Máster universitario en Ingeniería de Sistemas y de Control
Escuela Técnica Superior de ingeniería informática
UNED
Trabajo fin de máster

Eliminación de sombras en vehículos para su clasificación mediante Redes Neuronales Convolucionales

Autor: Juan Antonio Bellido Jiménez

Director: Gonzalo Pajares Martinsanz

Convocatoria 2018 – 2019

Madrid, Septiembre 2019

Agradecimientos

En primer lugar, me gustaría agradecer a Don Gonzalo Pajares todo el esfuerzo dedicado en la corrección y en la guía de este trabajo final de máster y su asignatura. Es un verdadero placer conocer a profesores a distancia que se preocupan tanto por el aprendizaje, y más cuando este se realiza a distancia. Me has instruido en un mundo, el de la inteligencia artificial, que me ha fascinado y en el que seguro voy a seguir investigando; muchísimas gracias

Por otra parte, me gustaría agradecer al profesor Don Javier Estévez, de la universidad de Córdoba, todo el tiempo que me ha dedicado, no solo al proyecto futuro que se avecina, sino a todos los consejos personales y profesionales que me ha facilitado. Soy un fan no solo de su trabajo como docente e investigador, sino de su trabajo musical con Subtónica; eres un profesor e investigador excepcional y he aprendido muchísimo como por ejemplo tu frase que llevo por bandera es: "*si a lo que nos divide le restamos importancia, el resultado será siempre positivo*"

A mis padres. No olvidéis jamás que os quiero y que tanto Sergio como yo estamos donde estamos por vuestro esfuerzo. Me siento muy agradecido y afortunado por teneros y haber aprendido de vuestras grandezas y puntos flacos, pero sobre todo, de seguir hacia delante a pesar de las adversidades.

Sergito, eres una persona que se supera día a día, nos has enseñado a todos que la actitud lo es todo y que el límite es algo ficticio. Eres todo un referente para mí. Tu camino y tus formas son las correctas, pronto seré espectador del grandísimo futuro que te espera.

Por último, me gustaría dedicar este trabajo a mi pareja, Almudena, por todos los esfuerzos realizados. La recompensa llegará porque seguiremos luchando por nuestro proyecto de vida y nuestra familia. Te quiero.

Índice general

Figuras

Tablas

1. Introducción

1.1 Preliminares y motivación

La inteligencia artificial (IA) ha sido una de las disciplinas tecnológicas más estudiadas en las últimas décadas. Su gran versatilidad hace que su uso se haya diversificado en numerosos ámbitos; hoy día, es complicado pensar en un proceso productivo o tarea cotidiana en la que no esté presente.

A pesar de la elevada mención en numerosas revistas tecnológicas, noticiarios e internet cabe preguntarse ¿qué es la inteligencia artificial (IA)? La respuesta viene en función de la perspectiva que se utilice al respecto. En cualquier caso, desde un punto de vista computacional práctico se puede llegar a la conclusión de que se trata de dotar a los sistemas de la capacidad necesaria para analizar multitud de datos (Big Data), identificar tendencias y generar patrones con el fin de formular predicciones y clasificaciones automáticas. El presente trabajo se centra precisamente en este último concepto, esto es, en las clasificaciones. Gracias a estos sistemas inteligentes, hoy es posible la realización del reconocimiento de matrículas en los parkings privados, la traducción en tiempo real de distintos idiomas con el uso de la cámara del móvil, o el seguimiento de personas a través de las cámaras de vigilancia en las grandes ciudades como Londres y Pekín.

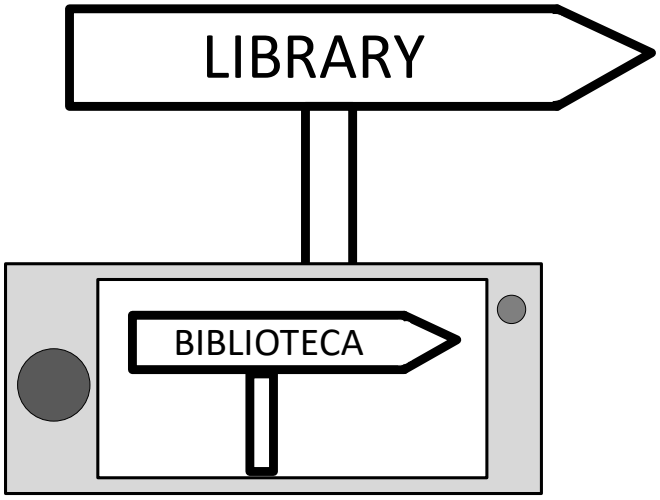


Figura 1. Ejemplo de traducción en tiempo real a través del Smartphone

Cabe destacar la gran cantidad de datos a tratar para el buen fin de las aplicaciones anteriormente mencionadas. Es por ello que existe un desarrollo continuo de herramientas metodológicas para el procesamiento analítico. En el campo de la IA (Russell y Norvig, 2010), destacan algunas técnicas, métodos y enfoques como los algoritmos genéticos (GA, Genetical algorithms), sistemas difusos (SD), sistemas híbridos (HS, Hybrid Sistems) o las redes neuronales (NN, Neural Networks) entre otros.

Una NN es un sistema específicamente diseñado para el tratamiento de información, cuya unidad básica de procesamiento está inspirada en la célula fundamental del sistema nervioso humano: la neurona. Tienen como objetivo asignar elementos a una serie de clases establecidas al efecto; todo ello basado en la capacidad de aprendizaje a través de la experiencia acumulada en un conjunto de ejemplos (aprendizaje automático o “machine learning”).

Dentro del aprendizaje automático, destaca el Aprendizaje Profundo (*Deep Learning*) mediante el cual es posible clasificar distintos elementos dentro de una estructura compleja como puede ser una imagen o sonido. El término *Deep* se refiere al nivel de profundidad en cuanto al número de capas de la red, tal y como se muestra en la Figura 2, con una estructura de red del tipo retro-propagación (Pajares y Cruz, 2017; Haykin, 1994) con diversas capas ocultas que proyectan un vector patrón de entrada a través de ellas hasta llegar a la salida, donde a cada entrada le corresponde una categoría según las clases establecidas.

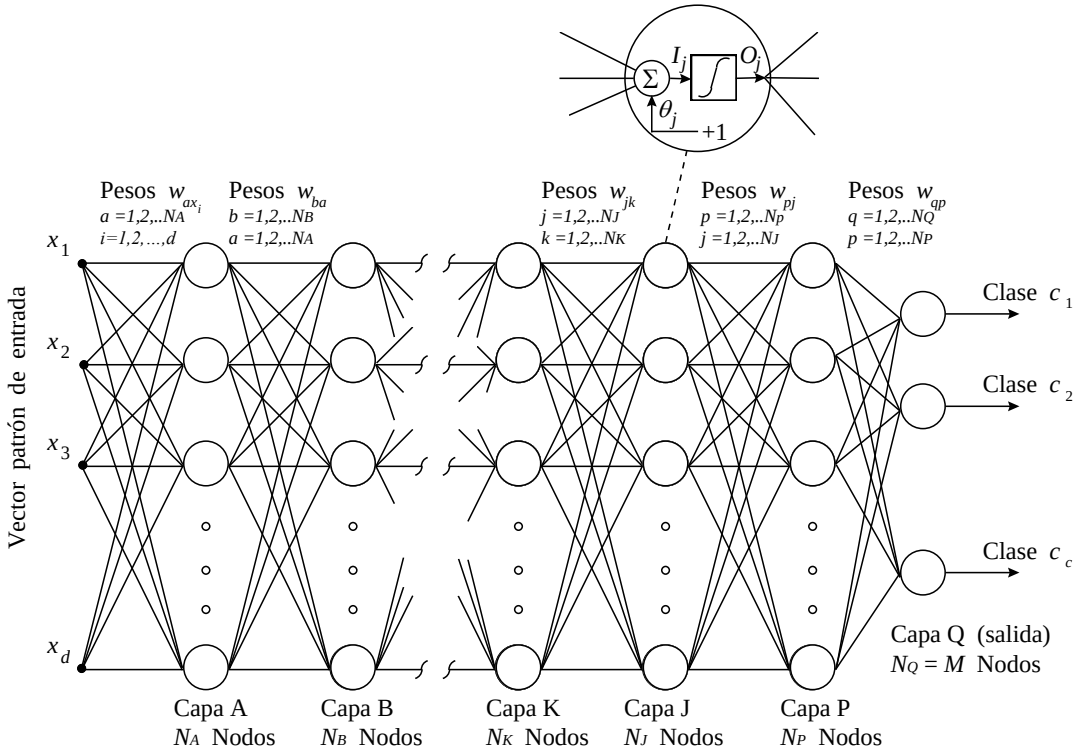


Figura 2. Composición de una red neuronal profunda del tipo retro-propagación.

Por otra parte, dentro de los ámbitos de aplicación destaca de forma especial su utilización en lo que se conoce como Ciudades Inteligentes (Smart Cities), y dentro de éstas, en el control del flujo de vehículos en las ciudades de cara a la eficiencia y sostenibilidad. Es precisamente en este ámbito donde se enfoca el presente trabajo mediante la utilización de las CNN (*Convolutional Neural Network*). La idea básica consiste en identificar sobre una secuencia de imágenes, esto es un vídeo, los objetos en movimiento de la escena, que corresponden esencialmente a los vehículos en una u otra dirección de una vía de circulación en la ciudad. Una vez identificado el movimiento, el objetivo consiste en determinar el tipo de objetos que lo ha producido, y dentro de éstos, discriminar entre distintos tipos de vehículos (coches vistos por delante y por detrás, autobuses, camiones o motos). De esta forma se tiene información no sólo del flujo sino también de los tipos de vehículos en circulación.

Para abordar este problema se han propuesto soluciones (Llerena, 2019) bajo el mismo planteamiento relativo a la detección de vehículos en movimiento. Si bien existe un problema serio en la identificación de sombras asociadas a los vehículos en movimiento. En efecto, en días soleados, los vehículos proyectan su propia sombra sobre la calzada, que además se encuentra en movimiento solidario con el vehículo. Esto hace que, a la hora de detectar los objetos en movimiento en la escena, las mencionadas sombras aparecen solapadas, haciendo que la detección de los vehículos sea más difícil o incorrecta. Con tal

propósito, el presente trabajo aborda esta problemática, cuyo objetivo final consiste en identificar las sombras asociadas a cada vehículo para proceder a su eliminación y posterior identificación sin sombra. En la Figura 3 se muestra una imagen representativa de una secuencia de vídeo en la que se observa con claridad la presencia de las sombras solidarias asociadas a los vehículos en circulación.

Figura 3. Vehículos en tránsito con proyección de sombras

Para dar solución a este problema se propone una estrategia que se sintetiza de acuerdo al diagrama mostrado en la Figura 4. A partir de un número determinado de imágenes de la secuencia de vídeo se genera el fondo de la escena sobre la que se realizará la identificación de sombras, aplicando lo que se conoce como sustracción de fondo (*background substraction*). Como el fondo puede variar debido principalmente a cambios producidos en la iluminación ambiental en entornos de exterior, cada cierto tiempo, o equivalentemente, se actualiza el fondo de la escena. En paralelo, se identifica el movimiento entre cada dos imágenes consecutivas. Con ello se procede a la localización y eliminación de las sombras de los objetos, para realizar la clasificación de los mismos sin sombras según las clases establecidas al respecto.

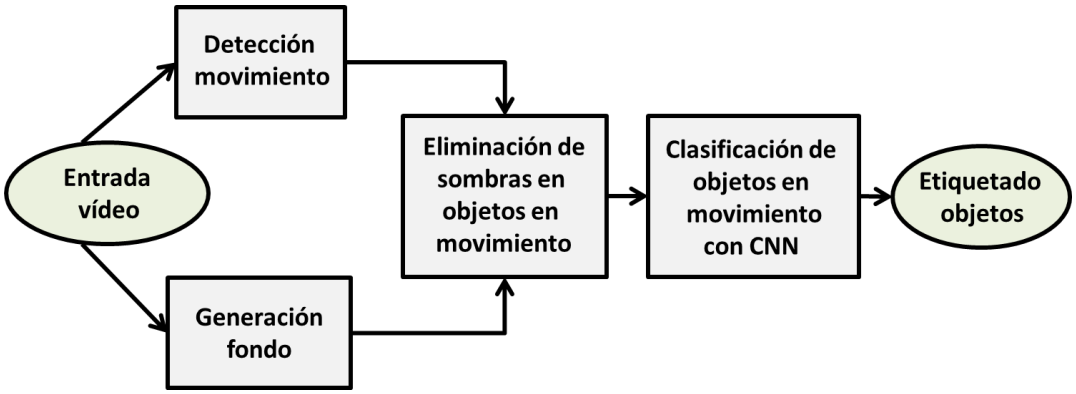


Figura 4. Planteamiento de la propuesta

Bajo el planteamiento anterior, la aportación principal que se realiza en el trabajo propuesto consiste en integrar las funcionalidades que implementan los módulos expuestos en la figura 4, de forma que convenientemente secuenciados desembocan en una mejora sobre la identificación de vehículos en tránsito en ciudades inteligentes.

1.2 Objetivos

El objetivo principal que se plantea en este trabajo es la identificación de vehículos en movimiento mediante la combinación de técnicas de detección en imágenes, y la posterior clasificación con CNN, integrando un módulo adicional para la eliminación de las sombras asociadas.

Los objetivos específicos propuestos se sintetizan como sigue:

1. Determinación del fondo de la escena mediante técnicas de sustracción de fondo (*background subtraction*).
2. Identificación de movimiento en escenas con vehículos en movimiento mediante técnicas basadas en flujo óptico.
3. Clasificación de objetos mediante CNN.
4. Análisis de los resultados entre las diferentes técnicas utilizadas.

1.3 Organización de la memoria

La presente memoria se organiza como sigue.

- La sección 2 describe técnicas de detección de movimiento mediante flujo óptico basado en el método de Lucas-Kanade.
- La sección 3 detalla el método usado para la detección de sombras, incluyendo como procesos aplicados los de la sustracción del fondo y la propia detección de sombras.
- La sección 4 profundiza en la teoría relativa a las CNN, con descripción de la redAlexNet, que es el modelo pre-entrenado para transferencia de conocimiento.
- La sección 5 muestra la estructura de las distintas etapas de la aplicación.
- La sección 6 analiza los resultados obtenidos.
- La sección 7 expone las conclusiones y trabajo futuro

2. Flujo óptico

El flujo óptico es una forma de representación sobre una imagen 2D del movimiento de los objetos producido en la escena 3D, ya sea por el desplazamiento del propio objeto o por el de la cámara. Obviamente en este planteamiento la variable tiempo es crucial en el proceso.

Existen tres tipos diferentes de flujo óptico dependiendo del punto de observación, tal y como demostró Gibson en su trabajo (Gibson, 1966). Por una parte está el flujo frontal, que indica aproximación; el flujo posterior, asociado al alejamiento y los flujos laterales, relativo al desplazamiento lateral.

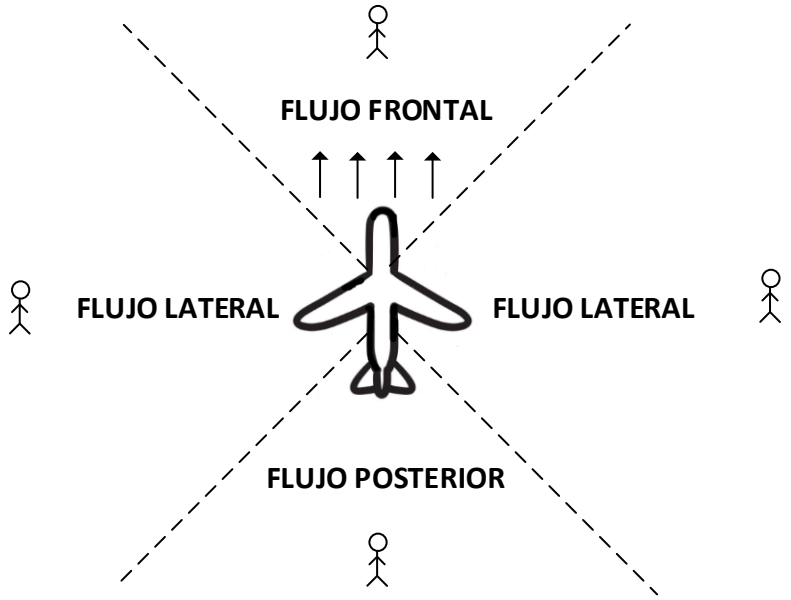


Figura 5. Tipos de flujo óptico

Con el flujo óptico se puede determinar la magnitud y sentido del movimiento de los objetos en la escena y, en particular, de un vehículo transitando por una vía de circulación, como es el caso que se plantea en este trabajo; para este fin se requiere una calibración previa del sistema de captura de imágenes (cámara y óptica) para establecer las transformaciones necesarias entre las imágenes y la escena.

Hay que tener en cuenta que los objetos que se encuentran más cercanos a la cámara aparentarán tener una velocidad mayor que aquellos que estén más alejados, efecto muy característico cuando vamos conduciendo en un coche; los árboles que están más cercanos desaparecen de nuestro rango visual antes que aquellos que se encuentran más alejados.

