinar el error como:

$$e_j(n) = d_j(n) - p_j(n) \qquad (23)$$

Siendo $d_j(n)$ el j-ésimo elemento de la salida deseada **d**(n), y $p_j(n)$ es el valor predicho a la salida de la neurona j de la capa de salida l=L.

4. **Fase backward:** Se calculan los gradientes locales de la red por medio del método de gradiente descendente. Los pesos se ajustan mediante la siguiente ecuación:

$$w_{ji}^{(l)}(n+1) = w_{ji}^{(l)}(n) + \alpha\left[w_{ji}^{(l)}(n-1)\right] + \eta\delta_j^l(n) y_i^{(l-1)} \qquad (24)$$

Siendo α la constante de momento, η la tasa de aprendizaje y $\delta_j^l(n)$ el gradiente local [4] y [15]. La tasa de aprendizaje está relacionada con la trayectoria que toma el algoritmo BP en el espacio de los pesos por el método del gradiente. Cuando η es pequeño la variación de los pesos ente iteración será pequeña, haciendo más "suave" la trayectoria, pero también haciendo más lento el entrenamiento. Por el contrario para η grandes, la variación de los pesos será mayor, acelerando el entrenamiento. Sin embargo, si η es demasiado grande, se corre el riesgo de que la red oscile y sea inestable.
Existen métodos que mantienen la tasa de aprendizaje constante y otros que la hacen adaptativa según la evolución de error de un *epoch* a otro.

Por otro lado, el término de momento α ayuda a evitar las oscilaciones producidas por las variaciones de los pesos en la función de error, que no decrece de manera monótona.

5. **Iteración:** se realizan los cálculos de los puntos 3 y 4 previos presentando las muestras de entrenamiento de cada *epoch* a la entrada de la red hasta que se cumple el criterio de parada. Este criterio de parada es esencial ya que por lo general, no converge [15]. Estos criterios pueden ser que la norma euclídea del vector gradiente se ha reducido hasta alcanzar un determinado umbral, la variación del error promedio por *epoch* es lo suficientemente pequeña o la generalización de la red ha alcanzado su máxima cota "*early-stopping*".

3.2.3.1 Optimizadores basados en Gradientes

La optimización tiene como objetivo minimizar o maximizar una determinada función ***f(x)***, modificando el ***x.*** En realidad, se habla de minimización, ya que la maximización es equivalente a la minimización de -*f*(***x***).

La función a minimizar (maximizar) se conoce como función **objetivo**. Cuando se está minimizando, se le denomina también ***cost function, loss function*** o ***error function***. El valor que consigue la minimización o maximización se identifica con el símbolo *, $\boldsymbol{x}^* = arg\,min\ f(\boldsymbol{x})$.

Existen diferentes clasificaciones en los problemas de optimización dependiendo del criterio utilizado, algunas de estas clasificaciones son:

- Existencia de restricciones (con/sin restricciones)
- Número de objetivos (mono/multi objetivo)
- Naturaleza determinista de las variables de decisión (problemas deterministas/estocásticos)
- Complejidad computacional (problemas clase P, clase NP)

Resulta bien conocido que la derivada de una función en un punto ***x*** proporciona la pendiente de la función en dicho punto. La derivada permite escalar un pequeño cambio en la entrada para obtener el correspondiente cambio en la salida: $f(x+\varepsilon) \approx f(x)+\varepsilon f'(x)$. La derivada es, por tanto, útil para minimizar una función porque nos indica cómo cambiar *x* para conseguir una pequeña mejora en la salida *y*. Por ejemplo, se sabe que $f(x-\varepsilon \mathrm{sign}(f'(x)))$ es menor que *f*(***x***) para un ε lo suficientemente pequeño. Se puede entonces disminuir *f*(***x***) moviendo *x* en pequeños pasos con el signo opuesto de la derivada. Esto se conoce como ***gradient descent*** *[4]*.

A menudo es necesario minimizar funciones que tienen múltiples (*n*) entradas, con una única salida (escalar). Con múltiples entradas, se debe utilizar el **gradiente** que mide cómo cambia *f* según aumenta la variable x_i en el punto ***x***.

La **derivada direccional** en la dirección ***u*** (vector unitario) es la pendiente de la función *f* en la dirección *u*. En otras palabras, la derivada direccional es la derivada de la función $f(\boldsymbol{x}+\alpha\boldsymbol{u})$ con respecto a α, evaluada en $\alpha = 0$. Usando la regla de la cadena, se puede ver que $\frac{\partial}{\partial\alpha} f(\boldsymbol{x}+\alpha\boldsymbol{u})$ evalúa $\boldsymbol{u}^T \nabla_{\boldsymbol{x}} \frac{\partial}{\partial\alpha} f(\boldsymbol{x})$ cuando α = 0.

Para minimizar *f*, es deseable encontrar la dirección en la que *f* disminuye de forma más rápida, lo que se puede hacer usando la derivada direccional:

$$\min_{\boldsymbol{u},\boldsymbol{u}^T\boldsymbol{u}=1} \boldsymbol{u}^T \nabla_{\boldsymbol{x}} f(\boldsymbol{x}) = \min_{\boldsymbol{u},\boldsymbol{u}^T\boldsymbol{u}=1} \|\boldsymbol{u}\|_2 \|\nabla_{\boldsymbol{x}} f(\boldsymbol{x})\|_2 \cos\theta \qquad (25)$$

donde θ es el ángulo entre ***u*** y el gradiente. Sustituyendo $\|\boldsymbol{u}\|_2 = 1$e ignorando factores que no dependen de ***u***, simplifica a minimizar $min_{\boldsymbol{u}} \cos\theta$. Esto se minimiza cuando ***u*** apunta en la dirección opuesta del gradiente. En otras palabras, el gradiente apunta directamente hacia arriba. Se puede decrecer *f* moviéndose en la dirección opuesta del gradiente, lo que se conoce como método ***steepest***

descent, descent gradient **(gradiente descendiente)**, que propone un nuevo punto, donde ε es la **razón de aprendizaje**, que es un valor escalar que determina la dimensión del paso.

$$\boldsymbol{x}' = \boldsymbol{x} - \varepsilon \nabla_x f(\boldsymbol{x}) \qquad (26)$$

3.2.3.2 Stochastic Gradient Descent (SGD)

En el ámbito del aprendizaje automático, el problema consiste en minimizar una función objetivo que tiene la forma indicada en la ecuación siguiente, donde el parámetro *w* que minimiza *J*(*w*) debe estimarse.

$$J(w) = \frac{1}{n} \sum_{i=1}^{n} J_i(w) \qquad (27)$$

J_i se asocia con la *i*-ésima observación en el conjunto de datos utilizados para el entrenamiento (ajuste). El término estocástico proviene del hecho relativo a la selección de las muestras (observaciones) para el ajuste de forma aleatoria. $J_i(w)$ es el valor de la ***loss function*** en el i-ésimo ejemplo y J(w) es el **riesgo empírico**.

El gradiente descendiente se usa para minimizar esta función de forma iterativa, con iteraciones *t*,

$$w(t+1) = w(t) - \varepsilon \frac{1}{n} \sum_{i=1}^{n} \nabla J_i(w) \qquad (28)$$

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

3.3 Redes Neuronales convolucionales (CNN)

Uno de los tipos de redes con capacidad para realizar lo que se conoce como aprendizaje profundo son la ya mencionadas CNN. Las redes neuronales convolucionales son redes multicapa que se componen de capas convolucionales y de sub-muestreo alternadas, presentando al final capas totalmente conectadas [24]. La figura 27 muestra un esquema estándar de una CNN:

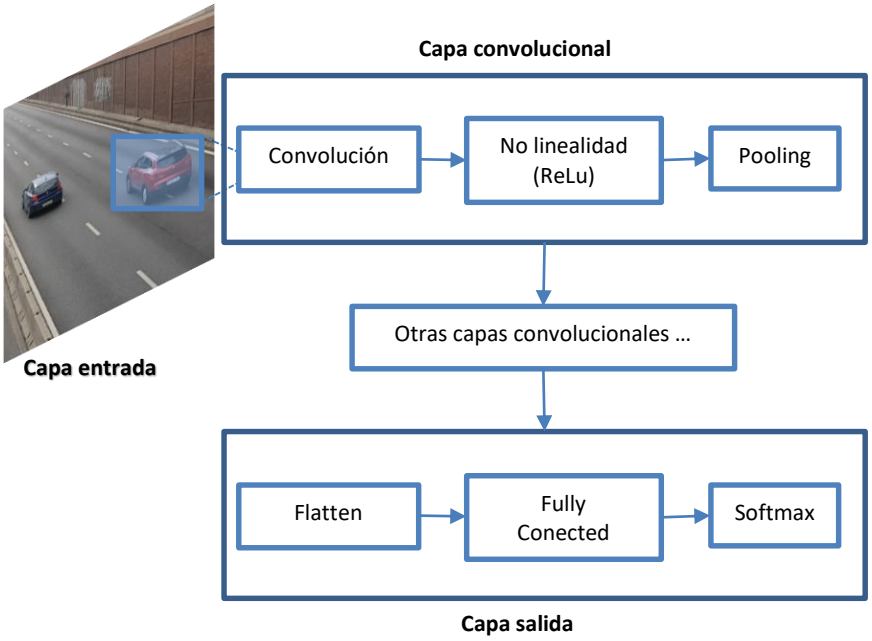


Figura 27: Arquitectura estándar de una CNN

Aunque en la figura 27 se muestra una arquitectura estándar, a continuación se describen las capas que se utilizan en la red AlexNet, ResNet50 y Vehicle, que son las arquitecturas que se utilizan en este trabajo; Las cuales son:

- **Capas convolucionales**
- **Capas de activación**
- **Capas de *Pooling***
- **Capas *flatten***
- **Capas totalmente conectadas**
- **Capas *softmax* o capa final**
- **Otras capas de interés**

3.3.1 Capas Convolucionales

Sin duda este tipo de capas son las de mayor relevancia en las CNN, de ahí el nombre de éstas. Por ello analizaremos en profundidad tanto el concepto de convolución como el proceso que se realiza en estas capas.

En líneas generales, la operación de convolución involucra dos funciones con valores reales como argumento. Para clarificar el concepto supóngase que se tiene una fuente de luz variable cuya intensidad se recibe mediante un sensor, que proporciona una salida en una determinada posición x en el tiempo t, esto es, $x(t)$. Tanto x como t son valores reales, de forma que, debido a la variabilidad de la fuente mencionada, se pueden obtener diferentes lecturas en diferentes instantes de tiempo. Si

a esto se añade que la captura de la señal por el sensor puede estar contaminada con un cierto ruido, para obtener una señal más limpia, lo acertado es realizar un promediado de la salida con varias medidas. Si además tenemos en cuenta que las medidas más recientes son más relevantes que las alejadas en el tiempo, el promediado puede ponderarse concediendo más relevancia a las recientes, lo cual puede hacerse mediante la función de promediado *w*(*a*), donde *a* representa el alejamiento de la medida en el tiempo. Si se realiza esta operación de promediado ponderado en cada instante de tiempo *t*, se obtiene una función nueva promediada como sigue.

$$s(t) = \int x(a)w(t-a)da \tag{29}$$

Esta operación se denomina **convolución**, que se denota como sigue,

$$s(t) = \int (x * w)(t) \tag{30}$$

Resulta necesario matizar que *w* necesita ser una función de densidad de probabilidad, para garantizar que la salida esté promediada. También, es necesario que *w* sea cero para todos los argumentos negativos, con el fin de evitar la toma de valores en el futuro, algo que no es materialmente posible.

En el ámbito de las CNN, el primer argumento *x* para la convolución se conoce como *entrada* (***input***) y el segundo *w*, como *núcleo* (***kernel***) de convolución. La salida *s* se define generalmente como *mapa de características* (***feature map***). Desde el punto de vista computacional, y siguiendo con el ejemplo de la fuente de luz, las medidas son discretas, de forma que se realizan medidas en intervalos de tiempo determinados, por ejemplo, cada segundo. De esta forma, el tiempo *t* puede tomar sólo valores enteros, de forma que considerando que *x* y *w* se definen sólo en el tiempo *t* con valor entero, se define la convolución discreta como sigue.

$$s(t) = (x * w)(t) = \sum_{a=-\infty}^{\infty} x(a)w(t-a) \tag{31}$$

En aprendizaje máquina la entrada es un vector o matriz multidimensional de datos, y el núcleo es, generalmente, un vector o matriz multidimensional de parámetros, que se ajustan, mediante el proceso de aprendizaje. Dichas estructuras multidimensionales se conocen como tensores. Puesto que cada elemento de la entrada y del núcleo deben almacenarse separadamente, los valores que no pertenecen a esos elementos deben considerarse nulos, lo que traducido en la práctica resulta en una suma finita de valores en un número finito de elementos. Por otra parte, las convoluciones se realizan sobre más de un eje a la vez, tal es el caso de las imágenes I, cuando constituyen las entradas, que se trata de estructuras bi-dimensionales a procesar con un núcleo de entrada bi-dimensional K.

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(m,n)K(i-m, j-n) \tag{32}$$

La convolución es conmutativa, lo que significa que la expresión anterior, puede escribirse de forma equivalente como sigue, que resulta en una menor variación en los valores de m y n, lo que resulta en una forma más eficiente desde el punto de vista de la implementación.

$$S(i,j) = (K * I)(i,j) = \sum_m \sum_n I(i-m, j-n) K(m,n) \quad (33)$$

La propiedad conmutativa de la convolución se interpreta como una forma de reflejar el núcleo respecto de la entrada, de forma que a medida que el índice *m* aumenta, el índice en la entrada crece a la vez que disminuye en el núcleo. En cualquier caso, la propiedad conmutativa no es relevante desde el punto de vista de las CNN. Sin embargo, lo que sí resulta relevante en este contexto es lo que se conoce como correlación cruzada, como sigue.

$$S(i,j) = (K * I)(i,j) = \sum_m \sum_n I(i+m, j+n) K(m,n) \quad (34)$$

A veces correlación cruzada y convolución son términos que se confunden en la terminología especializada, usándose indistintamente. En el contexto de aprendizaje máquina, el algoritmo de aprendizaje estima los valores apropiados del núcleo en cualquiera de los dos casos. La figura 28 muestra un ejemplo de convolución con el núcleo *K*, sin reflexión del núcleo, aplicado sobre un tensor 2-D, que bien puede representar una imagen, ***I***. La convolución se representa con solapamiento total del núcleo y obteniendo una imagen resultante, cuyos bordes externos se quedan sin valores, obteniendo una imagen de menor dimensión que la original, en el caso de la figura se pasa de tener dimensión 6×7 a 4×5 No obstante, si se desea obtener una imagen de la misma dimensión lo que hay que hacer es ampliar la imagen original en dos filas (arriba y abajo) y dos columnas (izquierda y derecha) con ceros, procesándola con el núcleo solapado. Esta operación es lo que se conoce como ***zero-padding*.**

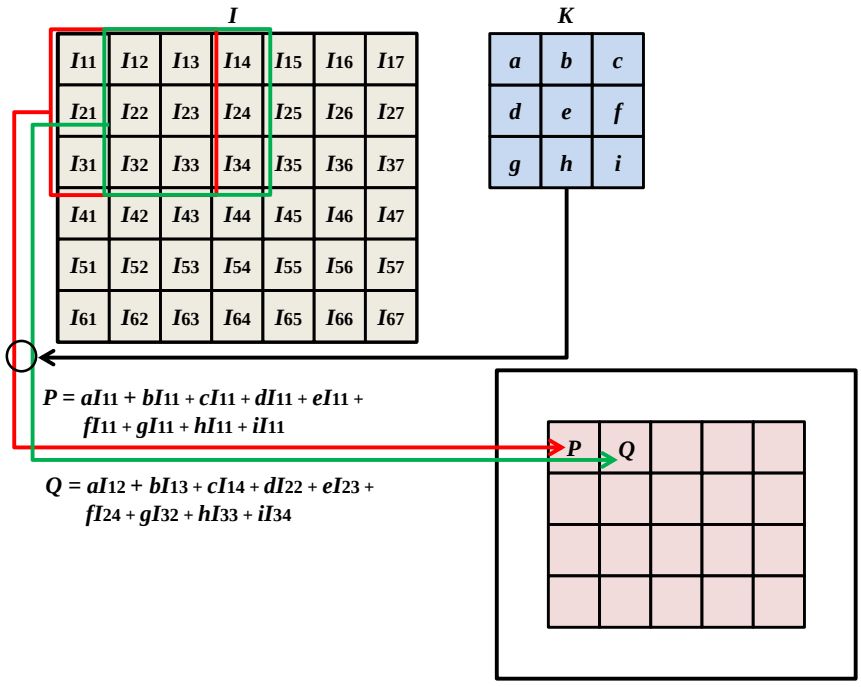


Figura 28: Ejemplo de convolución de tensor I bajo núcleo K con solapamiento total

Como se ha indicado previamente, la convolución de la figura 28 es de tipo 2-D. También podría ser de tipo 3-D, de forma que el núcleo sería un cuboide que se desplaza a través de las dimensiones alto, ancho y alto del mapa de características. Por ejemplo, si se tiene una imagen RGB y por tanto con tres

canales para la capa de entrada se utilizan núcleos de tipo cuboide y por tanto con tres dimensiones. Esto puede generalizarse a convoluciones de tipo N-D cuando el número de dimensiones es N.

En la figura anterior, los desplazamientos del núcleo son de una posición a la siguiente, pero en lugar de desplazarse una posición de izquierda a derecha y de arriba abajo, el desplazamiento podría ser de más de una unidad, es lo que se conoce como ***stride***.

Así pues, la colección de núcleos que definen una convolución discreta tiene una forma que se corresponde a alguna permutación del tipo siguiente (n, m, k_1,…, k_N), donde:

- n ≡ número de mapas de características de salida,
- m ≡ número de mapas de características de entrada,
- k_j ≡ dimensión del núcleo a lo largo del eje j.

La dimensión o_j de una capa de convolución a lo largo del eje j tiene las siguientes propiedades,

- i_j ≡ dimensión de entrada a lo largo del eje j,
- k_j ≡ dimensión del núcleo a lo largo del eje j.
- s_j ≡ distancia entre dos posiciones consecutivas del núcleo (*stride*) a lo largo del eje j.
- p_j ≡ número de ceros concatenados al comienzo y al final del eje j (zero-padding).

La figura 29 muestra un ejemplo ilustrativo sobre el proceso de convolución bi-dimensional (N = 2), con $i_1 = i_2 = 5$ como tamaño 5×5 de la imagen de entrada (I) y dimensión 3×3 ($k_1 = k_2 = 3$) del núcleo de convolución (K), con desplazamiento (*stride*) de uno en ambas direcciones ($s_1 = s_2 = 1$) y cero *padding* ($p_1 = p_2 = 0$). Generalmente, suele ser habitual que los valores de i, k, s y p en todos los ejes tengan el mismo valor

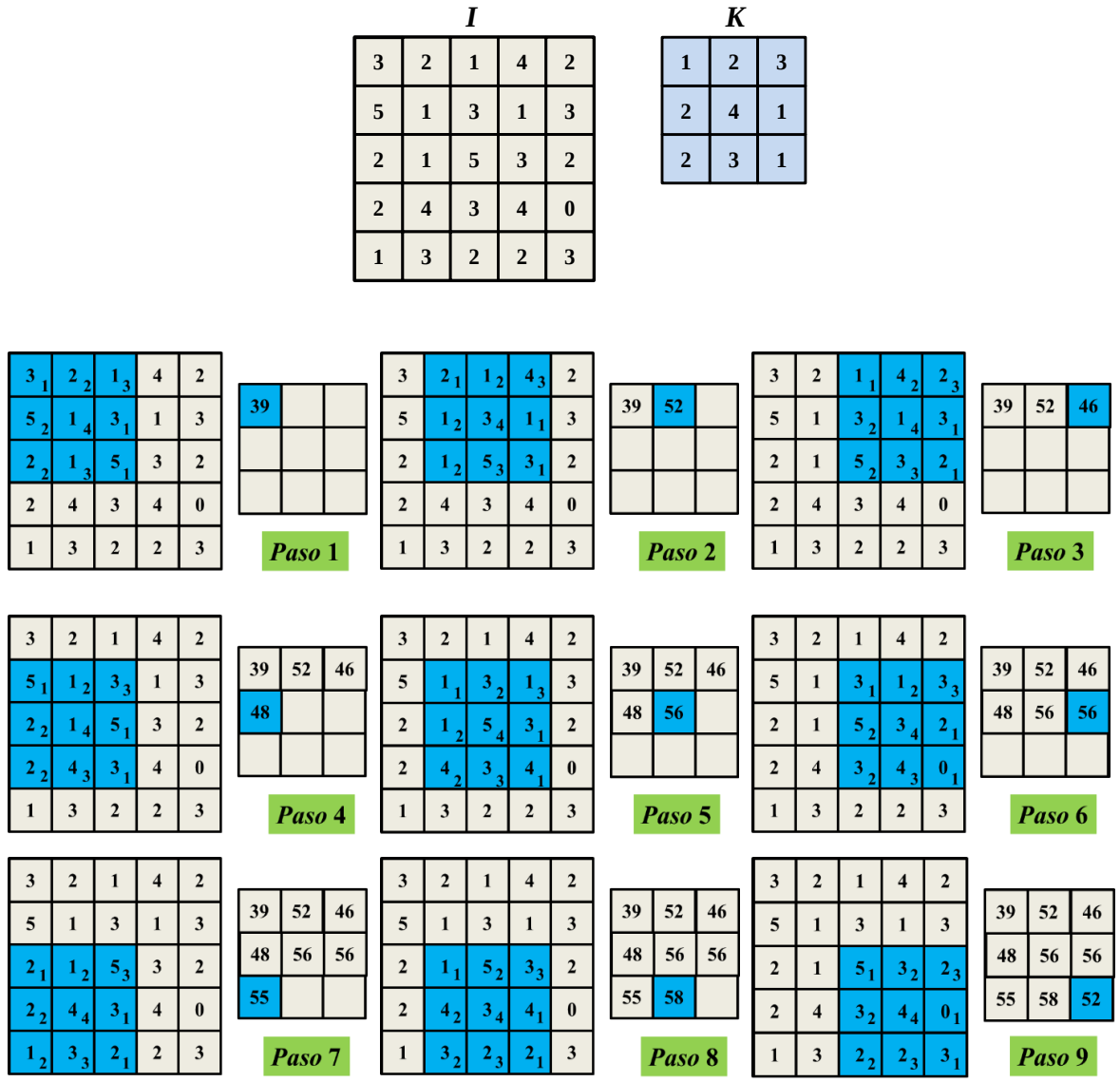


Figura 29: Ejemplo de una convolución discreta 1

con N = 2, $i_1 = i_2 = \mathbf{5}$, $k_1 = k_2 = \mathbf{3}$, $s_1 = s_2 = \mathbf{1}$, $p_1 = p_2 = \mathbf{0}$

La figura 30 muestra un ejemplo ilustrativo sobre el proceso de convolución bi-dimensional (N = 2), con $i_1 = i_2 = 5$ como tamaño 5×5 de la misma imagen de entrada (I) y dimensión 3×3 ($k_1 = k_2 = 3$) del mismo núcleo de convolución (K), con desplazamiento (*stride*) de dos en ambas direcciones ($s_1 = s_2 = 2$) y una extensión de *padding* ($p_1 = p_2 = 1$).

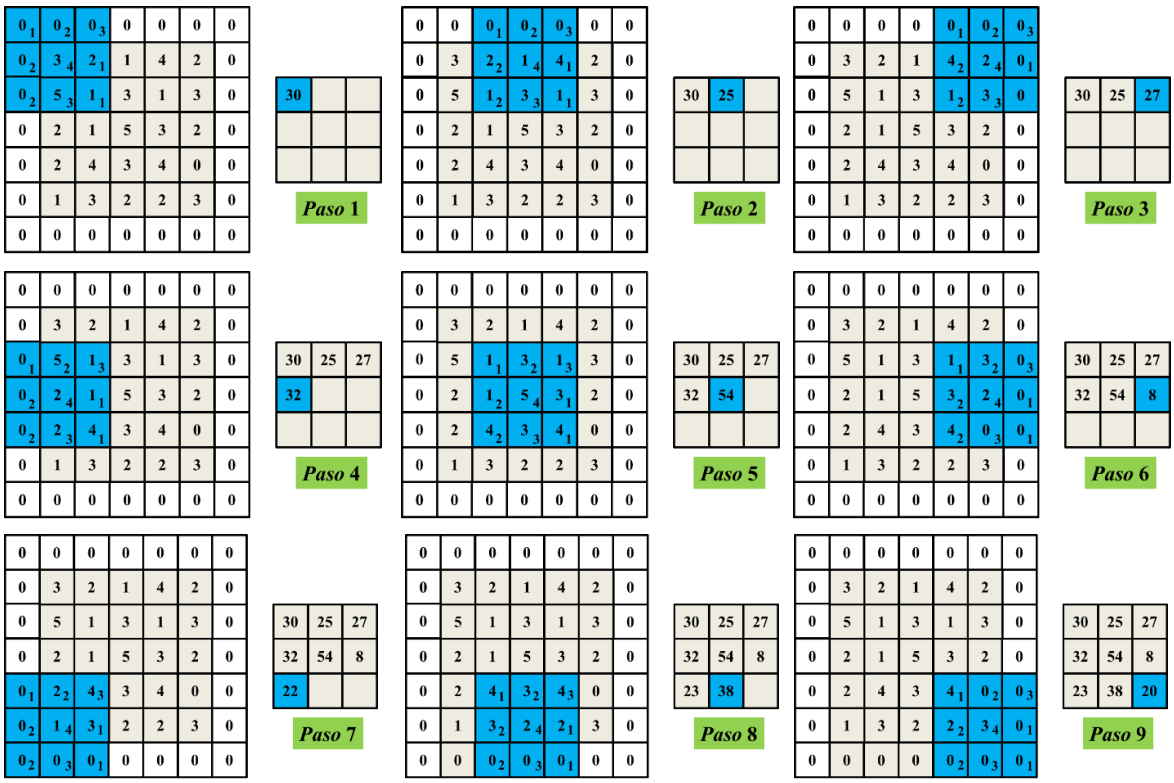


Figura 30: Ejemplo de una convolución discreta 2

Con N = 2, $i_1 = i_2 = 5$, $k_1 = k_2 = 3$, $s_1 = s_2 = 2$, $p_1 = p_2 = 1$

Obsérvese cómo la operación de desplazamiento del núcleo (*stride*) es una forma de submuestreo cuando el desplazamiento es superior a uno. Por otra parte, como puede deducirse de la figura 31, el desplazamiento del núcleo en saltos de dos puede interpretarse como la convolución con saltos de uno, pero conservando sólo los resultados de las salidas impares.

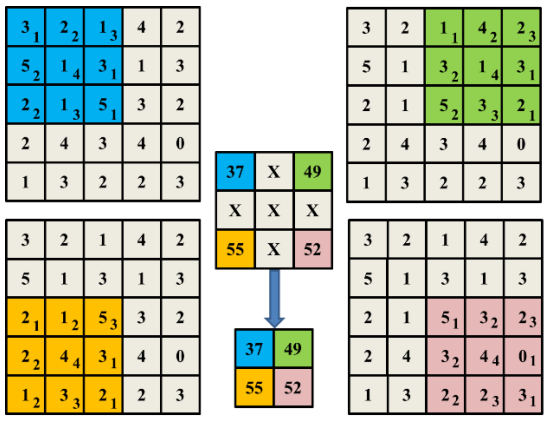


Figura 31: Forma alternativa de interpretar los desplazamientos del núcleo

En cada localización, se obtiene el producto entre cada elemento del núcleo y el elemento de entrada sobre el que se solapa y los resultados se suman para obtener la salida en la posición actual. El procedimiento se repite utilizando diferentes núcleos para formar tantos mapas de características

como se deseen. En la figura 32 se representa una operación de convolución a partir de tres mapas de características de entrada para obtener cuatro mapas de características de salida, utilizando una colección de núcleos w de dimensión 4×3×3×3. Siguiendo la línea superior, el mapa de características 1 se convoluciona con el núcleo $w_{1,1}$, el mapa de características 2, con el $w_{1,2}$ y el mapa de características 3 con el núcleo $w_{1,3}$ de forma que los resultados se suman elemento a elemento para formar el primer mapa de características de salida. El mismo procedimiento se repite para las dos líneas intermedias y la inferior de la figura para formar los mapas de características de salida 2, 3 y 4 respectivamente, de forma que los cuatro mapas de salida se agrupan de forma conjunta.

