UNIVERSIDAD NACIONAL DE LA PLATA

FACULTAD DE INFORMÁTICA

TESINA DE LICENCIATURA

TÍTULO: Detección de peatones en video usando algoritmos de aprendizaje automático

AUTORES: Camele, Genaro

DIRECTOR: Dr. Hasperué, Waldo

CODIRECTOR: Dr. Ronchetti, Franco

ASESOR PROFESIONAL:

CARRERA: Licenciatura en Sistemas

Resumen

La detección de peatones utilizando algoritmos de Machine Learning es un tema ampliamente abordado. En esta tesis se hará uso de SVM y redes neuronales como clasificadores, utilizando combinaciones de diferentes descriptores para esta tarea. Se analizaran varias métricas obtenidas por ambos modelos, esto permitirá determinar si los clasificadores propuestos alcanzan resultados aceptables en un contexto tan diverso como lo es un ambiente urbano. Usando SVM se realizará un análisis de la transferencia de aprendizaje, midiendo si al evaluar un conjunto de datos diferentes del que se entrenó alcanza métricas similares.

Palabras Clave

Clasificación y detección de peatones. Machine Learning. Máquinas de vectores de soporte. Redes neuronales. Deep Learning. Transferencia de aprendizaje.

Conclusiones

Los algoritmos de Deep Learning representan una mejora en las métricas finales. Sin embargo, los tiempos que conlleva entrenar y evaluar dichos modelos son mucho mayores a los del SVM, generando una discusión del balance entre los resultados obtenidos y la eficiencia en tiempo de ejecución para el desarrollo de la tarea. En cuanto a la transferencia, la base de datos Daimler resulta la más propicia para las tareas de clasificación, tanto para SVM como para redes neuronales, siendo la que mejor logró generalizar los ejemplos de evaluación.

Trabajos Realizados

Se realizaron experimentos para llevar a cabo distintos análisis sistemáticos de precisión, exhaustividad y medida-f de los modelos estudiados (SVM y redes convolucionales) y la capacidad de transferencia de aprendizaje que brinda un SVM lineal.

Trabajos Futuros

- *Plantear una medición sistemática del desempeño de los modelos estudiados en tareas de detección.*
- *Realizar nuevos experimentos a partir de nuevos subconjuntos de ejemplos de las bases de datos estudiadas.*
- *Analizar nuevos modelos de clasificación que provean o bien una mejor performance final o bien un menor tiempo de ejecución.*

Fecha de la presentación: FEBRERO 2019

Detección de peatones en video utilizando algoritmos de aprendizaje automático

Genaro Camele

Director: Dr. Hasperué, Waldo

Co-director: Dr. Ronchetti, Franco

Facultad de informática
Universidad Nacional de La Plata

Para obtener el grado de
Licenciado en Sistemas

Febrero 2019

Índice general

Capítulo 1

Introducción

1.1. Resumen

La detección de peatones es un tema ampliamente abordado en el área de detección de objetos utilizando algoritmos de Machine Learning. Esta es la motivación central de esta tesina, en la cual se hará uso de máquinas de vectores de soporte y redes neuronales como clasificadores, utilizando combinaciones de diferentes descriptores para llevar a cabo dicha tarea. Se realizará la medición de la precisión, exhaustividad y medida-f obtenidas por ambos modelos, esto permitirá determinar si los clasificadores propuestos alcanzan resultados aceptables en un contexto tan diverso como lo es un ambiente urbano donde se lleva a cabo la detección de peatones. También se analizará cuál resulta más conveniente en la práctica, analizando la performance tanto desde el punto de vista de la eficacia como de la eficiencia temporal.

En base al modelo de máquinas de vectores de soporte propuesto se realizará un análisis de la transferencia de aprendizaje con la que cuenta, midiendo si al evaluar un conjunto de datos diferentes al que se entrenó alcanza métricas similares.

1.2. Motivación

La inteligencia artificial está cada día más arraigada a nuestras vidas. Desde nuestro móvil proponiéndonos artículos o productos que son de nuestros interés, hasta algoritmos de predicción financiera que guían acciones en la bolsa de valores. Los grandes avances en el hardware y software hacen posible cómputos de altas prestaciones para satisfacer las necesidades cada vez más demandantes tanto del mercado como de los ciudadanos.

El procesamiento de imágenes no es una tecnología precisamente nueva, pero a medida que pasa el tiempo se han desarrollado técnicas y herramientas que permiten ampliar la frontera útil que esta rama de la ciencia ya tiene bastante extendida.

La detección de peatones es uno de los temas más importantes de la detección de objetos dentro del área de visión por computadora. En los últimos años ha atraído la atención de muchos investigadores, produciendo grandes avances debido a su impacto social y aplicaciones en seguridad vial [34] [56] [8] [41]. Estos avances fueron tan exitosos que se logró resolver el problema con tasas de aciertos satisfactorias, posteriormente, los investigadores comenzaron a enfocarse en problemas puntuales como la detección de personas parcialmente ocultas, o llevar a cabo la tarea en imágenes borrosas o con poca resolución [42].

La detección de peatones, como otras tareas de detección de objetos, se puede separar en tres etapas: la obtención de un descriptor de imágenes adecuado para representar a un peatón, un modelo de clasificación que distinga imágenes de peatones (positivas) de las que no lo son (negativas) en base al descriptor obtenido, y un sistema de ventana deslizante que analiza porciones de la imagen en busca de los peatones en una escena realizando varias ejecuciones del clasificador.

Día a día se desarrollan nuevas técnicas y herramientas que ayudan a resolver el problema en cuestión. Descriptores como HOG [12] (cálculo de gradientes entre píxeles vecinos de la imagen) y LBP [37] (variaciones de los píxeles que permiten obtener información acerca de la textura de la imagen) fueron, en los comienzos de la práctica, protagonistas de diversos estudios en la última década relacionados con la detección de peatones utilizando algoritmos de Machine Learning [31] [29] [13] [30] [57].

Durante todos estos años los investigadores han centrado su atención en mejorar las técnicas y resultados obtenidos en la clasificación de peatones [3] [27]. La aparición del concepto de Deep Learning [47], técnica que permite la obtención de descriptores de manera automática y nuevas funciones de aprendizaje automático en el mundo del Machine Learning, abrió las puertas a nuevas investigaciones [1] [52] [9].