El principal objetivo del flujo óptico es determinar el campo de movimiento, es decir, la realización de una representación del movimiento sobre una imagen mediante la utilización de vectores en cada píxel. La magnitud, dirección y sentido son un indicativo del movimiento de un píxel entre dos fotogramas consecutivos.

En la imagen de la figura 6 se muestra una representación gráfica de los vectores representativos del flujo óptico asociados con los objetos en movimiento, en este caso, sobre los más cercanos, al haber establecido que la representación se realice sólo para valores del módulo de los vectores superiores a un determinado umbral, cuyo valor es 10. En la propia

representación se observa tanto la dirección y sentido del movimiento como la magnitud, debido a las flechas de distinta longitud.

Figura 6. Flujo óptico en imagen real

Hay dos suposiciones a tener en cuenta y sobre las que se trabaja en el tema de flujo óptico. El primero es que la intensidad del movimiento no cambia de forma brusca entre dos fotogramas consecutivos, es decir, que la velocidad de captación es suficientemente alta como para detectar el movimiento, y la segunda que dos píxeles vecinos tienen un movimiento similar.

A método de ejemplo, se puede considerar un píxel $I(x, y, t)$ que se desplaza una distancia (dx, dy) de un fotograma a otro en un tiempo dt. Siempre y cuando la intensidad de éstos no haya cambiado, se puede decir que:

$$I(x, y, t) = I(x + dx, y + dy, t + dt) \tag{1}$$

Teniendo en cuenta la aproximación por series de Taylor se tendría:

$$\begin{aligned} I(x, y, t) &= I(x + dx, y + dy, t + dt) I(x + dx, y + dy, t + dt) \\ &= I(x, y, t) + \frac{\partial I}{\partial x}\delta x + \frac{\partial I}{\partial y}\delta y + \frac{\partial I}{\partial t}\delta t + \cdots \end{aligned} \tag{2}$$

$$\frac{\partial I}{\partial x}\delta x + \frac{\partial I}{\partial y}\delta y + \frac{\partial I}{\partial t}\delta t = 0 \tag{3}$$

Dividiendo entre dt se consigue derivar la ecuación de flujo óptico:

$$\frac{\partial I}{\partial x}v_x + \frac{\partial I}{\partial y}v_y + \frac{\partial I}{\partial t} = 0, donde \begin{cases} u = {}^{dx}/_{dt} \\ v = {}^{dy}/_{dt} \end{cases} \tag{4}$$

Otra forma de encontrarnos esta ecuación es:

$$I_x v_x + I_y v_y + I_t = 0 \, , donde \begin{cases} I_x = {dI}/{dx} \\ I_y = {dI}/{dy} \\ I_t = {dI}/{dt} \end{cases} \quad (5)$$

$I_x, I_y \, e \, I_t$ corresponden con el gradiente a lo largo de los ejes horizontal, vertical y el tiempo. Al haber una única ecuación y dos incógnitas, no se puede resolver de forma directa, y es por ello que se utilizan métodos como el de Lucas-Kanade (1981; Lucas, 1984) para su resolución, que es un método diferencial ampliamente usado en este ámbito. Para poder solucionar el problema presentado, se parte de una de las dos premisas comentadas anteriormente, el movimiento entre píxeles vecinos es similar o con muy poca diferencia entre ambos; es decir, que tanto el píxel a estudiar, como sus vecinos tienen el mismo flujo. Además, las ecuaciones se resuelven aplicando el criterio de mínimos cuadrados.

La principal ventaja de este método es que consigue obtener un modelo robusto frente al ruido, a diferencia de aquellos basados en un único píxel. Por otra parte, la principal desventaja es su mala precisión en los bordes de los objetos en movimiento, por lo que debe de considerarse como un método de rango local

De forma matemática, Lucas-Kanade se expresa como:

$$\begin{aligned} I_x(p_1)v_x + I_y(p_1)v_y &= -I_t(p_1) \\ I_x(p_2)v_x + I_y(p_2)v_y &= -I_t(p_2) \\ &\vdots \\ I_x(p_n)v_x + I_y(p_n)v_y &= -I_t(p_n) \end{aligned} \quad (6)$$

Donde $p_1, p_2 \ldots p_n$ son píxeles dentro de la vecindad. Por forma análoga, se puede expresar como matrices:

$$Av = b \text{ , de donde } A = \begin{bmatrix} I_x(p_1) & I_y(p_1) \\ I_x(p_2) & I_y(p_2) \\ \vdots & \vdots \\ I_x(p_n) & I_y(p_n) \end{bmatrix}, v = \begin{bmatrix} V_x \\ V_y \end{bmatrix}, b = \begin{bmatrix} -I_t(p_1) \\ -I_t(p_2) \\ \vdots \\ -I_t(p_n) \end{bmatrix} \quad (7)$$

Aplicando el principio de métodos cuadrados (*Bruce D., 1984*) se tiene:

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

$$\begin{bmatrix} V_x \\ V_y \end{bmatrix} = \begin{bmatrix} \sum_i I_x(p_n)^2 & \sum_i I_x(p_n) I_y(p_n) \\ \sum_i I_x(p_n) I_y(p_n) & \sum_i I_x(p_n)^2 \end{bmatrix}^{-1} \begin{bmatrix} -\sum_i I_x(p_n) I_t(p_n) \\ -\sum_i I_y(p_n) I_t(p_n) \end{bmatrix} \quad (8)$$

Una de las peculiaridades de la implementación anterior es que el peso de todos los píxeles es idéntico, sin embargo, existe la alternativa de usar distintos pesos. En la práctica,

es más útil el dar un peso mayor a aquellos píxeles que están más cercanos al píxel central, quedando como:

$$\begin{bmatrix} V_x \\ V_y \end{bmatrix} = \begin{bmatrix} \sum_i w_i I_x(p_n)^2 & \sum_i w_i I_x(p_n) I_y(p_n) \\ \sum_i w_i I_x(p_n) I_y(p_n) & \sum_i w_i I_x(p_n)^2 \end{bmatrix}^{-1} \begin{bmatrix} -\sum_i w_i I_x(p_n) I_t(p_n) \\ -\sum_i w_i I_y(p_n) I_t(p_n) \end{bmatrix} \quad (9)$$

Donde w_i corresponde al peso de cada píxel. Éstos suelen estar definido por una función Gaussiana de la distancia entre el píxel y el píxel central.

3. Detección de sombras

El proceso de detección de sombras consta a su vez de dos procesos diferenciados, a saber: generación del fondo de la escena mediante la técnica de *background subtraction* y la determinación de las sombras propiamente dichas.

3.1. Sustracción de fondo (*background subtraction*)

Uno de los métodos más básicos para este fin es la Diferenciación Temporal (DT). Esta técnica se basa en la comparación de diferentes instantáneas separadas por una constante de tiempo, con el fin de encontrar aquellas regiones que han cambiado de una a la otra dentro de la secuencia. Es inapropiada para aquellas secuencias de imágenes que contienen movimiento indeseado y para aquellos objetos que durante la captura se detienen ya que, si no existe diferencia entre dos instantes de tiempo, el objeto se vuelve invisible para el algoritmo.

Existen distintas variantes de este método, pero la más simple es la diferenciación bruta, y la introducción de una histéresis (Dubuisson y col., 1996), aunque en la implementación que se formaliza en el presente trabajo no se aplica este concepto. De esta forma, teniendo en cuenta que I_n es la intensidad del fotograma $'n'$, entonces:

$$\Delta_n(x,y) = |I_n(x,y) - I_{n-1}(x,y)| \tag{10}$$

Y la imagen en movimiento se define por:

$$M_n(x,y) = \begin{cases} I_n(x,y), & \Delta_n(x,y) \geq T \\ 0, & \Delta_n(x,y) < T \end{cases} \tag{11}$$

En el caso específico de la aplicación que se propone en el presente trabajo, el *background* se obtiene mediante el promediado de la *mediana* para todas las imágenes de la secuencia y sobre los tres canales de color (rojo, verde y azul).

3.2. Detección de sombras

La detección de sombras constituye un elemento esencial de la aplicación. Su detección se basa en un análisis de crominancia (información de color contenida en una señal de vídeo) de los distintos fotogramas contenidos en la secuencia de imágenes a analizar.

Se puede definir el color como aquella luz reflejada por un objeto y que es percibida por nuestro sentido de la vista. El color no deja de ser radiación de una longitud de onda determinada dentro del espectro visible tal y como se muestra en la Figura 7 (Pajares y Cruz, 2007).

Un objeto se define de un color específico porque es la radiación reflejada de la luz solar. La luz solar o luz blanca, contiene todos los colores, (longitudes de onda del espectro visible) de los cuales, una parte son absorbidos por el objeto, y el resto son reflejados.

Desde el punto de vista del ojo humano, todos los colores son una combinación de los tres primarios (rojo, verde y azul). Éstos pueden mezclarse para producir los secundarios:

magenta (rojo y azul), cian (verde y azul) y amarillo (rojo y verde). La mezcla de los tres, ya sean los primarios o los secundarios, en proporciones iguales produce el color blanco.

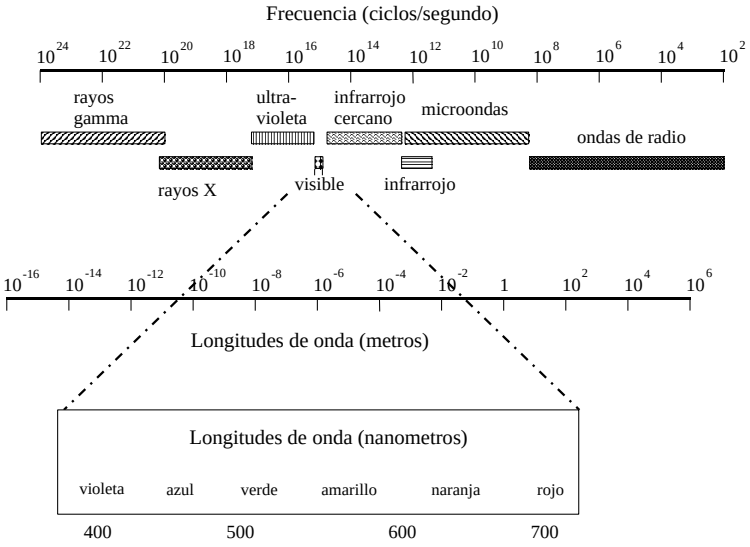


Figura 7. Radiación en el espectro visible en función de las longitudes onda y frecuencia electromagnética

Las características utilizadas para distinguir un color de otro son: ***brillo***, ***matiz*** y ***saturación***. El brillo incorpora la noción cromática de intensidad. El matiz es un atributo asociado con la longitud de onda dominante en la mezcla de longitudes de onda de la luz, viene a ser como el color percibido por el observador, por lo que cuando se dice de un objeto que es rojo, naranja o amarillo, realmente se está especificando su matiz. La saturación se refiere a la pureza relativa o a la cantidad de luz blanca mezclada con un matiz y el grado de saturación es inversamente proporcional a la cantidad de luz blanca añadida (Pajares y Cruz, 2007). Por otra parte, se puede definir la ***cromaticidad*** como el atributo conjunto de matiz y saturación.

3.2.1. Modelos de Color

Con el fin de unificar la definición de un color específico, se crearon unos estándares para representar cualquiera. A esto se le denomina modelo de color. En esencia, es una especificación de un sistema de coordenadas 3-D, donde cada color se representa por un punto.

a) Modelo RGB

Está formado por los colores de luz primarios: rojo (Red), verde (Green) y azul (Blue). La definición de un color se realiza mediante la adición de un porcentaje específico de cada uno de éstos. Este modelo es el más utilizado en monitores, televisiones y cámaras de fotos. Sin embargo, la intensidad en estos dispositivos tiene el rango [0, 255].

En la Figura 8 se aprecia el tetraedro de color para todos los valores posibles. Por conveniencia, se asume que todos los vectores han sido normalizados, es decir, todos los valores de R, G y B están en el rango [0,1]. Los colores primarios corresponden con los extremos en los ejes de coordenadas. Aquellos que se encuentran en las esquinas del plano generado por dos ejes de coordenada se corresponden con los colores secundarios. Por otra

parte, la escala de grises está ubicada en la diagonal que une los colores blanco y negro, mientras que todo el espacio ubicado dentro del tetraedro pertenece al resto.

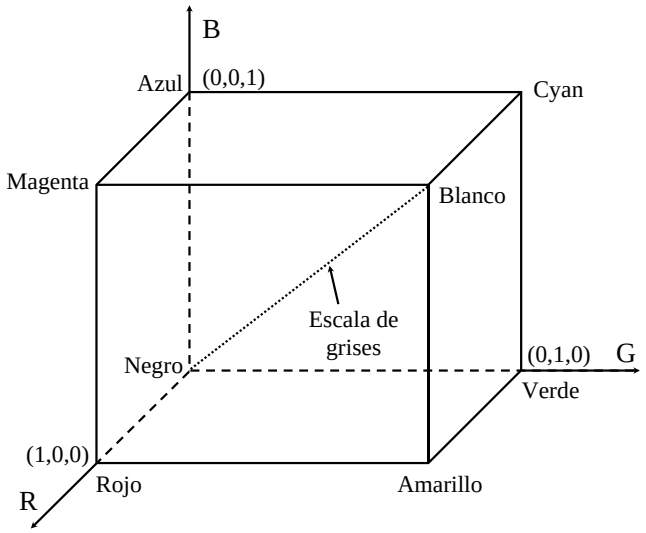
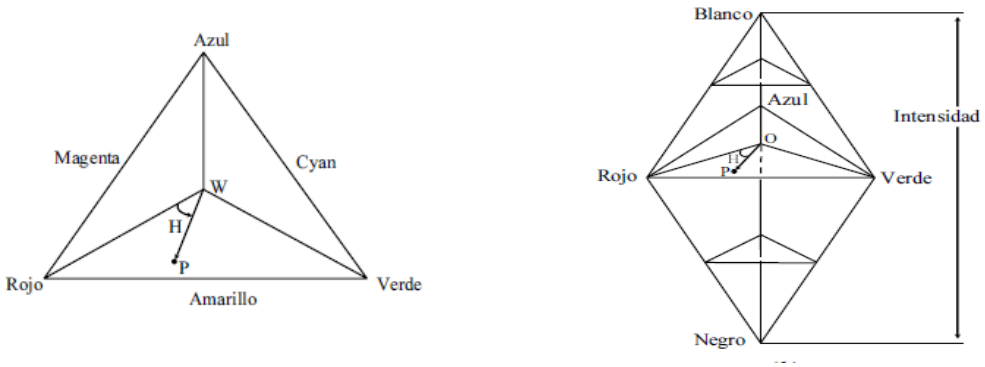


Figura 8. Tetraedro de color RGB

b) Modelo HSI

Este modelo define a los colores en función de su tono, saturación e intensidad (Hue, Saturacion and Intensity).

- El **tono** está relacionado con la longitud de onda del color dominante en el objeto. Cuando se indica que un color es amarillo, azul o verde, lo que se está indicando realmente es su tonalidad.
- La **saturación** es un indicativo de la cantidad de luz blanca mezclada con el color dominante. Con este atributo se puede diferenciar entre colores intensos o pálidos. Por ejemplo, el azul del cielo es un color poco saturado, mientras que el azul marino es un color más opaco, y por ende, más saturado.
- La **intensidad** hace referencia a la iluminación percibida del objeto; es decir, de la sensación de que un objeto refleje más que otro.



(a) (b)

Figura 9. Modelo de color HSI

c) Conversión de RGB a HSI

Los colores en el modelo HSI se definen con respecto a los valores normalizados de rojo, verde y azul, a partir de los colores primarios RGB.

$$r = \frac{R}{R+G+B}; g = \frac{G}{R+G+B}; b = \frac{B}{R+G+B} \tag{11}$$

Además, se debe cumplir que:

$$r + g + b = 1 \tag{12}$$

Por otra parte, la intensidad se calcula como:

$$I = \frac{1}{3}(R+G+B) \tag{13}$$

Para el cálculo del tono y la saturación se sigue el procedimiento descrito en (G. Pajares, 2001), de donde se tiene que:

$$H = cos^{-1}\left(\frac{\frac{1}{2}((R-G)+(R-B))}{\sqrt{(R-G)^2+(R-B)(G-B)}}\right) \tag{14}$$

$$S = 1 - \frac{\min[R,G,B]}{R+G+B} \tag{15}$$

El modelo HSI, es el que mejor reproduce el comportamiento visual de las personas, además de ser sensitivo a los cambios de luminancia provocados por las sombras. Es por ello que resulta de gran utilidad para esta tarea.

d) Modelo HSV

Este modelo es similar al HSI salvo que la intensidad varía de negro a blanco en un único prisma, a diferencia del doble prisma que se tenía anteriormente tal y como muestra la siguiente Figura 10.