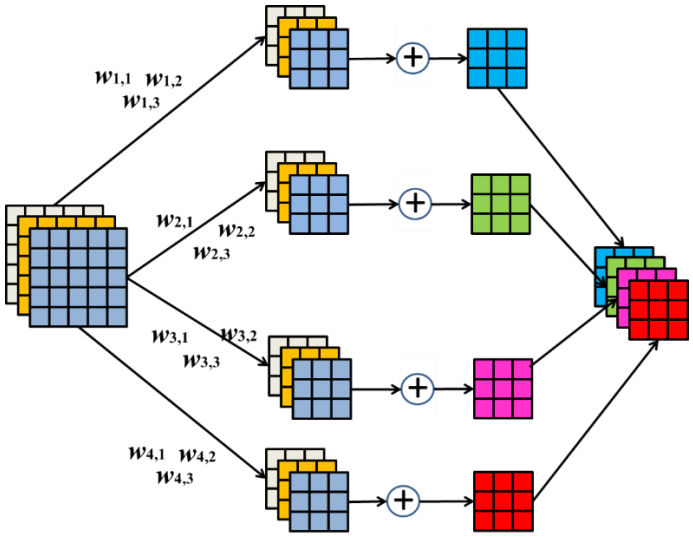


Figura 32: Forma alternativa de interpretar los desplazamientos del núcleo

Para realizar la convolución existen diferentes formas de desplazamiento del núcleo a saber: con y sin añadido de ceros en los extremos (*padding*), con distintos saltos de desplazamiento (*stride*). De este modo se identifican 6 casos relevantes de convolución:

- **Sin cero *padding* y una unidad de desplazamiento**
- **Cero *padding* y una unidad de desplazamiento**
- **"Half" (*same*) *padding***
- ***Full padding***
- **Sin cero *padding* y desplazamiento (*stride*) superior a la unidad**
- **Cero *padding* y desplazamiento superior a la unidad**

3.3.2 Capas de *pooling*

Tras la convolución aparece uno de los principales problemas de la minería de datos que es la dimensionalidad, para ello la capa de *Pooling*, o reducción, disminuye la cantidad de parámetros, al quedarse con las características más comunes. Esto se lleva a cabo extrayendo características esenciales, algunas de las cuales son el promedio o el máximo de una región. Por ejemplo, en la figura 33 se realiza una operación de máximo para obtener el resultado mostrado en la parte derecha [24].

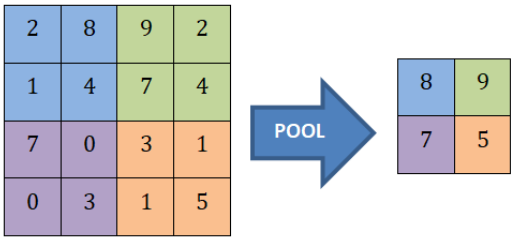


Figura 33: Operación de la capa *pooling*

Como se puede comprobar en la figura 35, el mapa de características del que partíamos, con dimensiones 32x32x10 acaba con dimensiones 16x16x10. Tras haber aplicado la operación de *pooling*, la profundidad del mapa de características no se ve afectada.

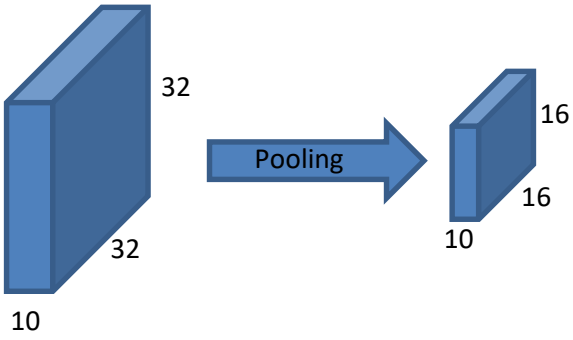


Figura 34: Operación de la capa *Pooling*

Desde el punto de vista de la convolución [4], esta operación puede interpretarse bajo la suposición de que alguna función se repite sobre subconjuntos (ventanas) de la entrada. Por tanto, teniendo en cuenta las relaciones previas, el *pooling* es una operación sin *zero padding*, de forma que la relación que describe el caso general y que sirve para cualquier tipo de *pooling*, es como sigue:

$$o = \left[\frac{i-k}{s} \right] + 1 \qquad (35)$$

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

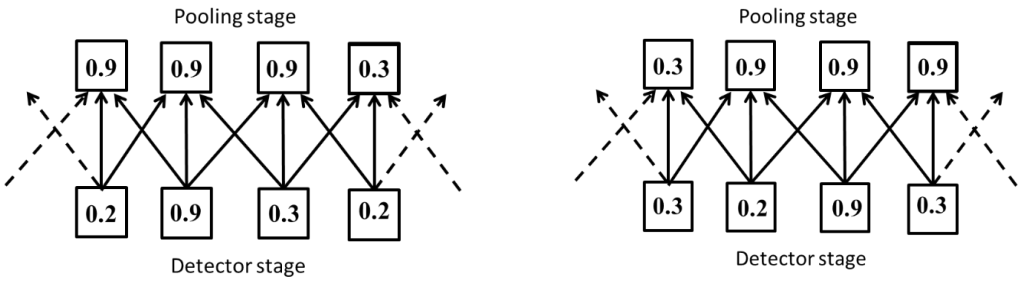


Figura 35: Ejemplo de Invarianza en *max pooling*

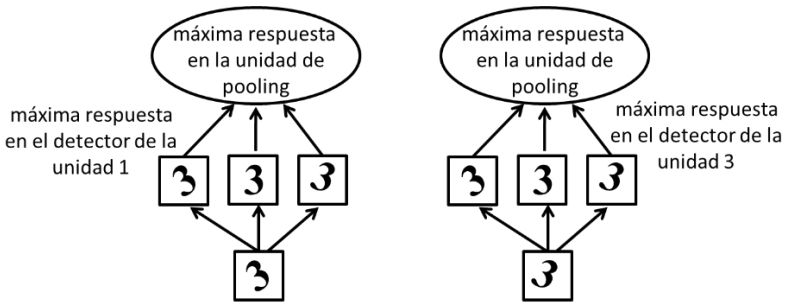


Figura 36: Ejemplo de invarianzas aprendidas

3.3.3 Capas de aplanamiento (*Flatten*)

En esta fase se parte de los mapas de características reducidas, procediéndose a "aplanar" los datos matriciales, es decir reordenamos los datos procedentes de los mapas de características bidimensionales con varias capas en un vector de n filas y 1 columna, donde n es el producto de las dimensiones de los mapas de características por el número de capas, y de este modo poder introducirlos a la red neuronal artificial.

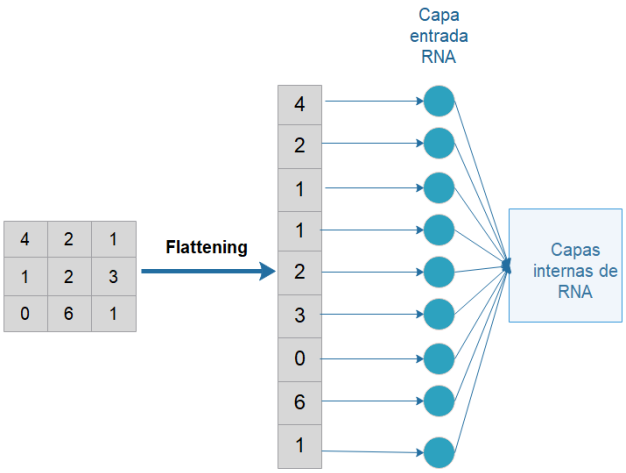


Figura 37: Proceso de aplanamiento o *flattening*

3.3.4 Capas totalmente conectadas (*full Conect*)

Tras las capas de convolución, *pooling* y *flatten*, se añaden dos capas totalmente conectadas que como su nombre indica se encarga de conectar todos los nodos (neuronas) entre sí. Esta arquitectura aparece reflejada en la figura 23, conde las neuronas ocultas conforman las capas totalmente conectadas.

3.3.5 Capas de salida o *Softmax*

Este tipo de capa se introduce en problemas de clasificación con el objetivo de obtener coeficientes de probabilidad de las diferentes clases a identificar, por ello en problemas de clasificación *Softmax* se implementa a través de una capa de red neuronal justo antes de la capa de resultado. La capa de *softmax* debe tener la misma cantidad de nodos que la capa de clases.

En definitiva, esta capa nos proporciona las probabilidades condicionales de que una determinada entrada pertenezca a cada una de las M-clases.

En la figura 38 se muestra una capa *SoftMax* tras la salida de una red de retropropagación con M capas, vector de entra x y dimensión d, donde se representa las probabilidades asociadas a cada una de las clases. El proceso de etiquetado se hace seleccionando la clase con mayor probabilidad $\sigma(j)$.

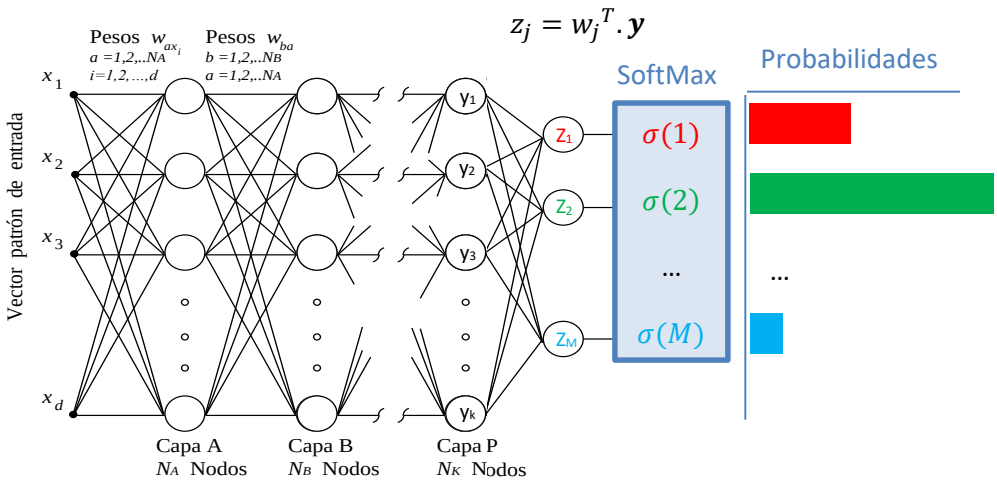


Figura 38: Capa *SoftMax*

$$z_j = w_j{}^T.\boldsymbol{y};\ \sigma(j) = \frac{e^{z_j}}{\sum_{i=1}^{M} e^{z_i}} \qquad (36)$$

La normalización de *softmax* es una forma de reducir la influencia de valores extremos o atípicos en los datos sin eliminar puntos de datos del conjunto.

3.3.6 Otras capas de interés

Dentro de las estructuras de las ANN podemos encontrar otro tipo de capas de alto interés, en este apartado se mencionan algunas de las capas relevantes que también se pueden utilizar con las CNN.

1. **Convoluciones dilatadas**
2. **Capas *Dropout***
3. **Normalización**

3.3.6.1 Convoluciones dilatadas

La motivación de esta capa surge de aumentar el campo receptivo de las unidades de salida sin tener que incrementar la dimensión del núcleo de convolución, lo que resulta muy efectivo si se apilan varias convoluciones dilatadas como se puede ver en el modelo *WaveNet* [25], que implementa un modelo autoregresivo para una señal de audio que utiliza convoluciones dilatadas para condicionar nuevos tramos de audio en un amplio tramo de contexto pasado.

Resultados de este artículo publicado en 2016, muestra como este modelo superaba a sistemas TTS (*Text To Speech*) como el de Google. La capacidad de esta red profunda para modelar formas de onda permite modelar cualquier tipo de audio, incluida la música. En 2017 Google incorporó este sistema para generar voces de ayudante de sistemas y finalmente en mayo de 2018 se lanzó desde la misma empresa la asistencia por nuevas voces generadas con este sistema.

Saliendo de aspectos curiosos, las capas convolucionales dilatadas no requieren de un mayor esfuerzo computacional y se puede representar gráficamente del siguiente modo:

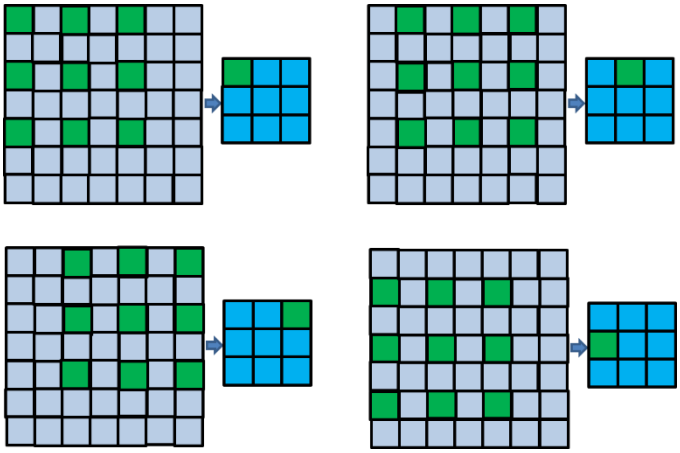


Figura 39: Convolución dilatada para $i = 7$, $k = 3$ y $d = 2$.

Esta convolución se puede formular del siguiente modo:

$$o = \left[\frac{i + 2p - k - (k-1)(d-1)}{s} \right] + 1 \qquad (37)$$

Donde la razón de dilatación viene definida por el hiperparámetro "d", de forma que si d es la unidad nos encontramos en el caso de una convolución regular.

Para comprender el proceso matemático partiendo de la relación en una convolución regular, se ha considerado un nuevo núcleo $\hat{k} = k + (k-1)(d-1)$.

3.3.6.2 Capas Dropout

Uno de los problemas más comunes en la fase de entrenamiento y validación de las redes neuronales, es el denominado "*overfitting*", que consiste en un ajuste elevado de los pesos de un gran número de neuronas. Este fenómeno produce una dispersión entre las muestras de validación y las de entrenamiento. Se puede representar tal y como aparece en la figura 40:

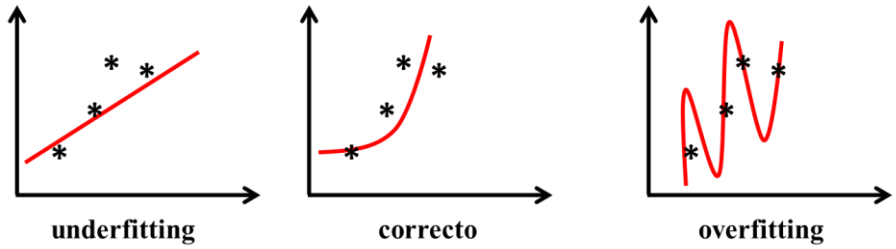


Figura 40: Tres tipos de ajuste en entrenamiento

En la imagen anterior muestra un ejemplo de "*underfitting*" y "*overfitting*" frente a un posible modelo correcto. El dilema consiste en determinar qué modelo es mejor, ¿uno polinómico que pasa por todos los puntos con un gran "esfuerzo"? o ¿un modelo de bajo grado que en el momento de la decisión esta no es adecuada? Como se ve en la imagen anterior en realidad un posible modelo correcto será aquel que no presenta ni el comportamiento de uno (*underfitting*) ni del otro (*overfitting*) sino una relación "suave" entre ellos.

Para evitar el *overfitting*, tendríamos que aplicar una regularización que consiste en modificar la función objetivo J, añadiéndole términos que penalicen pesos con valores elevados. Si añadimos un término $\lambda f(\theta)$ de forma que $f(\theta)$ crece a medida que el argumento θ también crece y λ es el coeficiente de regularización. El valor de este parámetro determina la protección contra el *overfitting*, de forma que si es cero, no se realiza ninguna acción protectora, por el contrario, si es demasiado grande, el modelo dará prioridad a mantener θ lo más pequeño posible, al tratar de encontrar los valores de los parámetros que que mejor se ajustan al conjunto de datos de entrenamiento.

El tipo más común de regularización es la L_2. Se puede implementar aumentando la función de error con la magnitud cuadrada de todos los pesos en la red neuronal. En otras palabras, para cada peso *w* en la red neuronal, se agrega $1/2\ \lambda w^2$ a la función de error. La regularización de L_2 tiene la

interpretación intuitiva de penalizar fuertemente los pesos con valores altos y preferir los pesos más discretos. Esto tiene la atractiva característica de "alentar a la red a usar todas sus entradas un poco en lugar de usar muchas de sus entradas mucho". De particular interés es que durante la actualización de los pesos basada en gradient descendente, el uso de la regularización de L_2 significa que cada peso tiende a cero. Debido a este fenómeno, la regularización de L_2 también se conoce comúnmente como disminución de peso.

No obstante, en el ámbito de las redes neuronales, para paliar el efecto del *overfitting* se propone anular (*dropout*) determinado tipo de neuronas; incluidas sus salidas para evitar el problema mecionado del sobre ajuste durante el proceso de entrenamiento, lo cual previene el hecho de *overfitting* de los pesos [26]. La selección de las unidades a anular se realiza de forma aleatoria. Así, la red resultante es la que mantiene las unidades que sobreviven al *dropout*. No obstante, en lugar de eliminar neuronas, también cabe la posibilidad de mantenerlas asignándoles un hipeparámetro de forma aleatoria que es en realidad una probabilidad *p* de supervivencia durante la fase de entrenamiento, que se utiliza para disminuir la influencia del peso en la fase de test. En la figura 41 se muestran nodos cancelados (en rojo) y con el peso de activación $w_{(5,6)}$ atenuado por la probabilidad *p* durante la fase de test.

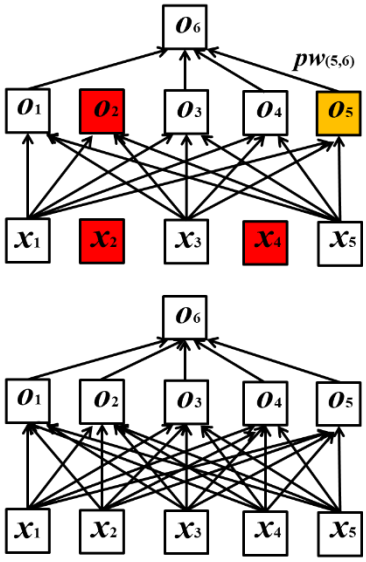


Figura 41: *Dopout*

En cualquier caso, el *dropout* puede aplicarse tras la salidad de una determinada capa como por ejemplo, tras un *pooling* como ocurre en el ejemplo gráfico mostrado a continuación:

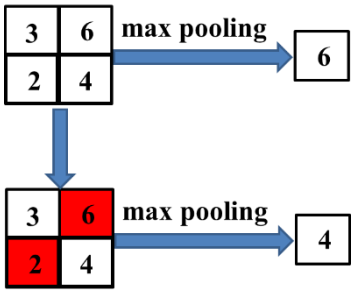


Figura 42: *Dropout y max pooling*

3.3.6.3 Normalización

Como se ha mencionado anteriormente con las funciones de activación, ReLU tiene la deseable propiedad de que no requiere normalización para prevenir la saturación [19]. Si al menos algunos ejemplos de entrenamiento producen una entrada positiva a una unidad ReLU, se producirá aprendizaje en esa neurona. No obstante, todavía se puede aplicar una normalización local como ayuda a la generalización. Si se denota por $a^i_{x,y}$ la actividad de una neurona obtenida por aplicación del núcleo *i* en la posición (*x*,*y*) y luego se aplica la nolinealidad ReLU, la actividad de la respuesta normalizada $b^i_{x,y}$ viene dada por la expresión, donde la suma se extiende sobre *n* mapas adyacentes generados por los núcleos en la misma posición (*x*,*y*) y *N* es el número total de núcleos en la capa.

$$b^i_{x,y} = a^i_{x,y} \Big/ \left(k + \alpha \sum_{j=max(0,i-n/2)}^{min(N-1,i+n/2)} \left(a^i_{x,y} \right)^2 \right)^{\beta} \qquad (38)$$

El ordenamiento de los mapas es arbitrario y se determina antes de que comience el aprendizaje.

Esta clase de respuesta normalizada implementa una forma de inhibición lateral inspirada por el tipo encontrado en neuronas reales, creando una competición para actividades elevadas frente a salidas de neurones obtenidas usando diferentes núcleos. Las constantes *k*, *n*, α y β son hiperparámetros cuyos valores se determinan usando un conjunto de validación. En [19] se proponen los siguientes valores $k = 2$, $n = 5$, $\alpha = 10^{-4}$ y $\beta = 0.75$. En la mismo trabajo, se indica que esta normalización se realiza en ciertas capas tras la aplicación de ReLU.

En [27] se propone otro esquem de normalización (Local Contrast Normalización) para una capa determinada (*K*), basada en la computación de substracciones y divisiones, forzando una clase de competición local entre mapas de características adyacentes. La operación de normalización sustractiva para $a^i_{x,y}$ (definida previamente) se obtiene,

$$b^i_{x,y} = a^i_{x,y} - \sum_{p,q} w_{pq} a^i_{x+p,y+q} \qquad (39)$$

donde w_{pq} es una ventana promediada Gaussiana de domensión $p \times q$ de modo que $\sum_{p,q} w_{pq} = 1$. La operación de normalización relativa a la división se obtiene como sigue,

$$d^i_{x,y} = b^i_{x,y} \Big/ max\left(c, \sigma_{x,y}\right) \qquad (40)$$

donde $\sigma_{x,y} = \sum_{p,q} w_{pq} \left(b^i_{x+p,y+q} \right)^{1/2}$ y $c = mean\left(\sigma_{x,y}\right)$. Por tanto, en la expresión anterior el denominador es la desviación estándar promediada de todas las características en una vecindad espacial. La normalización de contraste local está inspirada en modelos computacionales de la neurociencia [28].

3.4 Reconocimiento de regiones en imágenes.

Las nuevas tecnologías tales como vehículos autónomos, video vigilancia inteligente, detección facial y otras aplicaciones similares son cada vez más demandadas y requieren de una mayor precisión y velocidad de proceso. Estos sistemas no sólo requieren de reconocimiento y clasificación de objetos presentes en la imagen, sino que además se hace necesaria la localización del objeto. Esto hace que la detección de objetos sea una tarea aún más compleja que su predecesora en el ámbito de la visión por computador, la clasificación de imágenes [29].

De este modo el objetivo que persigue esta tecnología es lograr identificar regiones clasificables dentro del conjunto de una imagen.

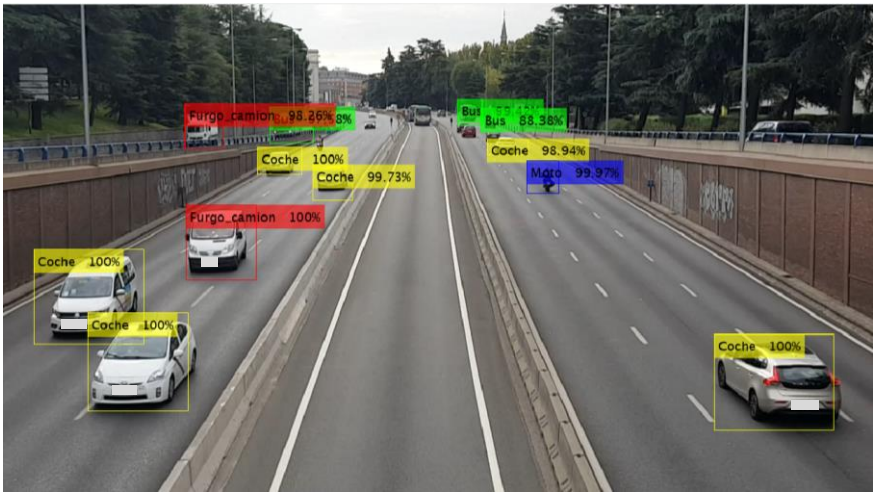


Figura 43: Diferentes clases encontradas en A-6 (Madrid)

Para alcanzar este objetivo necesitamos separar diferentes regiones dentro de la imagen e ir clasificándolo, por lo que podríamos entender el proceso en una primera fase de "*data mining*" y otra de clasificación.

En los apartados anteriores hemos explicado las RNA's como medio para la clasificación, en concreto las CNN, por lo que partiremos de este punto para la clasificación, pero estudiaremos como extraer las distintas técnicas extraen las regiones en una imagen.

Para ello se identifican cuatro técnicas principales, de las cuales en este trabajo nos centraremos en las tres primeras, dado que la última, redes de segmentación semántica "SegNet", presenta una diferencia sustancial con respecto a las tres primeras, ya que la identificación no la realiza a nivel de región sino a nivel de píxel por lo que no se analizan en este trabajo.

- Regions whit convolutional neuronal network (R-CNN)
- Fast R-CNN
- Faster R-CNN
- SegNet Segmentación semántica

Es importante comentar que en lo que se refiere a la arquitectura de una R-CNN en comparación a una CNN, la estructura de salida estará compuesta por una clase extra, el fondo, la cual no se mostrará

en el proceso de detección, pero resulta importante para la discriminación de regiones que pertenecen a este fondo de imagen o "*background*".

3.4.1 R-CNN

A diferencia de las CNN las R-CNN han de proporcionar 2 salidas en vez de una, una encargada de la clasificación y otra encargada de asociar dicha clasificación a una región determinada:

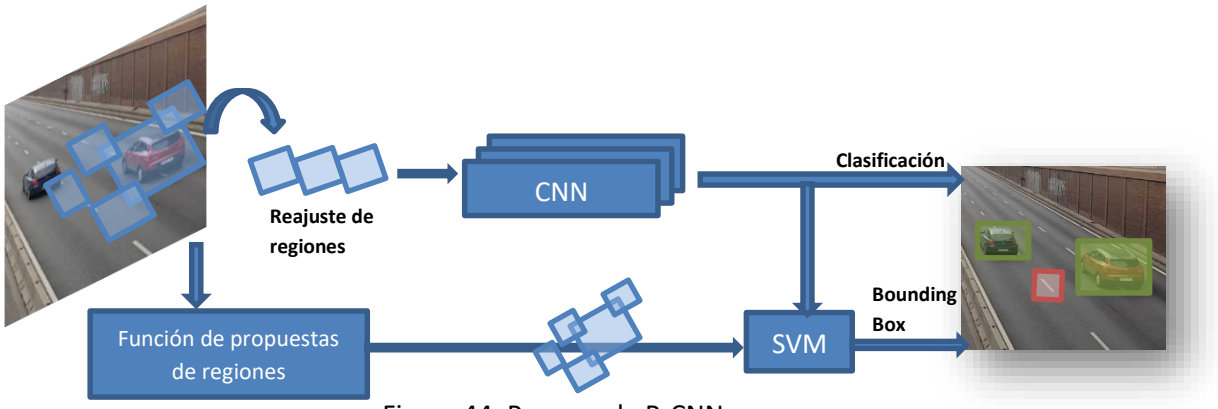


Figura 44: Proceso de R-CNN

Este proceso consiste en tres fases:

1. Captura de la imagen de entrada en búsqueda de posibles objetos usando un algoritmo como puede ser "*Edge Boxes*", que extrae aproximadamente 2000 regiones de la imagen (*regiones propuestas*), este parámetro se puede ajustar.
2. Se redimensionan las regiones propuestas y se ejecuta la CNN sobre las regiones propuestas.
3. Se toma la salida de la CNN y se realizan los siguientes procesos:
    a. Por cada salida clasificable y región asociada, se alimenta a una SVM (*Suport Vector Machine*) que ajusta la región para seguir obteniendo la clasificación.
    b. Se proporciona la información de identificación, clase y probabilidades

De este modo a la estructura general de la CNN no se ve modificada, solo añadimos los procesos de extracción de regiones y optimización de las mismas. Esto se realiza de manera automática en Matlab mediante el entrenamiento de un detector de objetos "*trainRCNNObjectDetector*".

Dado que se proporcionan muchas regiones y cada una de ellas hay que clasificarla el proceso puede resultar lento si la CNN es "pesada", por lo que se motiva la búsqueda de un método rápido de regiones (*fast*).

Como podemos ver la primera fase de este tipo de arquitectura (ROIs), es equivalente a la proporcionada en el segundo capítulo donde identificamos regiones, pero partiendo del movimiento en vez de un proceso de "*Machine learning*".

3.4.2 Fast R-CNN.

Motivado por el problema en tiempo de cómputo de las R-CNN, las Fast R –CNN buscan reducir dicho tiempo. Esta técnica asemeja a su predecesora en muchos aspectos, si bien mejora la velocidad de detección fundamentalmente en dos aspectos:

1. La extracción de características sobre la imagen se lleva a cabo antes de proponer las regiones de interés, por lo tanto, usando una sola CNN sobre el total de la imagen en lugar de las 2000 CNNs en las 2000 regiones sobrepuestas (R-CNN).
2. Se reemplaza el SVM por una capa *softmax*, de esta forma se extiende la red neuronal para predicciones en lugar de crear un nuevo modelo.

Así el proceso queda:

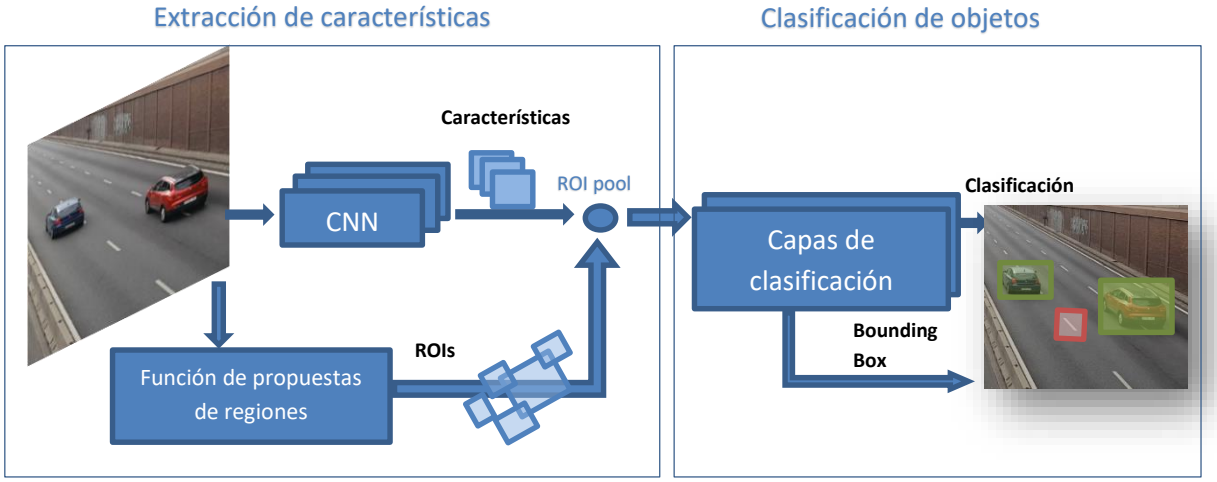


Figura 45: Esquema Fast R-CNN

Una imagen de entrada y múltiples regiones de interés (RoIs) son las entradas que aportamos a una red totalmente convolucional. Cada región de interés (RoI) se asocia con un mapa de características de tamaño fijo que se proyecta sobre un vector de características por capas totalmente conectadas (FC). La red tiene dos vectores de salida por región de interés: clasificación (*softmax)* y compensación a la regresión de la caja delimitadora por clase (*box regresión layer*) [29].