En la actualidad no es posible considerar una técnica como solución estándar para problemas de este tipo. además, la puesta en práctica en un entorno real suma dificultades técnicas como por ejemplo: el hardware necesario para el procesamiento de imágenes y algoritmos de entrenamiento, conseguir ejemplos de entrenamiento nuevos a fin de adaptar el clasificador a un contexto específico y a las variaciones a las que se puede ver expuesto el sistema (imágenes de peatones en situaciones dificultosas de reconocer como zonas oscuras, lluvia, diferentes grados de luz y colores, etc.), entre otras.

Por los motivos antes mencionados la utilización de un modelo entrenado con una base de datos estandarizada en un entorno real no siempre consigue buenos resultados. Esto conlleva

al análisis de la transferencia de aprendizaje que resulta de entrenar un modelo con un conjunto de datos y utilizarlo con otro. Para esto es necesario el análisis de diferentes descriptores y clasificadores adecuados para el problema planteado, así como el estudio de los resultados en diferentes conjuntos de datos, verificando la eficacia de la transferencia de aprendizaje. Durante el desarrollo de esta tesina, se publicó un artículo en el Congreso Argentino de Ciencias de la Computación (CACIC) sobre la efectividad de entrenar clasificadores con diferentes bases de datos al enfrentarse a otra base de datos con diferente información en cuanto a tamaños, colores y contexto de las imágenes que la componen para poder medir sistemáticamente este concepto [6].

1.3. Objetivos

En el presente documento se mostrará el proceso de reconocimiento de peatones en tareas tanto de clasificación como detección. Resolver esta problemática en la vida real abre el camino hacia un mundo con automóviles autónomos, vigilancia inteligente, prevención y mitigación de riesgos, entre muchas otras ventajas que están a la vanguardia en los laboratorios de todo el mundo. Se analizará, además, una comparativa de las bases de datos más utilizadas en el estado del arte para evaluar que tan eficaces son al enfrentarse a datos con diferentes distribuciones, este concepto se conoce como *transferencia de aprendizaje*.

El objetivo general de la tesina es la aplicación de diferentes algoritmos de aprendizaje automático para la clasificación de peatones en imágenes, evaluando en el proceso la transferencia de aprendizaje de las diferentes bases de datos utilizadas. En cuanto a objetivos específicos, se listan los siguientes:

- Analizar los descriptores existentes más utilizados para la clasificación de peatones.
- Analizar diferentes técnicas de procesamiento de imágenes como ser el uso de ventanas deslizantes, imágenes en pirámide, supresión no máxima, entre otras, para llevar a cabo la detección de peatones.
- Estudiar y aplicar técnicas de aprendizaje automático para la clasificación de peatones utilizando máquinas de vector de soporte y diferentes tipos de redes neuronales.
- Analizar la transferencia de aprendizaje al utilizar las bases de datos del estado del arte.
- Presentar la comparación entre resultados obtenidos con los algoritmos de clasificación, bases de datos y descriptores estudiadas.

1.4. Organización del documento

Capítulo 2: Se explican los conceptos básicos de la representación, almacenamiento y manipulación de imágenes en una computadora, los atributos, más adelante referenciados como *descriptores*, que podemos extraer de ellas; y como dicha información sirve de materia prima para algoritmos de clasificación. Se hace una pequeña introducción de estos últimos.

Luego se explica las técnicas y métricas utilizadas durante el proceso de detección de peatones.

Capítulo 3: Se brinda una introducción a las redes neuronales, su estructura, y como funcionan durante su etapa de entrenamiento y evaluación. Después de haber presentado este modelo básico, se pasa a explicar otros modelos más modernos que, veremos, resultan mucho más provechosos, eficaces y eficientes en la tarea planteada.

Capítulo 4: Se introducen los conjuntos de datos que se utilizarán para entrenar y evaluar los diferentes modelos. Se explica además, en qué consiste el proceso de transferencia de aprendizaje.

Capítulo 5: Se detallan todos los experimentos realizados para evaluar la transferencia de aprendizaje entre los distintos conjuntos de datos, y los experimentos realizados para medir la clasificación de peatones utilizando dichos datasets y diferente cantidad de ejemplos, descriptores y clasificadores.

Capítulo 6: Se finaliza este documento con las conclusiones sobre los resultados, clasificadores y las líneas de trabajos futuros del estudio con base a lo evaluado y los objetivos propuestos.

Capítulo 2

Visión por computadora

Existen distintos tipos de herramientas para la obtención de imágenes, desde cámaras digitales que permiten la representación de dos dimensiones de una escena física, hasta sensores avanzados que aportan más datos de la escena, alcanzando las tres dimensiones.

En este capítulo se revisarán los conceptos básicos de representación de imágenes en una computadora, qué información podemos extraer de las mismas (información que se conocerá más adelante con el término *descriptores*), modelos de clasificación que permitan determinar si una imagen corresponde a un peatón o no, métricas que permitan evaluar si esta última tarea se realiza con eficacia, y cuáles técnicas se aplican para detectar peatones en una imagen específica.

2.1. Conceptos básicos

Una imagen digital está compuesta por píxeles organizados en una matriz bidimensional. El valor en cada uno de los píxeles (o celdas de la matriz) corresponde a información respecto a su color, temperatura, intensidad, luminosidad, entre otros aspectos que dependen de cual sea el sistema de representación que se esté utilizando para almacenar la imagen.

Cuando hablamos de un video no es más que un conjunto de imágenes individuales que respetan una secuencia temporal. Estas imágenes que conforman el videometraje se los llama frame (cuadro). Las cámaras estándares actuales poseen la capacidad de grabar a 29 frames por segundo.

2.1.1. Sistemas de representación

Como se mencionó anteriormente, existen diferentes sistemas de representación. En esta tesina se utilizaron dos representaciones que almacenan información de color para cada píxel: escala de grises y RGB.

2.1.2. Escala de grises

En una imagen representada en escala de grises cada valor de las celdas de su matriz de representación corresponderá a un valor en el intervalo $[0-1]$, siendo 0 negro absoluto y 1 blanco. Los valores intermedios representan la intensidad en los distintos tonos de gris. La figura 2.1 muestra una imagen en escala de grises y los valores para una sub-porción de 9 píxeles.