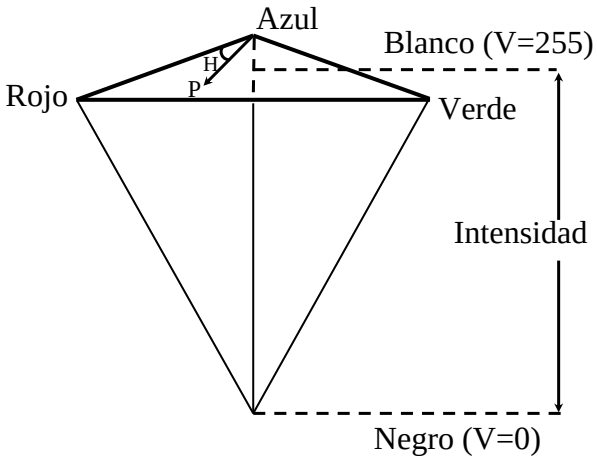


Figura 10. Diagrama del Modelo HSV

3.2.2. Localización de sombras

De forma empírica, en distintas imágenes y vídeos se aprecia que un píxel en el que existe sombra contiene una componente de intensidad, *I* o equivalentemente *V* menor; es decir, un píxel de color azul contenido en una sombra, pasará a ser un azul oscuro, pero sigue manteniendo su cromaticidad, aunque manifiesta una saturación de color en las correspondientes componentes *H* y *S*, tras la transformación del modelo de color RGB al HSI o equivalentemente al modelo HSV (Cucchiara y col., 2001, 2003). De esta forma se obtienen las correspondientes componentes para cada píxel (*x*,*y*) en el modelo HSV, es decir, $V_t(x,y)$ y $V_b(x,y)$, que son las intensidades de la imagen en el instante *t* y del *background* (indicada por el subíndice, *b*). Del mismo modo se obtienen las componentes *H* y *S* de la imagen en *t* y del *background*. De esta forma un píxel con sombra *SPt* en la posición (*x*,*y*) se determina como:

$$SP_t(x,y) = \begin{cases} 1, \ i \ \alpha \leq \frac{V_t(x,y)}{V_b(x,y)} < \beta \wedge |H_t(x,y) - H_b(x,y)| \leq \tau_H \\ \qquad \wedge |S_t(x,y) - S_b(x,y)| \leq \tau_S \\ 0, \qquad\qquad de\ otro\ modo \end{cases} \tag{16}$$

Con valores $0 < \alpha, \beta, \ \tau_H, \tau_S < 1$ se llega a interpretar que una sombra oscurece uno de los píxeles de la imagen, pero no demasiado. Valores de β entre 0.9 y 0.97 son típicos. Sin embargo, la utilización de valores superiores hace que la implementación sea demasiado sensible al ruido. Por otra parte, α suele variar en el rango entre 0,75 y 0,8, para los cuales, un punto sombreado se vuelve más oscuro en torno al 20%. Un valor típico para τ_H y τ_S es 0.15 (Elgammal, 1999). Por otra parte, este método ha sido comparado favorablemente con otros métodos existentes en la literatura para detección de sombras (Prati y col., 2001).

4. Redes neuronales convolucionales

Las CNN son un tipo especializado de redes neuronales (LeCun, 1989), con una topología basada en rejilla, específicamente diseñadas para el procesamiento de imágenes (Goodfellow, 2016; Dumoulin, 2016).

La base de estas redes se sitúa en la operación de **convolución**. También es muy común aplicar una operación denominada ***pooling***, que viene a ser una agrupación de píxeles mediante alguna operación realizada sobre una ventana en los resultados intermedios que se obtienen tras la convolución. Además, poseen las conocidas funciones de **activación,** que permiten transformar determinados valores mediante alguna de tales funciones, destacando las unidades **ReLU** (Rectified Linear Units).

Para evitar sobre-ajustes o infra-ajustes en el cómputo de los pesos se utilizan las denominadas capas **dropout.**

A veces conviene realizar también una operación de **normalización** de los datos para que todos contribuyan por igual en los diferentes cálculos y ajustes.

Es habitual que la última capa en una red de clasificación proporcione valores de probabilidad de pertenencia de los objetos a las clases, para ello se utilizan las denominadas capas ***softmax***.

Este apartado finaliza con la exposición del modelo de red pre-entrenada AlexNet (ImageNet, 2019; Russakovsky y col., 2015; Krizhevsky y col., 2012; BVLC AlexNet Model, 2019), que permite transferir conocimiento al sistema que se propone.

4.1. Operación de convolución

Esta operación involucra dos funciones con valores reales como argumento. Para clarificar el concepto, supóngase que se tiene un altavoz que emite sonidos de intensidad variable, proporcionando una salida en una determinada posición *x* en el tiempo *t*, *x*(*t*). Tanto *x* como *t* son valores reales, por lo que se pueden obtener distintas intensidades de sonido en diferentes instantes de tiempo. Si a esto se añade que la señal de sonido puede estar contaminada con un cierto ruido, para obtener una señal más limpia, lo propio 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.

Esto puede hacerse mediante la función de promediado $w(a)$, donde a representa el alejamiento de la medida en el tiempo.

$$s(t) = \int x(a)\, w(t-a) da \tag{17}$$

La operación convolución se denota como:

$$s(t) = \int (x * w)\,(t) \tag{18}$$

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 del sonido, 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, y considerando que *x* y *w* se definen sólo en el tiempo *t*, se define la convolución discreta como sigue.

$$s(t) = (x * w)(t) = \sum_{a=-\infty}^{\infty} x(a)w(t-a) \tag{19}$$

En aprendizaje máquina, la entrada es un vector o matriz multidimensional de datos. 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 por separado, los valores que no pertenecen a esos elementos son nulos, lo que se traduce 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, resultando.

$$S(i,j) = (I * K)(i,j) = \sum_m \sum_n I(m,n)K(i-m, j-n) \tag{20}$$

La convolución es conmutativa, lo que significa que la expresión anterior, puede escribirse de forma equivalente como sigue en la ecuación 21, resultando en una menor variación en los valores de *m* y *n*; es decir, consiguiéndose una implementación más eficaz.

$$S(i,j) = (K * I)(i,j) = \sum_m \sum_n I(i-m, j-n)K(m,n) \tag{21}$$

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* aumente, 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, lo que sí resulta relevante en este contexto es lo que se conoce como correlación cruzada, tal y como sigue.

$$S(i,j) = (K * I)(i,j) = \sum_m \sum_n I(i+m, j+n)K(m,n) \tag{22}$$

A veces, correlación cruzada y convolución son términos que se confunden en la terminología especializada. En el contexto de aprendizaje máquina, el algoritmo de aprendizaje estima los valores apropiados del núcleo en cualquiera de los dos casos. La Figura 11 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; por lo que se obtiene una imagen de menor dimensión que la original. En este caso concreto, se pasa de tener dimensión 6×7 a 4×5. No obstante, si se desea una imagen de la misma dimensión, el método a aplicar 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 ***cero padding***.

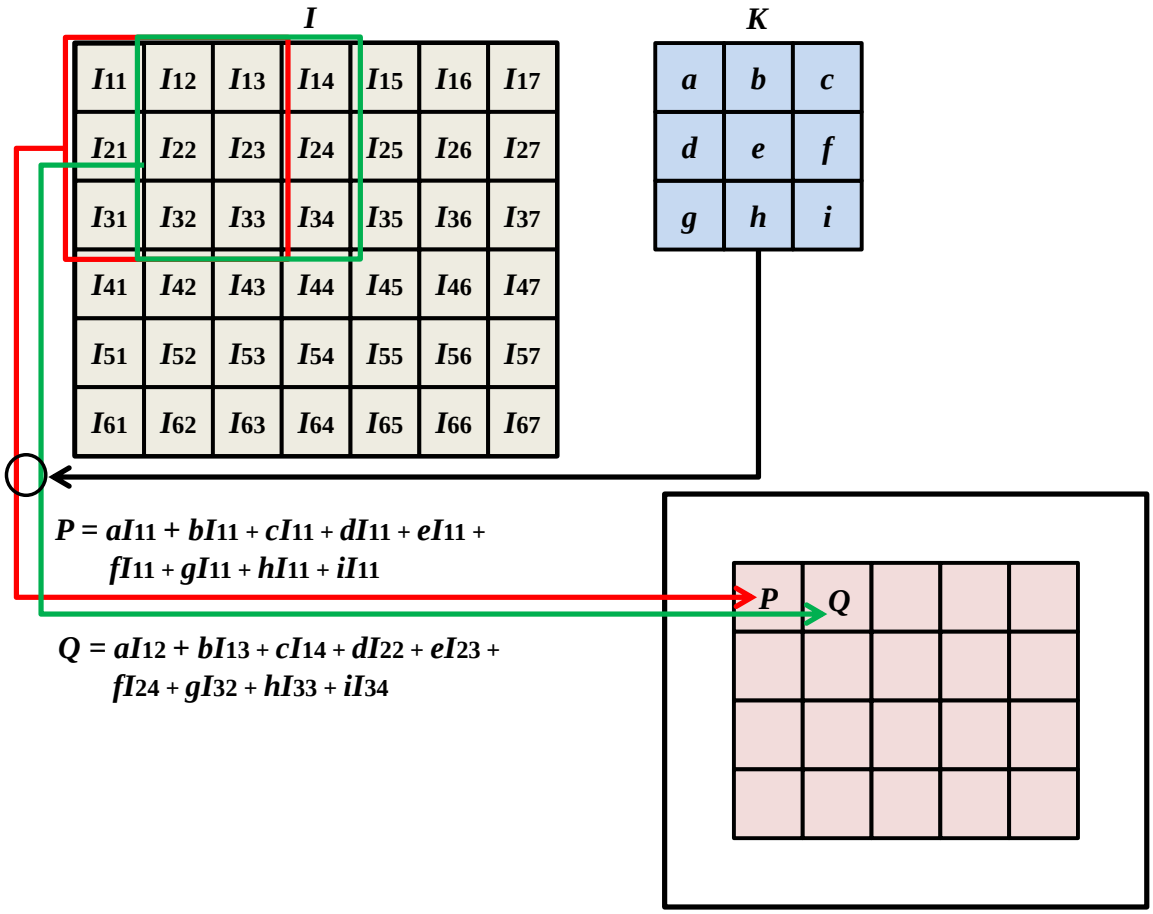


Figura 11. Ejemplo de convolución 2-D

Como se ha indicado previamente, la convolución representada en la Figura 11 es de tipo 2-D. Para disponer de un tipo 3-D se necesitaría disponer de una matriz tridimensional. Por ejemplo, una imagen RGB con sus respectivas componentes de color para cada píxel. Además, este término se puede generalizar a convoluciones tipo N-D, donde el número de dimensiones es *N*.

Los desplazamientos del caso anterior son unidimensionales, es decir se barren todas las columnas de la matriz. A pesar de ser un método bastante común, no es siempre igual, sino que se pueden producir saltos de dimensiones distintas a uno. A esto se le conoce como ***stride***.

Los distintos núcleos que definen una convolución discreta se pueden definir como una permutación del tipo (n, m, $k_1 \ldots k_n$), 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_i ≡ número de ceros concatenados al comienzo y al final del eje j (cero *padding*).

La Figura 12 muestra un ejemplo ilustrativo sobre el proceso de convolución bidimensional (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.

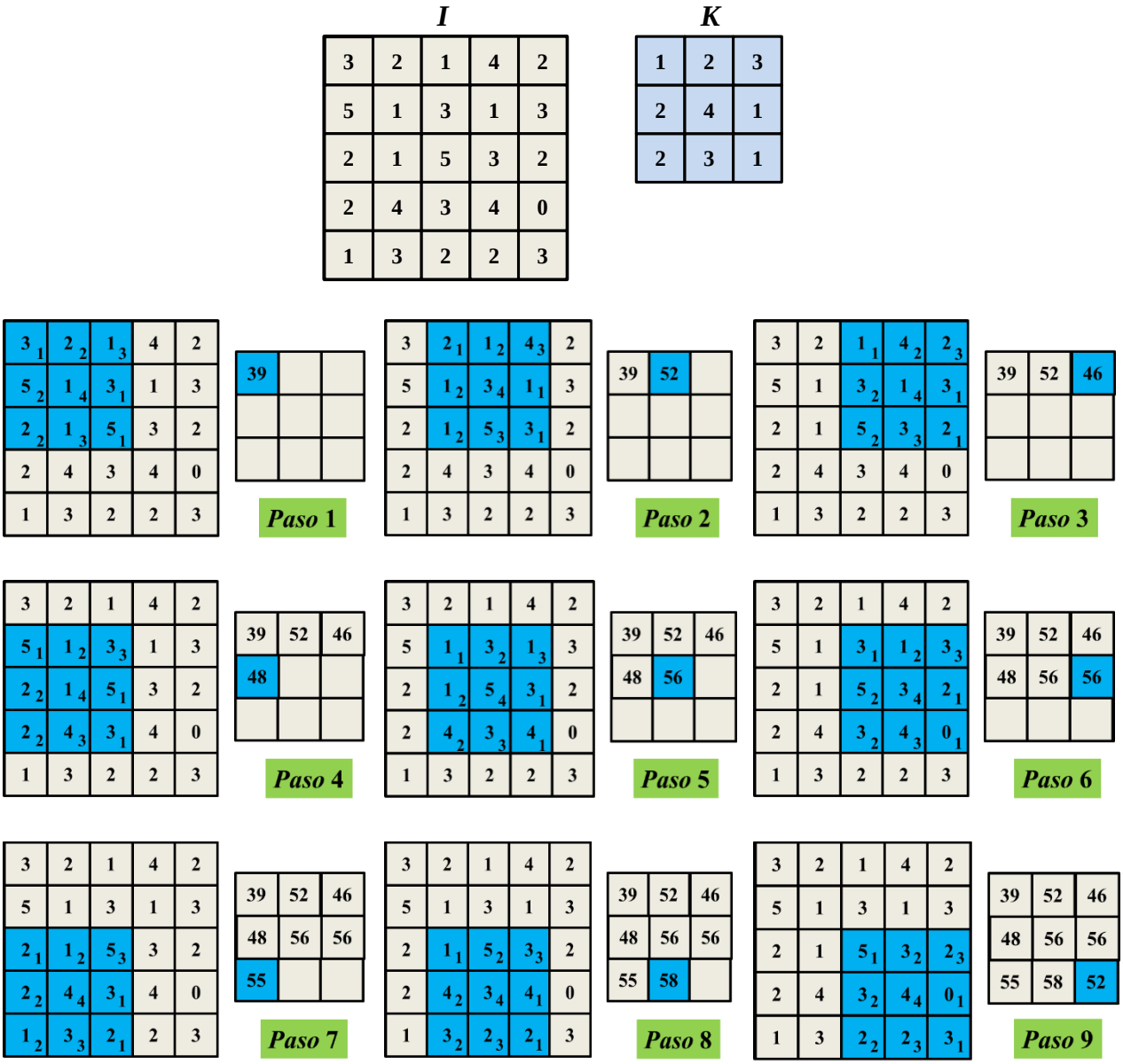


Figura 12. Ejemplo de una convolución discreta con N=2, $i_1 = i_2 = 5$, $k_1 = k_2 = 3$, $s_1 = s_2 = 1$, $p_1 = p_2 = 0$

La operación *stride* puede considerarse como una forma de submuestreo cuando el desplazamiento es superior a uno. Por otra parte, como puede deducirse de la Figura 13, 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.

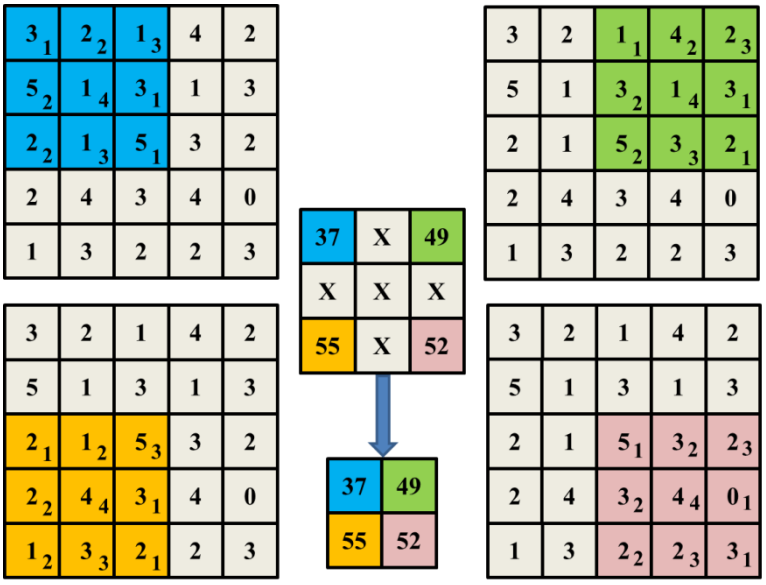


Figura 13. Forma alternativa de interpretar los desplazamientos del núcleo I

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 14 se representa una operación de convolución a partir de tres mapas de características de entrada con el fin de 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 para formar los mapas 2, 3 y 4 respectivamente. Finalmente, los cuatro mapas de salida son agrupados de forma conjunta.

En la Figura 15 se muestran tres ejemplos ilustrativos de convoluciones. Los tres casos tienen una imagen de entrada de profundidad 3 y dimensión 6×6. Los casos que aparecen serían:

- ***Caso* 1**: un filtro de dimensión 3×3×3 para obtener una imagen resultante de dimensión 4×4×1.
- ***Caso* 2**: un filtro de dimensión 1×1×3 para obtener una imagen resultante de dimensión 6×6×1.
- ***Caso* 3**: dos filtros de dimensión 1×1×3 para obtener una imagen resultante de dimensión 6×6×2. En este último, y a modo de ejemplo, si el número de filtros es 32 en lugar de 2 el resultado sería de dimensión 6×6×32.