El modelo Fast R-CNN mejora la velocidad de detección-clasificación. Llegado a este punto sólo queda optimizar el algoritmo selectivo de búsqueda para generar las regiones propuestas [29].

En este caso si partimos de una CNN concreta la estructura deberemos modificarla del siguiente modo:

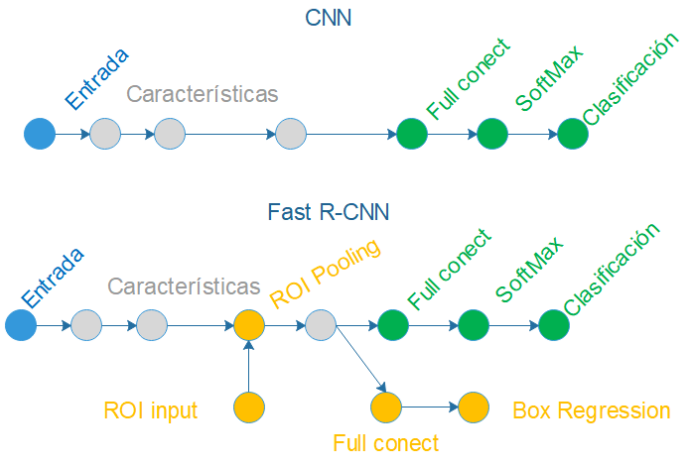


Figura 46: Evolución de estructura general CNN a estructura general Fast R-CNN

Donde las nuevas capas introducidas son:

- ***Box Regresion Layer***: Encargada de mejorar la posición del objeto en la imagen al aprender un conjunto de compensaciones del conjunto de recuadros de entrenamiento
- ***ROI Pooling layer***: Se encarga de agrupar el conjunto de características de la CNN para cada propuesta de región.
- ***ROI Input Layer***: La introducimos para poder tener una entrada de regiones preetiquetadas en el conjunto de datos de entrenamiento.

3.4.3 Faster R-CNN.

El objetivo principal de Faster R-CNN es reemplazar el lento algoritmo de selección de regiones propuestas con una red neuronal rápida. Para ello se introdujo la red de propuesta de regiones *(Region Proposal Network)* o RPN [30].

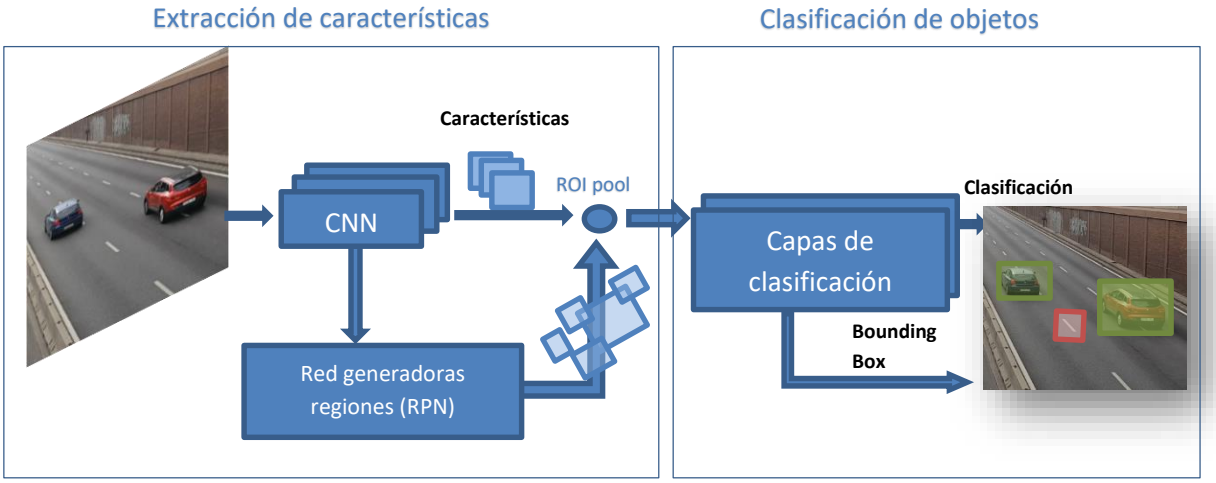


Figura 47: Esquema Faster R-CNNN

Como se menciona en [30], la red RPN realiza el siguiente proceso:

1. En la última capa de una CNN inicial, una ventana deslizante de 3x3 se mueve a lo largo del mapa de características y re-mapea en dimensiones inferiores.
2. Para cada localización de la ventana deslizante, se generan múltiples posibles regiones basadas en un radio fijo de fijación de cajas (cajas delimitadoras predeterminadas)
3. Cada propuesta de región consiste en:
    a. Una puntuación de "objetividad" para esa región.
    b. Cuatro coordenadas que representan la caja delimitadora de la región.

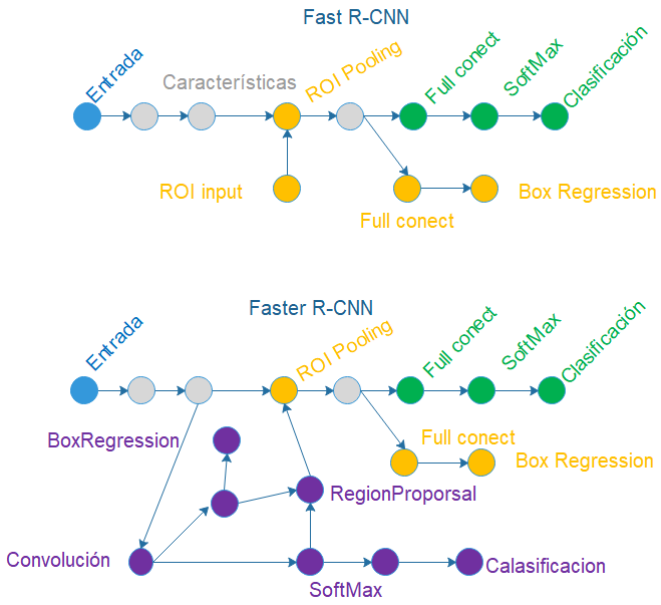


Figura 48: Evolución de estructura general Fast R-CNN a estructura general de Faster R-CNN

Las puntuaciones 2k (*2k scores*) representan la confianza *softmax* para cada una de las *k* cajas delimitadoras que están sobre un "objeto". Cabe señalar que, aunque la RPN procesa las coordenadas de los cuadros delimitadores, no clasifica los objetos potenciales, su único propósito es proponer regiones en los que puede encontrarse el objeto. Si la "objetividad" que presenta una determinada caja supera un determinado valor ("puntuación"), las coordenadas de esa caja pasan como una propuesta de región [30].

Una vez obtenidas las propuestas de regiones, las alimentamos a través de lo que es esencialmente una *Fast R-CNN*. Se añade una capa *pooling,* algunas capas conectadas y finalmente una capa de clasificación *softmax* y un regresor de cajas delimitadoras. En resumen, podemos considerar **Faster R-CNN** como una suma de **RPN** y **Fast R-CNN** [29].

En términos de estructura ahora nuestra CNN que en términos estructurales podía ser "serie" o "DAG", será siempre de tipo DAG a consecuencia de introducir la RPN.

Con esta configuración la red Faster R-CNN presenta una velocidad y precisión realmente novedosas, con un rendimiento verdaderamente significativo.

3.5 Arquitecturas de redes utilizas

En este apartado se muestran las arquitecturas de las redes utilizadas para el trabajo, donde se han utilizado 2 redes preen trenadas y modificadas y una tercera de elaboración propia denominada "*Vehicle*".

Estas redes son:

- AlexNet
- ResNet50
- Vehicle

La justificación de utilizar AlexNet es por su relevancia dentro del mundo de la visión por computador pues desde que en octubre de 2012 su creador *Alex Krizhevsky*, ganara la competición de *ImageNet* logrando un resultado de más de 10 puntos por debajo del error en clasificación con respecto al subcampeón, mostrando una arquitectura muy superior al resto. La relevancia de dicha arquitectura es tal que [19] es uno de los artículos más citados entre las publicaciones de visión artificial, con más de 36000 citaciones (Feb.2019).

En el caso de ResNet también ganó la misma competición en 2015, esta red fue propuesta por Microsoft y sus siglas provienen de la introducción de residuos entre los diferentes bloques de convolución. Marco también una revolución al introducir un método con el que la profundidad de una red no se veía afectada en términos de error [31].

3.5.1 AlexNet

La figura 49 muestra el esquema de la red AlexNet con especificación de las distintas capas de que consta.

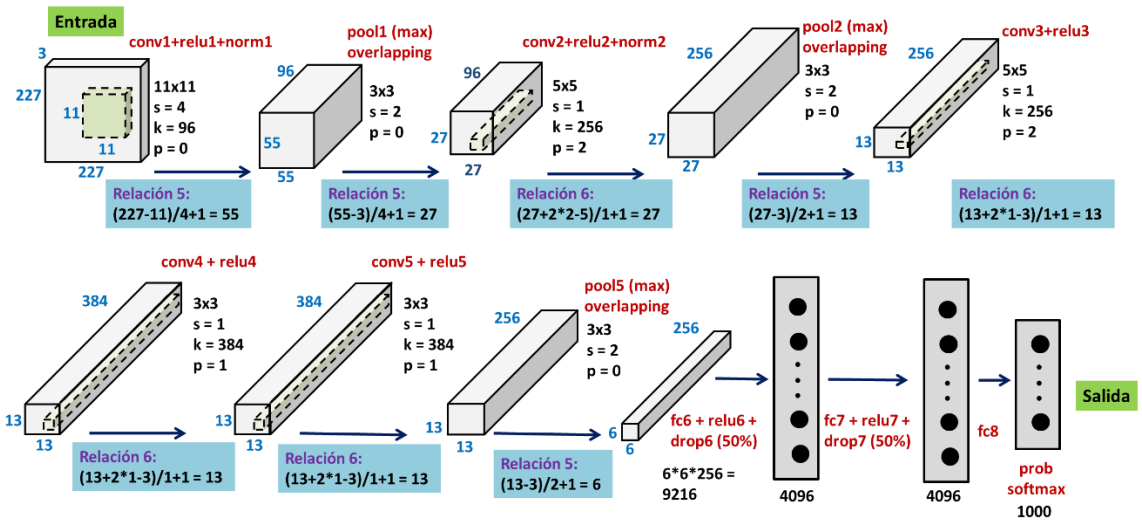


Figura 49: Arquitectura de capas AlexNet

Podemos ver como mantiene la arquitectura de una CNN sin tener una excesiva profundidad de red.

Para nuestro problema haremos una transferencia de conocimiento, para esto simplemente modificaremos la estructura de las últimas capas para poder hacer una clasificación de 4 clases diferentes; es decir solo 4 neuronas de salida.

3.5.2 ResNet50

La motivación estructural de este tipo de arquitectura proviene del estudio de la respuesta de redes ante la adición de más capas y por consiguiente una mayor profundidad de la red.

En el trabajo de [32], se muestra una comparación de la evolución del error en función de la profundidad de la red que proporciona parte de la motivación de dicho trabajo.

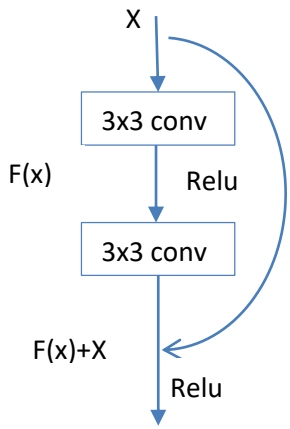


Figura 50: Bloque de residuo

Tal y como se muestra en [32], el problema principal reside en la optimización de las capas lo cual solucionan introduciendo el concepto de residuo [33].

Este proceso de bloque de residuos se añade dependiendo de la profundidad de la red pudiéndonos encontrar profundidades en redes preentrenadas ResNet de 34, **50**, 101 o 152.

Para mejorar la eficiencia en redes muy profundas se añade en la conexión residual una capa de normalización ("*batch normlization Layer*") que proporciona aceleración en el proceso de aprendizaje; esto aparece implementado también en redes como GoogleNet.

3.5.3 Vehicle

En este apartado exponemos una arquitectura propia generada a partir de la referencia [6] con el objetivo de alcanzar una velocidad mayor en el entrenamiento y detección para un numero bajo de clases.

La arquitectura general se define de acuerdo al esquema mostrado en la figura 51, que corresponde a la interfaz proporcionada por la herramienta "*deepNetworkDesigner*" de Matlab para el diseño de este tipo de redes.

En la figura 51 se pueden observar las distintas capas presentes en el diseño e identificadas según la terminología definida previamente: conv, relu, fc y *softmax*.

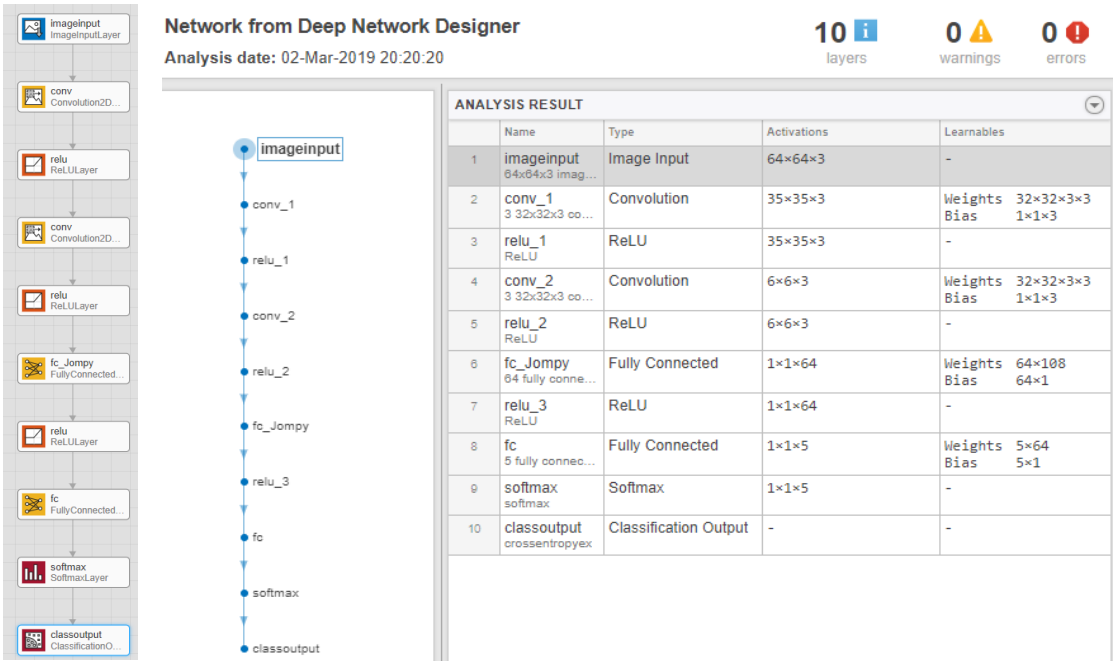


Figura 51: Arquitectura CNN Vehicle

Para finalmente obtener esta arquitectura se realizaron varias pruebas de entrenamiento utilizando diferentes tamaños de imagen de entrada 32x32x3, 64x64x3 y 128x128x3. Los resultados de la experimentación en la modificación de la capa de entrada se muestran en el capítulo 4 donde se somete a las diferentes arquitecturas a una fase de entrenamiento y validación.

CAPÍTULO 4: ENTRENAMIENTO

Para el proceso de entrenamiento y validación de AlexNet, ResNet50, Vehicle y las R-CNN, Fast R-CNN y Faster R-CNN asociadas a las CNN indicadas, se ha introducido el algoritmo de retropropagación, dicho algoritmo requiere para su funcionamiento un conjunto de datos que además de mostrar las imágenes, tengan definidas unas etiquetas de clasificación. Esto implica que el aprendizaje de las CNN sea supervisado, ya que el sistema aprende en función de lo que se le propone, de ahí la gran importancia de un correcto conjunto de datos de entrenamiento.

Existen diferentes plataformas y grupos de investigación que proporcionan datos de entrenamiento, es decir, imágenes etiquetadas y clasificadas, para este trabajo se solicitó acceso a la base de datos de "Cityscapes" y "GTI-UPM" (Grupo de tratamiento de imágenes de la Universidad Politécnica de Madrid) [35]. La primera posé un conjunto de más de 20000 fotogramas correspondiente a entornos urbanos de más de 50 ciudades diferentes, etiquetados a nivel de pixel con conjuntos diferenciables de varias categorías correspondiente al estudio de las *smart cities*, como pueden ser construcciones, humanos, vehículos, entre otras. [34].

El segundo conjunto de datos correspondiente a GTI de la UPM, posee diferentes conjuntos de datos etiquetados, para llevar a cabo los experimentos de este trabajo se solicitó acceso al conjunto de datos de vehículos el cual consta con más de 3400 imágenes clasificadas de vehículos en Madrid, Bruselas y Turin.

Con el objeto de trabajar a nivel de imagen y en clasificación de diferentes tipos de vehículos finalmente se optó por generar nuestro propio conjunto de datos (*dataset*) con características que aparecen en la tabla 3. En ella aparecen las cuatro clases consideradas (coche, moto, camión-furgo y bus), con el número de muestras o elementos indicados.

Clase	Coche	Moto	Camión-furgo	Bus
Nº elementos	148	28	102	93

Tabla 3: Base de datos propia

Consideramos que es un conjunto satisfactorio de imágenes dada la evolución en el proceso de entrenamiento y validación de las diferentes CNNs. Esto puede comprobarse en el conjunto de imágenes del apartado 4.1 que muestran la evolución de los procesos de entrenamiento, donde se observan mismas tendencias entre ajustes y validación, o lo que es lo mismo ausencia de *overfitting* y *underfitting* unidos a una validación mínima muy alta en todos los casos como también se puede observar en la tabla 5.

Puesto que se han utilizado diferentes arquitecturas se generó una función encargada de extraer de los diferentes vídeos etiquetados los diferentes conjuntos de imágenes redimensionadas según la arquitectura en cuestión para la arquitectura deseada, En la tabla 4, se muestran las correspondientes dimensiones en el modelo de color RGB (*Red, Green, Blue*) junto con el tamaño de los archivos

correspondientes (conjunto de imágenes clasificadas en carpetas por clases en formato .jpeg).

Arquitectura de Red	Dimensión de imagen RGB	Tamaño archivo
AlexNet	[227x227x3]	2.25MB
ResNet50	[224x224x3]	2.14MB
Vehicle	[32x32x3]	335KB
	[64x64x3]	556kB
	[128x128x3]	1.10MB

Tabla 4: Volumen de BBDD propia adaptado a las diferentes CNN

Para obtener este conjunto de datos se han utilizado diez vídeos de unos 180MB convenientemente etiquetados frame a frame hasta completar el conjunto de imágenes pertenecientes a las clases anteriores. Posteriormente de los videos se extraen 172 frames relevantes etiquetados de los que se obtienen las 371 imágenes anteriores.

Dependiendo el tipo de red utilizada el conjunto de datos de entrenamiento es diferente, diferenciando dos tipos, según las arquitecturas CNN y ROI CNN.

Para el primer grupo se proporciona el dataset como conjunto de tantas carpetas como clases a entrenar (4), en el que las imágenes han de tener la dimensión de la capa de entrada de la red a entrenar.

El grupo de ROI CNN hace referencia a las R-CNN, fast-CNN y Faster-CNN a las cuales se les proporciona una tabla en la que cada frame del video (previamente etiquetado), indica por cada una de las clases etiquetadas (4) la o las regiones por coordenadas donde se encuentra alguno de los vehículos definidos por las clases.

Para realizar el etiquetado del vídeo se utilizó la APP de Matlab, "*Ground Truth Labeler*" [6]

Cuando se realiza el entrenamiento se utiliza un subconjunto del conjunto de datos para la validación de la red, lo que permite determinar si se produce "*overfitting*", "*underfitting*" o se correlacionan bien.

Esta proporción de datos de entrenamiento y validación ha sido de un 80% para entrenamiento y un 20% para la validación de los mismos y para las 4 categorías de redes utilizadas (CNN, R-CNN, Fast R-CNN y Faster R-CNN).

4.1 Entrenamiento y validación de las CNN

El proceso de entrenamiento y validación de las diferentes CNN se realizó con Matlab de manera gráfica. En cada una de las gráficas mostradas aparecen los resultados relativos a precisión (*accuracy* en %) y error respecto del ajuste de la función objetivo (*loss*) a medida que el entrenamiento avanza.

Se muestran los resultados finales en términos de validación, junto con el tiempo de entrenamiento. Además, aparecen los ciclos que definen la etapa con indicación del número de *epochs* y las iteraciones

en cada una de ellas. Se muestra la razón de aprendizaje y cómo ésta se mantiene constante a lo largo de todo el proceso. Los procesamientos se llevan a cabo en una única CPU.

No se ha fijado ningún límite del tipo *patience*, que está puesto a infinito. Cuando se fija este parámetro a un valor determinado, lo que realmente se indica es que el proceso se detenga después de que tras el número especificado de *epochs* no se consigan mejoras significativas o relevantes en la precisión.

Alexnet:

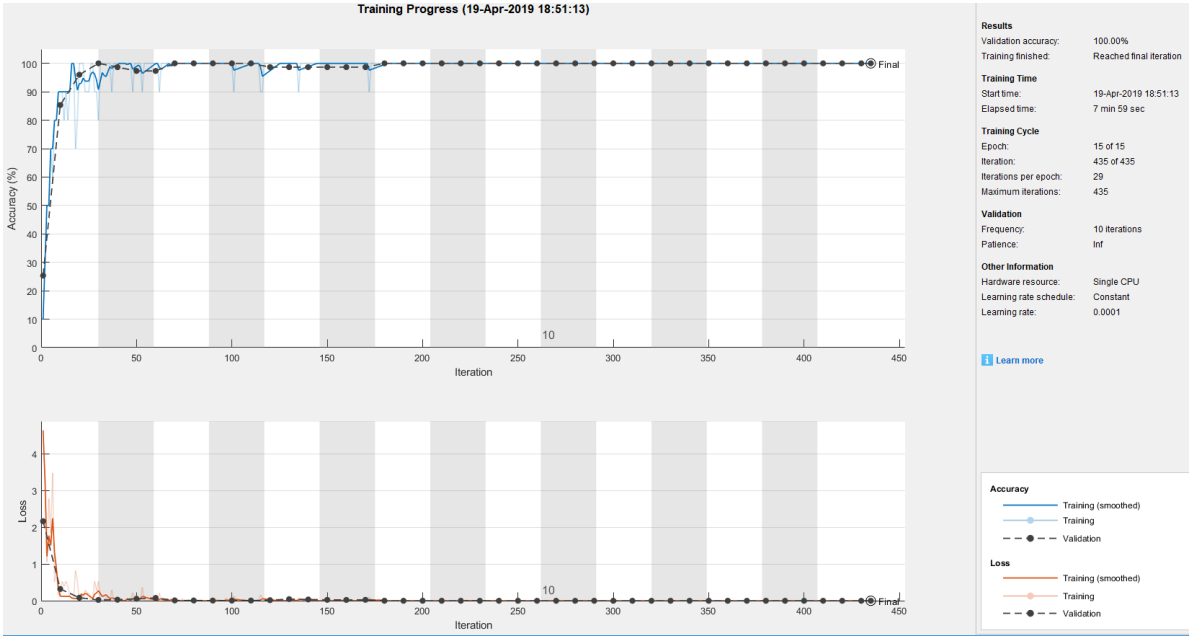
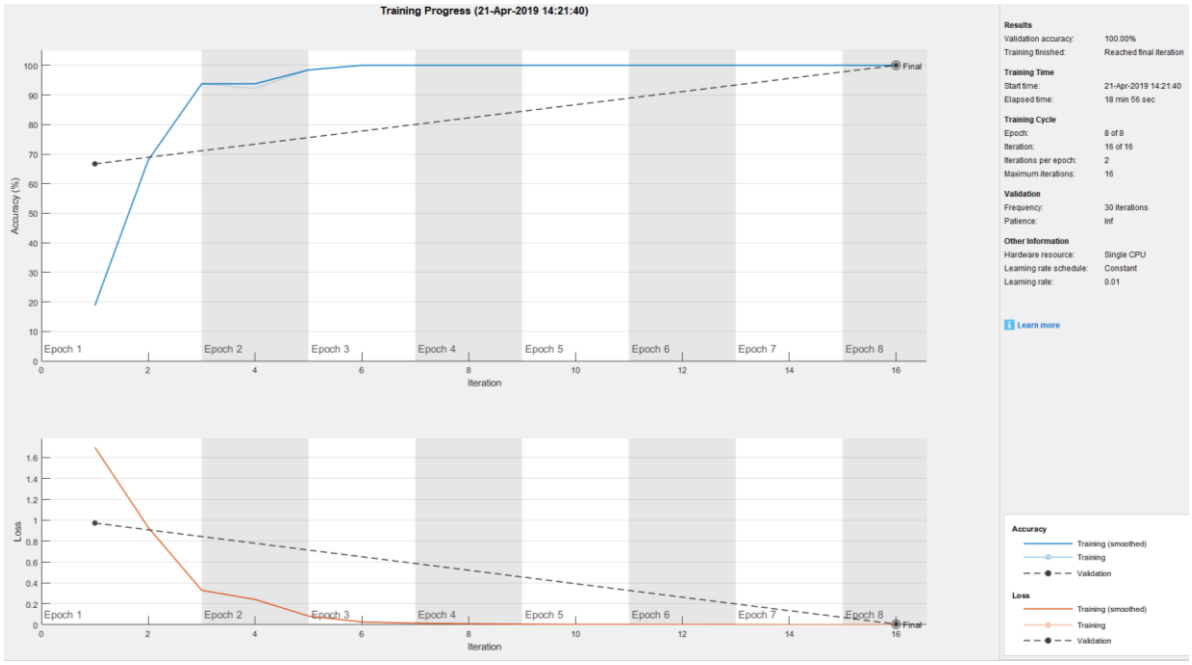


Figura 52: Evolución entrenamiento AlexNet

Resnet50:



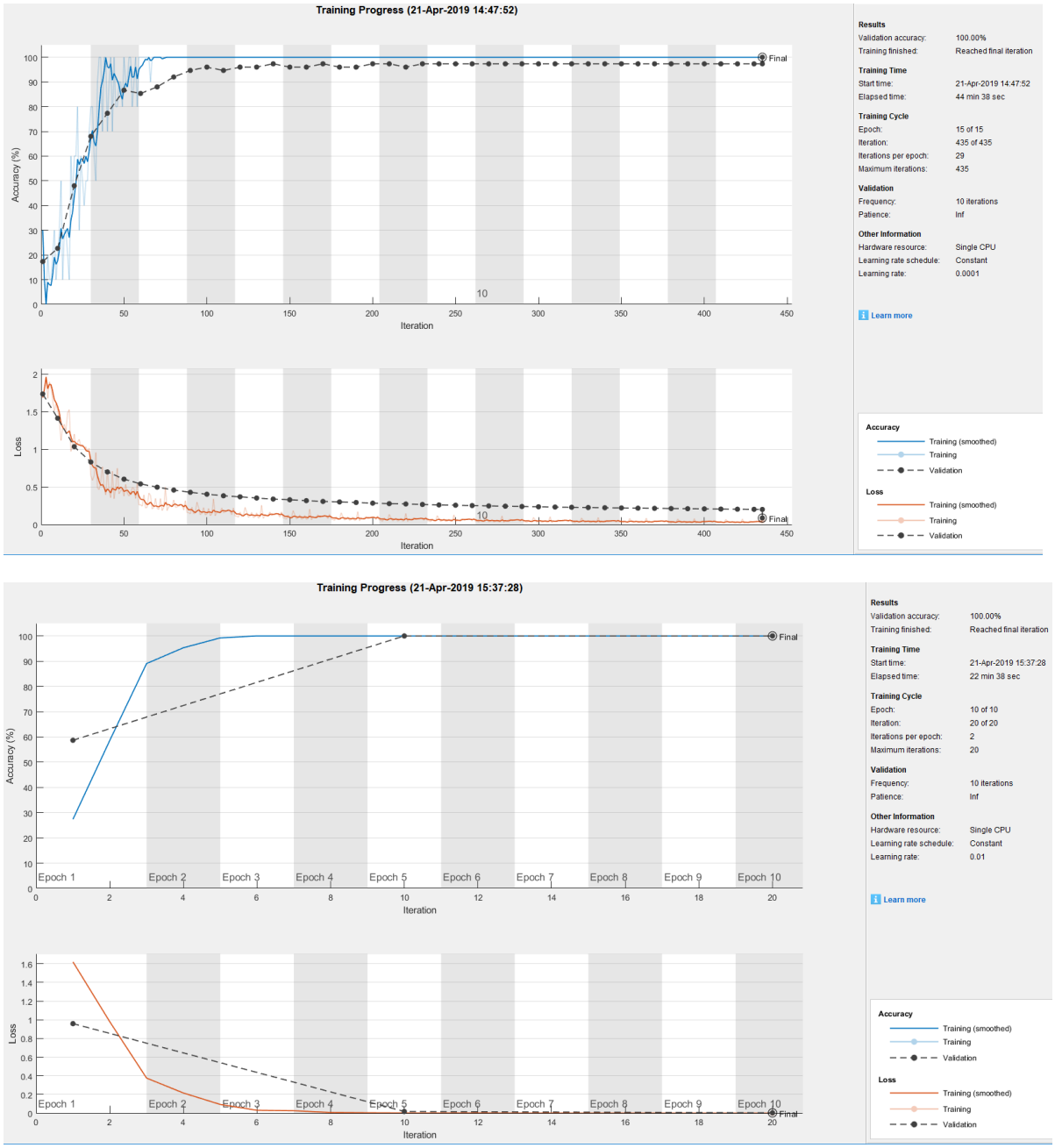


Figura 53: Evolución entrenamiento ResNet50

Vehicle:

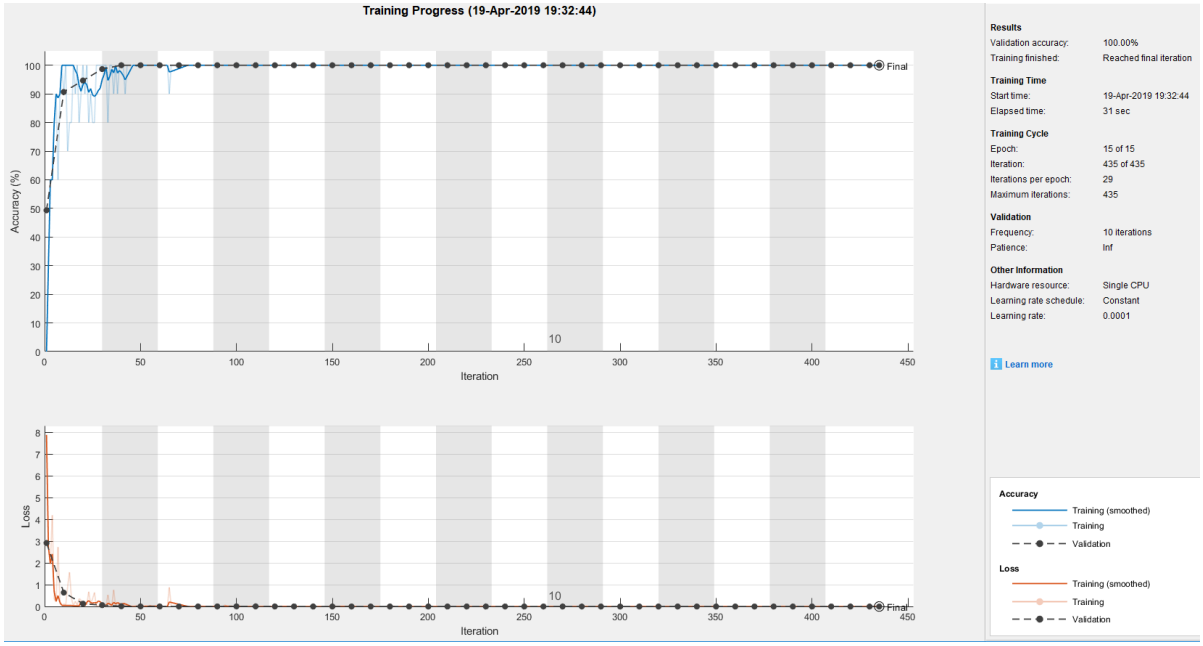


Figura 54: Evolución entrenamiento Vehicle 32x32

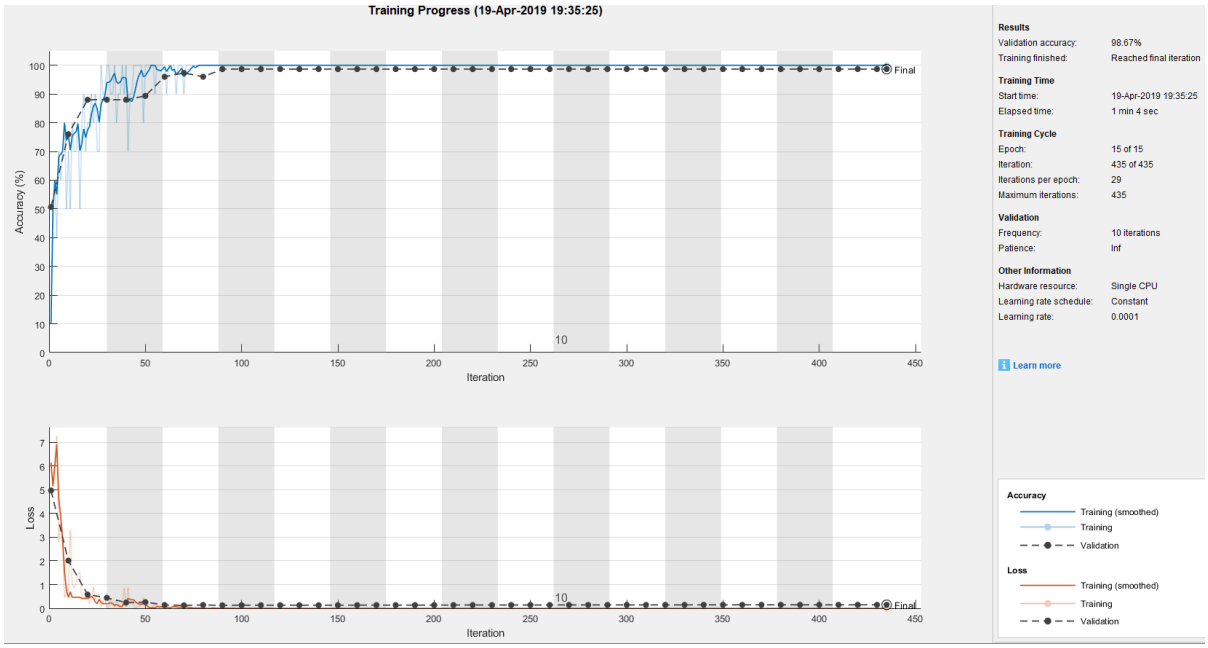


Figura 55: Evolución entrenamiento Vehicle 64x64

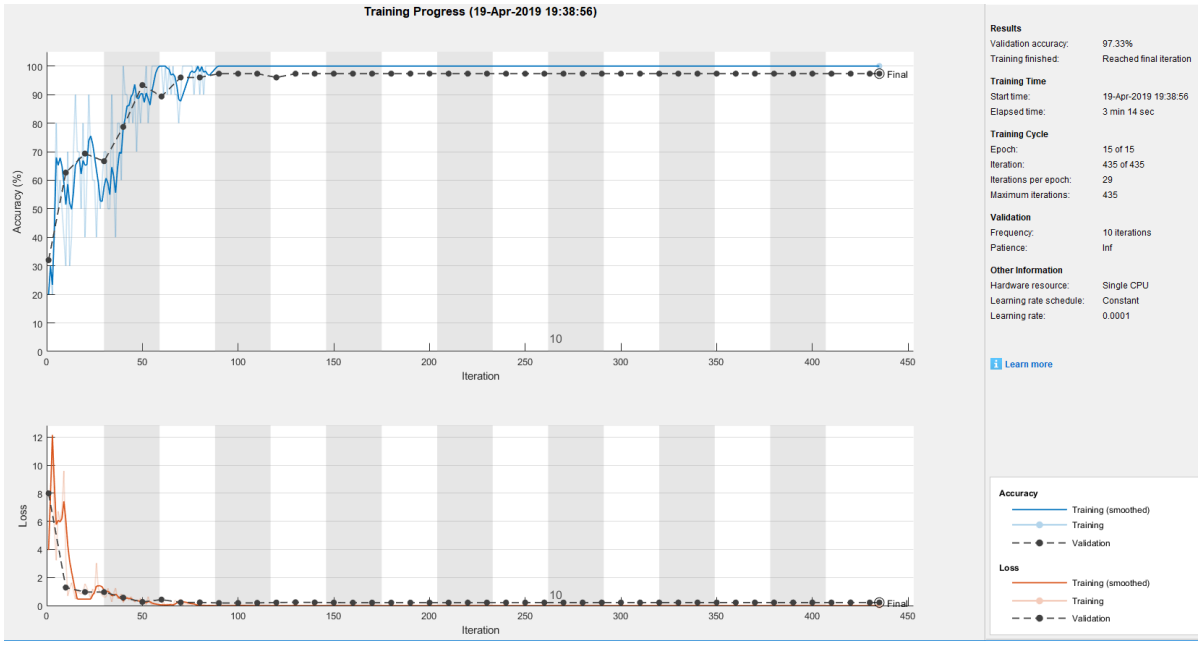


Figura 56: Evolución entrenamiento Vehicle 128x128

En la tabla 5 se muestra los resultados en el proceso de entrenamiento de las diferentes CNNs clasificadas por arquitectura, mostrando tamaño de la capa de entrada (dimensión), mínima validación en el proceso de entrenamiento y el tiempo empleado para dicho entrenamiento.

Arquitectura de Red	Dimensión	Validación [%]	Tiempo
Alexnet	[227x227x3]	100	8 min
Resnet50	[224x224x3]	100	18 min
Vehicle	[32x32x3]	100	31 s
	[64x64x3]	98.67	1min 4s
	[128x128x3]	97.33	3min 14s

Tabla 5: Resultados entrenamiento CNN's

Considerando los resultados de la tabla 5, en el caso concreto de la arquitectura de Vehicle, comprobamos cómo la dimensión de entrada que mejores resultados obtiene en cuanto a tiempo y validación es la correspondiente a 32x32x3, por lo cual para las redes de detección R-CNN, fast R-CNN y faster R-CNN, utilizaremos este tamaño de entrada (siempre hablando de la arquitectura de Vehicle).

4.2 Entrenamiento y validación de R-CNN, fast R-CNN y Faster R-CNN

En este caso no se dispone de un interfaz de representación por lo que los resultados del proceso de entrenamiento se han resumido mediante las tablas 6 y 8, donde se muestra la validación del proceso de entrenamiento para las diferentes arquitecturas y el tiempo empleado en este proceso.

En la tabla 6 se muestran los resultados de entrenamiento de las arquitecturas R-CNN para las diferentes arquitecturas, recordando que la arquitectura de Vehicle corresponde a la que mejores resultados obtuvo en el proceso de entrenamiento y validación en modo CNN.

R-CNN			
Arquitectura de Red	**Dimensión**	**Validación [%]**	**Tiempo**
AlexNet	[227x227x3]	96.88	2h17min43s
ResNet50	[224x224x3]	100	15h14min56s
Vehicle	[32x32x3]	98.99	1h4min56s

Tabla 6: Entrenamiento y validación R-CNN

En el caso de la tabla 7, se muestra el resumen de entrenamiento para las arquitecturas fast de R-CNN, donde podemos observar el dimensionamiento de la capa de entrada, la validación en el proceso de entrenamiento y el tiempo empleado para dicho entrenamiento.

Fast R-CNN			
Arquitectura de Red	**Dimensión**	**Validación [%]**	**Tiempo**
AlexNet	[227x227x3]	98.77	5h 30min 25s
ResNet50	[224x224x3]	85,94	2h 36mn 56s
Vehicle	[32x32x3]	99,37	2h 1m 29s

Tabla 7: Entrenamiento y validación Fast R-CNN

En el caso del proceso de entrenamiento de las arquitecturas faster R-CNN el proceso de entrenamiento tiene 4 fases, una encargada del entrenamiento de la RPN (figura 47), entrenamiento de la arquitectura completa (fast R-CNN) y dos fases de reentrenamiento de las dos anteriores.

Puesto que se distinguen diferentes fases en el proceso de entrenamiento de las faster R-CNN, se indican distintos tiempos de entrenamiento. Al conjunto de entrenamiento de la red faster tenemos que añadir el tiempo de preparación de las imágenes y regiones, por lo que en la tabla 8 se añade un tiempo total que es el conjunto de las fases que lo conforman junto con el proceso de preparación indicado. A este tiempo se denomina tiempo total (T.Tot) el cual aparece como última fila de la tabla 8.

	Faster R-CNN			
	Arquitectura	AlexNet	ResNet50	Vehicle
	Dimensión	[227x227x3]	[224x224x3]	[32x32x3]
Train RPN	Tiempo	-	4h 52min36s	4h 2min 14s
	Validación [%]	-	100	100
Train fast R-CNN	Tiempo	-	2h 19min 8s	1h 23min
	Validación [%]	-	96,88	95,31
Re-train RPN	Tiempo	-	3h 37min 8s	2h 6min 17s
	Validación [%]	-	100	100
Re-train fastR-CNN	Tiempo	-	1h 3min 48s	21min 34s
	Validación [%]	**-**	**84,38**	**92,97**
	T.Tot*	**-**	**25h 16m 9s**	**19h 3m 27s**

Tabla 8: Entrenamiento y validación Faster R-CNN

* Este tiempo total es la suma de las 4 fases de entrenamiento de las redes faster R-CNN más el tiempo que requiere el sistema para preparar la base de datos con la que es entrenada.

En el caso de AlexNet no se ha logrado su convergencia por lo que no se han incluido resultados de entrenamiento en la estructura propuesta de Faster R-CNN AlexNet.

CAPÍTULO 5: TRÁNSITO DE VEHÍCULOS

Dentro del campo de la ingeniería civil podemos encontrarnos la especialidad asociada al tráfico o tránsito de vehículos y o personas, enmarcado dentro de lo que se podría denominar como "Ingeniería de Tráfico", la cual se encarga de la planificación, diseño y operación de tráfico en las calles, carreteras, autopistas y otras infraestructuras con los diferentes medios de transporte. El objetivo es conseguir una movilidad segura, eficiente y conveniente tanto de personas como de vehículos.

A la hora de realizar el estudio de gestión del tráfico, es fundamental tener en cuenta las variables de tránsito [36], [37] y [38].

Estas variables de tránsito permiten modelar sistemas dinámicos con los que realizar simulaciones del comportamiento del tráfico, evolución de la polución u otras de vital importancia en el contexto de las "*smart cities*", o ciudades inteligentes.

5.1 Variables de tránsito de vehículos

A continuación, se introducen y describen las variables para el cómputo del tránsito, recordando que es posible encontrar otras variables más específicas y no menos importantes en la literatura como en [38].

Las principales variables que se estudian son:

- Intensidad o flujo de vehículos
- Velocidad
- Densidad

a) Intensidad o flujo (veh/h)

La intensidad de vehículos es la expresión que resulta al dividir el número de vehículos que atraviesan una determinada sección de la vía en un intervalo de tiempo determinado, denominado periodo de integración. Ésta es una variable muy fácil de obtener, puesto que, si pasan N vehículos por una sección, definida como una región específica en la imagen, durante un periodo de tiempo T, la expresión de la intensidad q resultante es:

$$q = \frac{N}{T} \quad (41)$$

Otra manera de entender este fenómeno es considerar que el periodo de integración es la suma de los intervalos de tiempo h_i entre vehículos sucesivos. Por tanto, si se sustituye en la expresión anterior dicha consideración, se obtiene:

$$T = \sum_{i=1}^{N} h_i \; ; q = \frac{1}{\frac{1}{N}\sum_{i=1}^{N} h_i} = \frac{1}{\bar{h}} \quad (42)$$

Donde $\bar{h}$ representa la media de los intervalos de tiempo entre los N vehículos consecutivos.

Por tanto, resulta coherente que el flujo de vehículos, medido en número de vehículos por unidad de tiempo, pueda obtenerse también realizando la inversa del tiempo medio que transcurre entre dichos vehículos, puesto que las unidades coinciden en ambos casos [36].

b) Velocidad

Esta variable se define como el espacio recorrido por un vehículo en un determinado tiempo. No obstante, cuando nos referimos a flujo de tráfico se requiere de la utilización de la velocidad media de los vehículos. De este modo se pueden diferenciar dos tipos diferentes, velocidad local media y velocidad momentánea media.

La velocidad local media $\bar{v}_l$, hace referencia a la media de las velocidades de un conjunto N de vehículos que atraviesan una sección específica de una vía. Esta velocidad local se puede expresar como la media armónica de intervalos de tiempo Δt_i, que tardan los N vehículos en recorrer una distancia fija Δx:

$$\bar{v}_l = \frac{\Delta x}{N} \sum {}^{1}/_{\Delta t_i} \quad (43)$$

Por otra parte, la velocidad momentánea o velocidad media espacial, se define como la media aritmética de las velocidades de los vehículos que se encuentran entre dos secciones determinadas y en un determinado momento. Tomando N vehículos, cada uno con una velocidad v_i, puede escribirse la siguiente expresión:

$$\bar{v}_m = \frac{N}{\sum {}^{1}/_{v_i}} = \frac{1}{N\Delta t} \sum \Delta x_i \quad (44)$$

Donde Δt corresponde al intervalo de tiempo que tarda un vehículo en recorrer Δx.

Entre ambas velocidades existe una relación deducida por Wardrop [36], asociada a las varianzas de las muestras de velocidades:

$$\bar{v}_l = \bar{v}_m + \frac{\sigma_m^2(v)}{\bar{v}_m} \quad (45)$$

De la expresión anterior se deduce fácilmente que la velocidad local es mayor que la velocidad

momentánea, excepto cuando la varianza es nula, lo cual implica que las velocidades de todos los vehículos serian la misma. Esta relación puede resultar interesante en el estudio de mejorar el tránsito de vehículos autónomos inteligentes por determinadas vías, donde estos vehículos puedan funcionar de manera colaborativa, utilizando este parámetro como función objetivo a minimizar para evitar colisiones, flujo constante, entre otras opciones.

c) Densidad

La densidad de vehículos K se define como el número de vehículos N que ocupan una longitud determinada Δx en un instante de tiempo concreto.

$$K = \frac{N}{\Delta x} \quad (46)$$

Si conocemos la información en un punto concreto podemos formular la **"ecuación fundamental del tránsito"** [38] que nos la relaciona con el flujo de vehículos y la velocidad momentánea.

$$K = \frac{q}{\bar{v}_m} \quad (47)$$

5.2 Procedimientos y tecnologías de medida

Aparte de las variables introducidas anteriormente existen otro tipo de variables que se utilizan, como son entre otras el tiempo de viaje o la ocupación de la vía.

Dependiendo de la metodología utilizada para la medida de variables se pueden utilizar directa, o indirectamente unas u otras. En este apartado introduciremos algunos procedimientos de medida clásicos para la obtención de dichas variables, así como las tecnologías principales que se utilizan con dichos procedimientos.

Tras cada uno de los procedimientos que se introducen a continuación, se encuentran diversas tecnologías que permiten obtener los valores asociados a cada variable concreta.

Las tecnologías o principios físicos más comunes que se utilizan para para llevar a cabo estas mediciones son las siguientes, que se encuentran descritas con detalle en [39]:

- **Inducción en lazo abierto**
- **Radar o microondas**
- **Infrarrojos**
- **Ultrasonidos**
- **Captador magnético**
- **Bluetooth**
- **Visión artificial**

Como se ha indicado previamente, el método empleado en el presente trabajo para la obtención de estas variables, se utilizan técnicas basadas en visión artificial.

A continuación, se describen las técnicas de medida según el criterio establecido.

a) Desde un punto fijo

Como su nombre indica la medición se realiza desde una posición fija del observador siendo uno de los procedimientos más antiguos que se utilizan para estudiar el comportamiento del tráfico. El propio observado anota directamente los vehículos que pasan por un determinado punto.

Más adelante aparecieron sistemas de tubo neumáticos como detectores que atravesaban la vía, posteriormente, se utilizaron los sistemas de lazo inductivo, siendo muy utilizados en la actualidad, o sistemas más recientes como son, los infrarrojos, radares y cámaras de vídeo.

Las medidas directas que se pueden realizar mediante la observación de punto fijo son **flujo** y **volumen** o el número de vehículos en la vía de estudio, así como el intervalo de tiempo entre vehículos, la velocidad local o velocidad media temporal y la media del tiempo de ocupación de la vía.

No obstante, una de las variables que no se pueden determinar utilizando este procedimiento es la densidad del tráfico, puesto que no tiene sentido para la medida desde un punto fijo debido al desconocimiento de unidades de longitud en el escenario de cómputo a la hora de realizar la medida. Por tanto, para conocer la densidad será necesaria la utilización de la ecuación fundamental del tráfico.

b) En una sección corta

Como sección corta se define un tramo con una longitud inferior a 10m [38]. Bajo esta consideración se utilizaron tubos neumáticos paralelos colocados uno cerca de otro dentro del margen de distancia definido por el concepto de sección corta. Aunque se sigue utilizando esta tecnología, actualmente son más populares los lazos inductivos, si bien es cierto que también se utilizan tecnologías de microondas, infrarrojos y visión artificial.

Las variables de tránsito que pueden ser medidas directamente por este procedimiento son velocidades, volúmenes de vehículos e intervalos de tiempo entre vehículos. Por otro lado, debido a la existencia de “medidores continuos” (ej: los lazos inductivos) y a que proporcionan una lectura continua, es posible determinar la ocupación de la vía.

c) En una sección larga

Una sección de vía larga, se entiende aquella con una longitud superior a 0,5km [38]. De esta forma se puede obtener información del tráfico en un tramo concreto de la vía y en un instante determinado. La tecnología habitual utilizada con este procedimiento es la fotografía aérea [38] en diferentes instantes de tiempo. La principal variable que se puede medir con este procedimiento es la densidad.

d) Observador en movimiento.

El procedimiento de observador en movimiento se plantea con dos enfoques, uno como observador móvil y otro como vehículo flotante. El primero de los enfoques consiste en un observador móvil desplazándose a velocidad constante en un cierto tramo contando vehículos. Éste puede desplazarse en cualquiera de los sentidos con respecto al flujo, incrementándose el número de vehículos a contar si se circula en sentido contrario al flujo.

En el caso de vehículo flotante, el vehículo se incorpora al flujo de vehículos contando la diferencia entre vehículos que adelantan menos los que son adelantados.

e) De grandes áreas con ITS

Las zonas en las que se utiliza este procedimiento requieren de sistemas inteligentes de transporte (ITS), lo cual implica que los vehículos de dichas zonas están especialmente equipados con sistemas de comunicación para poder transmitir la información.

Con los avances de los nuevos conceptos como el de "*smart cities*", o la comunicación 5G esta metodología adquiere mayor relevancia por cuestiones de conectividad, por lo que la adquisición de información de variables relativas al tránsito de vehículos, son proporcionadas por los propios vehículos gracias a tecnologías de comunicación de alta velocidad como el 5G.

5.3 Tecnología de medida por visión artificial

A lo largo del presente trabajo se han presentado técnicas basadas en visión por computador con las que poder realizar tareas de identificación de vehículos y clasificación de los mismos, en este apartado se describe la forma de realizar la identificación de vehículos mediante las técnicas descritas y el proceso de conteo de vehículos y la obtención de variables de tránsito.

El procedimiento utilizado es de punto fijo, enfocando una sección de la vía que se define sobre cada frame del video, no necesariamente centrada, pero sí en un único sentido de circulación.

El procedimiento considera sólo los vehículos identificados en el campo visual, cuyo centroide ha pasado por una determinada región acotada dentro del mencionado campo visual.

Esta región se puede justificar en base a la velocidad de la vía donde se ha presupuesto que todo vehículo identificado circula dentro de un rango de velocidades estipulado por el valor ΔV. Para estimar el espacio que define esta región de conteo, necesitamos tener en cuenta la frecuencia con la que la cámara toma imágenes T, de este modo $\Delta x = \Delta V . T$.

Por otro lado, la correlación espacio real-espacio píxel es un campo bien estudiado [4] que permite realizar un calibrado del sistema con respecto a una vía.

Para un modelo simplificado en el que se parte de la condición de una visualización de la vía desde su perpendicular a una altura determinada es posible encontrar la relación gracias a una homotecia, es decir un simple cambio de escala, por lo que $\Delta x_p = k\Delta x$, donde k > 0. Este factor de escala puede determinarse con unas secciones calibradas en terreno real Δx y posteriormente comparadas con la dimensión en píxel registrada en la imagen Δx_p para dicha sección real.

En el caso bajo estudio, se trata de un modelo muy simplificado de calibración del sistema con los parámetros extrínsecos, que considera relaciones geométricas, y los intrínsecos, íntimamente relacionados con las cámaras.

Existen multitud de metodologías de calibración como TSAI, calibración por puntos coplanares, Song de Ma, entre otros, algunos de estos métodos se describen en [4].

En el presente trabajo se toman como secciones de conteo la región definida por las rectas separadas 10 unidades de píxel trazadas a 1/3 respecto del alto de la imagen a partir de la base de ésta del tamaño del campo visual.

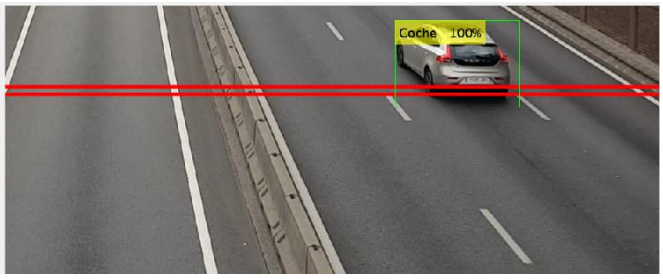


Figura 57: Ejemplo de región de conteo acotado por líneas rojas.

En la figura 57 se observan las líneas de conteo junto con una imagen donde se ha identificado un vehículo en movimiento.

En la figura 58 se muestra un ejemplo ilustrativo de conteo de vehículos y tiempo transcurrido utilizando un vídeo a 30 frames por segundo. El diagrama de barras de la parte inferior muestra el número de clases contabilizadas durante el tiempo indicado.

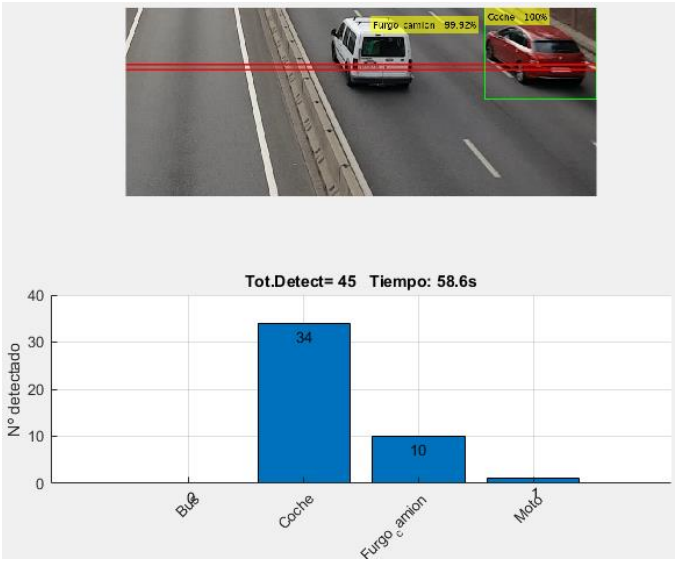


Figura 58: Conteo y clasificación

Todo elemento móvil identificado* en la figura 58 es un vehículo en movimiento definido por su centroide.

** Se especifica identificado ya que en una primera fase de identificación de regiones puede darse el caso de detectar movimiento de otros objetos en la imagen producidos por perturbación de la cámara. Tras el proceso de clasificación de regiones estas imágenes son identificadas como fondo y no se contabilizan.*

Dado que el flujo vehicular se mide en vehículos/hora, utilizamos la ecuación (41) y al realizar la conversión segundos-hora obtenemos el ejemplo de la tabla 9:

Elemento	**Flujo [veh/h]**
Vehículos total	2764.5
Bus	0
Coche	2088.7
Furgo-camión	614.33
Moto	61.43

Tabla 9: Ejemplo de tabla de flujo de vehículos

La clasificación se ha realizado con una CNN con arquitectura de AlexNet, e identificación de movimiento por SSIM. En el capítulo de resultados se realiza una comparación exhaustiva entre las diferentes técnicas mencionadas en este trabajo.

Con los datos anteriores se pretende mostrar al lector el procedimiento utilizado para la obtención de la variable vehicular de flujo de vehículos.

Con respecto al resto de variables definidas en relación a los vehículos, dado que es necesario un proceso de calibrado del sistema de visión para obtener la relación Δx, anteriormente mencionada, en lo que respecta al presente trabajo simplemente se mencionan sus posibilidades por la ausencia de tal calibrado en las autopistas donde se han obtenido los vídeos.

CAPÍTULO 6: RESULTADOS

En este capítulo se describen los resultados obtenidos mediante las técnicas aplicadas, así como un resumen de las diferentes metodologías implementadas.