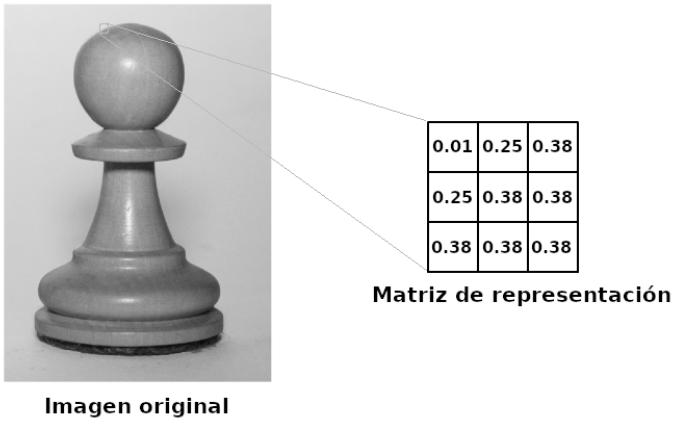


Figura 2.1 Imagen en blanco y negro.

2.1.3. RGB

Por sus siglas en inglés, Red, Green & Blue (Rojo, Verde y Azul) describe la composición de la luz a partir de estos tres colores. Una imagen en RGB está definida por un tensor de $N * M * 3$, siendo N y M el ancho y alto, y para cada uno de estos píxeles, tres valores correspondientes al rojo, verde y azul. En la figura 2.2 se muestra una imagen a color y los valores de una sub-sección de 9 píxeles de la misma para cada uno de los colores mencionado.

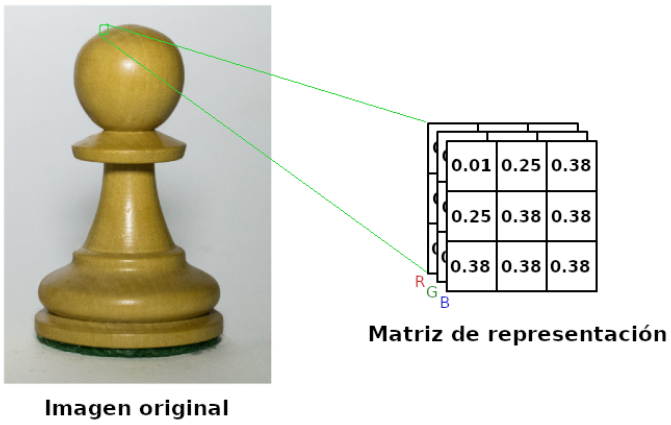


Figura 2.2 Imagen en escala RGB.

2.2. Descriptores

Un descriptor (también llamado descriptor visual) es información que describe alguna característica particular de la imagen como podría ser la textura, el color, profundidad, entre otros. A continuación se detallan los dos descriptores que se utilizarán para un algoritmo de aprendizaje automático que será explicado más adelante.

2.2.1. Histograma de gradientes orientados

El histograma de gradientes orientados (HOG por sus siglas en inglés) se calcula en base al gradiente de la imagen [31] [29] [13] [27]. El gradiente indica cuanto varía una determinada magnitud física (en este caso, el color de los píxeles) al desplazarse en una determinada dirección (píxeles vecinos). Dado que dicho gradiente tiende a ser más grande en los bordes de los objetos debido a la diferencia de colores que se presenta, es apto para detectar la forma de los mismos.

El cálculo de HOG en una imagen consta de 4 parámetros principales: cantidad de celdas, cantidad de bloques de celdas, la cantidad de *bins* que posee el histograma y la normalización que se le aplica a cada bloque contemplado. El procedimiento es el siguiente: en base al gradiente, representado como ángulo-magnitud, se divide la imagen en celdas de igual tamaño. En la figura 2.3a se muestra la imagen original, y a su derecha (figura 2.3b), el resultado del proceso de división por celdas.

(a) Imagen original. (b) Imagen por celdas. (c) Imagen por bloques.

Figura 2.3 Representación del proceso de división de la imagen en celdas.

Para cada una de estas celdas se calcula un histograma con 9 barras (también llamados *bins*). Cada barra corresponde a un rango de ángulos, y cada punto del gradiente contribuye a dicho barra en forma proporcional a su magnitud. En la figura 2.4 se puede observar hacia donde baja la intensidad de los píxeles para una celda determinada de la imagen. La magnitud del gradiente es cuanto varía el color y la dirección es el ángulo en el que se presenta dicho cambio. Cada uno de estos 9 *bins* que constituyen el histograma son los ángulos de orientación posibles que van desde 0° a 180° cuyo valor final para cada uno de los ángulos será la suma de los valores absolutos de las magnitudes. Nótese que los bins no abarcan los grados que van desde 180° a 360°, dado que se utilizan los llamados *gradientes sin signo* ya que el gradiente y su negativo son representados por el mismo número, mientras que los denominados *gradientes con signo* van desde 0° a 360°. Si bien tanto el tamaño de las celdas y los bloques, como la cantidad de *bins* de los que consta el histograma no es una constante, empíricamente se ha demostrado que bloques de $2*2$ celdas, celdas de $8*8$ píxeles y los gradientes sin signo ofrecen una mejor performance en la detección de peatones [12].