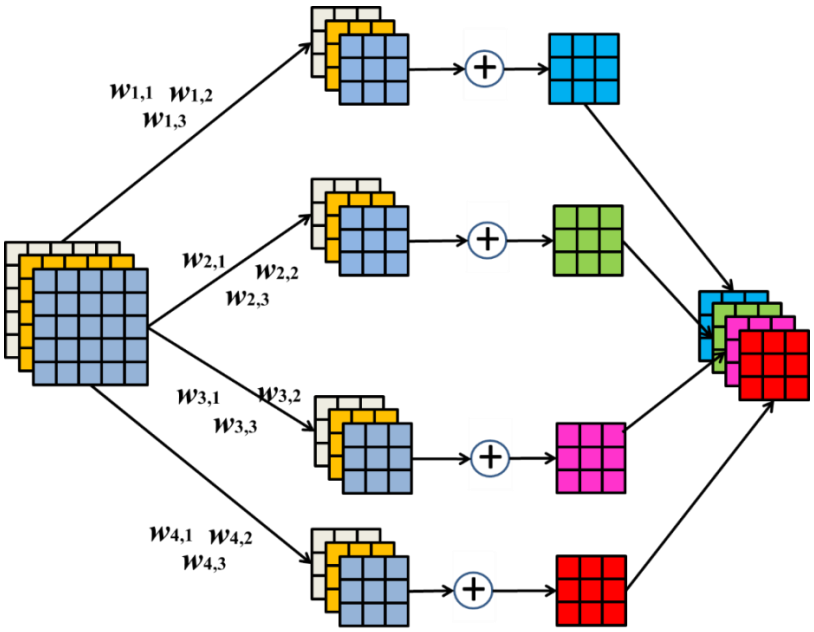


Figura 14.Forma alternativa de interpretar los desplazamientos de núcleo II

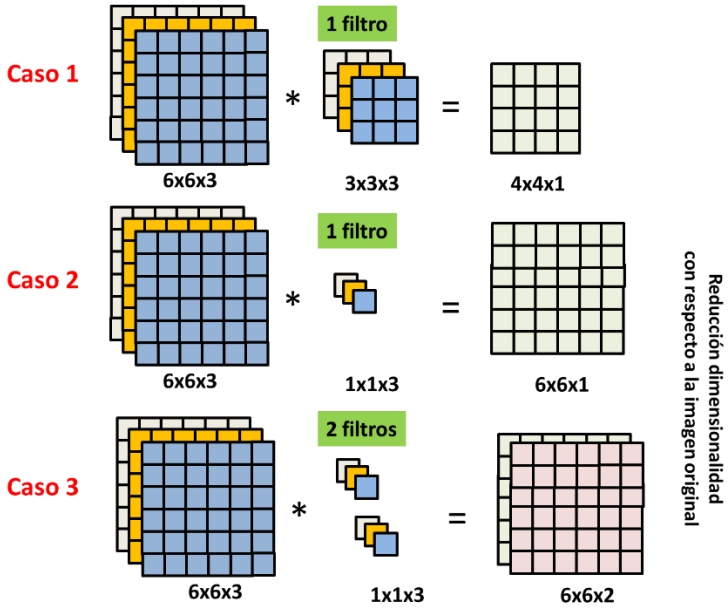


Figura 15. Forma alternativa de interpretar los desplazamientos del núcleo III

4.2. *Pooling* (agrupación o concentración)

Es una operación de operativa similar a la de convolución cuyo objetivo es la reducción de dimensionalidad, con el fin de reducir la carga computacional. Es muy útil para la extracción de características dominantes presentes.

Existen dos tipos de operaciones de esta naturaleza, clasificándose en función de la operación efectuada para este fin. Por un lado, está el ***Max. Pooling***, usando la operación 'máximo'; por otro lado, está el ***Avg. Pooling***, utilizando la operación de promediado estadístico (Boureau, 2010*a*,*b*; Saxe, 2011; Zhou, 1988).

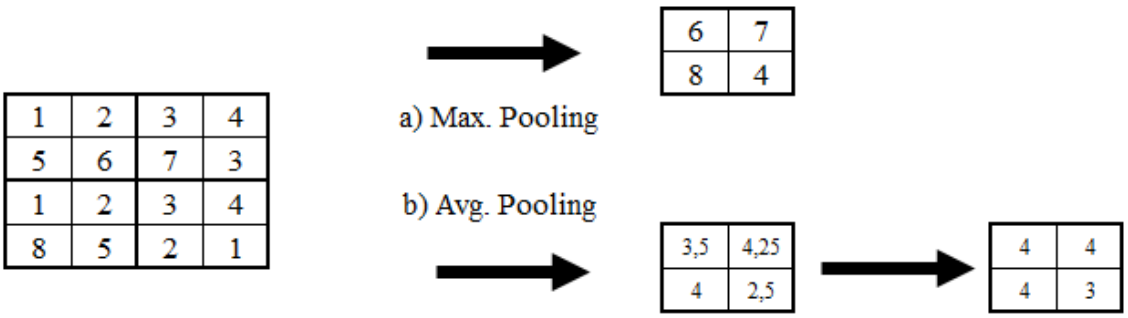


Figura 16. Tipos de operación de pooling: a) Max. Pooling b) Avg. Pooling

La principal ventaja de la utilización de esta operación es que confiere una invarianza a pequeñas traslaciones en la entrada, además de no necesitar una potencia de cálculo elevada, sino todo lo contrario, es bastante rápida.

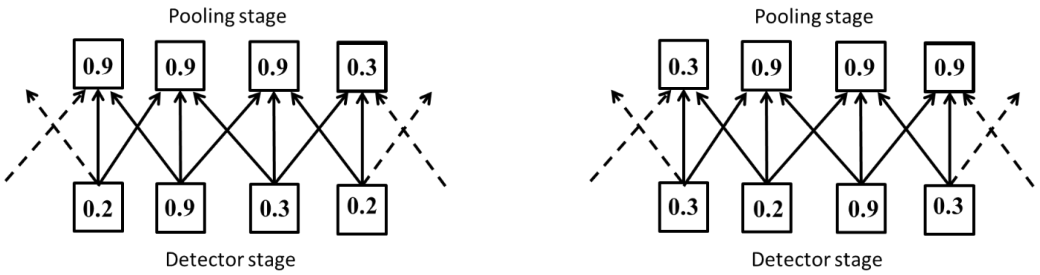


Figura 17. Invarianza en max pooling.

4.3. Funciones de activación y Rectified Linear Units (ReLU)

Una función de activación clásica es la función sigmoidal definida como sigue:

$$f(a,x,c) = \frac{1}{1+e^{-a(x-c)}} \tag{23}$$

Dependiendo del signo del parámetro ***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 Figura 18 muestra la representación de sendas funciones sigmoide con distintos parámetros:

- (a) a = 2; c = 4;
- (b) a = −2; c = 4).

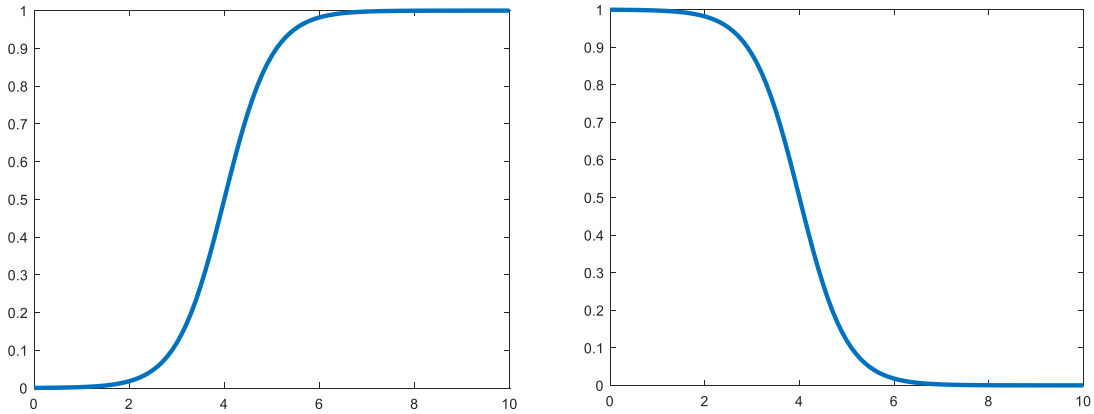


Figura 18. Funciones sigmoides

La función sigmoide presenta dos problemas: a) **saturación del gradiente**, así cuando el valor de la función de activación se aproxima a los extremos 0 "o" 1, el gradiente de la función tiende a 0, lo que repercute en el ajuste de los pesos de las redes; b**) Pesos positivos de forma continua**, el valor medio de la función de salida no es 0, lo que origina que los pesos tienden a ser positivos. Estas dos cuestiones provocan una convergencia lenta de los parámetros, afectando a la eficiencia del entrenamiento.

La función tangente hiperbólica (*tanh*) es una variante de esta función, en la que se proporcionan salidas reales en el rango [-1, 1]. Se puede definir como $tanh(x) = 2sigmoid(2x) - 1$, sin embargo, presenta el mismo problema de saturación del gradiente que la función sigmoide. La Figura 19 muestra la representación de la función tangente hiperbólica.

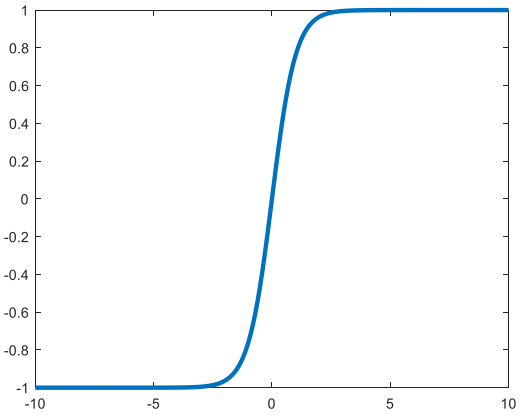
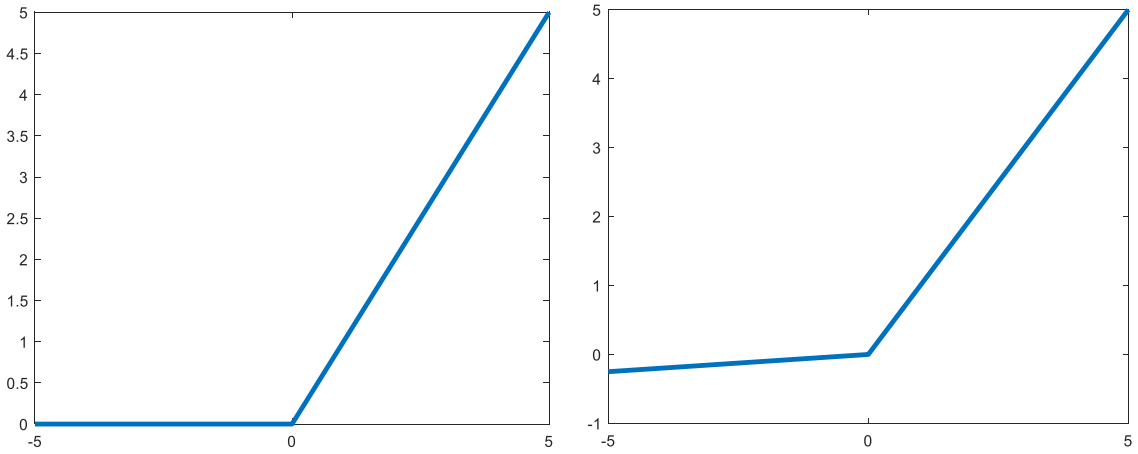


Figura 19. Funciones tanh

La función **ReLU (Rectified Linear Unit)** $f(x) = max(0, x)$, representada en la Figura 20(*a*) tiene las siguientes características: a) **Gradiente no saturado**, por el hecho de que x > 0; de esta forma, 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, ante entradas con gradientes negativos altos, la neurona puede quedar en estado de desactivación (neurona muerta). Esto se puede evitar al inicializar cuidadosamente los pesos o utilizar ReLU con "fugas", es decir, una

ReLU en la que su salida es multiplicada de forma lineal por un valor pequeño cuando la entrada es negativa, f(x) = *max*(0.01*x*, *x*), Figura 20.(b).



(a) (b)

Figura 20. Funciones ReLU

4.4. Capas *Dropout*

Un problema que puede surgir en las redes neuronales es lo que se conoce como "overfitting". En redes neuronales se ajustan un número elevado de pesos en numerosas neuronas. La Figura 21 muestra claros ejemplos de *underfitting* y *overfitting* frente a un posible ajuste correcto. La cuestión es qué ajuste es mejor, el que pasa por todos los puntos mediante un polinomio con un grado elevado (derecha), o una línea recta (izquierda). En el primer caso, a pesar de que el modelo se ajusta bien a los datos, este no es adecuado en el momento de decisión.

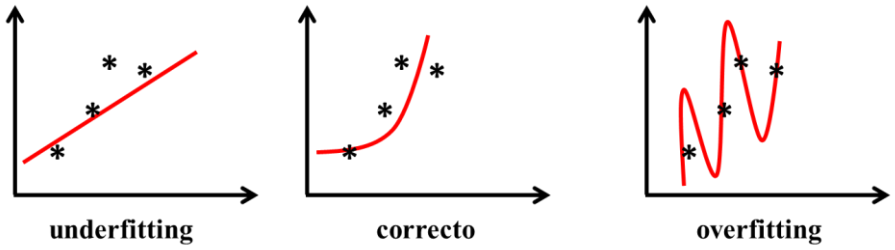


Figura 21. Tipos de ajuste

Un método para evitar el *overfitting* es aplicar regularización, es decir, modificar la función objetivo a minimizar añadiendo términos adicionales que penalizan pesos con valores elevados. Así, por ejemplo, dada la función objetivo J se le puede añadir el término $\lambda f(\theta)$ de forma que $f(\theta)$ crece a medida que el argumento θ también, siendo λ 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. 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 mejor se ajustan al conjunto de datos de entrenamiento.

No obstante, en el ámbito de las redes neuronales, para paliar el efecto del overfitting se propone anular determinado tipo de neuronas, incluidas sus salidas, durante el proceso de entrenamiento (Srivastava, 2014; Krizhevsky, 2012). A este proceso se le conoce como ***Dropout***. La selección de las unidades a anular se realiza de forma aleatoria, así, la red

resultante es la que mantiene las unidades que sobreviven a este proceso. No obstante, en lugar de eliminar neuronas, también cabe la posibilidad de mantenerlas asignándoles un parámetro *p* de forma aleatoria que indica la probabilidad de que las neuronas se queden activadas durante la fase test del entrenamiento. Este método solo se usa en la fase de entrenamiento, durante la fase de pruebas no se desactiva ninguna neurona.

En la Figura 22 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.

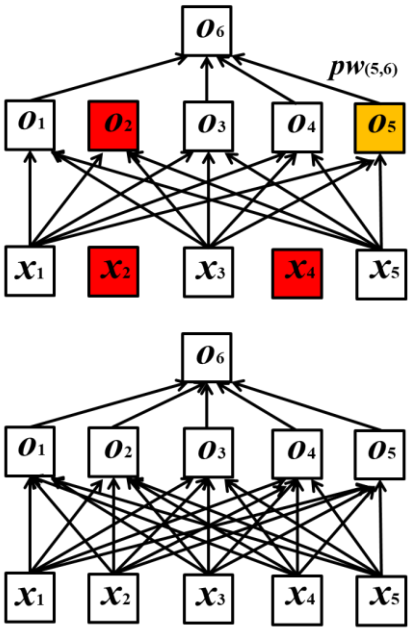


Figura 22. Dropout

4.5. Normalización

Como se ha mencionado previamente, la operación ReLU no requiere normalización para prevenir la saturación (Krizhevsky, 2012). 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 no linealidad ReLU, la actividad de la respuesta normalizada $b^i_{x,y}$ viene dada por la expresión,

$$b^i_{x,y} = \frac{a^i_{x,y}}{\left(k + \propto \sum_{j=\max(0,i-n/2)}^{\min(N-1,i+\frac{n}{2})} (a^i_{x,y})^2\right)^2} \qquad (24)$$

donde la suma se extiende sobre *n* mapas adyacentes generados por los núcleos en la misma posición (*x*,*y*) y *N* es el número total de núcleos en la capa.

El ordenamiento de los mapas es arbitrario y se determina antes de que comience el aprendizaje. Esta clase de respuesta normalizada implementa una forma de inhibición lateral inspirada en neuronas reales, creando una competición para actividades elevadas frente a salidas de neuronas obtenidas usando diferentes núcleos. Las constantes *k*, *n*, α y β son

parámetros cuyos valores se determinan usando un conjunto de validación. En Krizhevsky, A. (2012) se proponen los siguientes valores $k = 2$, $n = 5$, $\alpha = 10^{-4}$ y $\beta = 0.75$. En el mismo trabajo, se indica que esta normalización se realiza en ciertas capas tras la aplicación de ReLU.

En Jarrett (2009) se propone otro esquema 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,

$$b^i_{x,y} = a^i_{x,y} - \sum_{p,q} w_{pq} a^i_{x+p.y+q} \tag{25}$$

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

$$d^i_{x,y} = \frac{b^i_{x,y}}{max\left(c, \sigma_{x,y}\right)} \tag{26}$$

donde $\sigma_{x,y} = \sum_{p,q} w_{pq} \left(b^i_{x+p,y+q}\right)^{\frac{1}{2}}$ y $c = mean\left(\sigma_{x,y}\right)$. Por tanto, en la expresión (2.10) 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 (Lyu y Simoncelli, 2008).