6.1 Consideraciones preliminares

Partiendo del objetivo principal de la obtención del flujo de vehículos en una vía, en la figura 59 se muestra un esquema a alto nivel del funcionamiento del sistema implementado.

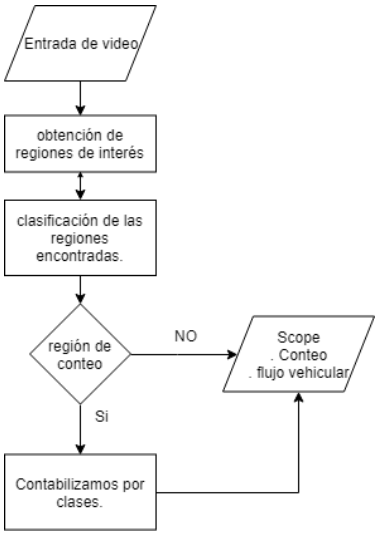


Figura 59: Arquitectura general del proceso del sistema de flujo vehicular

En la aplicación desarrollada, el interfaz general es el que se muestra en la figura 60, cuyo contenido es similar al indicado en la figura 58.

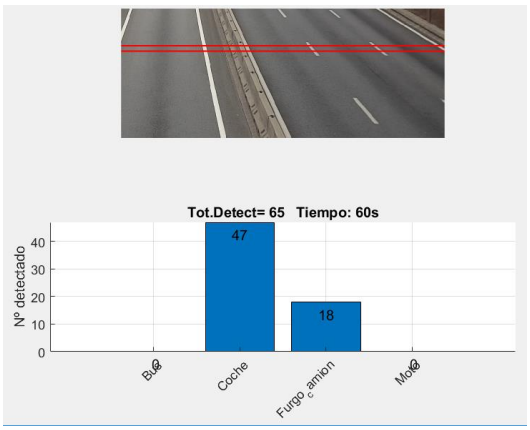


Figura 60: Interfaz general de salida

En la parte superior del interfaz, aparecen las imágenes que componen el vídeo y las regiones de conteo (líneas rojas). En la parte inferior se muestra un diagrama de barras donde aparecen contabilizadas las diferentes clases, el conteo general y el tiempo utilizado teniendo en cuenta que el vídeo está obtenido a razón de 30 fps.

Finalmente, el sistema proporciona el flujo vehicular total expresado en unidades de [veh/h], tal y como se indica en el capítulo cinco.

Conviene reseñar, no obstante, que el objetivo principal del presente trabajo se enfoca en la identificación y clasificación de vehículos, habiendo puesto el mayor énfasis en este aspecto.

Para resumir todos los procesos con los que se ha trabajado, la figura 61 muestra un esquema donde aparecen todos los métodos utilizados para las tres arquitecturas de red indicadas en el capítulo tres.

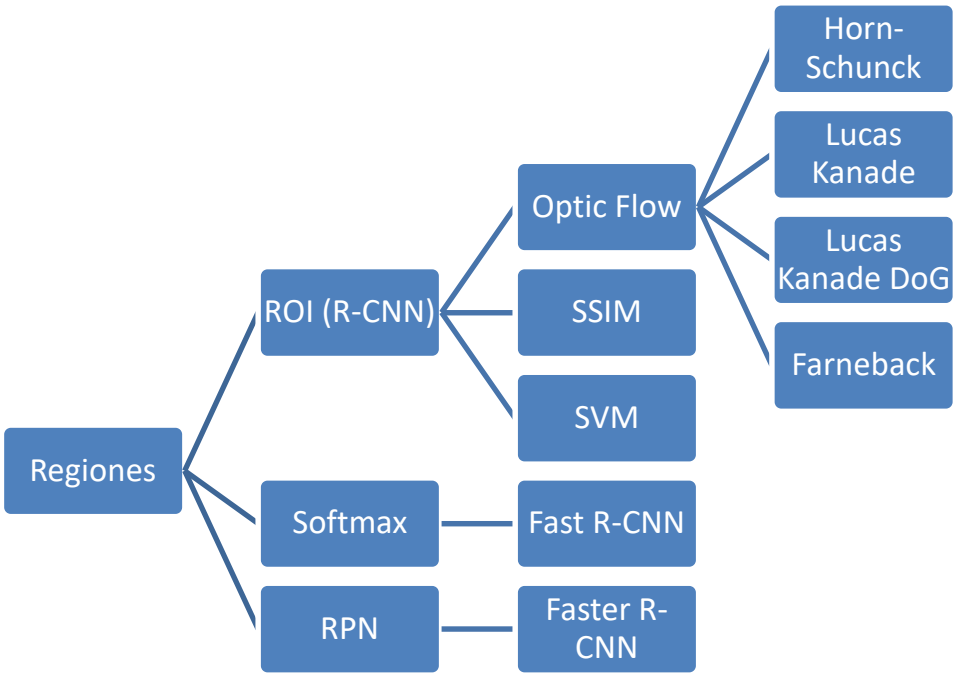


Figura 61: Métodos utilizados para la detección y clasificación

Una vez determinado y clasificado el elemento detectado solo se contabiliza si está presente dentro de una región determinada como se indica en el capítulo cinco. Esto aparece reflejado en el diagrama general de la figura 61.

Tras el entrenamiento de las redes, a continuación, se muestran y analizan los resultados de clasificación obtenidos y estructurados en tres fases: a) detección y clasificación de vehículos con las arquitecturas indicadas; b) cómputo del flujo de vehículos y c) análisis de casos excepcionales y singulares.

Para el análisis de resultados se han utilizado 10 vídeos capturados en entornos reales con un dispositivo móvil tipo Samsung Galaxy 7, con una duración promedio cada uno de 2', de forma que en total hace un total de 20' de vídeo. Teniendo en cuenta que la velocidad de captura es de 30 fps, esto supone un total de 36000 imágenes disponibles en total. Si bien para el análisis de los resultados se han seleccionado aleatoriamente 20 tramos de las secuencias con una duración cada uno de 2

segundos de duración, de esta forma en cada tramo se analizan 60 imágenes y en total 1200 imágenes. El hecho de considerar tramos de vídeo es debido a la necesidad de cómputo del flujo óptico, lo que sólo es posible con imágenes consecutivas en el tiempo.

Para el entrenamiento de las redes se han utilizado 130 ejemplos para cada una de las categorías utilizadas, a saber: bus, coche, furgo-camión y moto.

En el Anexo I se especifican y describen los recursos utilizados durante los diferentes experimentos.

En la tabla 10 se muestra un resumen de validación de entrenamiento de las redes que se indican.

Arquitectura	**CNN**	**R-CNN**	**Fast R-CNN**	**Faster R-CNN**
Alexnet [%]	100	96.88	98.44	…..
Resnet50 [%]	100	100	85,94	84,38
Vehicle [%]	100	98.99	99,37	92.97

Tabla 10: Resumen de validaciones de redes

6.2 Resultado detección y clasificación de vehículos

En este apartado se muestra el resultado obtenido en la clasificación de diferentes imágenes con las tres arquitecturas de CNN utilizadas. Posteriormente se muestran resultados al aplicar los diferentes detectores compuestos por las arquitecturas de las CNN, antes imágenes donde aparecen elementos clasificables.

En la figura 62 se muestra un ejemplo del interfaz de clasificación. En la parte superior aparece la imagen a clasificar con la asignación a la clase con mayor probabilidad de pertenencia y la probabilidad obtenida en la clasificación para dicha clase. En la parte inferior se muestra un diagrama de barras con las probabilidades de pertenencia a cada una de las clases que dispone el clasificador.

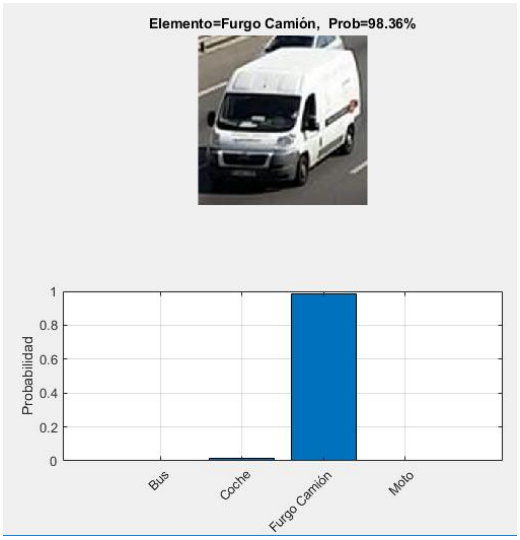
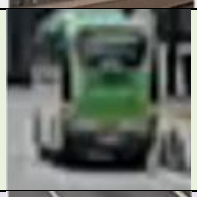
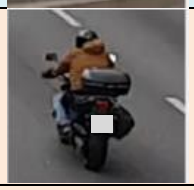


Figura 62: Ejemplo de interfaz de clasificación

En la tabla 11 se muestra un conjunto de imágenes clasificadas con las tres arquitecturas de CNN y las probabilidades de pertenencia a cada una de las clases. Se trata de un conjunto de imágenes seleccionadas de forma aleatoria del conjunto de imágenes disponibles. Se puede observar cómo en todos los casos las probabilidades máximas coinciden con la imagen en cuestión. Este proceso de determinar la probabilidad máxima de pertenencia para asignar una etiqueta es lo que realiza la capa de salida de las CNN como consecuencia del paso por la capa *softmax*.

		Arquitectura		
Imagen	**Clases**	**AlexNet**	**Resnet50**	**Vehicle**
	**Bus [%]**	**100**	**100**	**99.9996**
	Coche [%]	0	0	0
	Camión-Fur. [%]	0	0	0
	Moto [%]	0	0	0.0004
	**Bus [%]**	**100**	**99.9001**	**99.9694**
	Coche [%]	0	0.0178	0
	Camión-Fur. [%]	0	0.0784	0.0001
	Moto [%]	0	0.0029	0.0305
	**Bus [%]**	0	0.0001	0
	Coche [%]	**100**	**99.9894**	**99.9964**
	Camión-Fur. [%]	0	0.0105	0.0036
	Moto [%]	0	0	0
	**Bus [%]**	0	0.0001	0
	Coche [%]	**100**	**99.9426**	**99.4188**
	Camión-Fur. [%]	0	0.0572	0.5812
	Moto [%]	0	0.0001	0
	**Bus [%]**	0	0.0004	0
	Coche [%]	**100**	**98.6244**	**99.9936**
	Camión-Fur. [%]	0	1.3751	0.0061
	Moto [%]	0	0.0001	0
	**Bus [%]**	0	0	0
	Coche [%]	0.0001	0.0058	0.1428
	Camión-Fur. [%]	**99.999**	**99.9942**	**99.8572**
	Moto [%]	0	0	0
	**Bus [%]**	0	0.0086	0
	Coche [%]	0	0.0003	0.0002
	Camión-Fur. [%]	**100**	**99.991**	**99.9998**
	Moto [%]	0	0	0
	**Bus [%]**	0	0.0005	0.1440
	Coche [%]	0	0.0006	0
	Camión-Fur. [%]	0	0.0009	0
	Moto [%]	**100**	**99.9981**	**99.85**
	Bus [%]	0	0	0.4601

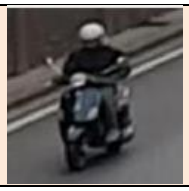
	**Coche [%]**	0	0	0
	Camión-Fur. [%]	0	0	0
	Moto [%]	**100**	**100**	**99.5399**

Tabla 11: Ejemplos de clasificación con cada una de las arquitecturas de CNN

Utilizando la tabla 11 como ejemplo de validación gráfica de las diferentes redes, en la tabla 12 se muestra una misma escena para la determinación de los vehículos y clases en movimiento que proporcionan las diferentes arquitecturas de reconocimiento de regiones y clasificación, incluyendo en este grupo las correspondientes a SSIM-CNN y FlujoOptico-CNN como una arquitectura de R-CNN.

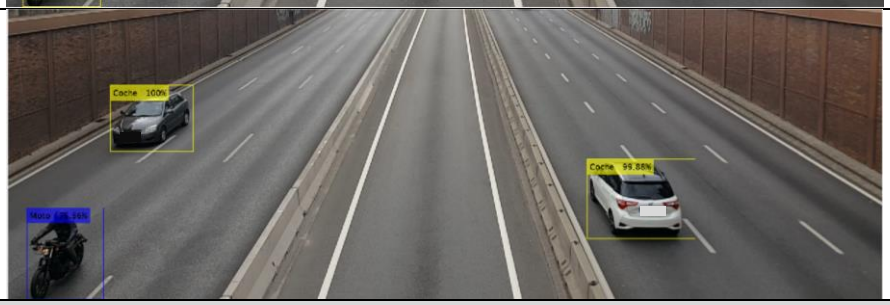
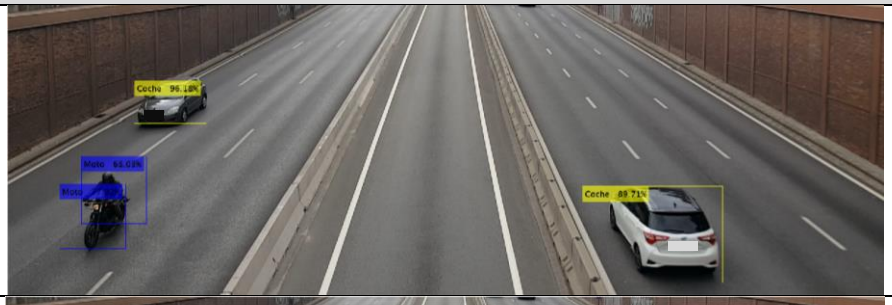
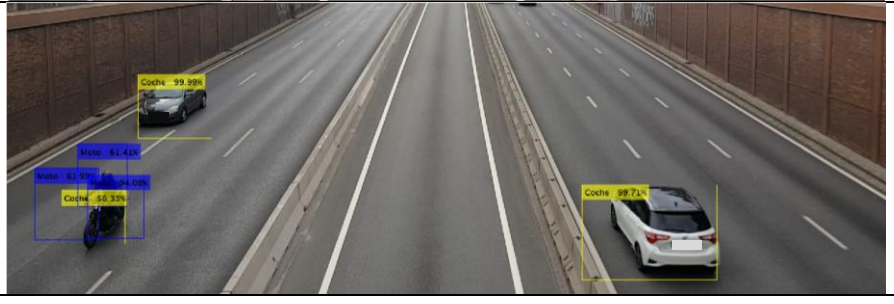
Puesto que tanto para la determinación del flujo óptico como para el SSIM se necesita una consecución de imágenes, tal y como se indicó en el capítulo 2, para el análisis de los resultados mostrados en la tabla 12 se han utilizado 2 frames contiguos en el tiempo, mientras que para las R-CNN, Fast R-CNN y Faster R-CNN y se utiliza solo el primero de los dos frames indicados anteriormente.

El total de detectores implementados en este trabajo es de 24, 12 asociados a flujo óptico (3 arquitecturas de CNN y 4 métodos de determinación de flujo óptico), 3 por el método SSIM, 3 arquitecturas de R-CNN, 3 arquitecturas de fast R-CNN y finalmente 3 arquitecturas de faster R-CNN.

Para el detector FlujoOptico-CNN se utilizó el método de Lucas Kanade (LK) en base a la resolución obtenida en la tabla 8.

Obsérvese cómo para determinados modelos de red no se han obtenido resultados satisfactorios en términos de clasificación (resultados por debajo del 10% de aciertos), lo que se indica como inefectiva en la tabla 12. Esto significa que el modelo correspondiente necesita un mayor análisis con el fin de conseguir posibles mejoras sobre él.

SSIM-CNN	
AlexNet	
ResNet50	

Vehicle	
Flujo óptico (LK)-CNN	
AlexNet	
ResNet50	
Vehicle	
R-CNN	
AlexNet	
ResNet50	

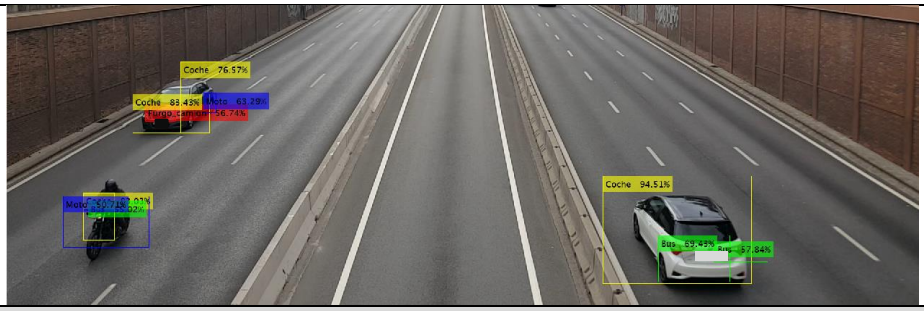
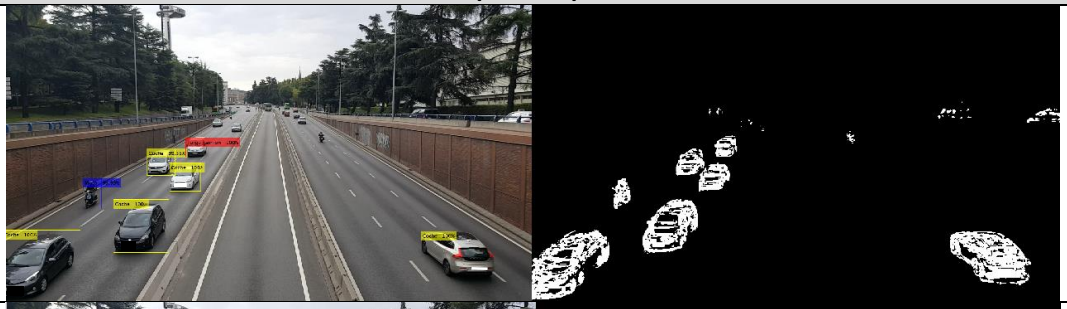
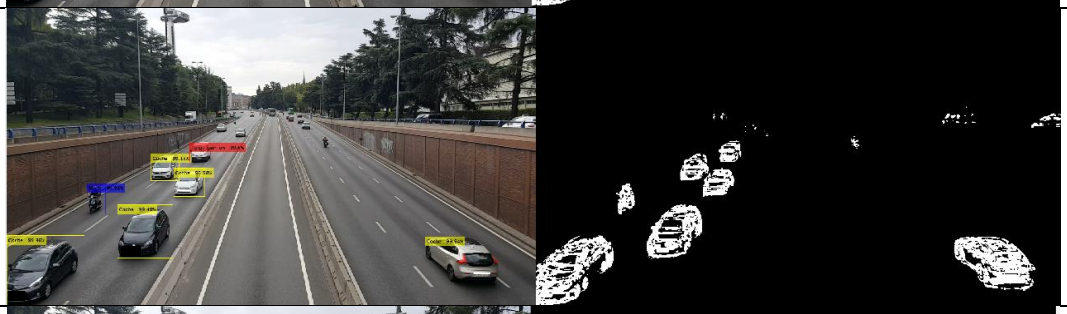
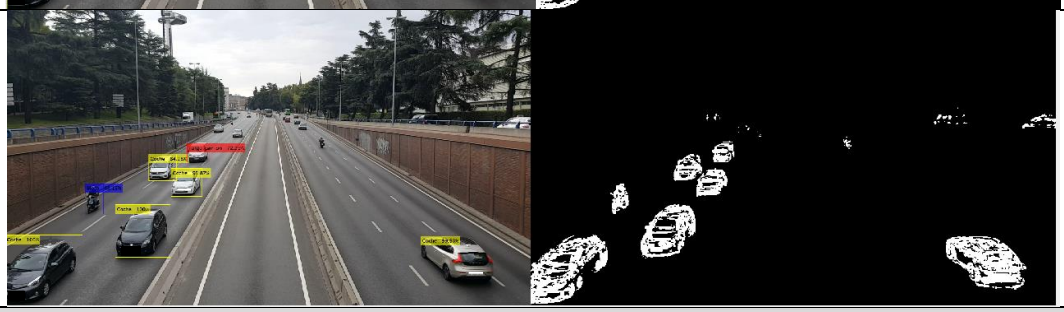
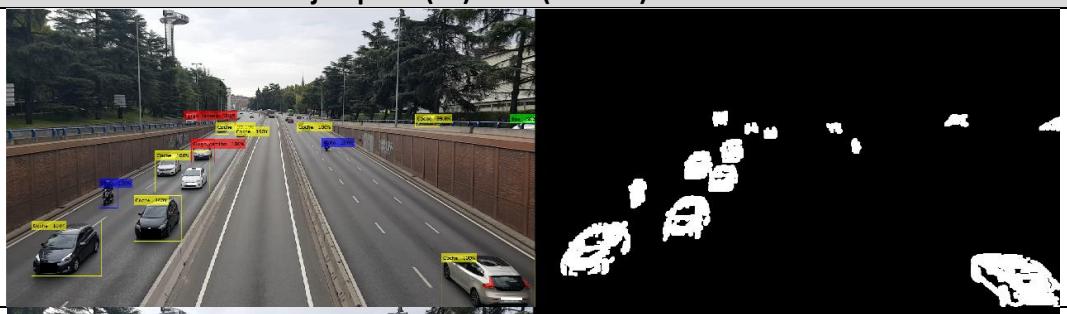
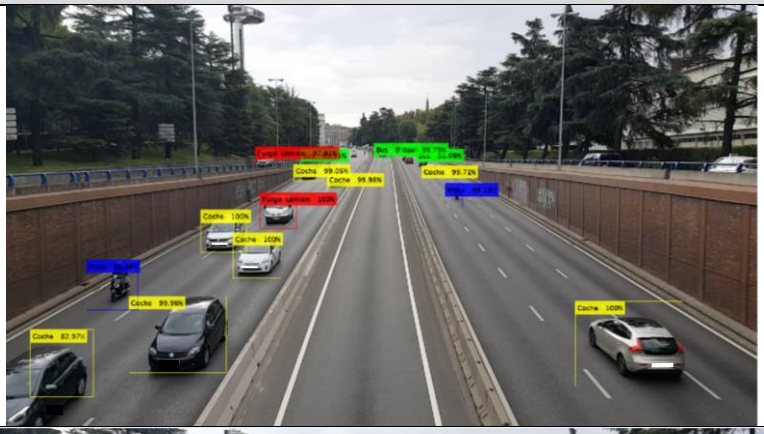
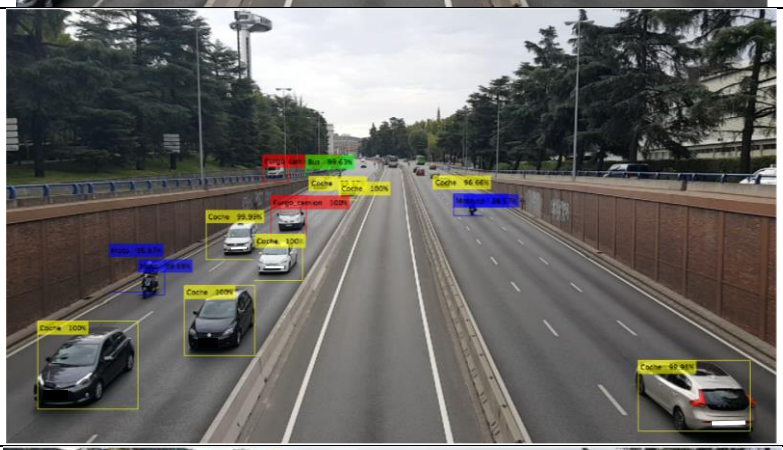
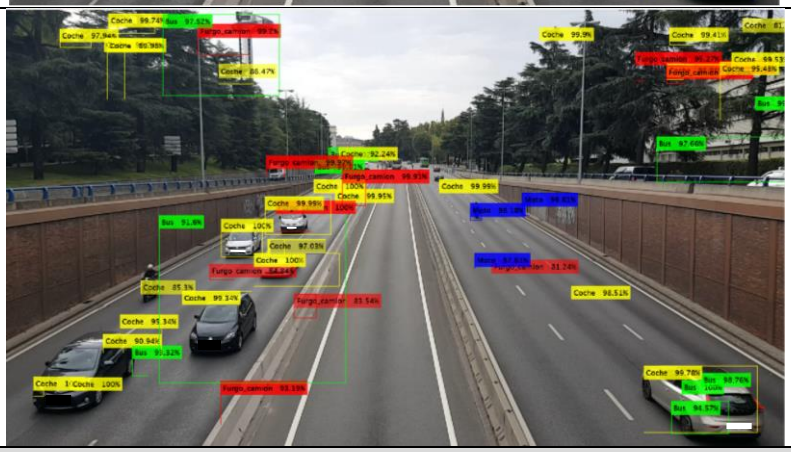
Vehicle	
Fast R-CNN	
AlexNet	--Inefectiva--
ResNet50	--Inefectiva--
Vehicle	
Faster R-CNN	
AlexNet	--Inefectiva--
ResNet50	--Inefectiva--
Vehicle	

Tabla 12: Comportamiento de los diferentes detectores ante una misma escena controlada*

En la tabla 12 se indica que la escena es controlada. Entendemos por escena controlada aquella en la que las áreas de las regiones en movimiento entran dentro de los márgenes especificados tal y como se describen en el capítulo 2, para eliminar ruido en el proceso de segmentación.

En la Tabla 13 se muestra el comportamiento de los detectores-clasificadores ante situaciones de escenas complejas, donde entendemos por escena compleja aquella en la que aparece una acumulación de objetos en la escena, donde no aparecen apenas espacios entre ellos y en el caso de los vehículos con circulando en paralelo.

SSIM-CNN (a=1000)	
AlexNet	
ResNet50	
Vehicle	
Flujo óptico (LK)-CNN (a=1000)	
AlexNet	
ResNet50	

Vehicle	
R-CNN	
AlexNet	
ResNet50	
Vehicle	
Fast R-CNN	

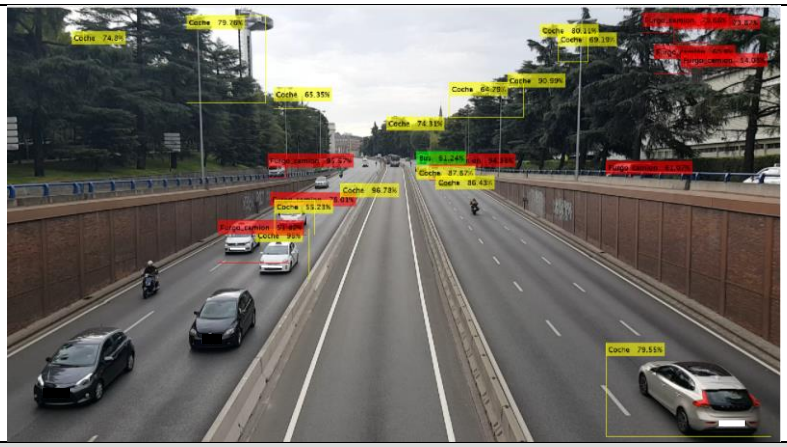
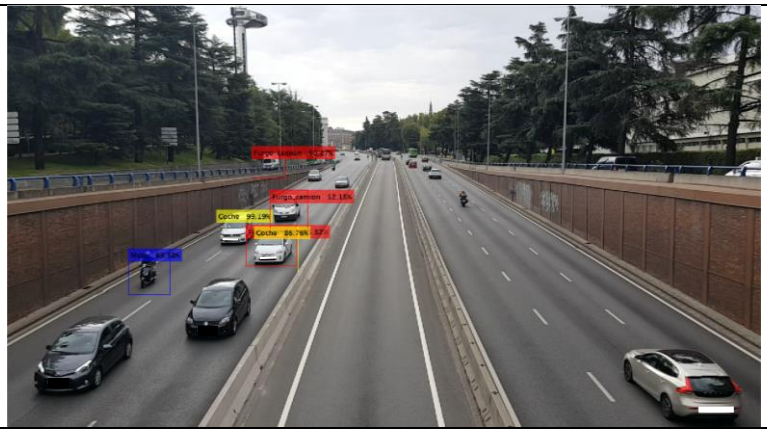
AlexNet	
ResNet50	--Inefectiva--
Vehicle	
Faster R-CNN	
AlexNet	--Inefectiva--
ResNet50	--Inefectiva--
Vehicle	

Tabla 13: Identificación de clases en escena compleja

A partir de los resultados mostrados en la tabla 13 se deduce en términos generales el desempeño de los diferentes detectores, donde podemos destacar el buen comportamiento de los SSIM-CNN y flujoOptico-CNN, donde tras variar el margen del filtro de áreas menores detectables “a”, podemos reconocer vehículos en la lejanía. En el caso de los detectores FlujoOptico-CNN se pueden observar cómo si no se controla la dilatación morfológica, ésta puede producir solapamientos. Esto aparece bien reflejado en el apartado 6.4 de casos excepcionales observados. En cuanto a las R-CNN destaca el buen comportamiento en las redes AlexeNet y Resnet50 frente a Vehicle, pues muestra varios falsos positivos.

Con relación a las arquitecturas fast y faster destacaremos el comportamiento de la arquitectura Vehicle frente a las de AlexNet y ResNet50, dado que estas dos últimas no han logrado identificar regiones, mientras que Vehicle mejora su comportamiento de R-CNN para el caso de faster R-CNN.