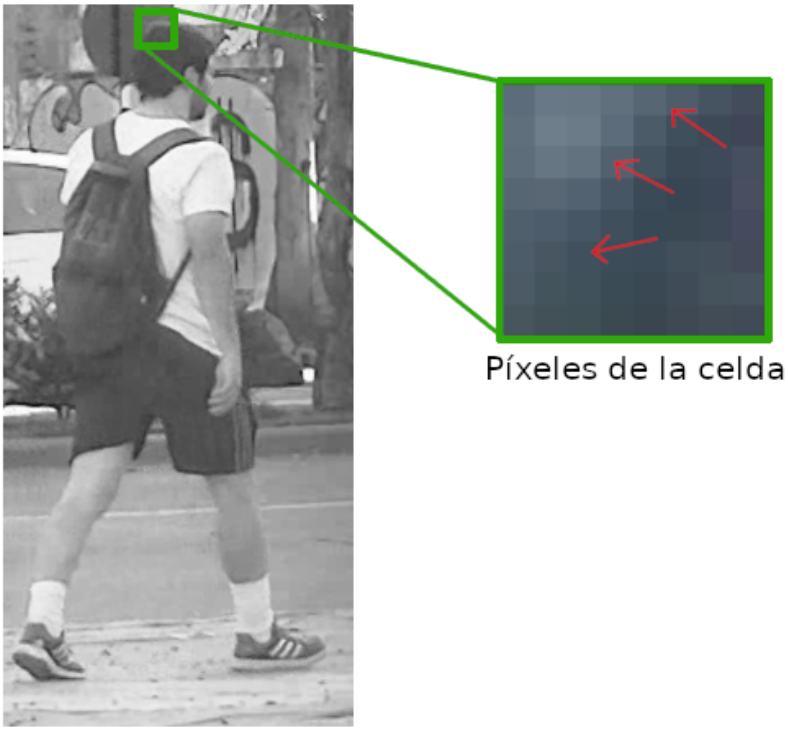


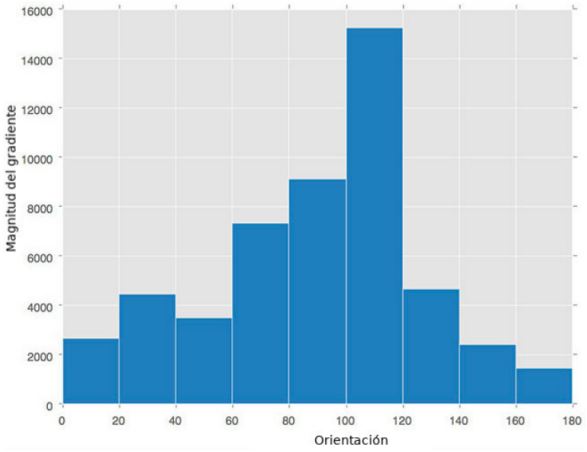
Figura 2.4 Representación de la orientación de la intensidad para los píxeles de una celda específica.

Una vez calculados los gradientes, se realiza una normalización a nivel de bloque de $N * M$ celdas, en la figura 2.3c se muestra gráficamente con bloques de $2 * 2$ celdas.

Los bloques normalizados pierden posteriormente su estructura 2D y se concatenan en un vector unidimensional de modo de formar el descriptor final (figura 2.5b).

La visualización del descriptor para una imagen dada se puede apreciar en la figura 2.5a, donde cada marca blanca indica el ángulo de dirección y cuanto más larga se representa dicha marca, mayor fue el cambio entre las tonalidades de grises de los píxeles continuos. Si bien hay métodos y librerías que permiten el cálculo de HOGs en escala RGB, por motivos de eficiencia se trabajará solo sobre imágenes en escala de grises.

(a) Visualización de un HOG.



(b) Histograma final del proceso de cálculo del descriptor HOG.

Figura 2.5 Visualización e histograma final del descriptor HOG.

Normalización por bloques

Como se mencionó anteriormente, los bloques computados por cada celda de la imagen, son sometidos a un proceso de normalización. Hay cuatros procesos bien conocidos en el estado del arte: la normalización L1, la L1-sqrt, la normalización L2 y por último la normalización L2-hys.

Siendo v el vector de información no normalizado, las cuatro funciones citadas se definen de la siguiente manera:

- **Normalización L1**

$$f = \frac{v}{||v||_1 + e}$$

- **Normalización L1-sqrt**

$$f = \sqrt{\frac{v}{||v||_1 + e}}$$

- **Normalización L2**

$$f = \sqrt{\frac{v}{||v||_2^2 + e^2}}$$

- **Normalización L2-hys** Consiste en limitar todos los valores generados por una normalización L2 al valor máximo 0,2; y volviendo a normalizar el vector resultante.

Para el trabajo aquí presente se utiliza la normalización L2-hys debido a los resultados obtenidos y que es uno de los más utilizados [21].

2.2.2. Patrón binario local

Los patrones binarios locales, ahora en más mencionado como LBP correspondientes a su nomenclatura en inglés, son descriptores de textura [30]. Se calculan a base de una imagen que debe estar en escala de grises, se selecciona un numero r de vecinos que rodean al píxel central. Al igual que con los HOG, se computa por celdas de $N * M$ píxeles, siendo N y M enteros mayores a 0. En la figura 2.6 se muestra el procedimiento con una celda de $3 * 3$ y, por consiguiente, un $r = 8$ ya que en una celda $3 * 3$ un píxel no podría tener más de 8 vecinos. Si el vecino posee un valor mayor, entonces su valor será 0, caso contrario, 1.

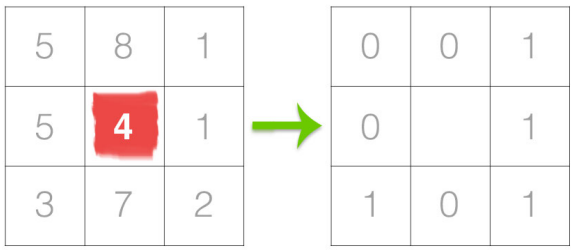


Figura 2.6 Proceso de cálculo de los valores de los píxeles vecinos al píxel central que está siendo computado, para obtener el descriptor LBP.

Con $r = 8$ se pueden obtener $2^8 = 256$ combinaciones posibles para una celda. El resultado de los bits calculados en el paso anterior se almacenan en un arreglo de tamaño r, y las posiciones se asignan en el sentido de las agujas del reloj empezando por el píxel en la posición superior derecha del píxel central.

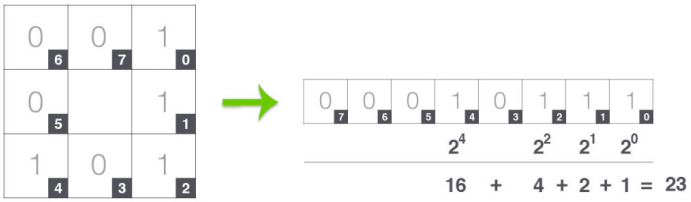


Figura 2.7 Transformación de los valores en un arreglo formando un número binario de r dígitos.

Por último, dicho valor se pasa a decimal y se asigna en la posición del píxel central correspondiente en la matriz resultante del procedimiento, tal como se muestra en la figura 2.8. Este proceso se aplica a todos los píxeles de la imagen.