4.6. Softmax

La función sofmax o función exponencial normalizada, que aparece generalmente en las últimas capas ocultas, definida en (2.11). Se emplea para proyectar un vector n-dimensional, x de valores reales en un vector n-dimensional sofmax(x) de valores reales en el rango [0,1],

$$softmax(x)_i = \frac{exp(x_i)}{\sum_{j=1}^{n} exp(x_j)} \text{ para } i = 1,..,n \text{ y } x = \left(x_1, x_2, \dots, x_n\right) \in \mathbb{R}^n \tag{27}$$

En definitiva, se aplica la función exponencial a cada elemento xj del vector de entrada x y se normalizan esos valores por la suma de las exponenciales, lo que asegura que la suma de las componentes del vector de salida softmax(x) = 1. Es posible, todavía, modificar los exponentes de las exponenciales con el fin de crear funciones de distribución de probabilidad más concentradas alrededor de posiciones de los valores de entrada mayores, lo cual se consigue como sigue con $\alpha x_i > 0$o $-\alpha x_i > 0$.

$$softmax(x)_i = \frac{exp(\alpha x_i)}{\sum_{j=1}^{n} exp(\alpha x_j)} \text{o bien } softmax(x)_i = \frac{exp(-\alpha x_i)}{\sum_{j=1}^{n} exp(-\alpha x_j)} \text{ para } i = 1,..,n \tag{28}$$

4.7. Arquitectura de la red AlexNet

En la Figura 23 se muestra la estructura de la red AlexNet con ilustración gráfica de las operaciones involucradas con indicación de las dimensiones de filtros para las operaciones de Convolución (*conv*), ReLU (*relu*), Normalización (*norm*), Pooling (*pool*), dropout (*drop*) o *softmax*. Además, se indican los números de relación, que establecen la relación entre entre las dimensiones de salida de cada capa (*o*) considerando los parámetros de entrada (*i*, *k*, *p*, *s*).

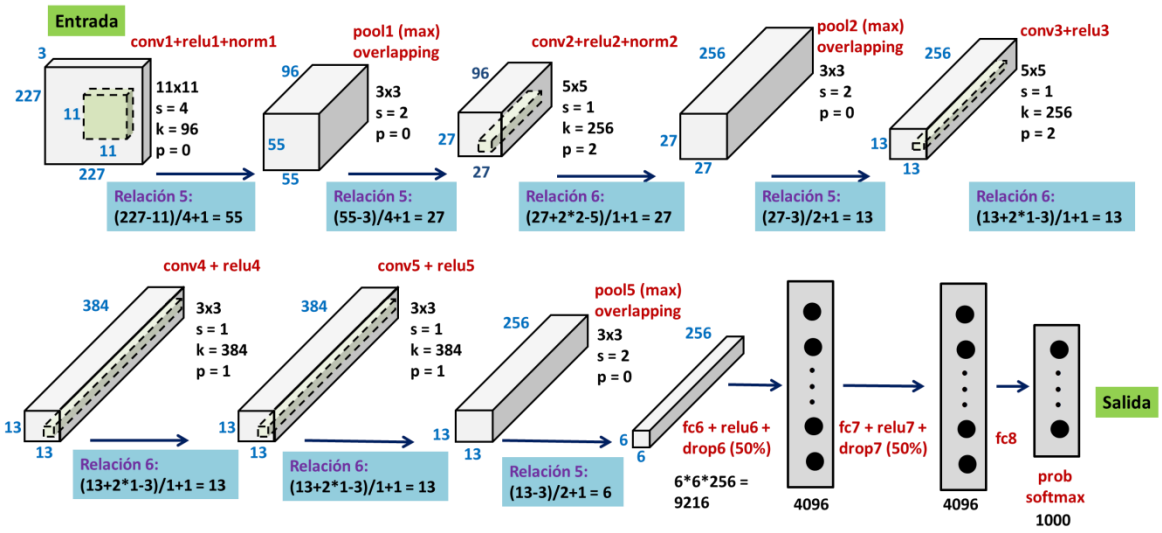


Figura 23. Estructura de capas de AlexNet

5. Diseño de la aplicación

El objetivo general de la implementación consiste en localizar y etiquetar todos los vehículos que aparecen en un vídeo de una cámara de videovigilancia de carretera. Las fuentes provienen de distintos vídeos realizados en puentes de autovías y carreteras, con distintas condiciones climáticas y en distintas horas del día. Esto hace que la visibilidad de los vehículos, el nivel de sombras y la cantidad de tráfico varíen de uno a otro. El diseño tiene en cuenta estos aspectos e intenta conseguir la mayor robustez posible.

De forma más específica, el objetivo de este trabajo tiene como núcleo esencial localizar y eliminar las sombras de los distintos vehículos, ayudando, de esta forma, a la clasificación dentro de la CNN.

Para el diseño de la aplicación se han utilizado los siguientes recursos:

1) Hardware:
   - Procesador: Intel ® Core™ i5-4460U CPU @3.20GHz 3.20GHz
   - Memoria RAM: 16 GB
   - Almacenamiento interno: SSD 250GB
   - Tipo de sistema: Sistema operativa de 64bits, procesador x64
   - Tarjeta gráfica: Nvidia GeForce GTX 760

2) Software:
   - Sistema operativo: Windows 10 Professional 64bits
   - Procesamiento y desarrollo: Matlab (MathWorks, 2018), así como los toolboxes de Deep Learning, Image Processing y Computer Vision con toda su potencialidad.

5.1. Estructura de la aplicación

Tal y como se muestra en el diagrama de la Figura 4, el diseño consta de una serie de funcionalidades que se traducen en los siguientes módulos:

- Entrada de video.
- Detección de movimiento.
- Generación de fondo (*obtención del background*).
- Eliminación de sombras en objetos en movimiento
- Clasificación de objetos con CNN.

5.1.1. Entrada de video

Consiste en determinar la fuente de entrada de imágenes a partir de un vídeo, obteniendo, además, toda la información asociada al mismo tales como: imágenes por segundo (*frames*) expresado como *fps*, resolución de cada *frame* en píxeles o número de fotogramas.

Sobre cada *frame* se aplica un suavizado gaussiano (Pajares y Cruz, 2007), cuya finalidad es reducir el ruido que se haya podido introducir por movimiento durante la grabación. Este filtro realiza un promediado de cada píxel con sus vecinos.

Cada uno de los *frames* que conforman el vídeo en su conjunto, se encuentran en el modelo de color RGB, siendo transformados al modelo HSV con el fin de obtener las correspondientes componentes de intensidad (V), matiz (H) y saturación (S) requeridas en el correspondiente proceso de detección de sombras.

Para este fin se utilizan las siguientes funciones de Matlab:

- ***VideoReader***. Permite crear un objeto con las características propias del vídeo a procesar. Haciendo referencia al susodicho se obtienen las características del vídeo indicadas previamente (*fps*, resolución).
- ***Read***. Obtiene un fotograma específico.
- ***Rgb2hsv***. Obtiene el modelo HSV de una imagen.
- ***Fspecial.*** Para la obtención del filtro gaussiano

5.1.2. Detección del movimiento

Para la detección de movimiento se calcula el flujo óptico entre dos fotogramas consecutivos mediante la aplicación del método de Lucas-Kanade descrito en la sección dos. Se utiliza la función *EstimateFlow,* que utiliza las imágenes de intensidad sobre las que se aplican los procesos previstos por el método de Lucas-Kanade.

En la imagen de la Figura 24(a) se muestra el resultado de aplicar Lucas-Kanade entre dos imágenes consecutivas. Se observa en los vehículos en movimiento los vectores de flujo asociados con su correspondiente dirección y sentido, así como la magnitud que viene representada por las longitudes de los vectores que les representan.

En función de la magnitud del movimiento se detectan las zonas con una magnitud suficiente, procediendo a la binarización de la imagen como sigue. Dada la imagen que representa dicha magnitud (M) la imagen de movimiento binaria (B) se obtiene como sigue $B = M > (\mu_M + 2\sigma_M)$ donde μ_M y σ_M son**,** respectivamente, la media y la desviación estándar de *M*, de esta forma se obtiene la imagen mostrada en la Figura 24 (b).

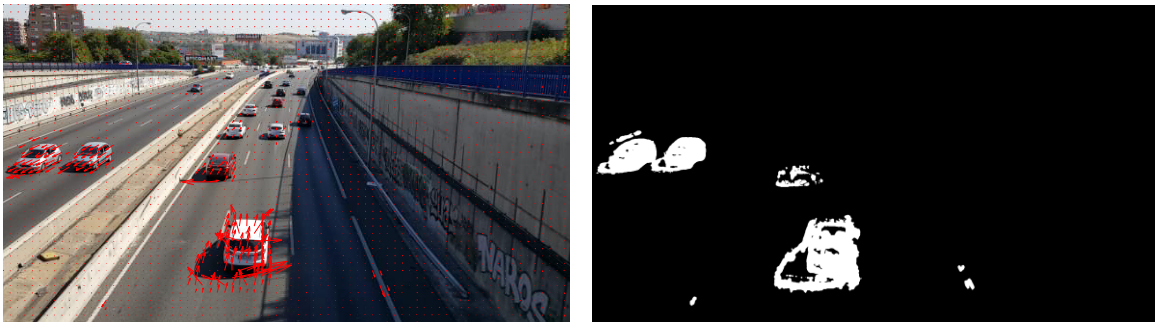


Figura 24. (a) Detección del flujo óptico mediante Lucas-Kanade; (b) Imagen binarizada a partir de la magnitud del flujo óptico

Sobre la imagen binaria se lleva a cabo un etiquetado de componentes conexos mediante la función *bwlabel* que aplica el procedimiento descrito en Haralick y Shapiro (1992). Se obtienen las regiones mostradas en la imagen de la Figura 25, a las que se les extraen sus correspondientes áreas mediante la función *regionprops*. Con las regiones así obtenidas se seleccionan solo aquellas que superen un determinado valor de umbral en número de píxeles,

en este caso establecido a 500 (determinado de forma empírica). Además, se obtiene el mínimo rectángulo que contiene a cada región, conocido como *bounding box*.

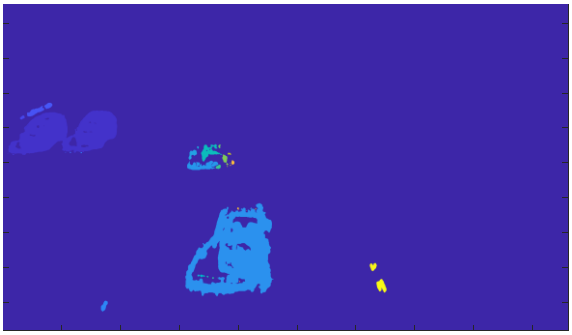


Figura 25. (a) Etiquetado de componentes conexos

5.1.3. Generación de fondo

Para llevar a cabo este proceso se aplica el procedimiento descrito en la sección 3.1 mediante el promediado de la *mediana* para todas las imágenes de la secuencia y sobre los tres canales de color (rojo, verde y azul). Para ello se utiliza un número prefijado de *frames* consecutivos. En la Figura 26 se muestra una imagen de fondo utilizando 100 *frames*. Obsérvese cómo mediante este proceso es posible eliminar los objetos en movimiento, en este caso vehículos, presentes en cada uno de los *frames* consecutivos. Como consecuencia de esta operación, la escena aparece un tanto difuminada, lo cual es una consecuencia lógica del proceso de promediado de la mediana aplicado.

El proceso de generación de fondo se lleva a cabo cada cierto tiempo, por ejemplo, cada 100 *frames*, ya que en entornos de exterior, las condiciones de iluminación medioambientales son, o pueden ser, altamente cambiantes, máxime teniendo en cuenta que en las imágenes objeto de estudio, con abundantes sombras en días soleados, cambian con la rotación terrestre y en particular en días con nubes variable.

Figura 26. Fondo de la escena con 100 frames

5.1.4. Detección y eliminación de sombras

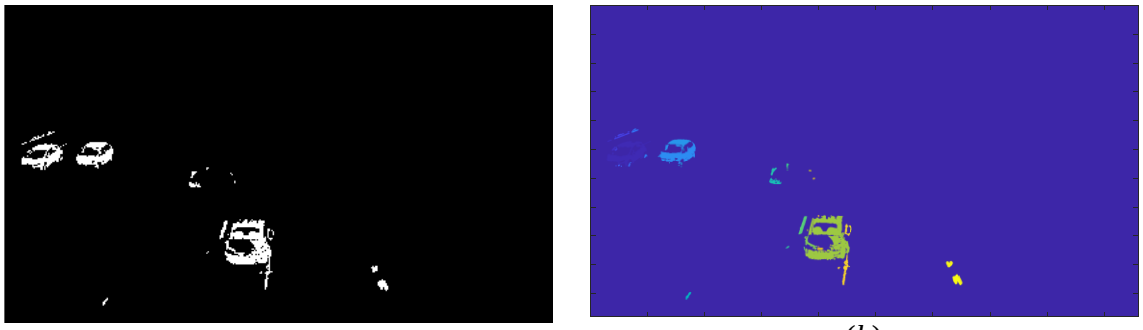
En relación a la detección de sombras, se aplica el procedimiento descrito en la sección 3.2.2. Aplicando exclusivamente el primer término de comparación $\alpha \leq \frac{I_t(x,y)V}{B_t(x,y)V} < \beta$, se realiza una única detección de sombras que no forman parte del fondo de imagen, por lo que se obtendrían solo las correspondientes a los vehículos. La Figura 27 muestra sendas imágenes de sombras detectadas sobre la imagen de referencia. En la derecha aparecen las

sombras binarizadas, mientras que en la de la izquierda se muestran las mismas solapadas sobre la imagen original.

(*a*) (*b*)

Figura 27. Detección de sombras con el primer término de Cucchiara y col. (2002, 2003)

Una vez detectados los objetos en movimiento y las sombras, se procede a eliminar éstas de los objetos, para lo cual se aplica una sencilla operación lógica sobre las imágenes binarias correspondientes. Siguiendo con la imagen de ejemplo, de lo que se trata es de eliminar las sombras de la Figura 27(b) sobre los objetos detectados según la imagen de la Figura 24(b). De esta forma se obtiene la imagen de la Figura 28(a), ya sin las sombras. De nuevo, sobre esta imagen resultante se aplica el procedimiento de etiquetado de componentes conexas indicado en la sección 5.1.2, obteniendo la imagen de la Figura 28(b) y de nuevo se obtiene el área y el *bounding box*, descartando igualmente las áreas inferiores a 500 píxeles.



(*a*) (*b*)

Figura 28. (a) Imagen binarizada con eliminación de sobras; (b) Imagen sin sombras etiquetada

A partir de las regiones etiquetadas, tanto en las imágenes que contienen sobras como en las que no, según se ha descrito previamente y se infiere en las imágenes de las figuras Figura 25 y Figura 28(*b*), se utilizan los correspondientes *bounding boxes* para proceder al recorte con las mismas coordenadas de imagen proporcionadas por el rectángulo correspondiente. Estos recortes, convenientemente redimensionados al tamaño requerido por el modelo AlexNet (227x227×3 píxeles) son los que se proporcionan a la red para proceder a su clasificación.

En las imágenes mostradas en la Figura 29 se muestran los correspondientes recortes a partir de los objetos etiquetados en las figuras Figura 25 y Figura 28(b), respectivamente. En la parte superior, los vehículos aparecen con sombras, mientras que, en la inferior sin ellas, observándose claramente las diferencias en los recortes e incluso cómo en el primer caso aparecen dos vehículos unidos por circular en paralelo y extenderse la sombra de uno de los vehículos sobre el otro; algo que ya no ocurre una vez procesado. En el caso del coche

blanco, también se observa el efecto de la supresión de la sombra, no incluyéndose debido a la eliminación de esta.