El hecho de que estas dos redes no muestren resultados en su forma fast y faster, puede ser consecuencia de dos factores principales, que a su vez están relacionados entre sí. Por un lado, el método de selección de regiones y por otro dónde inyectarlas. La profundidad de las redes, especialmente en las redes muy profundas como ResNet50, puede ser la causa de que las capas donde inyectamos las propuestas de regiones ROI no sea la más adecuada o en el caso de las Faster, que la arquitectura de la RPN, no adquiera características claves de regiones y por consiguiente no genere buenas propuestas de regiones, o en el caso de sí hacerlo, éstas no se inyecten en una posición adecuada de la estructura de red.

6.3 Resultados del flujo de vehículos

Con el objetivo de validar y comparar los diferentes métodos, a continuación, se muestran los resultados obtenidos. La comparación se lleva a cabo seleccionando datos de forma manual tal y como se indica en la tabla 14. En esta tabla se muestra el flujo de vehículos para las categorías indicadas, obtenido manualmente y medido en número de vehículos/hora (veh/h). Esto se ha realizado sobre una sección de vídeo de 60s.

Flujo de vehículos determinado manualmente [veh/h]					
Clases	**Bus**	**Coche**	**Camión-furgo**	**Moto**	**Tot.**
Flujo	120	2040	660	60	2880

Tabla 14: Flujo de vehículos manual.

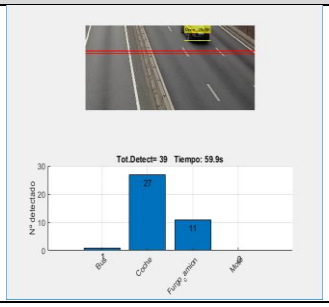
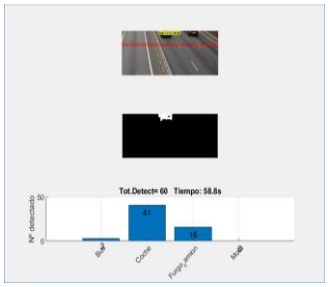
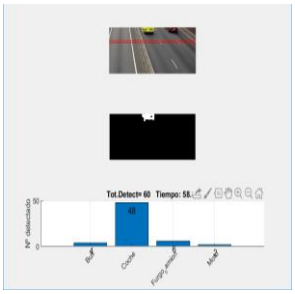
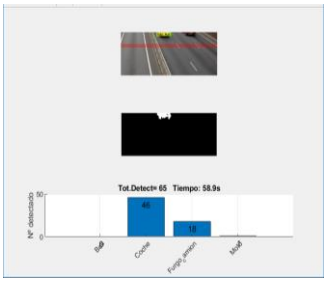
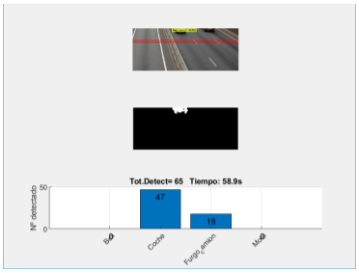
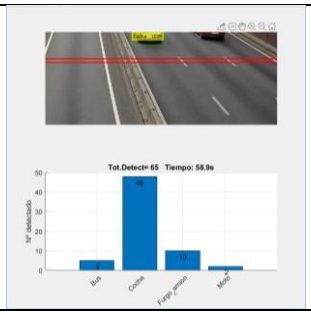
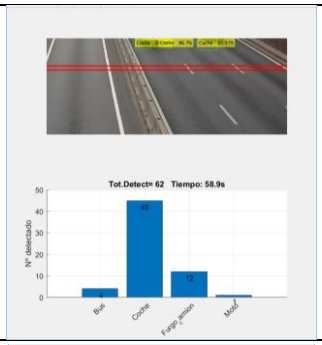
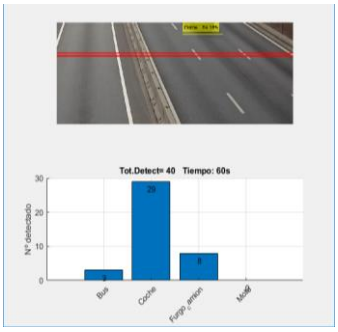
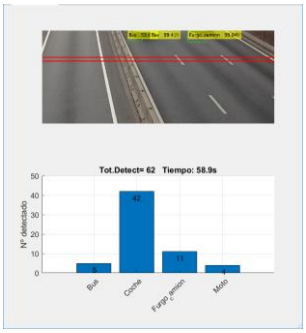
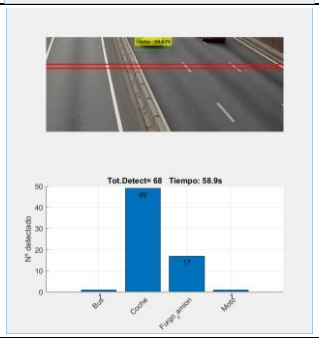
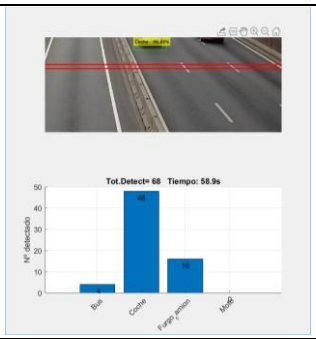
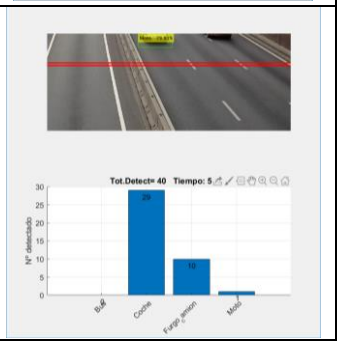
Se ha procesado el mismo vídeo de 60s bajo los diferentes métodos estudiados, los resultados obtenidos del conteo, clasificación y flujo de vehículos aparecen resumidos en la tabla 15. El flujo de vehículos se indica en [veh/h] y se añade una columna con el tiempo de cómputo empleado en segundos (s). El sistema empleado para el cómputo se indicó en el apartado 6.1 de consideraciones preliminares y puede consultarse en el anexo I de este trabajo.

		AlexNet		**Resnet50**		**Vehicle**	
		Flujo	**Time P.**	**Flujo**	**Time P.**	**Flujo**	**Time P.**
R-CNN	**Optic flow (Hs)**	3600	435.43	3600	543.82	3600	304.57
	Optic flow (LK)	3900	329.40	3900	516.81	3900	375.09
	O. flow (LK-DoG)	3720	772.98	3720	1171.59	3720	960.83
	O.flow (Far)	4080	673.06	4080	1094.61	2400	775.23
	SSIM	2880	392.86	2880	556.07	2880	333.37
	SVM	-	-	-	-	-	-
Fast R-CNN		-	-	-	-	-	-
Faster R-CNN		-	-	-	-	-	-

Tabla 15: Flujo de vehículos y tiempo cómputo para diferentes métodos

En la tabla 15 aparecen diferentes líneas donde no se ha indicado el flujo de vehículos. Esto es consecuencia del tiempo de cómputo asociado al procesado utilizando el sistema indicado en el Anexo I. Este problema es bien conocido, y en el mundo industrial se utilizan GPUs que aceleran considerablemente el proceso, con vista a implementarse en tiempo real.

Los resultados mostrados en la tabla 15 se obtienen gráficamente utilizando el interfaz y contando los vehículos que atraviesan las líneas marcadas, que se resumen en la tabla 16.

Método	AlexNet	Resnet50	Vehicle
HS			
LK			
LK-DoG			
Franeback			

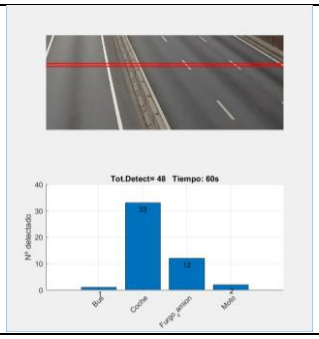
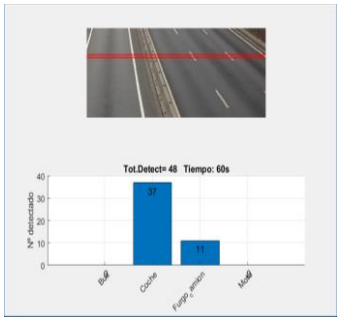
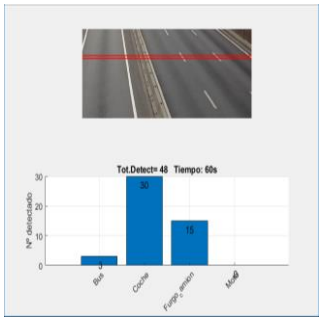
SSIM			

Tabla 16: Resultados gráficos de conteo de vehículos

6.4 Casos excepcionales observados

A partir del análisis de resultados previo se han puesto de manifiesto una problemática, existente en la realidad, que se valora a continuación. Se puede decir que la mayoría de los problemas encontrados son consecuencia de la identificación de regiones por las tres causas principales siguientes: a) solapamiento de dos regiones que representan a dos vehículos diferentes por efecto de la dilatación morfológica, obteniéndose una única región; b) superposición de dos vehículos que circulan a la misma velocidad bien en paralelo o uno seguido de otro; c) existencia de sombras, que al moverse solidariamente con cada vehículo alcanzan a otro, aun estando separados entre sí.

a) Solapación por dilatación **b) Superposición en escena** **c) Sombras**

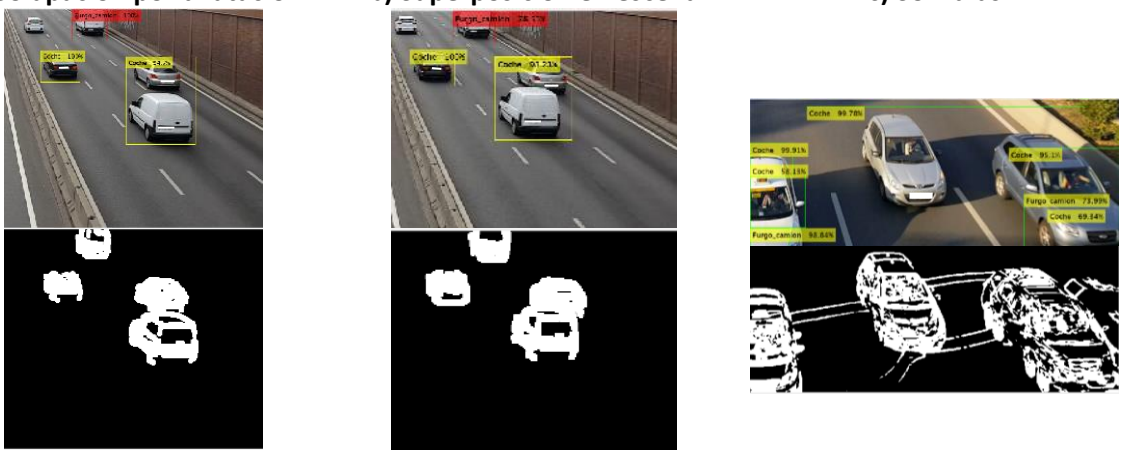


Figura 63: Tres casos excepcionales

La principal diferencia entre las escenas de la figura 64 a) y b) reside en que la primera se han unido las imágenes por la dilatación introducida tras la binarización del flujo óptico, mientras que la b) se produce por superposición de la escena y la segmentación no lo puede diferenciar. Para el caso c) se muestra el efecto de la sombra sobre los vehículos dificultando la labor de segmentación y clasificación.

Otro efecto en relación con los anteriores es la discriminación de áreas por solapamiento con sombras. Es decir, una sombra asociada a un vehículo une a dos o más vehículos distintos de la imagen a clasificar, si esta área supera los márgenes introducidos en el capítulo dos, esta área se discrimina. En la figura 64 se muestra un ejemplo de esta superposición de sombras discriminada por el filtro de

áreas introducido en el capítulo dos, recordando que este filtro de áreas discrimina áreas mayores que un determinado valor constante “b” y menores que otro determinado valor “a”. La obtención de dichos valores son resultado de un análisis de las escenas habituales en el lugar de captura del vídeo, por lo que podemos estandarizar localmente el tamaño de las áreas de las regiones clasificables.

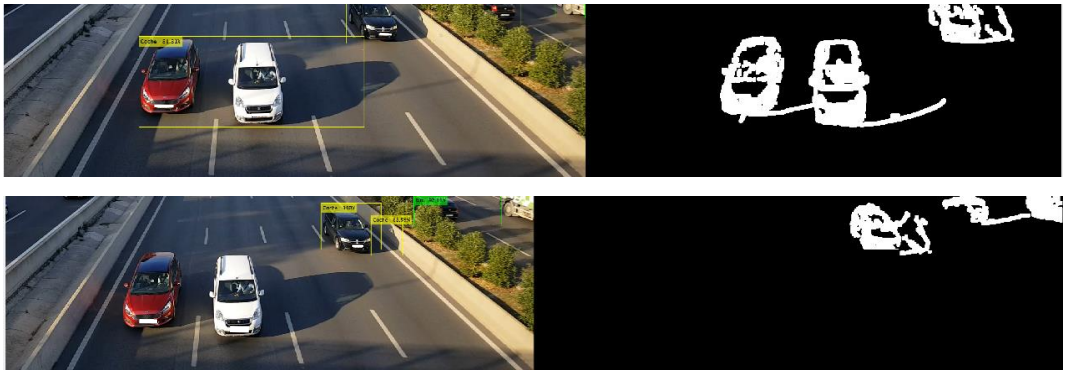


Figura 64: Discriminación de áreas grandes por superposición de sombras

Otros problemas encontrados provienen de la captura de las imágenes bajo proyección de perspectiva. En efecto, dado que, desde una posición cercana de la cámara con respecto a los vehículos en movimiento, puede suceder que al perder el contexto, un vehículo grande y “liso” en lo que se refiere a una textura homogénea, como podría ser el techo de un autobús, no se perciben como una región de interés, provocando que no se identifique su correcta clasificación en relación a las distintas clases disponibles. Una perspectiva más alejada contendría un contexto con suficientes características que permitirán su identificación.



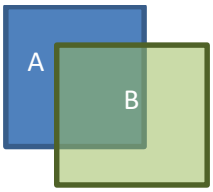
Figura 65: Perdida de contexto en clasificación

Debido a la mencionada proyección de perspectiva, regiones con forma rectangular en la escena se proyectan en la imagen 2D de forma trapezoidal, haciendo que las partes de la región próximas a la cámara aparecen con anchos mayores que las partes alejadas.

Esta situación puede resolverse mediante la calibración apropiada del sistema de captura de imágenes, que comprende tanto la propia cámara como la óptica asociada [4].

En el caso de los detectores R-CNN en sus tres formas (R, fast y faster) en ocasiones aparece solapamiento de regiones. En la figura 67 se muestran sendos casos de solapamiento de regiones utilizando una R-CNN. En el primero de ellos (a), se muestra como se identifican dos regiones sobre el mismo vehículo asociando ambas a la misma clase y acertando en la clasificación. En el caso (b) se muestra un ejemplo de identificación de dos regiones sobre el mismo vehículo identificándolo como dos clases diferentes, produciéndose un fallo en este caso.

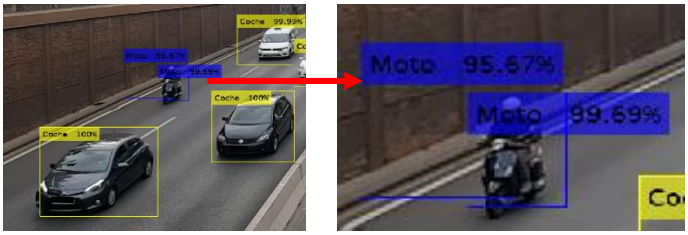
Este efecto puede ser ajustado mediante un radio de superposición [6] donde dos regiones pueden ajustarse bajo los siguientes criterios:



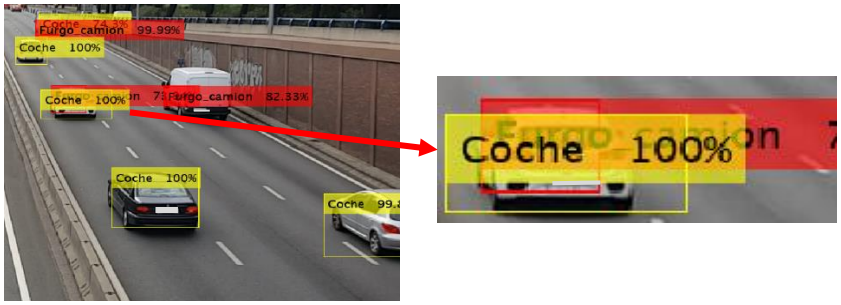
$$Unión: \frac{Area(A \cap B)}{Area\ (A \cup B)} \quad (48)$$

$$Min: \frac{Area(A \cap B)}{\min(\ area(A),\ area(B))} \quad (49)$$

El radio de superposición toma un valor en el rango de 0 y 1, donde 0 significa que no existe solapamiento (no se intersectan las áreas) y 1 que están completamente superpuestas. Este valor dependerá del criterio seleccionado (Unión o mínimo) y, por consiguiente, como regla general, el resultado será diferente según el criterio utilizado.



Caso a) misma clase



Caso b) diferentes clases

Figura 66: Ejemplo doble identificación

Otro efecto observado en algunos casos con las arquitecturas R-CNN y asociadas con fast y faster, es que al descontextualizar las escenas donde se desea identificar regiones aparezcan falsos positivos. Un ejemplo de este efecto se muestra en la figura 67 donde se han identificado y clasificado dos falsos positivos en la escena, que aparecen en la parte superior de la escena.



Figura 67: Falsos positivos en regiones y clasificación

CAPÍTULO 7: CONCLUSIONES Y TRABAJO FUTURO

En relación a los diferentes temas abordados en este trabajo y conclusiones parciales obtenidas, a continuación se indican los aspectos más relevantes que merecen especial consideración en relación a los resultados obtenidos, así como posibles líneas de trabajo futuro.

7.1 Conclusiones generales

En cuanto a los objetivos específicos planteados en el capítulo 1, éstos se han alcanzado con resultados satisfactorios, lo que permite concluir que el trabajo desarrollado cumple con las expectativas planteadas para la prueba de concepto desarrollada, como paso previo a un posible producto comercial.

La propuesta plantea un desarrollo basado en diferentes módulos:

a) Movimiento

En el proceso de detección de movimiento por flujo óptico, una vez determinado éste por alguno de los cuatro métodos desarrollados, el mapa de vectores de flujo óptico se binariza en base al módulo de dichos vectores asociados a los píxeles. En este trabajo se ha optado por utilizar el módulo de dicho vector únicamente.

Dentro de la determinación de flujo óptico, se ha trabajado con cuatro métodos diferentes con los que se ha podido comparar el comportamiento del flujo óptico ante situaciones estándar de escenas con movimiento de vehículos y sin ellos. Esto ha permitido comprobar la robustez de cada uno de los cuatro métodos ante situaciones de ruido por movimiento de la cámara.

En el capítulo 2 se presentan dos métodos con los que obtener las regiones donde se ha producido movimiento. Ambos métodos tienen integrado un filtro de áreas que nos permiten discriminar regiones no válidas. Para poder asignar los márgenes de estos filtros es necesario realizar un estudio previo de los objetos en movimiento a identificar, por ello resulta determinante acotar las áreas de detección dentro de unos márgenes controlados. Sin este estudio previo de áreas medias de los objetos a detectar, las técnicas presentadas en el capítulo 2 pueden proporcionar a las siguientes fases de clasificación, un número inadecuado de regiones para el proceso de clasificación.

El método SSIM es generalmente utilizado para la comprobación de la “claridad” de señales transmitidas, del mismo modo que puede ser el PSNR, por lo que resulta muy interesante como aplicación para la detección de movimiento, dado que los resultados obtenidos en el capítulo 2 son

comparables a los de flujo óptico en cualquiera de los cuatro métodos, con la ventaja de la velocidad en el procesado con respecto a la obtención de flujo óptico.

Uno de los problemas encontrados en la determinación de las variables de tránsito de vehículos, es la estimación de velocidades de los vehículos. El flujo óptico proporciona dicha información en el espacio de píxeles y para poder realizar un cambio al espacio real, se requiere de una calibración del sistema que permita asignar la correspondencia de distancia en píxeles a distancia real. Esto no se ha podido realizar en el trabajo por la imposibilidad de acceder a las vías donde se tomaron los vídeos y por no poder fijar el sistema de captura (cámara) en unas condiciones controladas de posicionamiento.

En el proceso de conteo de vehículos, un problema observado en el método planteado, es el "efecto rebote", es decir doble conteo del mismo vehículo. Este efecto es principalmente producido por la falta de calibración del sistema.

b) Redes Neuronales

En el estudio de las capas de las CNN, resulta interesante mencionar el funcionamiento de la capa *softmax* que proporciona un grado de probabilidad de pertenencia a cada una de las clases a clasificar. Con este grado de probabilidad en las arquitecturas presentadas, se asigna una etiqueta concreta al objeto o región a clasificar a partir de la clase con mayor probabilidad de pertenencia.

Se ha trabajado con dos arquitecturas diferentes de CNN pre-entrenadas y una tercera de elaboración propia. Partiendo de estas tres se ha generado el conjunto de los experimentos de clasificación, resultando muy interesante a la vez que complicado el análisis a nivel dinámico de las capas.

En el apartado de las arquitecturas fast y faster CNN solo se han obtenido resultados satisfactorios para el caso de la arquitectura Vehicle (fast y faster) no siendo así con las arquitecturas de AlexNet y ResNet50. Esto es consecuencia de la dificultad asociada tanto a la implementación de las ROI y las RPN, como en qué punto de la estructura de la RNA se extraen características y se inyectan regiones. Esto requiere de un estudio más detallado para cada caso concreto, en el que se debe analizar a nivel de capas los canales que proporcionan características deseadas y por consiguiente de las capas de activación (ReLu) donde se observen grandes variaciones entre clases identificables y otras. Esto se hizo, obtenido mapas de características como los de la figura 68.

Dado que en cada proceso de entrenamiento para este tipo de arquitecturas y con el sistema expuesto en el Anexo I, los tiempos de cómputo se extienden entre 7 y 25 h para opciones simples de entrenamiento, no se ha dispuesto del tiempo necesario para adecuar la ubicación de las ROI y las RPN en las arquitecturas de AlexNet y ResNet50. En la figura 69 se muestra un ejemplo de análisis de la primera capa convolucional de AlexNet indicando el comportamiento de la capa de activación contigua para el canal con máxima respuesta a una imagen de entrada en fase de clasificación.

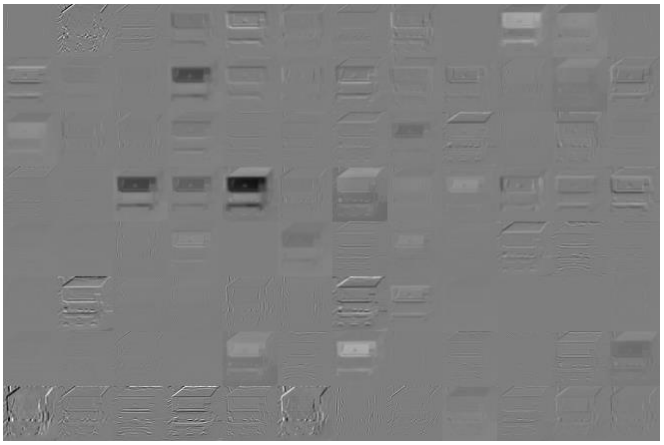


Figura 69: Mapa de características en primera capa convolucional AlexNet

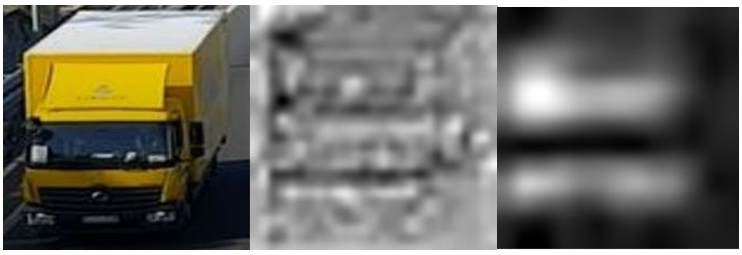


Figura 68: a) Imagen entrada; b) Canal con máxima activación; c) Imagen de activación del canal con máxima respuesta

Otro aspecto a estudiar en la estructura en la faster es el "*AnchorBoxes*" y en el caso de la fast la "*ROI Max pooling layer*", que pueden ser determinantes en el correcto funcionamiento de las RPN y las ROI.

Todos estos aspectos junto con los parámetros de entrenamiento, resultan críticos para el correcto comportamiento de estas arquitecturas.

Aun con todos los problemas que presenta el estudio de este tipo de arquitecturas, la red Vehicle fast y faster funcionan cumpliendo los principales propósitos para las que fueron diseñadas (reducción del volumen de datos de entrenamiento, minimizar el volumen del detector y acelerar el proceso de clasificación y detección para conjuntos de datos con pocas clases).

En lo referido a la base de datos generada para el entrenamiento de los clasificadores se puede concluir, en términos generales, que son aceptables. Esto se fundamenta en base a la correlación entre los datos de validación y el entrenamiento mostrados en las gráficas de entrenamiento.

A lo largo del trabajo se ha podido comprobar cómo para este caso concreto, dividir la arquitectura de detección de movimiento y clasificación proporciona el método más rápido en tiempo de cómputo. Dentro de los métodos de detección de regiones en movimiento utilizar el SSIM es más rápido que la detección de movimiento por flujo óptico, siendo ambos igual de efectivos.

7.2 Trabajo futuro

En este apartado se plantean diferentes líneas de trabajo futuro, especialmente relacionadas con las conclusiones obtenidas anteriormente, del mismo modo que en el apartado anterior subdividiremos esta sección en módulos:

a) Movimiento

Ampliar el sistema planteado en el trabajo con la diferenciación entre carriles permitiría proporcionar entre otros datos, mapas de densidad de tráfico por carriles, los cuales podrían ser buenas herramientas predictivas para el mantenimiento de la vía o la prevención de congestión, entre otras acciones. En la línea de la congestión de tráfico, esta información a tiempo real podría utilizarse para la implementación de sistemas de desvío de tráfico entre carriles, para homogeneizar el tráfico y poder utilizar modelos basados en fluidos estables y nos viscosos simplificando en gran medida los modelos.

Sería interesante obtener un vector de flujo óptico promedio de desplazamiento de los vehículos en la vía objeto de análisis, con el fin de sesgar los flujos ópticos detectados y de este modo suprimir las posibles oscilaciones que pueda sufrir el sistema óptico de captura de imágenes consecutivas, dado que solo se aceptarían el conjunto de vectores dentro de un rango de valores de módulo y ángulo. Para ello sería necesario utilizar métodos que resuelvan el problema de apertura como es el método de Horn-Schunck [4].

Trabajar sobre un sistema de autocalibrado que partiendo de la información registrada de vehículos circulando y condiciones de la vía, se ajustara a las necesidades concretas del entorno, no sólo permitiría la obtención de la velocidad y dirección de los vehículos en el espacio real, sino que mejoraría las regiones de conteo propuestas en el capítulo 5.

Para plantear este sistema podría resultar interesante trabajar sobre homología entre imágenes consecutivas en las que se identifican diferentes vehículos o incluso con un vehículo de apoyo al despliegue que contenga una mallado identificable y deferenciable en cada uno de sus cinco lados visibles (frontal, lateral izquierdo, lateral derecho, trasero y techo), con tamaños controlados, de los cuales obtener características-descriptores (por ejemplo, esquinas de cuadriculas mas fondo, donde este último fuera una cualidad para diferenciar los cinco lados del vehículo) y por correlación homológráfica poder realizar el ajuste del sistema. Este vehículo podría ser en sí mismo una clase en el sistema de clasificación, de tal forma que el propio sistema se pudiera autoajustar periódicamente solo con el tránsito de éstos vehículos en las zonas donde se realiza el despliegue de los sistemas de captura de imágenes, es decir, el sistema al identificar este tipo de vehículo pasara a un modo de calibrado con autoajuste. Este método aumentaría la seguridad de los técnicos y el impacto de la implantación, ya que por un lado un técnico no tendría que invadir la vía fuera de un vehículo en circulación y por consiguiente no se tendría que paralizar el tráfico rodado para balizar la región de trabajo.

Para mejorar el método de conteo de vehículos sería interesante utilizar descriptores para definir los centroides de las regiones en movimiento con el objetivo de diferenciar si el objeto ha sido ya contabilizado o no. En esta misma línea utilizar filtros de Kalman permitirían estimar la trayectoria de

los vehículos (centroides) y por consiguiente conocer que un vehículo en una posición (x,y) en un tiempo t, es el mismo que se encontraba en (x_0,y_0) en t_0. Este mismo proceso podría realizarse con una red LSTM entrenada con las trayectorias de varios vehículos.

Si somos capaces de estimar la dinamica de un objeto en movimiento dentro de un campo visual, nos permitiría realizar un sistema de control por visión artificial con el cual realizar seguimiento de objetos. Una aplicación práctica podría ser un sistema que permita a un invidente orientarse con respecto a diferentes objetos (fijos o en movimiento), identificados en su entorno y poder alcanzarlos.

b) Redes Neuronales

Aunque en nuestros modelos de CNN no se ha incluido una capa de convolución dilatad, resulta muy interesante el comportamiento de estas para obtener información de contexto lo cual nos podría ayudar a distinguir entre un movimiento relativo o absoluto tal y como definimos al inicio del capítulo 2.