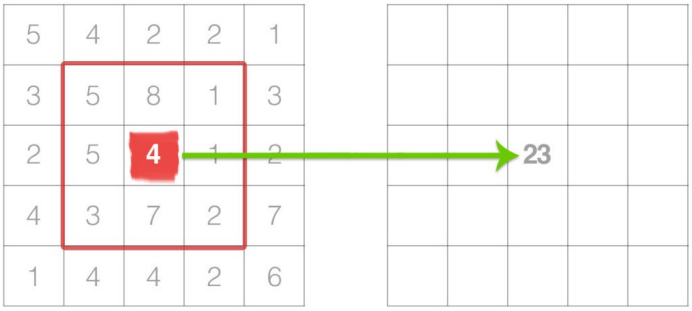
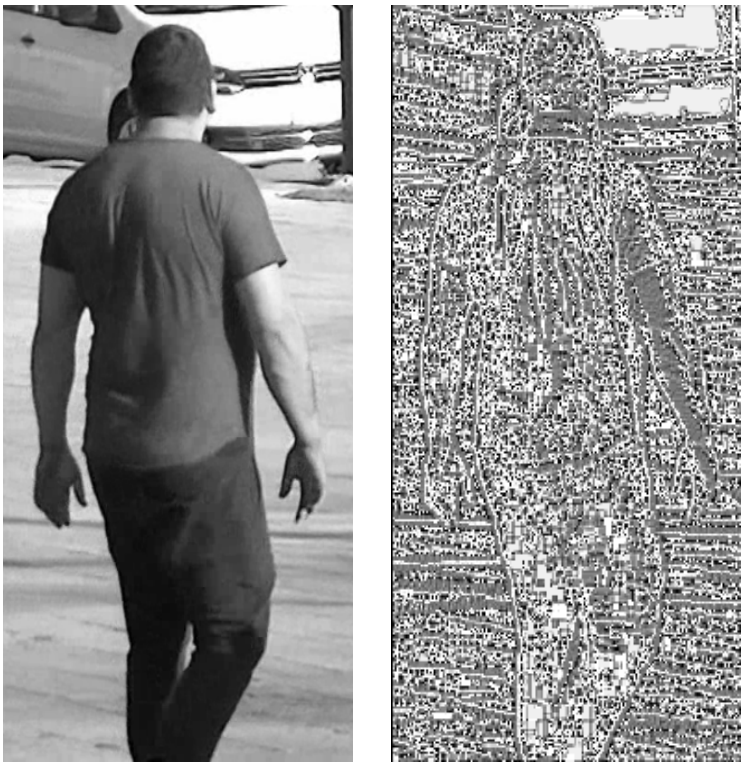


Figura 2.8 Cálculo final de la celda en un único valor que corresponde a la posición del píxel central en la matriz resultante.

Finalmente, se genera un histograma con 2^r *bins*, donde cada píxel computado durante el cálculo del descriptor incrementa en 1 el bloque correspondiente al número binario obtenido en el proceso. Con el r propuesto, el histograma resultante tendrá 256 *bins* ya que $2^8 = 256$, yendo del 0 al 255. Se puede apreciar de forma visual el descriptor generado en la figura 2.9b, tomando como referencia una imagen en particular 2.9a.



(a) Imagen original. (b) Visualización LBP.

Figura 2.9 Imagen y la representación gráfica del descripto LBP

Al igual que con los HOG los histogramas se concatenan para formar el descriptor final (figura 2.10). La performance de este descriptor en la tarea de detección y clasificación de peatones fue evaluada en el estado del arte: en [22] los investigadores se valen del dataset INRIA (que es presentado en el capítulo 4 de transferencia de aprendizaje) para demostrar que al combinar HOG y LBP con un SVM (será visto a continuación en la sección de modelos de clasificación) se logran resultados mucho mejores que los obtenidos usando un único descriptor. En [55] hacen uso del mismo conjunto de datos y la combinación de ambos descriptores, pero utiliza AdaBoost [20] como clasificador, nuevamente demuestran que ambos descriptores son mas efectivos que los resultados obtenidos en los papers que hace uso de HOG como único descriptor en sus experimentos [12].

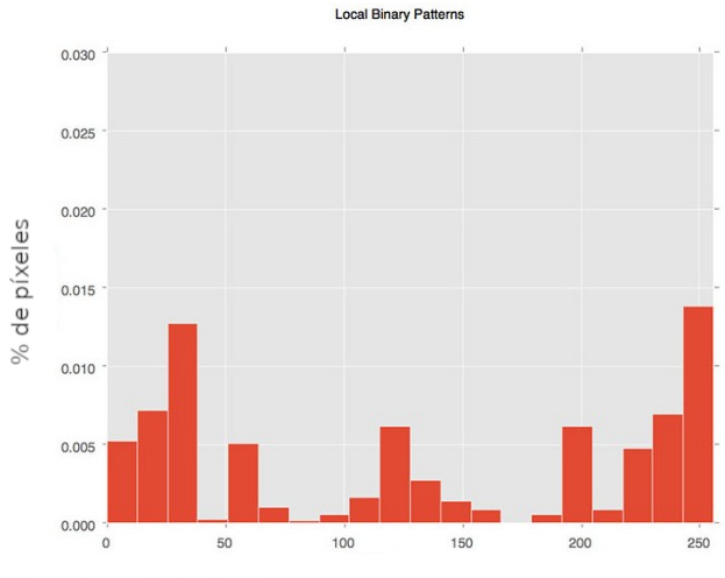


Figura 2.10 Histograma final del proceso de cálculo del descriptor LBP.

2.3. Modelos de clasificación

Un modelo de clasificación no es más que un algoritmo que permite determinar a qué grupo (más adelante llamado *clase*) pertenece un conjunto de datos. A lo largo de esta tesina se estudian dos modelos de clasificación: una máquina de vectores de soporte (explicado a continuación y en la sección 2.3.1) y redes neuronales (explicado en el capítulo 3 de Deep Learning).

Una máquina de vectores de soporte (SVM), es un algoritmo de aprendizaje supervisado, que representa los valores de las muestras de aprendizaje en el espacio, luego busca un hiperplano que genere la separación más amplia posible entre las diferentes clases de muestras [38]. Si bien se puede utilizar un SVM para clasificación de múltiples clases, en la tesina solo se hará uso del algoritmo para clasificaciones binarias que serán referenciados como clase *1* y clase *-1*.

Aunque el SVM original propuesto tiene un kernel lineal, existen soluciones propuestas por los autores Bernhard E. Boser, Isabelle M. Guyon y Vladimir N. Vapnik que utilizan funciones polinómicas a fin de realizar una separación no lineal de los datos [5]. Este tipo de SVM no será abarcado en este trabajo.