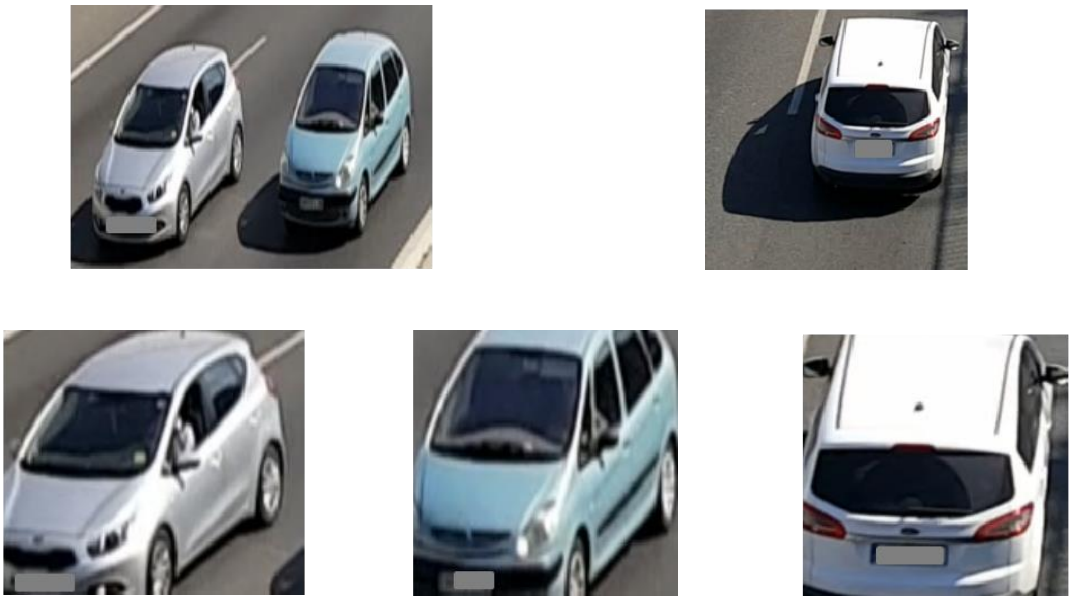


Figura 29. Recortes de entrada a la red AlexNet

5.1.5. Entrenamiento de la CNN.

Como es obvio, antes de proceder a la clasificación de los recortes antes mencionados, es necesario proceder al reentrenamiento con las imágenes asignadas a las clases seleccionadas, cuya dimensión como se ha indicado previamente (227×227x3 píxeles). De esta forma, la CNN, que en este caso se basa en Alexnet, realiza un reajuste de los pesos iniciales, procediendo así a la transferencia del conocimiento desde el modelo pre-entrenado al nuevo modelo. En el presente trabajo se han utilizado las siguientes ocho clases: *coche trasera*, *coche delantera*, *bus*, *camión-furgoneta*, *moto*, *líneas*, *asfalto* y *muro*. En la Figura 30 se muestran una serie de ejemplos representativos de estas clases utilizadas durante el entrenamiento, mientras que en la Figura 31 se grafican los resultados del entrenamiento realizado.

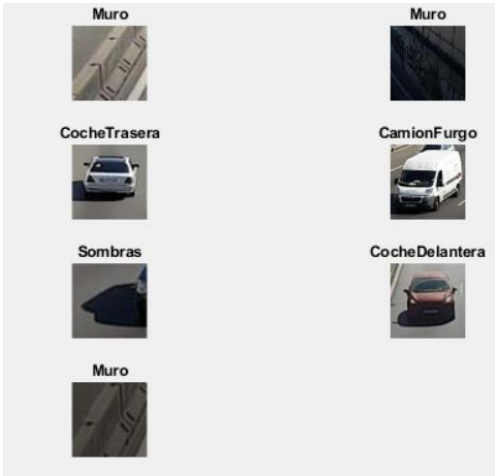


Figura 30. Ejemplos de muestras de entrenamiento para la CNN

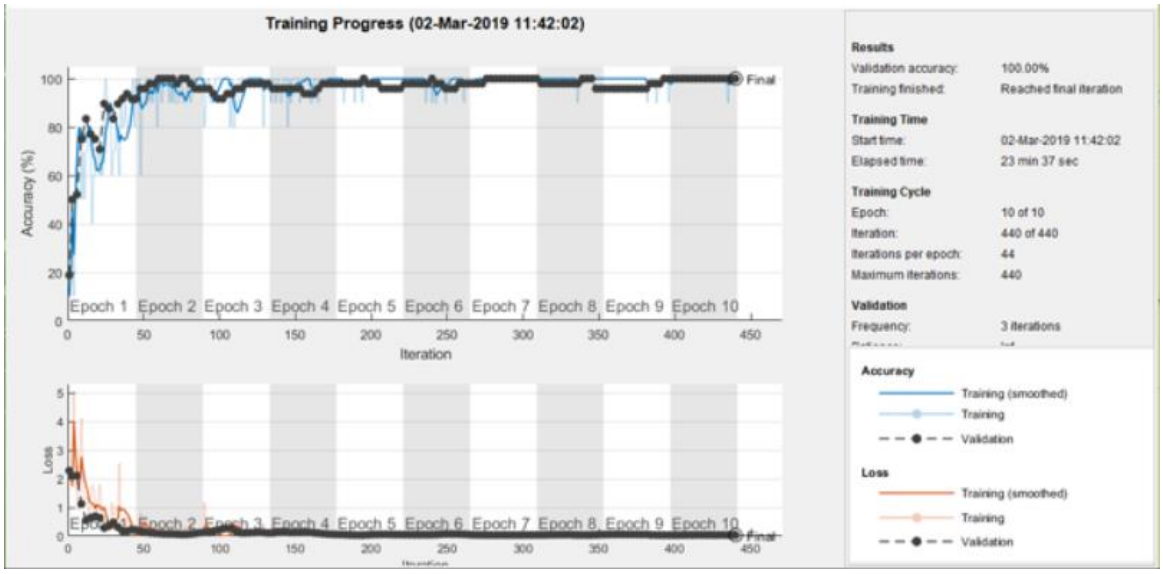


Figura 31. Resultados del entrenamiento de la CNN

5.1.6. Clasificación con la red neuronal

Esta es la etapa de la aplicación en la que se realiza el etiquetado de los objetos en movimiento según las ocho clases indicadas previamente. Lo ideal es que toda la algoritmia previa a la clasificación haya filtrado única y exclusivamente a los vehículos, descartando las sombras, el asfalto, las líneas y los muros. En la Figura 32 se muestran los objetos en movimiento identificados y etiquetados como vehículos vistos frontalmente (recuadro azul) y por su parte trasera (recuadro rojo). En (*a*) sin eliminación de sombras y en (*b*) con las sombras eliminadas, observándose el efecto de la eliminación.

(*a*) (*b*)

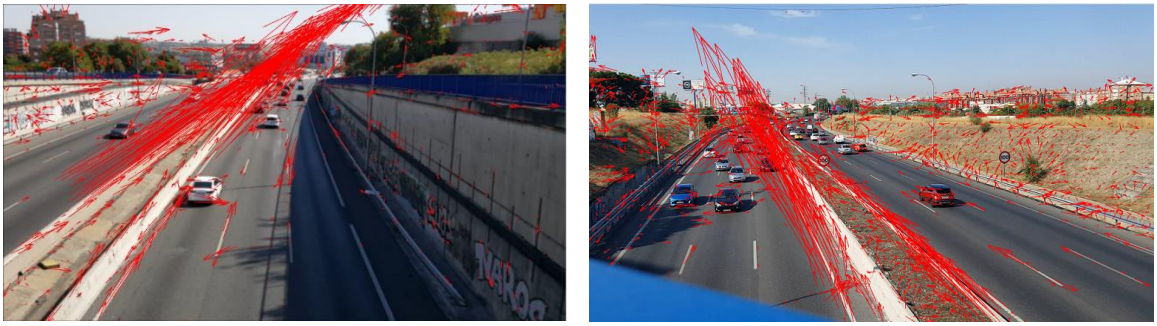
Figura 32. Clasificación y detección de uno de los fotogramas de la secuencia.

6. Resultados

A continuación se analizan los resultados más relevantes derivados del proceso, siguiendo los mismos pasos y módulos que comprenden el diseño de la aplicación. Para ello se han analizado 10 videos con sombras a razón de 30 *fps* y con una duración promedio de 1.5 minutos lo que supone un total de aproximadamente 2700 imágenes.

6.1. Detección del movimiento

En la Figura 33 se aprecia el resultado del flujo óptico en el primer fotograma procedente de dos vídeos diferentes. Tal y como se ha comentado en la sección 2, el método de Lucas-Kanade realiza el cálculo del campo de movimiento con dos imágenes; si solo se dispone de una de ellas, el resultado es el que aparece en la imagen mostrada. Es por este motivo por el que las primeras capturas del proceso, y por supuesto la primera de todas ellas, se desechan, excluyéndolas del análisis.

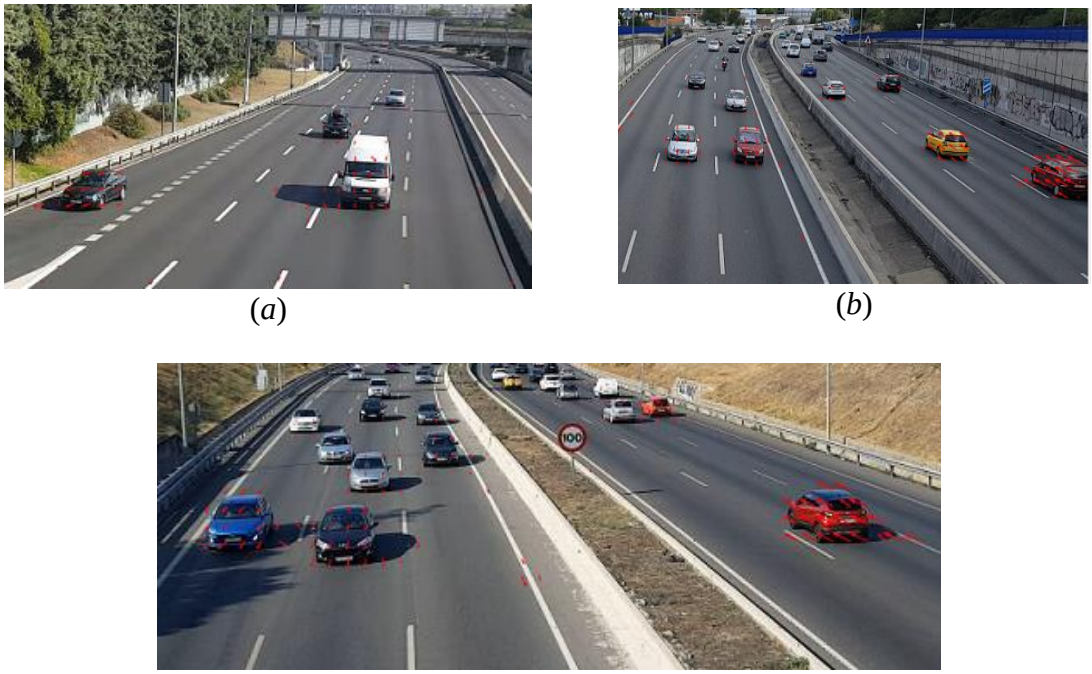


(*a*) (*b*)

Figura 33. Flujo óptico del primer fotograma

Por otra parte, en la Figura 34 se muestra el flujo óptico sobre los objetos en movimiento de tres imágenes distintas. Los vectores de flujo se representan gráficamente solapados sobre los vehículos en movimiento para cada escena. Las flechas indican la dirección del movimiento, y la cabeza de las mismas, el sentido. Además, la longitud de las flechas está asociada en una relación directa con la magnitud del vector del flujo óptico y por tanto con la intensidad del movimiento; esto es, a mayor longitud mayor magnitud. De esta forma se puede identificar claramente cómo en los vehículos más próximos, mayor es la intensidad, dejando de representarse en los más lejanos, donde la magnitud del flujo óptico es imperceptible. Esto no significa, evidentemente, que dichos vehículos no se estén moviendo a la misma velocidad que los otros (consecuencia directa de la discusión realizada en la sección dos).

Desde el punto de vista del interés que suscita el presente trabajo respecto de las sombras, en los diferentes objetos en movimiento se observa cómo las sombras son detectadas como objetos en movimiento. Las sombras se desplazan solidarias con los vehículos que las generan y a la misma velocidad y por tanto con el mismo flujo óptico, de ahí que sea un aspecto ciertamente relevante el hecho de proceder a su detección en esta fase.



(*a*) (*b*)

(*c*)

Figura 34. Flujo óptico bajo distintas condiciones lumínicas

En algunas ocasiones, aparecen zonas detectadas con un movimiento aparente que, en la realidad, son objetos estáticos. Tal es el caso de la línea de carretera de la Figura 35. Estas zonas son detectadas debido a vibraciones o movimientos no deseados durante la grabación del vídeo. Existen diversas posibilidades para solucionar este problema. En primera instancia se puede pensar en realizar una solución mediante un tratamiento software, pero también pueden utilizarse equipos que absorben las vibraciones y movimientos no deseados.

Figura 35. Fallos en el flujo óptico debido a alteraciones durante la grabación

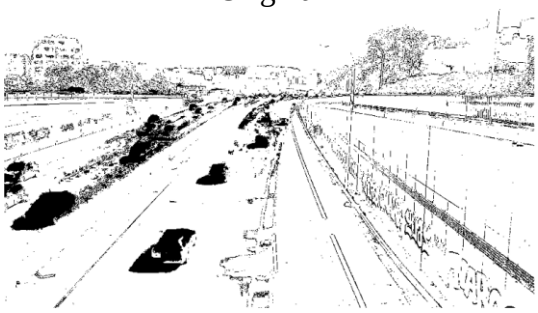
6.2. Detección de sombras

La detección de sombras utilizando el método de Cucchiara y col. (2001, 2003) depende de cuatro variables configurables ($\alpha, \beta, \tau_H \; y \; \tau_S$), tal y como se indica en la sección 3.2.2. En la Figura 36 se muestran resultados en los que se utilizan distintos valores de estos parámetros, observándose cómo afectan a la detección. A partir de los resultados observados, se deduce que no existe una clara combinación de los mismos que permita determinar el alcance y validez de los mismos. No obstante, se identifica que el parámetro más relevante es β, que determina una más clara diferenciación de las sombras en función de su valor.

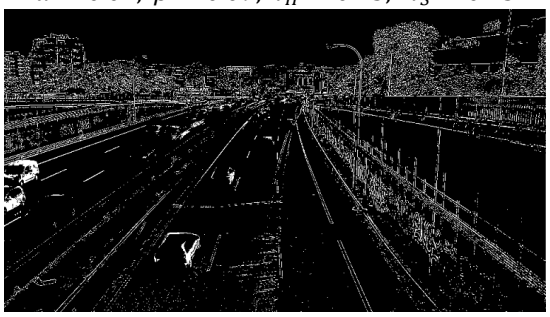
Finalmente, tras diversas pruebas de ensayo y error se determina que con $\beta = 0.09$ se obtienen resultados satisfactorios para el conjunto de imágenes de vídeos analizadas.

Original

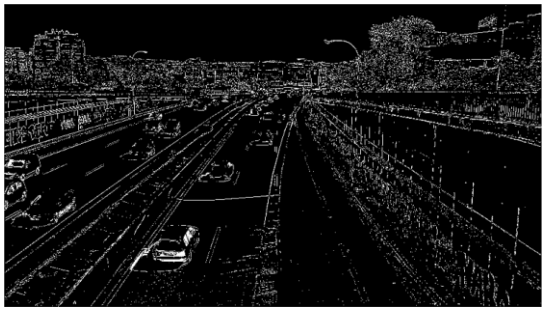
$\alpha = 0.01;\ \beta = 0.09;\ \tau_H = 0.15;\ \tau_S = 0.15$



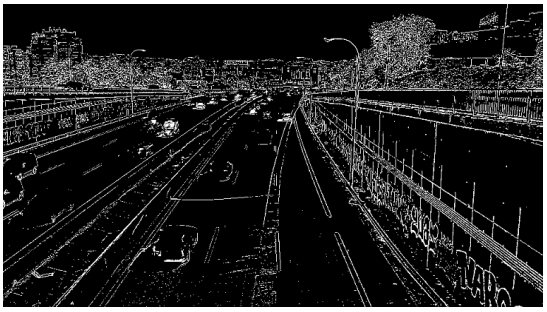
$\alpha = 0.01;\ \beta = 0.6;\ \tau_H = 0.15;\ \tau_S = 0.15$



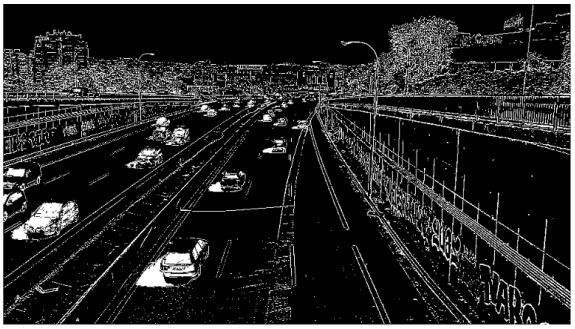
$\alpha = 0,12;\ \beta = 0,6;\ \tau_H = 0.15;\ \tau_S = 0.15$