Dada la consideración de las CNN como sistemas inteligentes, resultaría interesante incluir una capa de inferencia difusa con la que determinar la clase a la que pertenece el objeto o la región a clasificar. Esto podría hacerse mediante un modelo de inferencia Mamdani en el que las probabilidades de salida de la capa *softmax* serían las entradas al motor de inferencia, construido con un conjunto de funciones de pertenencias de entrada que podrían ser, “alta”, “media alta”, “media”, “media baja” y “baja”, mientras que la salida podría estar compuesta por tantas funciones de pertenencia como clases a identificar. Para ello resulta interesante revisar trabajos como el de Moreno [40]. Por otro lado, sería interesante la comparación de resultados en clasificación con redes Neuro Fuzzy como se proponen en el trabajo de Swheta [41] para el caso de reconocimiento de caras.

Profundizando sobre el funcionamiento del algoritmo de retropropagación, comentar que resultaría interesante estudiar el comportamiento al cambiar el método de optimización SGD por métodos heurísticos tales como “enfriamiento simulado” o “algoritmos genéticos” que podrían proporcionar otros resultados, evitar el problema de los posibles mínimos locales en la función de coste o incluso trabajar en paralelo con múltiples objetivos.

Una línea con la que mejorar las bases de datos en las que no se dispone de la opción de aumentarlas con datos reales, puede ser la ampliación con nuevos datos procedentes de los existentes a los que se ha aplicado unas ciertas transformaciones o manipulaciones. Estas transformaciones pueden ser cambios en la iluminación, rotaciones y traslaciones de las perspectivas, cambios en la nitidez o añadido de sombras, entre otras.

Para esto se podría generar una función encargada de analizar el número de archivos asociados a cada clase y ampliase aquellos directorios con menor número de archivos hasta alcanzar el número de la clase que contenga el número más alto de elementos. Esta función podría realizar algunas de las transformaciones indicadas u otras de manera aleatoria, o dotarla de ciertos parámetros de control para indicar al sistema el tipo de transformación concreta.

En el trabajo nos hemos centrado especialmente en las CNN como sistema clasificador, sería interesante continuar el estudio en modelos predictivos fundamentados en la extracción de características por capas convolucionales y quizá asociados a redes tipo LSTM a las que conectar dichas características tras una capa *flaten*.

En la misma línea de la predicción a un nivel de imagen también resultaría interesante continuar el estudio con redes de segmentación semántica con el objetivo de estimar la clasificación de píxeles vecinos y comparar el comportamiento con una red GAN ("*Generative Adversarial Network*").

Además de lo anterior, como trabajo a seguir explorando en el futuro, en el Anexo II se incluyen algunos resultados obtenidos en la exploración del comportamiento de AlexNet como arquitectura pre-entrenada, bajo el entrenamiento de imágenes RGB en las que los canales no definen el color, sino características geométricas y morfológicas, pudiendo explorar otro tipo de características. Los objetivos de explorar esta línea se fundamentan en la reducción de la arquitectura de CNN y el volumen del conjunto de datos necesario para el entrenamiento. El hecho de poder codificar dichas características en los canales RGB de entrada de estas arquitecturas abren la posibilidad de su utilización para otro tipo de problemas de clasificación como pueden ser señales procedentes de electroencefalogramas, señales y/o imágenes procedentes del estudio del plasma en un estellator, como el TJ-II, tendencias de los mercados bursátiles, entre otros.

REFERENCIAS

Nota: *Las referencia mostradas en esta sección se encuentran en el formato ISO 690 y bajo el orden de aparición en el documento.*

[1] GIBSON, James J. The ecological approach to visual perception: classic edition. Psychology Press, 2014.

[2] J. Antonio A., "Psicología de la percepción Visual", Universidad de Barcelona (VISCA). http://www.ub.edu/pa1/

[3] GIBSON, James Jerome. The senses considered as perceptual systems. 1966.

[4] PAJARES, Gonzalo; DE LA CRUZ, Jesús M. Visión por Computador: Imágenes digitales y aplicaciones. Madrid: In RA-MA, 2001.

[5] MARGALEF, José Bayo. Percepción, desarrollo cognitivo y artes visuales. Anthropos Editorial, 1987.

[6] THE MATHWORKS MATLAB (2019) https://es.mathworks.com/

[7] HARALICK, Robert M.; SHAPIRO, Linda G. Computer and robot vision. Reading: Addison-wesley, 1992.

[8] CUCCHIARA, Rita, et al. The Sakbot system for moving object detection and tracking. En Video-based surveillance systems. Springer, Boston, MA, 2002. p. 145-157.

[9] RUMELHART, David E.; HINTON, Geoffrey E.; WILLIAMS, Ronald J. Learning internal representations by error propagation. California Univ San Diego La Jolla Inst for Cognitive Science, 1985.WANG, Zhou, et al. Image quality assessment: from error visibility to structural similarity. IEEE transactions on image processing, 2004, vol. 13, no 4, p. 600-612.

[10] RICHTER, Thomas; KIM, Kil Joong. A ms-ssim optimal jpeg 2000 encoder. En 2009 Data Compression Conference. IEEE, 2009. p. 401-410.

[11] Nvidia.https://developer.nvidia.com/rtx/ngx

[12] JAIN, Anil K.; MAO, Jianchang; MOHIUDDIN, K. M. Artificial neural networks: A tutorial. Computer, 1996, no 3, p. 31-44.

[13] AGATONOVIC-KUSTRIN, S.; BERESFORD, R. Basic concepts of artificial neural network (ANN) modeling and its application in pharmaceutical research. Journal of pharmaceutical and biomedical analysis, 2000, vol. 22, no 5, p. 717-727.

[14] JIMÉNEZ, M. Redes neuronales y preprocesado de variables para modelos y sensores en bioingeniería. Ingeniería En Electrónica. Universidad Politécnica de Valencia, 2012.

[15] NIELSEN, Michael A. *Neural networks and deep learning*. San Francisco, CA, USA:: Determination press, 2015

[16] BISHOP, Christopher M. Pattern recognition and machine learning. springer, 2006.

[17] GARCÍA, Pedro Pablo García. Reconocimiento de imágenes utilizando redes neuronales artificiales. Facultad de Informática, Universidad Complutense de Madrid, 2013, vol. 5.

[18] KRIZHEVSKY, Alex; SUTSKEVER, Ilya; HINTON, Geoffrey E. Imagenet classification with deep convolutional neural networks. En Advances in neural information processing systems. 2012. p. 1097-1105.

[19] SZEGEDY, Christian, et al. Inception-v4, inception-resnet and the impact of residual connections on learning. En Thirty-First AAAI Conference on Artificial Intelligence. 2017.

[20] SAK, Haşim; SENIOR, Andrew; BEAUFAYS, Françoise. Long short-term memory recurrent neural network architectures for large scale acoustic modeling. En Fifteenth annual conference of the international speech communication association. 2014.

[21] PALACIO, F. Herramientas en GNU/Linux para estudiantes universitarios-Redes Neuronales con GNU/Linux. The Public's Library and Digital Archive. Online [Jun 2017], 2003.

[22] FRANCISCO JOSE P.B. "Herramientas en GNU/Linux para estudiantes universitarios" https://www.ibiblio.org/pub/linux/docs/LuCaS/Presentaciones/200304curso-glisa/redes_neuronales/curso-glisa-redes_neuronales-html/index.html

[23] DERTAT, A. (2017). Applied Deep Learning - Part 4: Convolutional Neural Networks. https://towardsdatascience.com/applied-deep-learning-part-4-convolutional-neural-networks-584bc134c1e2

[24] VAN DEN OORD, Aäron, et al. WaveNet: A generative model for raw audio. SSW, 2016, vol. 125.

[25] SRIVASTAVA, Nitish, et al. Dropout: a simple way to prevent neural networks from overfitting. The Journal of Machine Learning Research, 2014, vol. 15, no 1, p. 1929-1958.

[26] JARRETT, Kevin, et al. What is the best multi-stage architecture for object recognition. En 2009 IEEE 12th international conference on computer vision. IEEE, 2009. p. 2146-2153.

[27] LYU, Siwei; SIMONCELLI, Eero P. Nonlinear image representation using divisive normalization. En 2008 IEEE Conference on Computer Vision and Pattern Recognition. IEEE, 2008. p. 1-8.

[28] JOYCE, X. (2017). Deep Learning for Object Detection: A Comprehensive Review. https://towardsdatascience.com/deep-learning-for-object-detection-a-comprehensive-review-73930816d8d9

[29] REN, Shaoqing, et al. Faster r-cnn: Towards real-time object detection with region proposal networks. En Advances in neural information processing systems. 2015. p. 91-99.

[30] LARRANAGA, Pedro; INZA, Inaki; MOUJAHID, Abdelmalik. Tema 8. redes neuronales. Redes Neuronales, U. del P. Vasco, 1997, p. 12-17.

[31] HE, Kaiming, et al. Deep residual learning for image recognition. En Proceedings of the IEEE conference on computer vision and pattern recognition. 2016. p. 770-778.

[32] TARG, Sasha; ALMEIDA, Diogo; LYMAN, Kevin. Resnet in resnet: Generalizing residual architectures. arXiv preprint arXiv:1603.08029, 2016.

[33] CORDTS, Marius, et al. The cityscapes dataset for semantic urban scene understanding. En *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016. p. 3213-3223.: https://www.cityscapes-dataset.com/

[34] Grupo de tratamiento de Imágenes UPM: https://www.gti.ssr.upm.es/

[35] GRANADOS, Andres Felipe; MARIN, Jorge Ivan. Detección de flujo vehicular basado en visión artificial. Scientia et technica, 2007, vol. 3, no 35, p. 163-168.

[36] DOMÍNGUEZ VARELA, Artur. Análisis de la congestión en Áreas Metropolitanas: Aplicación al caso de Barcelona. 2007.

[37] DEYÀ LLOPÌS, Joan. Análisis de los sistemas de gestión del tráfico mediante velocidad variable aplicando a la vía C-31N velocidad variable. 2012.

[38] ÁNGEL J. M, "Temario a oposiciones escala superior DGT (Spain)" : www.dgt.es/Galerias/la-dgt/empleo-publico/oposiciones/doc/2014/TEMA-3.7.docx

[39] MORENO, Robinson Jiménez; SANCHEZ, Oscar Avilés; OVALLE, Diana M. RGB-D Training for Convolutional Neural Network with Final Fuzzy Layer for Depth Weighting. Contemp. Eng. Sci., 2017, vol. 10, no 29, p. 1419-1429.

[40] MEHTA, Shweta, et al. Face recognition using neuro-fuzzy inference system. International Journal of Signal Processing, Image Processing and Pattern Recognition, 2014, vol. 7, no 1, p. 331-334.

ANEXO I. RECURSOS UTILIZADOS

Este anexo se recoge las características de Hardware, así como la herramienta se software utilizada.

Hardware:

- Procesador**:** Intel ® Core™ i7-8550U CPU @1.80GHz 1.99GHz
- Memoria RAM: 12 GB
- Almacenamiento interno: SSD 512GB
- Tipo de sistema: Sistema operativa de 64bits, procesador x64
- Tarjeta gráfica: Nvidia GeForce 940MX

Software:

- Sistema operativo: Windows 10 Home 64bits
- Procesamiento y desarrollo: Matlab (MathWorks, 2019), así como los toolboxes de *Deep Learning*, *Image Processing* y *Computer Vision* con toda su potencialidad.
- Funciones y Script desarrollados en orden de ejecución según la estructura del presente trabajo:
    - **Flujo_optico.m**: utilizado para el estudio del comportamiento de los diferentes métodos de obtención de flujo óptico y calibración de áreas en movimiento a detectar.
    - **imagenesFlow.m**: Función encargada de generar una ventana en la que aparezca simultáneamente el comportamiento de cuatro métodos de determinación de flujo óptico, asociado a una región de un vídeo.
    - **Identificacion2.m**: Función encargada de procesar una imagen para devolver una imagen binarizada, dilatada y acotada a unas dimensiones de áreas específicas bajo los criterios de identificación del flujo óptico con los cuatro métodos estudiados. Además, esta función nos proporciona información de la segmentación de las diferentes regiones identificadas, así como etiquetas de los elementos conexos, o el número de regiones diferentes identificadas.
    - **OpticFlow_detection.m**: Encargado identificar regiones en movimiento, pudiendo seleccionar cuál de los cuatro métodos de flujo óptico se desea utilizar. Todas estas regiones identificadas se muestran sobre el vídeo con recuadros (*bounding Box*) y en paralelo con las regiones en movimiento identificadas, dilatadas y binarizadas.
    - **Movimiento_SSIM.m:** Script que permite la visualización de un vídeo en estructura de SSIM junto con un mapa del mismo binariazo que representa las regiones donde se ha producido movimiento. En ambas imágenes se identifica el movimiento.
    - **Ejemplo_SSIM_Deteccion.m:** Script encargado de mostrar por pantalla un video seleccionado junto con recuadros de los objetos que se están moviendo. Esto visualiza

en paralelo con el vídeo procesado en el que solo aparecen las regiones donde se ha producido movimiento.

- **Identificación.m:** Esta función se encarga de procesar una imagen para devolver una imagen binarizada en la que se han eliminado regiones muy pequeñas y muy grandes todas ellas procedentes de la detección de movimiento por SSIM. Además, esta función nos proporciona información de la segmentación de las regiones, así como etiquetas de los elementos conexos, o el número de regiones diferentes identificadas.
- **funcionesActivación.mxl:** Live script encargado de generar las gráficas de las funciones de activación mencionadas en el apartado 3.2.1.1 del presente trabajo.
- **reduccion_video.m:** función encargada de reducir el tamaño de un vídeo a una escala determinada y almacenarlo con un nombre deseado.
- **Read_video.m:** script que nos permite seleccionar un vídeo y visualizarlo.
- **imdsTo4Ddata.m:** Función encargada de generar un vector de 4D para el entrenamiento de redes tipo DAG.
- **Dataset4D.mxl:** Script encargado de partiendo de una BBDD "*imds*" de Matlab, generar un vector 4D de entrada de imágenes y un vector de salida con las clases correspondientes
- **extraer_imagenes_gTruth.m:** función encargada de extraer de un vídeo etiquetado con diferentes clases, procedente de la App de Matlab "*ground truth labeler*", imágenes con una dimensión determinada a elección por el usuario y ordenarlas en carpetas correspondientes a las diferentes clases etiquetadas.
- **Alexnet_RCNN.xml:** Live script con la construcción de la arquitectura de AlexNet para la clasificación de las clases estudiadas en este trabajo y encargado de proporcionar dicha arquitectura en formato *lgraph* para su entrenamiento como CNN y RCNN.
- **Resnet50_RCNN.xml:** Live script con la construcción de la arquitectura de ResNet50 para la clasificación de las clases estudiadas en este trabajo y encargado de proporcionar dicha arquitectura en formato *lgraph* para su entrenamiento como CNN y RCNN.
- **Vehicle_RCNN.xml**: Live script con la construcción de la arquitectura de Vehicle para la clasificación de las clases estudiadas en este trabajo y encargado de proporcionar dicha arquitectura en formato *lgraph* para su entrenamiento como CNN y RCNN.
- **AlexNet_FastRCNN.xml:** Live script con la construcción de la arquitectura del detector AlexNet fast R-CNN, que nos proporciona su estructura en formato *lgraph* para su entrenamiento como fast R-CNN.
- **ResNet50_FastRCNN.xml:** Live script con la construcción de la arquitectura del detector ResNet50 fast R-CNN, que nos proporciona su estructura en formato *lgraph* para su entrenamiento como fast R-CNN.
- **Vehicle_FastRCNN.xml:** Live script con la construcción de la arquitectura del detector vehicle fast R-CNN, que nos proporciona su estructura en formato *lgraph* para su entrenamiento como fast R-CNN.
- **AlexNet_FasterRCNN.xml:** Live script con la construcción de la arquitectura del detector AlexNet faster R-CNN, que nos proporciona su estructura en formato *lgraph* para su entrenamiento como faster R-CNN.

- **ResNet50_FasterRCNN.xml:** Live script con la construcción de la arquitectura del detector ResNet50 faster R-CNN, que nos proporciona su estructura en formato *lgraph* para su entrenamiento como faster R-CNN.
- **Vehicle_FasterRCNN.xml:** Live script con la construcción de la arquitectura del detector vehicle faster R-CNN, que nos proporciona su estructura en formato *lgraph* para su entrenamiento como faster R-CNN.
- **Red_Jompy.m**: Script encargado de entrenar una CNN con unas dimensiones de capa de entrada determinada con datos procedentes de carpetas que contienen las diferentes clases posibles a identificar. Nos devuelve un clasificador entrenado.
- **EntrenamientoRedesRCNN.m:** Script encargado de entrenar cualquiera de las arquitecturas de redes generadas, a partir de la base de datos generada del etiquetado de vídeos con la aplicación de Matlab "*ground truth labeler*". Este script se encarga de almacenar los detectores en una carpeta determinada y asociar a estos detectores un nombre en función de su arquitectura. En este script se pueden controlar los parámetros de entrenamiento y entrenar de manera consecutiva diferentes arquitecturas con distintos parámetros de entrenamiento.
- **My_clasificacion.m:** este script nos permite seleccionar un determinado clasificador CNN y una imagen concreta y proporcionarnos gráficamente la clasificación obtenida de dicha imagen y la probabilidad de pertenencia a cada una de las clases posibles mediante un diagrama de barras.
- **estudio_capas_red.m:** Este script nos permite analizar el comportamiento de cada una de las capas de una CNN seleccionada, con una imagen determinada; proporcionando información del canal con mayor activación en cada capa y mostrando dicha información. Esto se realiza sobre las capas convolucionales y las capas de activación, con el fin de proporcionarnos información para decidir donde inferir para generar las arquitecturas fast y faster R-CNN.
- **Validacion.m**: función encargada de testear un determinado detector R-CNN.
- **prueba_detectores_imagenes.m:** Este script nos permite realizar pruebas de identificación y clasificación de regiones en imágenes, con cualquiera de las arquitecturas de detectores entrenados**.**
- **OFlow_deteccion_clasificacion.m**: Este script nos permite clasificar las diferentes regiones que presentan movimiento en un video, pudiendo seleccionar por menú el método de flujo óptico deseado para detección y el tipo de arquitectura de CNN para la clasificación, para un determinado vídeo. Este script también permite generar un vídeo con los resultados obtenidos.
- **Optic_flow_CNN_traficFlow.m:** Este script permite contabilizar, clasificar vehículos que se encuentran en movimiento mediante cualquiera de los cuatro métodos de determinación de flujo óptico y mediante una CNN entrenada que podemos seleccionar. Esta información es proporcionada gráficamente a la par que se muestra el video a analizar con los diferentes vehículos identificados y el tiempo transcurrido, lo cual permite obtener el flujo de tráfico. Este script también nos permite grabar un vídeo con los resultados obtenidos.

- **Ejemplo_SSIM_Deteccion_clasificion.m:** Script encargado de identificar, en un video determinado, regiones en movimiento por el método SSIM y clasificarlas mediante una CNN que podemos seleccionar.
- **Conteo_vehiculos_SSIM.m:** Función encargada de contabilizar vehículos que se están moviendo, clasificarlos por categorías y proporcionarnos el tiempo transcurrido con el que determinar el flujo de tráfico de vehículos. Los datos se muestran simultáneamente con el video y las categorías simultáneamente junto con un diagrama de barras que muestra el conteo de las diferentes clases de vehículos. La CNN asociada se puede seleccionar en un menú inicial.
- **TraficFlow_Detectores.m:** Este script permite contabilizar, clasificar vehículos que se encuentran en movimiento mediante cualquiera de las tres arquitecturas diferentes (R-CNN, fast R-CNN y faster R-CNN) para las tres CNNs implementadas, es decir, mediante cualquiera de los 9 detectores diferentes. Esta información es proporcionada gráficamente a la par que se muestra el video a analizar con los diferentes vehículos identificados y el tiempo transcurrido, lo cual permite obtener el flujo de tráfico. Este script también nos permite grabar un vídeo con los resultados obtenidos.
- **folderImag2HVD.m:** Esta función se encarga de seleccionar una carpeta determinada donde se alojan diferentes subcarpetas de imagen RGB, para realizar una transformación de dichas imágenes a los formatos HVD (horizontal vertical diagonal), ACB (Apertura Cierre y Bordes), Seg (segmentación por canales RGB con algoritmos Sobel, log y canny), RGB 32x32, RGB 64x64 y RGB 128x128.
- **Imag2ca.**m: Función encargada de transformar una imagen RGB en una imagen construidas con las características horizontales, verticales y diagonales procedentes de la aplicación de la transformada wavelet con funciones rbior5.5 tras un ajuste de tamaño para que estos canales de características coincidan en tamaño con la capa de entrada de AlexNet al aplicar el primer orden de la transformada wavelet.
- **Imag2ACB.m:** Esta función se encarga de transformar una imagen RGB a una imagen de características proporcionada por transformaciones morfológicas de Apertura, Cierre y Bordes con dimensión aceptable por la arquitectura de AlexNet.
- **Imag2seg.m:** Esta función se encarga de transformar una imagen RGB a una imagen de características formada por la segmentación de cada uno de los canales. Para ello permite seleccionar los algoritmos de Canny, Sobel o laplaciano de Gauss.

Datos:

- Videos: Se han utilizado 10 vídeos, de unos 180MB tomados a 30 fps.
- Entrenamiento: Se han utilizado 371 imágenes relevantes extraídas de 172 frames significativos de los vídeos. Todas las imágenes se etiquetaron convenientemente con la clase apropiada a la que pertenecen.

Los ejemplos que aparecen en el trabajo son ejemplos ilustrativos de este conjunto de imágenes.

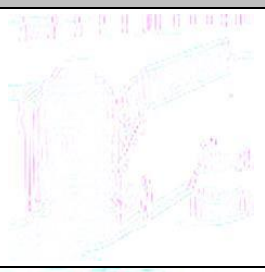
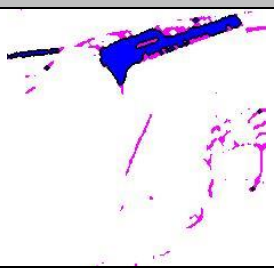
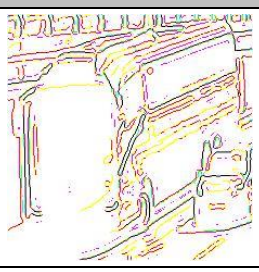
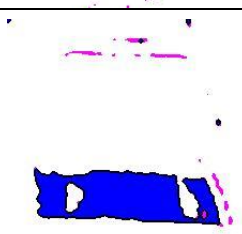
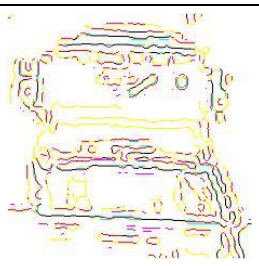
ANEXO II. IMÁGENES DE CARACTERÍSTICAS

En el planteamiento del TFM, se comenzó en la línea de utilizar nuevos canales basados en características geométricas y morfológicas de entrada en las arquitecturas de las CNN con el objetivo de reducir la arquitectura de este tipo de redes y del conjunto de datos necesarios para las fases de entrenamiento, así como el estudio del comportamiento de una CNN pre-entrenada ante el aprendizaje de nuevas características. Estas nuevas imágenes están compuestas por características codificadas en los canales RGB de archivos .jpeg.

En esta línea se plantean 3 nuevas propuestas de imágenes de características.

- HVD: Tres canales compuestos por características Horizontales, Verticales y Diagonales producidas por la transformada wavelet de una imagen con redimensionado de su tamaño al de la capa de entra de la red neuronal que se desea trabajar.
- ACB: Características provenientes de la Apertura, Cierre y Borde de una imagen dada, es decir de operaciones morfológicas.
- SEG: Características procedentes por segmentación, es decir, los 3 canales RGB separados y segmentados independientemente son introducidos en una nueva estructura compuesta por tres canales compuestos por la segmentación de cada uno de los canales de origen RGB.

En la tabla 17 se muestra una comparación de imágenes RGB y su transformación a cada una de las nuevas estructuras propuestas.

RGB	HVD	ACB	SEG
			
			

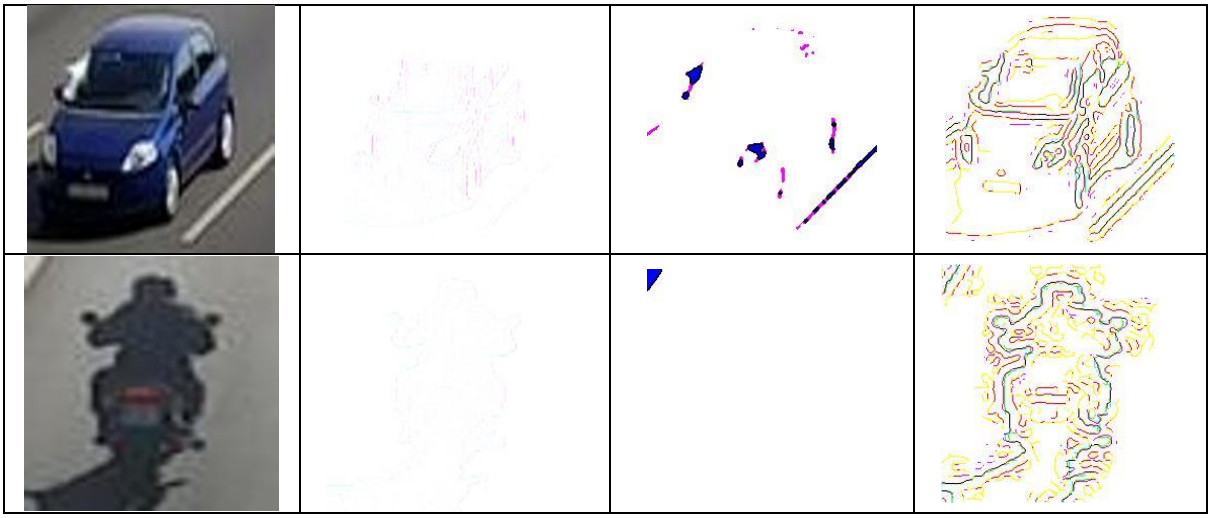


Tabla 17: Comparación de imágenes en RGB,HVD, ACB y SEG

Para generar estas nuevas imágenes se han creado 3 funciones encargadas de transformar un conjunto de dados (imágenes) en RGB a imágenes de estas características ordenadas en carpetas del mismo modo que el conjunto de datos de entrada. Esta similitud en la reconstrucción de las bases de datos facilita la comparación entre imágenes RGB y en cualquiera de las formas propuestas.

Para comprobar el comportamiento de estas nuevas propuestas de imágenes con las CNN se utilizó la red pre-entrenada AlexNet para la clasificación de 8 clases diferentes en el contexto de las ciudades inteligentes. Las clases propuestas son:

- Asfalto
- Bus
- Camión Furgo
- Coche Delantera
- Coche trasera
- Líneas
- Moto
- Muro

Se entrenó esta arquitectura para la clasificación de las 8 clases diferentes con 4 conjuntos de datos idénticos en número de elementos y donde se han obteniendo los resultados de las siguientes figuras.