2.3.1. Máquinas de vectores de soporte (SVM)

Vectores

Se denomina vector de dimensión n a un conjunto de n números reales. El conjunto de todos los vectores de dimensión n se representa como $\Re^n$. Un vector V perteneciente al espacio $\Re^n$ se podría representar como:

$$V = (V_1, V_2, \ldots, V_n)$$

Donde $v \in \Re^n$. Así, si un vector se encuentra en un plano R^2 se puede definir por sus coordenadas (x, y) representándose como:

$$\overrightarrow{V} = V = (V_x, V_y)$$

Siendo sus coordenadas V_x, V_y. Si un vector se encuentra en un plano $\Re^3$ se puede definir por sus coordenadas (x, y, z) y su representación sería:

$$\overrightarrow{V} = V = (V_x, V_y, V_z)$$

Y así respectivamente para cualquier espacio dimensional con el que se trabaje.

Hiperplano

Un hiperplano es un subespacio del espacio ambiente (el espacio en el que vamos a trabajar), de codimensión 1, es decir, que siendo un espacio ambiente de dimensión n, nuestro hiperplano tendrá una dimensionalidad $n-1$.

Si estamos frente a un espacio unidimensional, el hiperplano no es más que un punto que divide la recta en dos, si fuera bidimensional, el hiperplano constaría de una recta que divide nuestro plano original en dos sub-espacios, y así para cada una de las infinitas dimensiones.

Este hiperplano se representa con la siguiente ecuación lineal:

$$\overrightarrow{w} \cdot \overrightarrow{x} - b = 0$$

Donde $\overrightarrow{w}$ es el vector normal (un vector que es perpendicular al plano tangente, el plano tangente a una superficie en un punto dado es el plano que simplemente "toca" la superficie en ese punto) del hiperplano y $\frac{b}{||\overrightarrow{w}||}$ el corrimiento del hiperplano desde el origen hacia $\overrightarrow{w}$.

La incorporación de este concepto es importante ya que conlleva el mecanismo funcional fundamental de las máquinas de vector de soporte.

SVM lineal

Suponiendo que se tienen n puntos en el plano dados de la forma:

$$(\overrightarrow{x_1}, y_1) \dots (\overrightarrow{x_n}, y_n)$$

Siendo $\overrightarrow{x_i}$ un vector m-dimensional y y_i un número indicando la clase a la que pertenece el punto $\overrightarrow{x_i}$. Se busca obtener el hiperplano de separación óptima entre los pertenecientes a una clase y los $\overrightarrow{x_i}$ de la clase restante, dicha separación está definida tal que la distancia entre el hiperplano y el $\overrightarrow{x_i}$ más cercano para cada una de las clases, sea la máxima.

De esta separación depende la correcta clasificación. Existen dos maneras de definir el margen óptimo: el denominado *Margen duro* y *Margen suave*, ambos se describen a continuación:

Margen duro (Hard-Margin)

Si el conjunto de datos es linealmente separable, se generan dos hiperplanos paralelos, uno para la clase 1 y otro para la clase -1 de manera que la distancia entre ellos sea lo más grande posible. La región entre ambos hiperplanos se llama margen y el hiperplano de mayor margen, intuitivamente se define como aquel hiperplano que se encuentra exactamente en la mitad y logra la separación óptima entre ambas clases.

Ambos hiperplanos pueden representarse con las siguientes ecuaciones:

Cualquier valor del dataset que pertenezca a la clase 1:

$$\overrightarrow{w} \cdot \overrightarrow{x} - b = 1$$

Cualquier valor del dataset que pertenezca a la clase -1:

$$\overrightarrow{w} \cdot \overrightarrow{x} - b = -1$$

La distancia entre ambos hiperplanos se define por $\frac{2}{||\overrightarrow{w}||}$, por lo que, para maximizar dicha distancia, se busca minimizar $||\overrightarrow{w}||$. Antes de obtener las distancias, se debe evitar contemplar los puntos que se encuentran dentro del margen. Para ello, agregamos restricciones a ambas ecuaciones:

Para todo $y_i = 1$**:**

$$\overrightarrow{w} \cdot \overrightarrow{x} - b \geq 1$$

Para todo $y_i = -1$**:**

$$\overrightarrow{w} \cdot \overrightarrow{x} - b \leq 1$$

La distancia mínima entre cualquier punto y un hiperplano se calcula a partir de la siguiente ecuación:

$$\frac{|a_1 y_1 + a_2 y_2 + ... a_n y_n - d|}{\sqrt{a_1^2 + a_2^2 + ... a_n^2}}$$

En la figura 2.11 se puede apreciar gráficamente el proceso de obtención del margen óptimo:

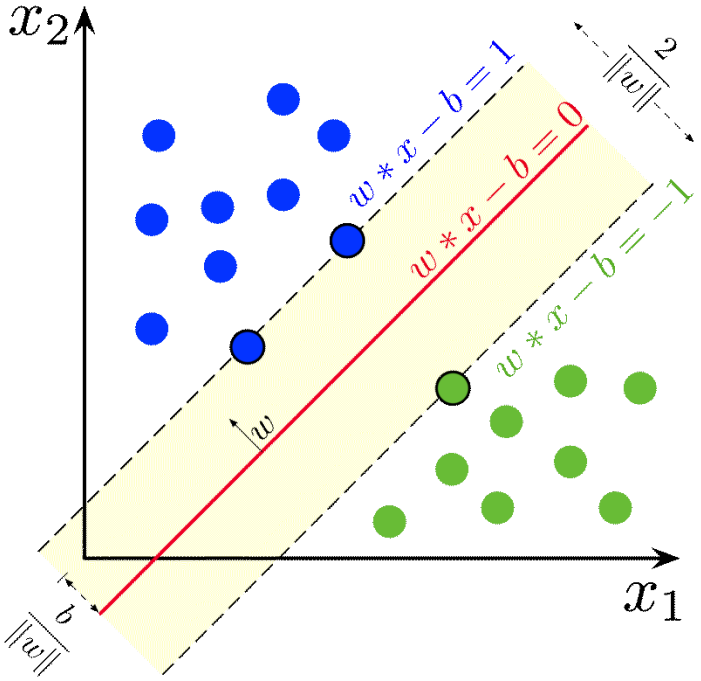


Figura 2.11 Hiperplano que separa perfectamente ambas clases en el ejemplo.