$\alpha = 0,01;\ \beta = 0,09;\ \tau_H = 0.6;\ \tau_S = 0.15$



$\alpha = 0.01;\ \beta = 0.09;\ \tau_H = 0.15;\ \tau_S = 0.9$

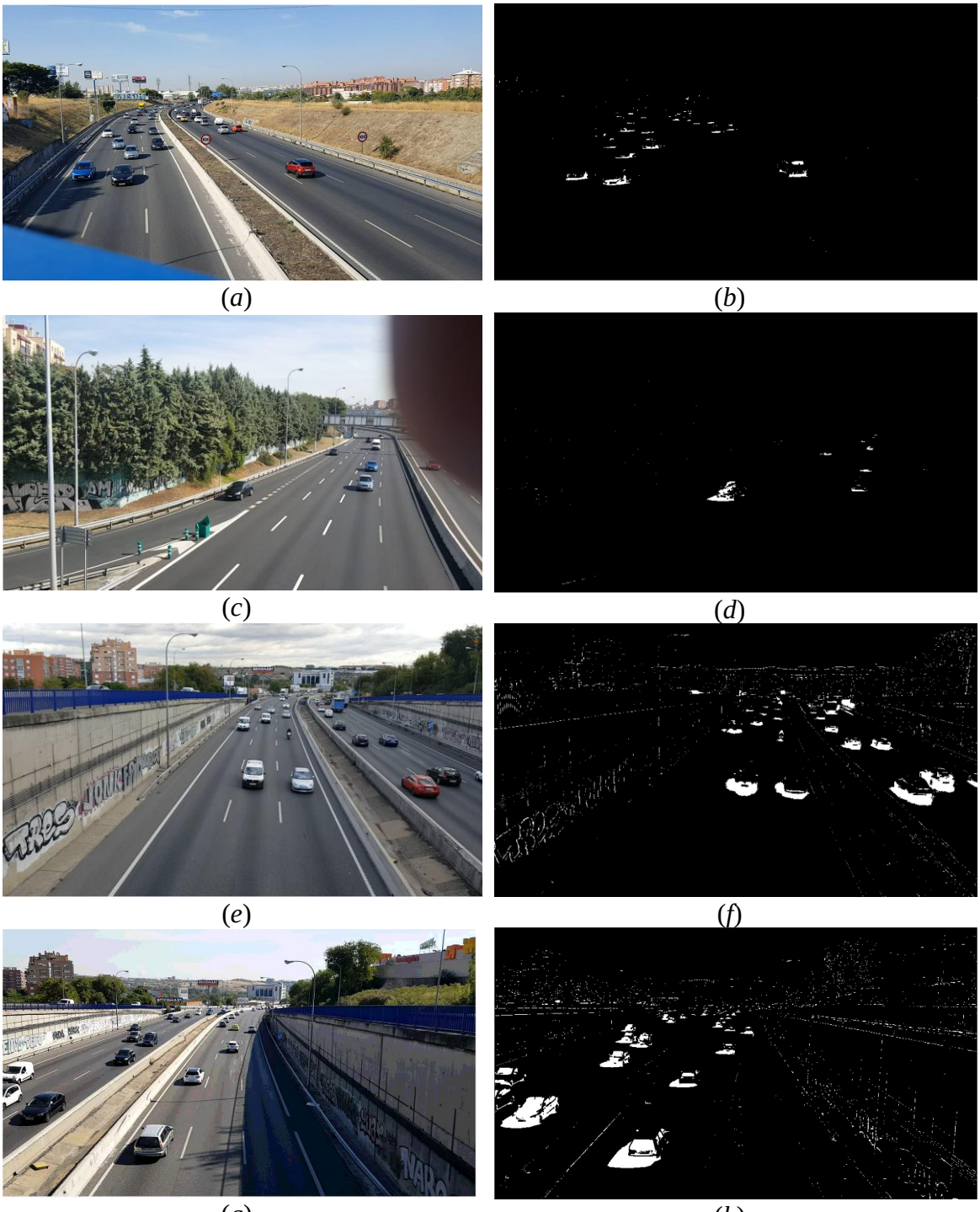


$\alpha = 0,01;\ \beta = 0,09;\ \tau_H = 0.6;\ \tau_S = 0.9$

$\alpha = 0,01;\ \beta = 0,6;\ \tau_H = 0.6;\ \tau_S = 0.9$

Figura 36. Ejemplo de detección de sombras con distintos valores de $\boldsymbol{\alpha, \beta, \tau_H, \tau_S}$

Por las razones expuestas previamente, se decide utilizar únicamente el término β, cuyos resultados se aprecian en la Figura 37. Tal y como se puede ver, la detección a partir de este único término $\frac{V_t(x,y)}{V_b(x,y)} < \beta$ se enfoca principalmente en el problema que se intenta atacar. De esta forma, se simplifica el algoritmo, se reduce computacionalmente la solución y los resultados son mucho más eficaces para el fin propuesto de detectar y eliminar las sombras que producen movimientos no deseados.



(*a*) (*b*)

(*c*) (*d*)

(*e*) (*f*)

(*g*) (*h*)

Figura 37. Ejemplos de detección de sombras con el primer término de Cucchiara y col. (2001, 2003).

6.3. Entrenamiento de la CNN.

A continuación, se detallan algunos aspectos relativos al proceso de entrenamiento, describiendo y definiendo los parámetros correspondientes al modelo y configuración de la red en referencia a la Figura 31:

1. *Muestras de entrenamiento y validación*. Indican, respectivamente, el número de muestras utilizadas para el entrenamiento y la validación, obtenidas a partir del total disponibles. En

este trabajo, del total de 338 muestras disponibles, el 70% (237) se utilizan para el entrenamiento y el 30% para la validación (101).

2. *Epochs e iteraciones*. Representa el paso de todas las muestras disponibles para el entrenamiento y por tanto las 237 indicadas previamente. Se han establecido 10 *epochs* con 44 iteraciones cada uno, resultando un total de 440 iteraciones.

3. *Precisión en la validación* (*validation accuracy*). Precisión en el conjunto de imágenes utilizadas para validación, esto es en las 101 imágenes dedicadas a tal fin.

4. *Dimensión por lotes* (*mini-batch size*). Define la cantidad de imágenes que se utilizan en cada iteración. En este caso se ha fijado en 10.

5. *Razón de aprendizaje* (*learning rate*). Determina la rapidez con la que la red aprende según el método de actualización de los parámetros que constituyen el objetivo del aprendizaje. Se ha fijado a un valor constante de 10^{-4}. Se han realizado diversos experimentos en este sentido disminuyendo la razón de aprendizaje del orden del 0.2% cada epoch sin lograr mejoras significativas.

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

El proceso de entrenamiento se completa según se indica en la Figura 31, observándose una precisión en cuanto a validación de las muestras de test del 100%, lo que permite concluir que el modelo utilizado resulta ciertamente eficiente. No obstante, durante la fase de clasificación todavía se producen errores en la clasificación de los objetos en movimiento lo que hace pensar que a pesar de este resultado el modelo es mejorable, desde el punto de vista de la utilización de más muestras de entrenamiento.

En relación a esta misma figura se observa que el tiempo total empleado en el proceso ha sido de 23 minutos y 37 segundos con indicación de que la ejecución se ha llevado a cabo en una CPU simple, sin procesamiento en GPU. Se trata de un tiempo razonable bajo las consideraciones especificadas y teniendo en cuenta que estos procesos se llevan normalmente a cabo sin interferir en el proceso de clasificación mientras éste se realiza. No obstante, con las consiguientes mejoras en el equipamiento hardware y en particular mediante el uso de una GPU, estos tiempos pueden reducirse fácilmente.

A modo ilustrativo, en la imagen de la Figura 38 se muestra una representación gráfica de los pesos actualizados de la red en la primera capa de convolución para los 96 filtros de la red AlexNet, observándose una cierta similitud, en términos generales, con los filtros de Gabor (Daugman, 1980), que simulan el comportamiento biológico de las células de la retina según los experimentos y modelos neurocomputacionales establecidos en el sistema de visión de los mamíferos (Hubel y Wiesel, 1959, 1962, 1968).

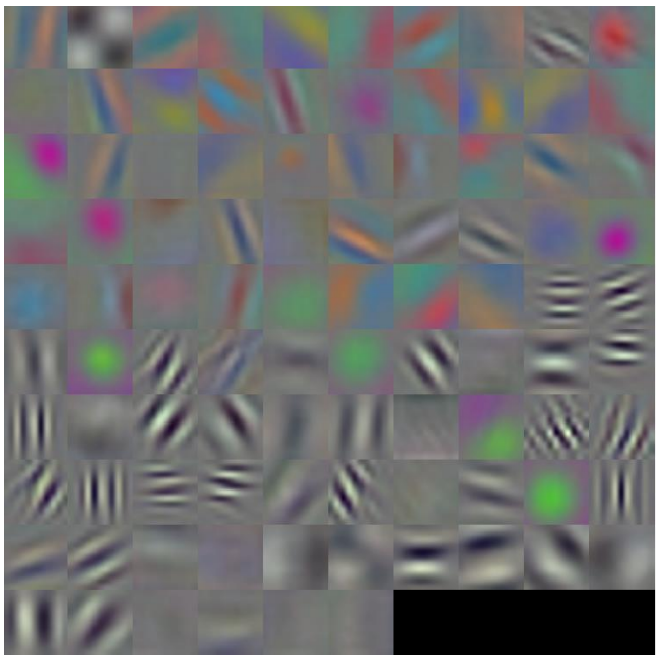


Figura 38. Representación de los pesos en la primera capa de convolución de AlexNet.

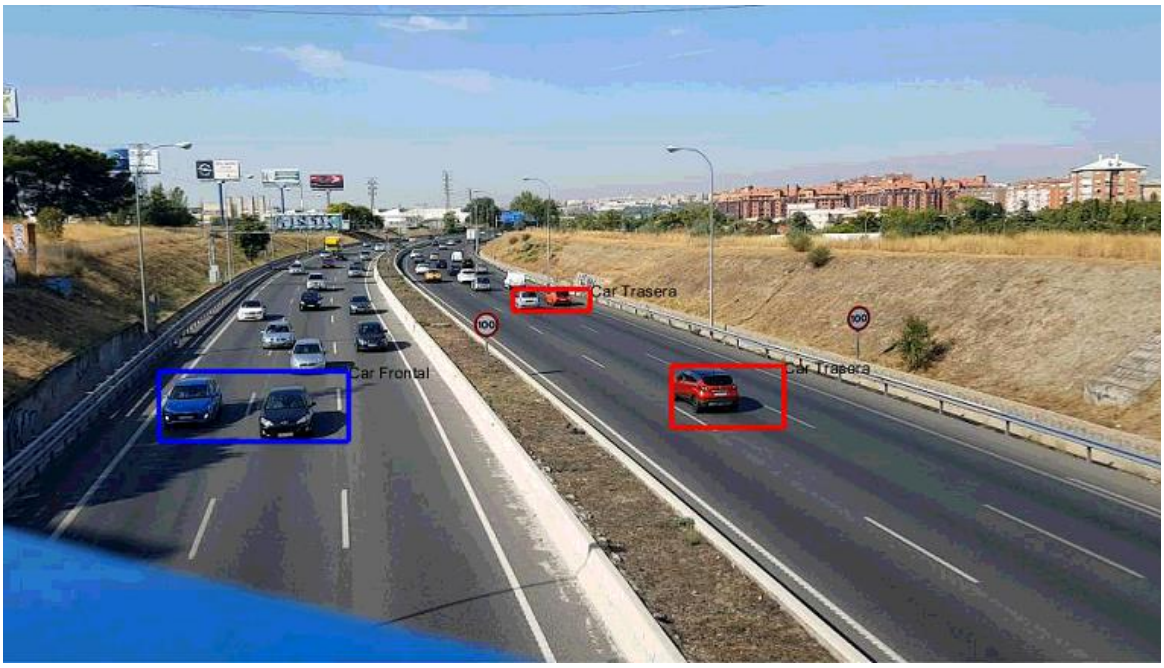
6.4. Clasificación con la red neuronal

A continuación, se exponen los resultados más relevantes en relación a la clasificación de los objetos en movimiento, considerando las regiones sin eliminar las sombras y tras su eliminación. Por ejemplo, para la imagen mostrada en la Figura 32 (sección 5.1.6) se obtienen los siguientes resultados. En la imagen sin eliminación de sombras, Figura 32 (*a*) se detectan los dos coches con vista frontal como un único vehículo con una probabilidad de 0.98, tratándose en este caso de un claro error, a pesar de que los ha identificado como coches frontales con la mencionada probabilidad. El coche blanco en la parte derecha se identifica como coche visto por detrás con una probabilidad del 94%. Mientras que en relación a la clasificación de la imagen con las sombras eliminadas de la Figura 32 (*b*) los dos vehículos de la izquierda se han separado claramente con la eliminación de las sombras, identificándose correctamente como vistos por delante y con probabilidades de 1.0 en ambos casos, mejorando así los resultados en ambos sentidos. En el caso del vehículo visto por detrás se observa un mejor ajuste de la región sobre el propio vehículo, obteniéndose, también, una probabilidad del 1.0 en cuanto a la clasificación. Por tanto, en relación a esta imagen, cabe concluir que el hecho de eliminar las sombras en los objetos identificados mejora el resultado de la clasificación en calidad, en determinación de vehículos individuales sobre grupos, en un mejor ajuste de las regiones del vehículo real, y cuantitativamente, en términos de mejoras de la probabilidad de detección.

Otro ejemplo similar al comentado es el que se presenta en las figuras Figura 39 y Figura 40 . En la primera, aparecen dos imágenes, haciéndose una comparación entre la clasificación de la CNN con y sin aplicar algoritmo de eliminación de sombras.

En la zona central izquierda de la Figura 39(a) se aprecia una detección de un vehículo frontal único, aunque realmente son dos circulando en paralelo. En la misma imagen se muestra el mismo caso, pero para el carril opuesto. Por otra parte, en la Figura 39(b) se muestran estos ejemplos en el mismo instante de tiempo, pero con una disgregación del

grupo de vehículos de forma satisfactoria. A pesar de ello, es característico que la clasificación por parte de la CNN es errónea



(*a*) Detección con sombras

(*b*) Detección sin sombras

Figura 39. Áreas detectadas por la CNN sin eliminación de sombras (a) y con algoritmo de eliminación de sombras (b)

En las Figura 40 (*a*) y (*b*) se puede ver el vector de probabilidades de la CNN para sendos grupos de vehículos. Además, en las Figura 40 *c*, *d*, *e* y *f*, se muestran estos mismos vectores para la detección individualizada de cada uno.

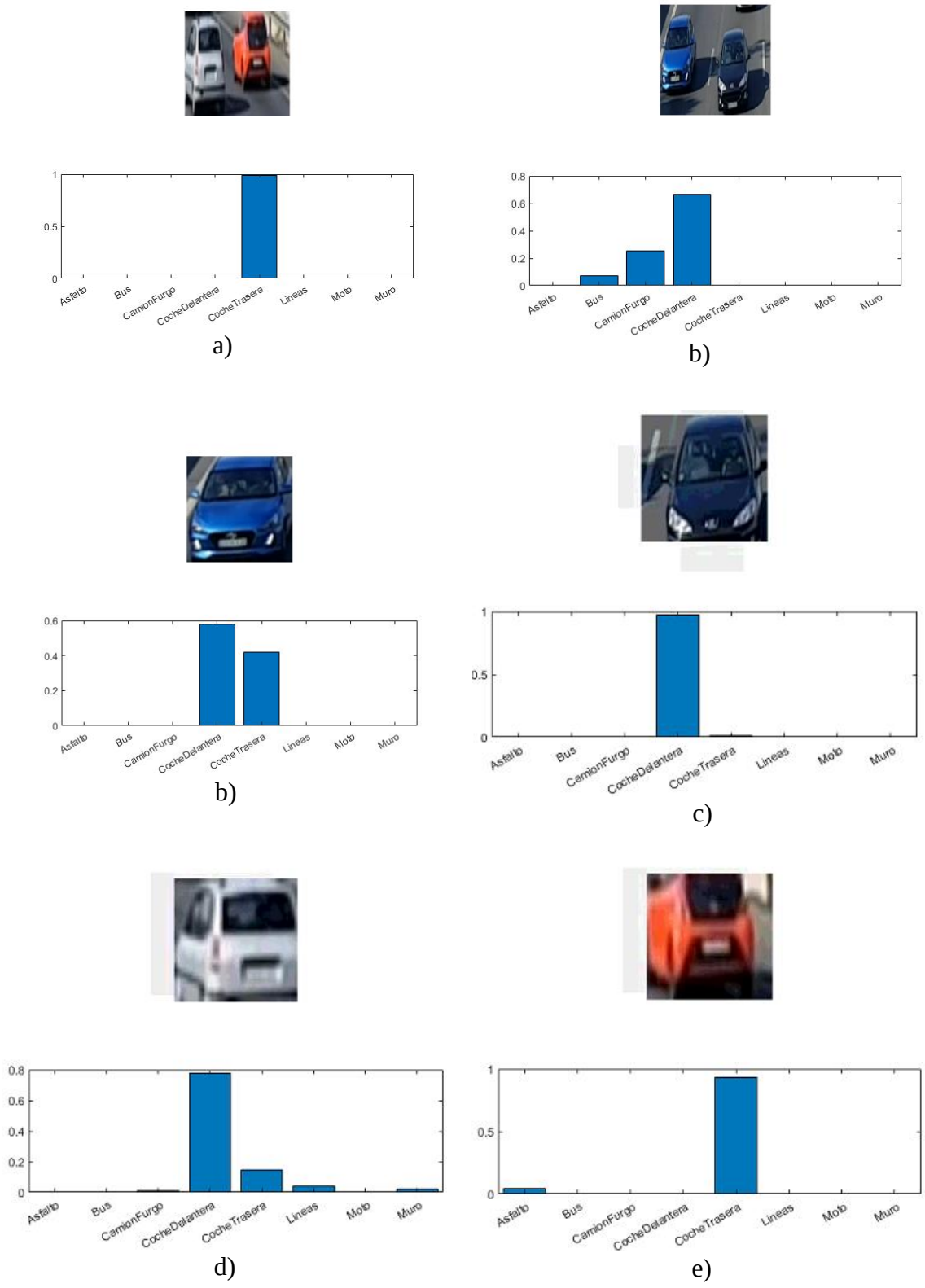
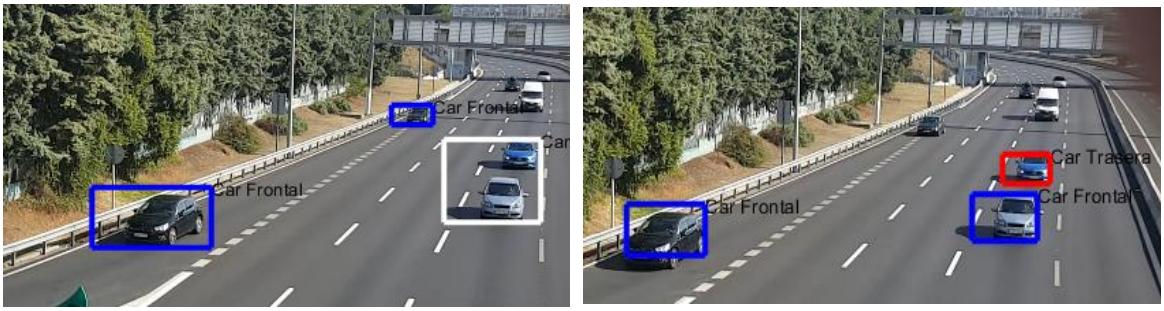


Figura 40. Vector de probabilidades de los vehículos para un caso erróneo en la etapa de clasificación de la CNN

Finalmente, otro de los casos con resultados no satisfactorios en su totalidad es el que se presenta en las imágenes de la Figura 41. En la zona central derecha de la imagen (*a*) se puede ver cómo dos vehículos son detectados como uno solo, además de ser clasificado como camión en lugar de coche frontal. Por otra parte, al aplicar la eliminación de sombras, estos dos vehículos son detectados por separado, si bien uno de ellos no está bien clasificado, puesto que tiene la etiqueta de vehículo trasera en lugar de vehículo frontal. La separación de las sombras ha funcionado correctamente en ambos casos, no así la identificación, lo cual podría solventarse con un mayor nivel de entrenamiento.