Alexnet en HVD

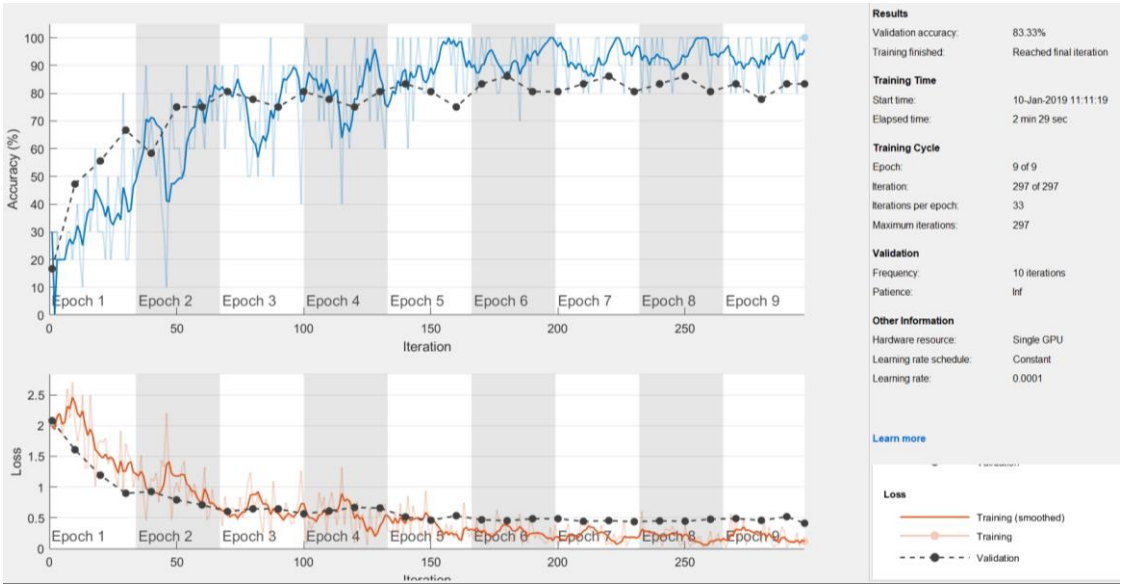
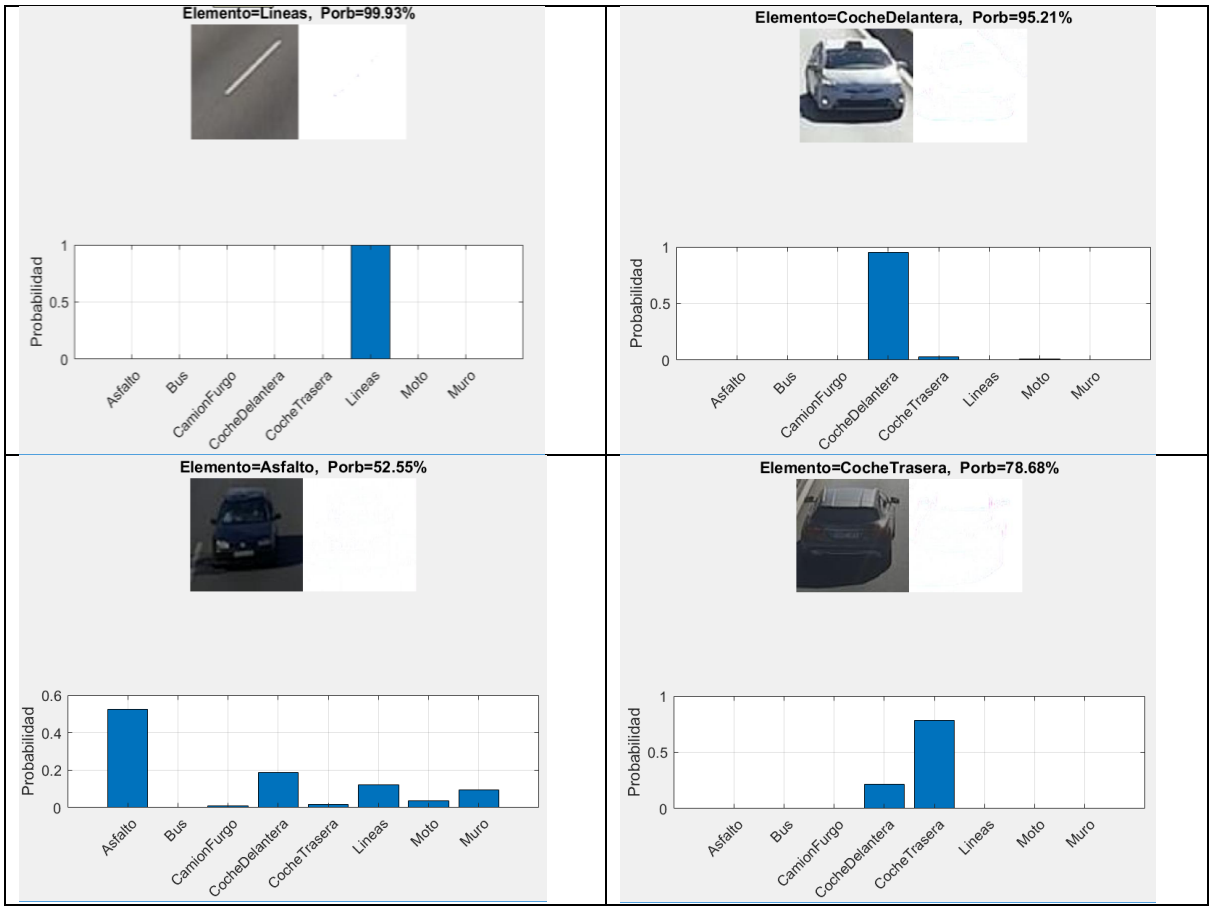


Figura 70: Proceso de entrenamiento AlexNet en HVD

Comprobación HVD:



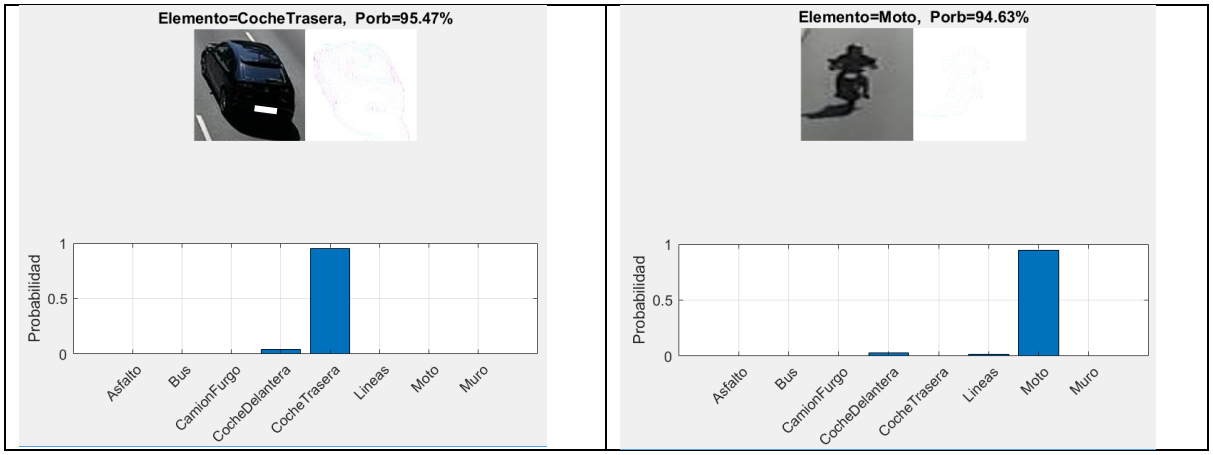


Tabla 18: Seis ejemplos de clasificación con AlexNet en HVD

Alexnet en ACB

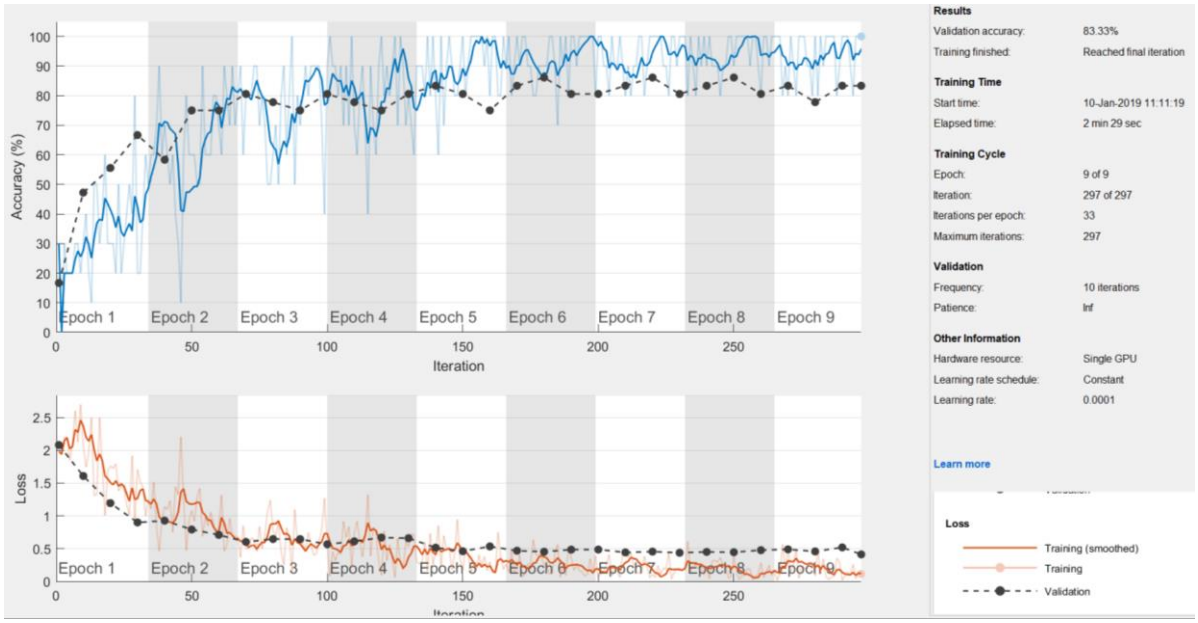


Figura 71: Proceso de entrenamiento AlexNet en ACB

Comprobación ACB

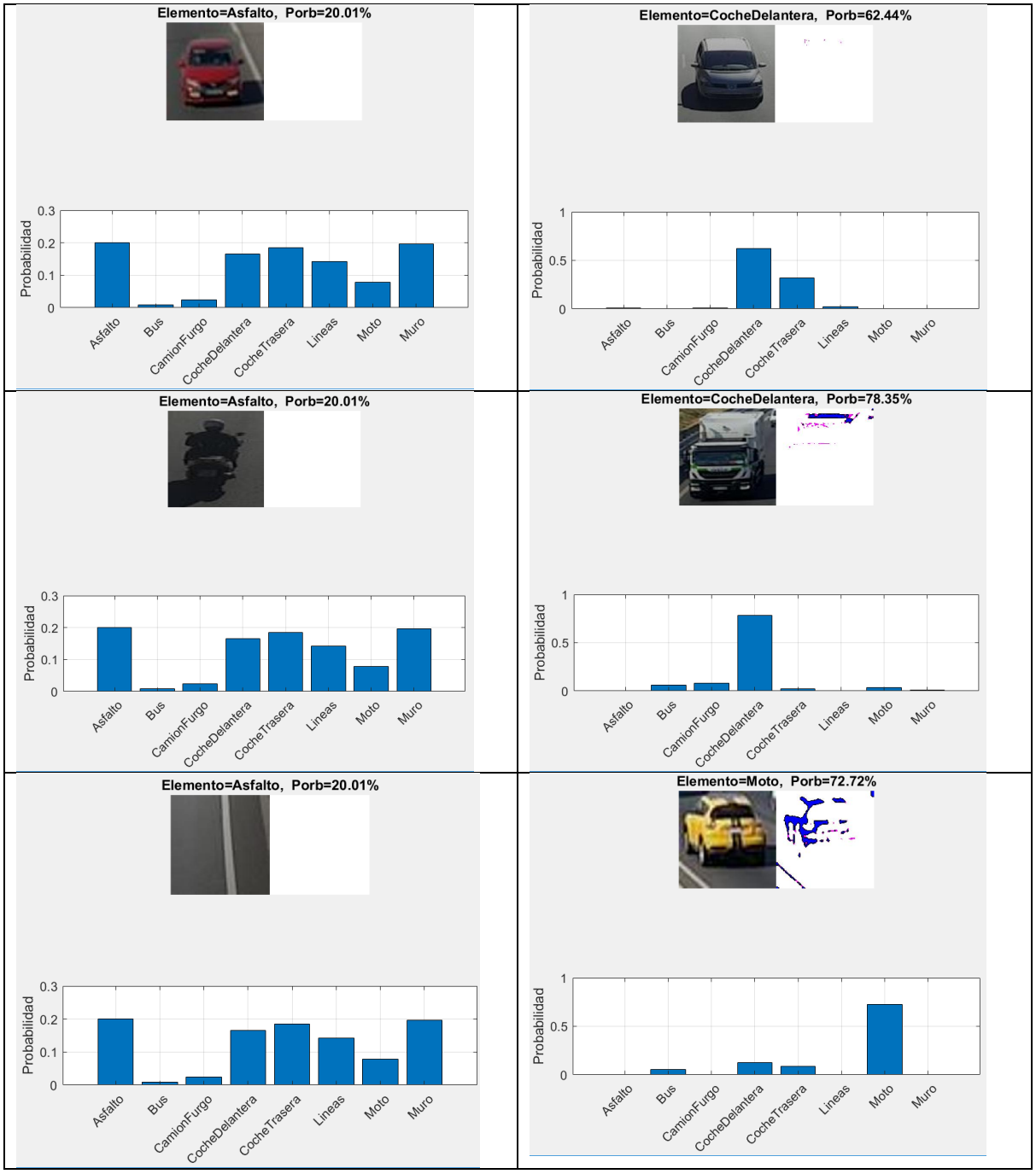


Tabla 19: Seis ejemplos de clasificación con AlexNet en ACB

AlexNet en SEG

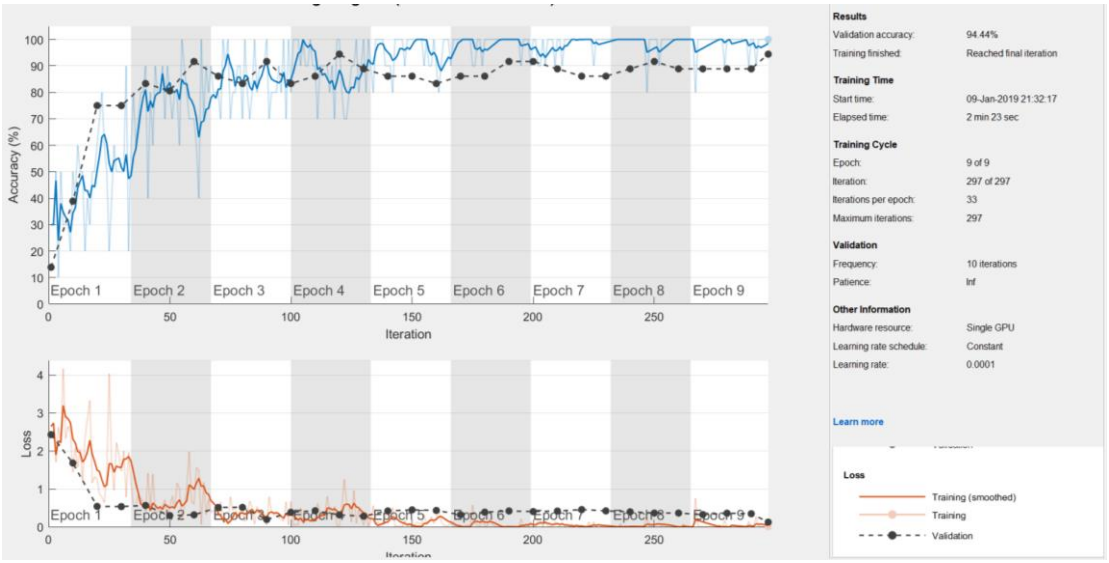
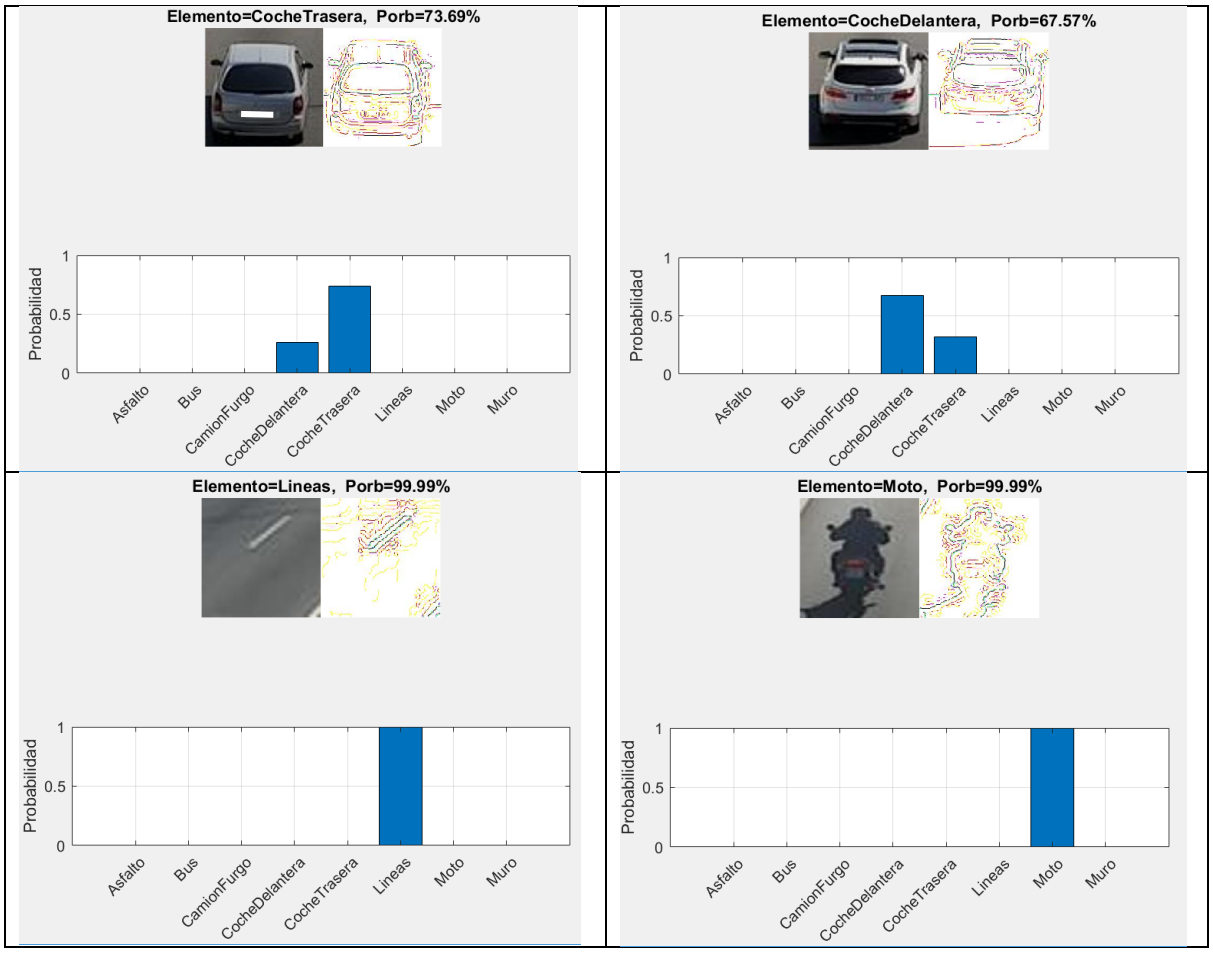


Figura 72: Proceso de entrenamiento AlexNet en SEG



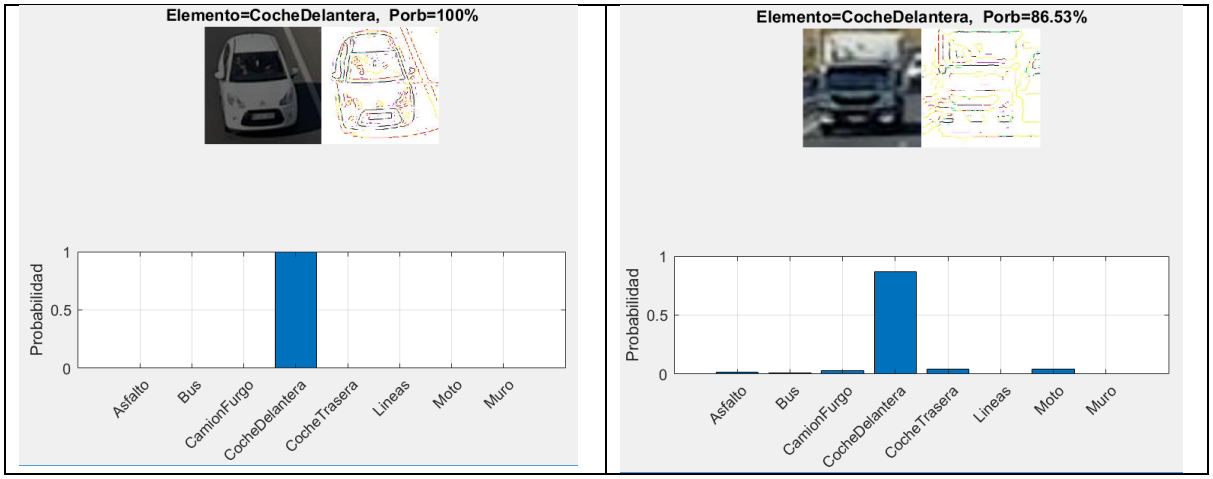


Tabla 20: Seis ejemplos de clasificación con AlexNet en SEG

AlexNet en RGB

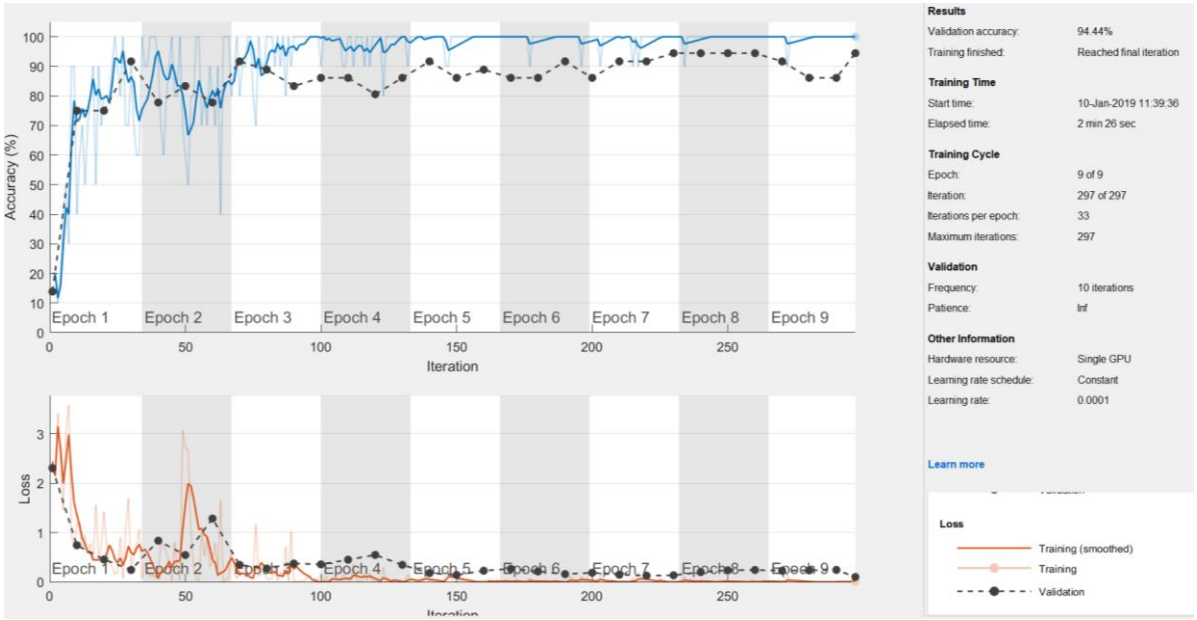
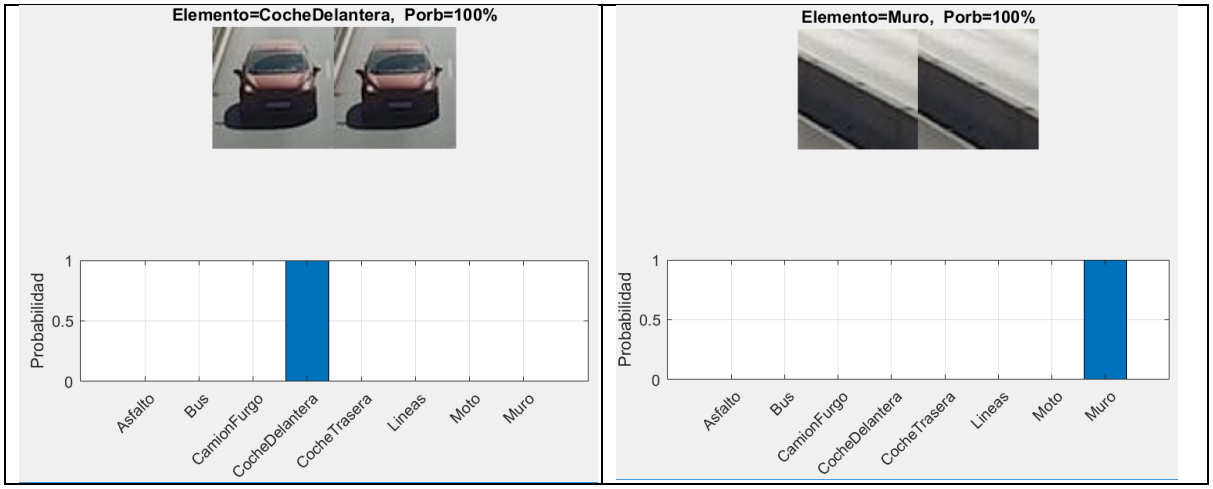


Figura 73: Proceso de entrenamiento AlexNet en RGB



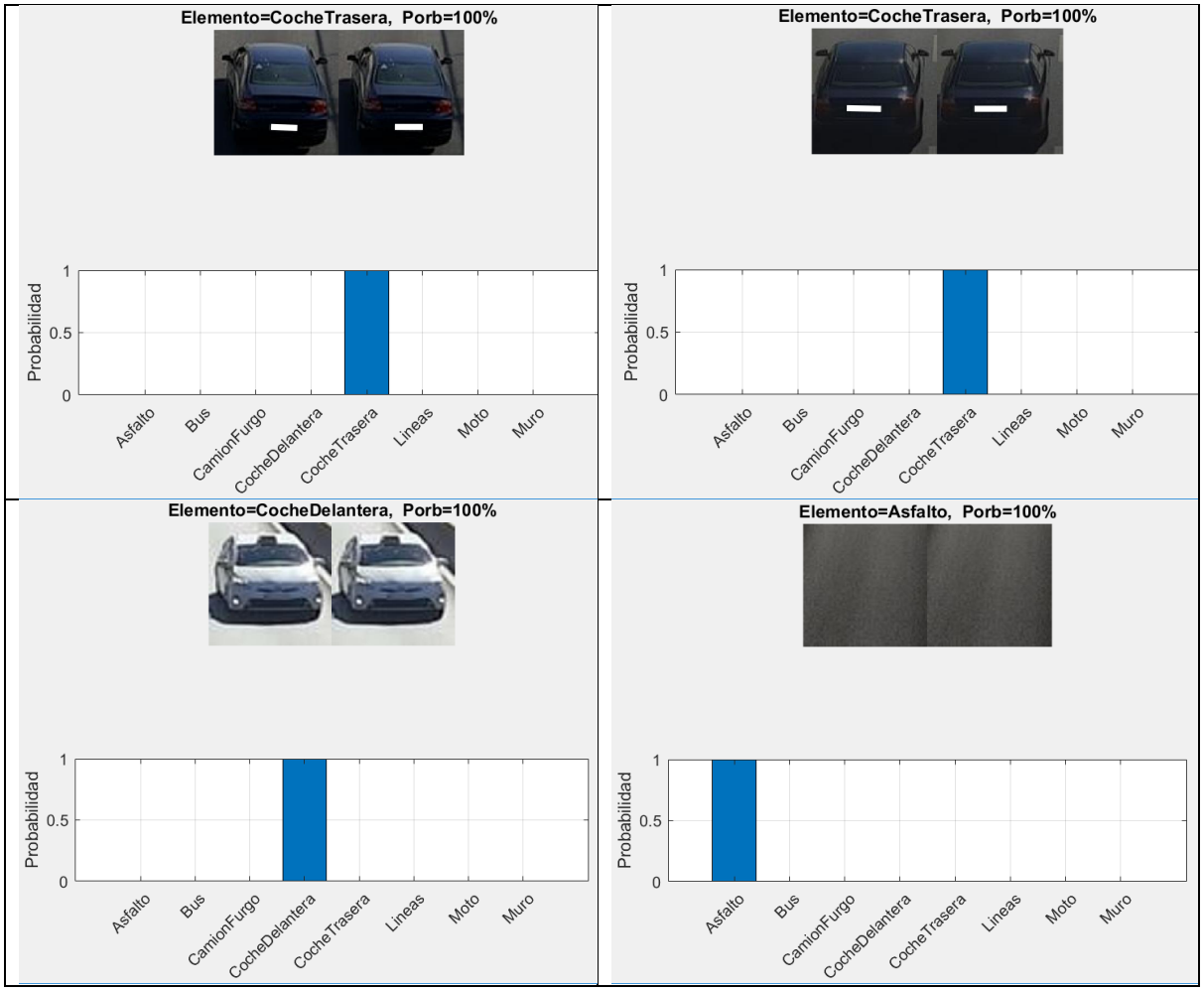


Tabla 21: Seis ejemplos de clasificación con AlexNet en RGB

En las tablas de ejemplos de clasificación se puede ver en cada una de las imágenes el interfaz de clasificación con los resultados obtenidos. En la parte superior, se compara la imagen en RGB con la imagen en la forma de características y se indica a modo de título la clase a la que se ha asignado con el grado de probabilidad de pertenencia a dicha clase, que se muestra en la parte inferior.

Se puede comprobar en las imágenes de entrenamiento cómo SEG y RGB son las que obtienen los mejores resultados, consiguiendo en este proceso de entrenamiento, validaciones mínimas superiores al 94% en ambos casos.

Para el caso de ACB los resultados de los ejemplos mostrados indican una gran incertidumbre en la fase de clasificación. Buena referencia de ello se muestra en la fase de entrenamiento, donde se produce *overfitting* con una aparente saturación y una validación “baja”.

En los ejemplos de clasificación con HVD comprobamos como el sistema es capaz de realizar buenas clasificaciones, especialmente con líneas y elementos con buena iluminación, pero dado que se obtienen mejores resultados con SEG y RGB, el objetivo se centra en estos dos casos.

Dadas las certezas mostradas en las tablas de ejemplos para el caso de AlexNet en RGB y SEG, finalmente se decide apostar por el trabajo con datos en formato RGB, considerando el estudio en HVD y SEG alternativas interesantes para el futuro.