Margen suave (Soft-Margin)

Este margen se utiliza para datasets que no son linealmente separables. Para ello se introduce una función de costo llamada Hinge loss, dada por la siguiente función:

$$max(0, 1 - y_i(\overrightarrow{w} \cdot \overrightarrow{x} - b))$$

Siendo y_i la clase a predecir (en nuestro caso 1 o -1) y $(\overrightarrow{w} \cdot \overrightarrow{x} - b)$ la clase predicha por el modelo. La función da 0 cuando $\overrightarrow{x}$ se halla del lado correcto del margen, para los valores incorrectos el valor resultante es la distancia al margen. Teniendo en cuenta esto, lo que necesitamos es reducir dicho costo, lo que significaría que nuestro clasificador posee un margen que abarca (por lo menos, en su gran medida) correctamente los valores $\overrightarrow{x}$, para ello se aplica la siguiente función:

$$[\frac{1}{n}\sum_{i=1}^{n} max(0, 1 - y_i(\overrightarrow{w} \cdot \overrightarrow{x} - b))] + \lambda ||\overrightarrow{w}||^2$$

Siendo λ la compensación entre incrementar el margen y asegurarse que $\overrightarrow{x}$ se encuentra del lado que le corresponde, cuanto menor es el valor de la variable, más despreciable es este segundo argumento de la función, siendo el vector soporte aproximadamente el promedio del costo de todos los puntos representados.

Entrenamiento de un SVM

Durante el trabajo alimentaremos al SVM con los descriptores extraídos de imágenes de peatones y no peatones, dicha información será entregada al clasificador, que dispondrá los valores en un plano de dimensionalidad muy alta, para encontrar un plano que aborde todas las dimensiones y que genere la separación más pronunciada de ambas clases: peatones, y no peatones. Este proceso se conoce como **etapa de entrenamiento**.

Todos los ejemplos del conjunto de datos tienen establecida la clase a la que pertenece. Cuando se entrena un modelo con las clases preestablecidas estamos frente a un **entrenamiento supervisado**, existen casos como el clustering donde el objetivo es formar grupos de datos a partir de patrones que resultan desconocidos a la hora de entrenar el modelo; este último proceso se denomina **entrenamiento no supervisado** y no va a ser abordado en esta tesina.

Luego, llega la llamada **etapa de evaluación**, donde se extraen los mismos datos de imágenes que no formaron parte del conjunto de entrenamiento. Dicho conjunto es conocido como muestras o dataset de evaluación.

La información obtenida de los descriptores se representan en el plano y, a partir de la evaluación del lado del margen (vector soporte) en el que se ubiquen los datos se obtiene la clase a la cual pertenece un dato, ejemplo o imagen.

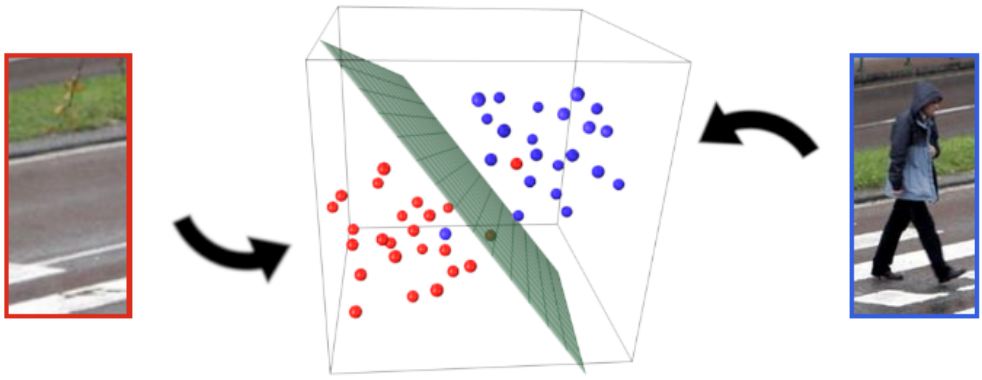


Figura 2.12 Representación ilustrativa de los descriptores extraídos de los ejemplos positivos (en azul) y negativos (en rojo) separados por el vector soporte óptimo en el SVM.

2.3.2. Redes neuronales

Una red neuronal es un modelo computacional que data de la década del 40 cuando el neurólogo Warren McCulloch y el lógico Walter Pitts desarrollaron un modelo matemático basado en neuronas biológicas para afrontar procesos computacionales [43]. Este tema será abordado con más detalle en el capítulo 3 sobre Deep Learning ya que la amplia variedad de temas amerita una separación específica para poder explicar los modelos basados en redes neuronales que fueron producto de esta tesis.

2.4. Clasificación de peatones

El proceso de clasificación consiste en, dado un modelo entrenado, ya sea un SVM o red neuronal, obtener un conjunto de datos de evaluación (datos diferentes a los que se entrenó originalmente el modelo), extraer la misma información (por ejemplo, el histograma de gradientes orientados y/o el resultante de la extracción del LBP) y dársela al modelo para que determine si se trata de un peatón o no. Para ejemplificar la clasificación, si se quisiera clasificar peatones se necesitaría un modelo, supóngase, SVM. Luego, se debería obtener un conjunto de datos de entrenamiento, que constaría de una serie de imágenes de diferentes peatones y otro subconjunto de imágenes que *no* lo son. Se extraen los descriptores de cada una de estas imágenes y dichos valores se cargan al SVM, que luego de representar todos los datos en el plano calculará un hiperplano que mejor separe los ejemplos positivos (peatones) de los negativos (cualquier otra imagen). Ahora que el algoritmo posee una separación óptima de las clases se puede evaluar información de nuevas imágenes que el SVM determinará

si pertenecen a la clase *peatón* o *no peatón*, permitiendo realizar una clasificación de las mismas.

2.5. Detección de peatones

En el apartado anterior se describió el proceso de clasificación: dada una imagen, el clasificador permite distinguir la clase a la que pertenece, peatón o no peatón.

A la hora de enfrentarse a una situación real, el peatón podría no conformar completamente la imagen, por el contrario, habría que localizar su posición en una imagen donde podemos encontrar múltiples objetos muy variantes. Es aquí donde hay que aplicar un conjunto de técnicas que permiten usar nuestro clasificador entrenado para tareas de detección.

2.5.1. Ventanas deslizantes

La idea de una ventana deslizante es ir recorriendo la imagen a partir de un Bounding Box (de ahora en adelante BB) de tamaño fijo, este término hace referencia a un simple recuadro de N píxeles de ancho por M píxeles de alto que irá recorriendo toda la imagen. En su recorrido, ese BB será evaluado por el clasificador que nos dirá si se trata de un peatón o no. En la figura 2.13 se puede apreciar el proceso.