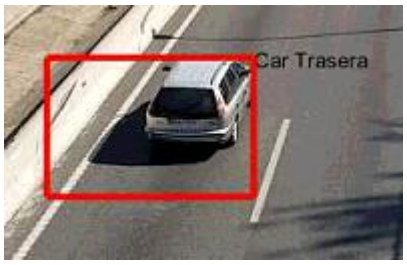


a) b)

Figura 41. Áreas detectadas por la CNN sin eliminación de sombras (a) y con algoritmo de eliminación de sombras (b)

A continuación, se reflejan un total de 8 casos donde la eliminación de sombras ha sido satisfactoria:

1. En la primera imagen, las sombras son clasificadas como parte del vehículo, mientras que, en la segunda, éstas son eliminadas.
2. Se aprecia la detección de tres vehículos como uno único, mientras que, en la segunda imagen, éstos son detectados de forma aislada
3. Este tercer caso es idéntico al primero. Aunque es característico, que la región de vehículo frontal no recoja a todo éste.
4. Este cuarto caso es idéntico al tercero.
5. Este caso es idéntico al primero, con la diferencia del sentido de circulación de ambos.
6. Caso idéntico al quinto.
7. En este caso, con la detección sin eliminación de sombras, el vehículo negro es clasificado como un autobús. Sin embargo, tras la aplicación del filtro, la CNN sí clasifica como vehículo frontal a éste.
8. Para el caso de la moto, debido a las condiciones de bajo contraste lumínico entre la carretera y la sombra, ambas clasificaciones son prácticamente idénticas.

	Detección con sombras	Detección sin sombras
1		

2


Car


Car Frontal
Car Frontal
Car Frontal

3


Car Trasera


Car Trasera

4


Car Trasera


Car Trasera

5


Car Frontal


Car Frontal

6


Car Frontal
Car


Car Frontal

7


Bus

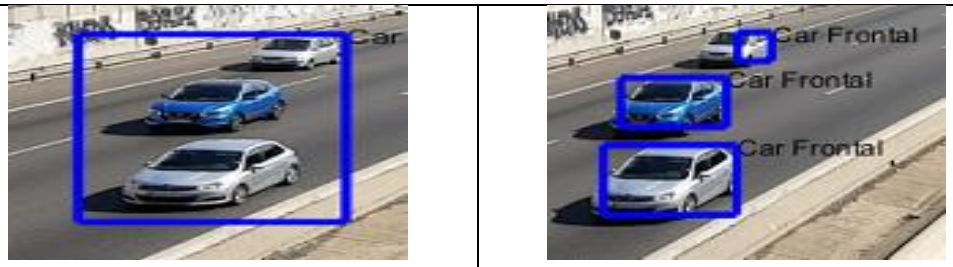

Car Frontal

8

Figura 42.Ejemplos comparativos de la detección con y sin sombras de la CNN

A continuación, se exponen una serie de resultados de tipo cuantitativo tras el análisis aleatorio sobre 10 secuencias de vídeos en distintos instantes de tiempo. Los datos analizados son los siguientes:

1) Separación de grupos de vehículos en varios. Se lleva a cabo el análisis de la eficiencia del algoritmo para individualizar un grupo de vehículos. En lugar de que la región que los contenga sea una, se debe tener una para cada vehículo. Se considera **acierto** los vehículos del grupo que estén bien separados, sin embargo, para aquellos que no se hayan detectado, se indica **fallo**. Por ejemplo, en la figura 43 se tienen tres aciertos. Si uno de ellos no hubiera sido detectado tras la eliminación de sombras, se tendrían dos aciertos y un fallo.



a) b)

Figura 43. Separación de grupos en varios vehículos.

Tras analizar 90 grupos de vehículos se obtienen los resultados mostrados en la Tabla 1. Para el 61,86% de los vehículos se obtiene un acierto, frente al 38,14% con fallo. Este hecho se debe a la baja resolución de la imagen para algunos vehículos, sumado al pequeño tamaño y la inexactitud de regionalización producida por la eliminación de la sombra, tal y como se puede ver en el vehículo superior de la Figura 43.

Aciertos		**Fallos**	
60 vehículos	61.86%	37 vehículos	38,14%

Tabla 1.Aciertos y fallos en separación de agrupaciones de vehículos por sombras

2) Vehículos individuales erróneamente clasificados. Se evalúa la eficiencia de clasificación de la CNN previa y posterior a la eliminación de sombras. Se darán por **acierto** aquellos vehículos cuya etiqueta sea correcta.

Vehículos con sombras	Aciertos	324 vehículos	62.43%
	Fallos	195 vehículos	37.57%
Vehículos sin sombras	Aciertos	263 vehículos	64.78%
	Fallos	143 vehículos	35.22%

Tabla 2. Vehículos individuales erróneamente clasificados.

A partir de los resultados mostrados en la Tabla 2 se aprecia que el porcentaje de aciertos es ligeramente superior en la detección sin sombras. También es característico la diferencia en el número de vehículos en ambos casos. Como se explicó en la sección tres, la región que contiene tanto al vehículo como a la sombra es reducida tras filtrar a estas últimas. La posibilidad de que el algoritmo no lo detecte como una región candidata es mayor, puesto que la cantidad de vectores del objeto en movimiento es reducida. De hecho, del total de los 563 vehículos analizados, 157 no son detectados como regiones candidatas, frente a 43 vehículos que sólo son detectados tras la eliminación sombras.

3) Vehículos clasificados erróneamente sin eliminación de sombras que han sido clasificados correctamente tras eliminarla. Se considera **acierto** aquellos vehículos que se hayan clasificado bien tras el filtro de sombras, sin embargo, como **fallo** se tendrán aquellos que mantienen el error. Por otra parte, aquellos que se detectan erróneamente sin el filtro, y posteriormente no se detectan, no se contabilizan.

Aciertos	20 vehículos	20.20%
Fallos	79 vehículos	79.80%

Tabla 3. Vehículos clasificados erróneamente sin eliminación de sombras que han sido clasificados correctamente tras eliminarla

Con estos datos se puede decir que se mejora en un 20% los aciertos de los vehículos cuya clasificación por la CNN en aquellos casos cuya clasificación ha sido incorrecta.

4) Vehículos clasificados correctamente que al eliminar las sombras mantienen el acierto. Se dan como **acierto** aquellos vehículos que se hayan clasificado bien tras el filtro de sombras, manteniendo la correcta clasificación por parte de la CNN. Sin embargo, como **fallo** se tendrán aquellos que hayan producido una etiqueta errónea. Por otra parte, aquellos que se detectan erróneamente sin el filtro, y posteriormente no se detectan, no se contabilizan.

Aciertos	229 vehículos	87.07%
Fallos	34 vehículos	12.93%

Tabla 4. Vehículos clasificados erróneamente sin eliminación de sombras que han sido clasificados correctamente tras eliminarla

Únicamente el 13% de los vehículos bien clasificados son erróneos. Como se comentó anteriormente, una detección no muy eficiente de las sombras, frente a la baja resolución y tamaño de estos vehículos puede hace que la región que se clasifique no contenga a todo el vehículo, produciendo su fallo en el etiquetado.
Como mejora se tendría que tener, no solo un entrenamiento de la CNN por partes del vehículo, sino también, una mejora de resolución del vídeo a analizar. Además

de una mejora en la determinación de sombras para situaciones lumínicas sin mucho contraste.

7. Conclusiones y trabajo futuro

7.1 Resultados

En el presente trabajo se ha propuesto una solución para clasificación de vehículos en una vía de acceso a la ciudad, que se plantea como una solución de concepto con vistas a la monitorización, gestión y control del flujo de vehículos en el contexto de las ciudades inteligentes del futuro a través de secuencias de imágenes, en este caso, de vídeo. Si bien el trabajo se ha centrado en una problemática crucial, como es la detección de sombras asociadas a los vehículos, y por tanto en movimiento solidario con ellos, llevando a interpretaciones y clasificaciones erróneas.

Con tal propósito se ha desarrollado una aplicación que integra diversas metodologías en el ámbito de la visión por computador y el aprendizaje profundo.

Por un lado, se utiliza el método de Lucas-Kanade para determinar lo que se conoce como flujo óptico y a partir de ahí se determina la magnitud de los objetos en movimiento a través de los vectores que determinan el mencionado flujo óptico. En paralelo, y sobre cada uno de los *frames* que componen la secuencia de vídeo se determinan las sombras existentes en la imagen mediante la aplicación de un procedimiento con tal finalidad. Una vez detectadas las sombras se eliminan éstas de los objetos en movimiento, obteniendo así una imagen binaria que permite el etiquetado de las regiones resultantes y por ende localizar los objetos en movimiento sobre cada *frame* original.

Por otro lado, para cada uno de los objetos en movimiento se realiza un recorte sobre la imagen original, sirviendo cada uno de ellos como entrada a una red neuronal convolucional. Esta red se ha diseñado a partir del modelo AlexNet ya entrenado, realizando un proceso de re-entrenamiento con imágenes propias y para ocho clases, en lugar de las mil que proporciona el modelo original. De esta manera se consigue transferir aprendizaje general a una situación específica.

Los resultados obtenidos permiten deducir que el modelo diseñado es válido para el tipo de aplicación propuesta, confirmando la eficiencia de las redes neuronales convolucionales, tal y como se viene planteando en diversos ámbitos.

Como conclusión final indicar que los objetivos propuestos se han cumplido de forma satisfactoria.

7.2 Trabajo futuro

A pesar de lo expresado previamente, siempre es posible la realización de mejoras al respecto, habiendo identificado las siguientes:

1) Probar nuevos métodos de detección de movimiento con el fin de afinar mejor a la hora de obtener el flujo óptico.

2) En relación a la detección de sombras es conveniente integrar nuevas funcionalidades para determinar la existencia de sombras sobre el asfalto y la identificación del asfalto mismo, para ello se propone la investigación de lo que

se conoce como clasificación semántica donde las redes neuronales convolucionales están llamadas a jugar un papel determinante.

3) Ampliar la base experimental para considerar otro tipo de redes convolucionales, incluyendo las R-CNN que tienen como fundamento trabajar sobre regiones (R) específicas, siendo en este caso los objetos en movimiento.

4) Establecer las condiciones necesarias para conseguir una mayor eficiencia en la identificación de los vehículos, en función de las velocidades de circulación o distancia a la cámara. En ambos casos como paso previo a la implantación del sistema en un entorno real.

8. Bibliografía

1. BVLC AlexNet Model (2019). Disponible on-line: https://github.com/BVLC/caffe/tree/master/models/bvlc_alexnet (accedido Febrero 2019).

2. Boureau, Y., Bach, F., LeCun, Y., and Ponce, J. (2010a). Learning mid-level features for recognition. In Proc. IEEE Int. Conference on Computer Vision and Pattern Recognition (CVPR'10).

3. Boureau, Y., Ponce, J., and LeCun, Y. (2010b). A theoretical analysis of feature pooling in vision algorithms. In Proc. IEEE Int. Conference on Machine learning (ICML'10).

4. Cucchiara R., Grana C., Neri G., Piccardi M., Prati A. (2002) The Sakbot System for Moving Object Detection and Tracking. In Remagnino P., Jones G.A., Paragios N., Regazzoni C.S. (eds.) Video-Based Surveillance Systems, pp. 159-171, Springer, Boston, MA.

5. Cucchiara, R., Grana, C., Piccardi, M., Prati, A. (2003). Detecting moving objects, ghosts and shadows in video streams. IEEE Transactions on Pattern Analysis and Machine Intelligence, 25, 1337-1342.

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

7. Dubuisson, M.P, Lakshmanan, S., Jain, A.K. (1996). Vehicle segmentation and classification using deformable templates. IEEE Trans. Pattern Anal. Machine Intell., 18(3), 293-308.

8. Dumoulin, V., Visin, F. (2016) A guide to convolution arithmetic for deep learning. arXiv:1603.07285v2. Disponible on-line: https://arxiv.org/abs/1603.07285 (accedido Febrero 2019).

9. Elgammal, D., Harwood, L., Davis, L. (1999). Non-parametric model for background subtraction. In Proc. Of FRAME-RATE, Corfu, Greece, 1999.

10. Goodfellow, I., Bengio, Y., and Courville, A. (2016). Deep learning. MIT Press. Disponible on-line: https://www.deeplearningbook.org/ (accedido Febrero 2019).

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

12. Haykin, S. (1994). Neural Networks: A comprehensive foundation. IEEE Press, NY, USA.

13. Hubel, D. H., Wiesel, T. N. (1959). Receptive fields of single neurons in the cat'sstriate cortex. Journal of Physiology, 148, 574–591.

14. Hubel, D. H., Wiesel, T. N. (1962). Receptive fields, binocular interaction, andfunctional architecture in the cat's visual cortex. Journal of Physiology (London),160,106–154.

15. Hubel, D., Wiesel, T. (1968). Receptive fields and functional architecture of monkeystriate cortex. Journal of Physiology (London), 195, 215–243.

16. ImageNet (2019). Disponible on-line: http://www.image-net.org (accedido Junio 2019).

17. Jarrett, K., Kavukcuoglu, K., Ranzato, M.A., LeCun, Y. (2009). What is the best Multi-Stage Architecture for Object Recognition?. In Proc. ICCV 2009, 8 pages.

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

19. LeCun, Y. (1989). Generalization and network design strategies. Technical ReportCRG-TR-89-4, University of Toronto. 326, 345747. Disponible on-line: http://yann.lecun.com/exdb/publis/pdf/lecun-89.pdf (accedido Febrero 2019).

20. Llerena, J.P. (2019). Detección de vehículos y flujo de tráfico con redes neuronales convolucionales en visión artificial. Trabajo Fin Máster. MÁSTER EN INGENIERÍA DE SISTEMAS Y CONTROL. Universidad Complutense-UNED.

21. Lucas, B.D. (1984). Image matching by the method of differences (PhD dissertation), Carnegie Mellon University, Disponible on-line: https://www.ri.cmu.edu/pub_files/pub4/lucas_bruce_d_1984_1/lucas_bruce_d_1984_1.pdf (accedido Junio 2019).

22. Lucas, B.D., Kanade, T. (1981). An iterative image registration technique with an application to stereo vision. Proceedings of Imaging Understanding Workshop, pp. 121-130.

23. Lyu, S., Simoncelli, L. (2008). Nonlinear image representation using divisive normalization. In CVPR, pages 1-8.

24. Matlab (2019). The MathWorks. Disponible on-line: https://es.mathworks.com/ (accedido Junio 2019).

25. Pajares y Cruz (2007). Visión por computador: Imágenes digitales y aplicaciones. RA-MA, Madrid.

26. Prati, A., Cucchiara, R., Mikic, I., Trivedi, M.M. (2001) Analysis and Detection of Shadows in Video Streams: A Comparative Evaluation. Proc. IEEE Int. Conf. Computer Vision and Pattern Recognition, 6 pages.

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

28. Russell, S., Norvig, P. (2009). Artificial intelligence: a modern approach (3.ª edición). Prentice Hall, Upper Saddle River, N.J.

29. Saxe, A., Koh, P. W., Chen, Z., Bhand, M., Suresh, B., and Ng, A. (2011). On random weights and unsupervised feature learning. In L. Getoor and T. Scheffer, editors, Proceedings of the 28th International Conference on Machine Learning (ICML-11), ICML '11, pages 1089–1096, New York, NY, USA. ACM.

30. Srivastava, N., G. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov (2014) Dropout: A Simple Way to Prevent Neural Networks from Overfitting. Journal of Machine Learning Research, 15, 1929-1958.

31. Zhou, Y. and Chellappa, R. (1988). Computation of optical flow using a neural network. Proc. IEEE International Conference on Neural Networks, 1988, pages 71–78.