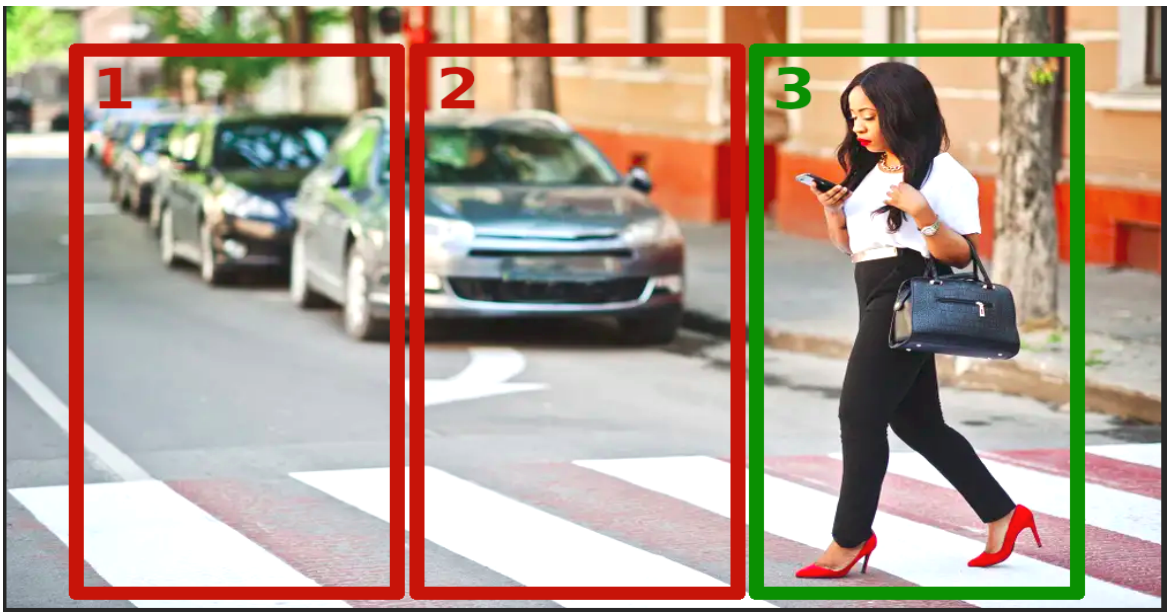


Figura 2.13 Cada ventana deslizante tiene su número. En la ventana deslizante número 3 el clasificador dio positivo.

2.5.2. Imagen en pirámide

Dada una imagen se la recorre usando un BB y se van fijando los peatones que el clasificador detecte. Ahora, puede ser posible que las imágenes con las que se entrenó dicho clasificador hayan estado en una escala diferente (por ejemplo, el peatón se veía más chico), por ello, para conseguir mejores resultados hay una técnica que consiste en realizar el proceso de ventana deslizante en diferentes escalas de la imagen original. Para ello, se hace un múltiple re-escalado de la imagen a n escalas inferiores que serán un $\varepsilon * i$ más pequeñas, siendo i el número de imagen re-escalada actualmente. Por ejemplo, si se quiere hacer una imagen en pirámide de 3 escalas con $\varepsilon = 1.5$, entonces $i = 3$. Por poner un ejemplo práctico, para la primera iteración ($i = 1$) la imagen será $1.5 * 1 = 1.5$ más pequeñas, es decir, su ancho y alto será casi la mitad de chico (ancho $=$ ancho$/1.5$ y alto $=$ alto$/1.5$), y así sucesivamente en las siguientes iteraciones. Se itera sobre cada una, y frente a cada positivo adquirido por el clasificador, hay que aplicar un coeficiente $coef = i$ a las coordenadas x e y , y al ancho y alto del BB detectado para poder representarlo en la imagen original. Este procedimiento aumenta considerablemente la probabilidad de detectar un peatón.

2.5.3. Intersección sobre la unión (Intersection Over Union)

La intersección sobre la unión, de ahora en adelante IOU por sus siglas en inglés, es una métrica de evaluación para saber qué tanto se superponen dos áreas definidas. Se calcula a partir de: el área del BB del objeto real (que es brindado por las bases de datos de conjuntos de entrenamiento junto con cada imagen) y del BB actual de la ventana deslizante que dio positivo para el clasificador (figura 2.14).

La ecuación es: $IOU = \frac{\text{área que se superpone entre ambos BB}}{\text{área de la unión de ambos BB}}$ siendo un valor cercano a 0 una superposición muy pobre y valores cercanos a 1 una superposición muy marcada.

Esta métrica permite definir un umbral para considerar cuando nuestro clasificador realizó una buena detección, durante el trabajo se fijó un $IOU \geq 0.5$ para definir un positivo.

Figura 2.14 Cálculo del IOU a partir del BB del peatón y la ventana deslizante en su posición actual.

2.5.4. Supresión no máxima

¿Qué ocurre cuando se detecta varias veces al mismo peatón (porque el IOU sobrepasó el umbral que definimos para varias ventanas deslizantes frente a un mismo peatón), generando múltiples BB positivos en la imagen? En la figura 2.15 se puede apreciar la situación descrita.

Para resolver este problema, se aplica un algoritmo popularmente llamado *Non Maximal Suppression*, de ahora en adelante referenciado como NMS, que elimina cada BB que se superpone con otro BB un IOU mayor a un umbral específico, dejando un solo BB por cada grupo de estos que se superponía (figura 2.16).

Independientemente del clasificador y conjuntos de datos utilizados para la detección, NMS nos permite eliminar detecciones redundantes. Por ejemplo, en [19] se utiliza un SVM sobre una base de datos llamada PASCAL [18] que provee imágenes de diferentes tipos de objetos como muebles, vehículos, personas, entre otros; y hacen uso de NMS para mejorar los resultados obtenidos.

Figura 2.15 Un peatón detectado varias veces, lo que generó que varios BB redundantes se grafiquen sobre la imagen.

Figura 2.16 Un único BB resultante después del proceso de NMS.

2.5.5. Seguimiento (Tracking)

Como se mencionó en la sección 2.1 del capítulo 2, un video no es más que un conjunto de imágenes continuas llamadas frames. En cada uno de estos se muestra un imagen que es, en la mayoría de los casos, ligeramente distinta a sus contiguas. Sin embargo, estos leves cambios en la imagen conllevan la generación de diferentes descriptores, en consecuencia se

puede dar el caso en el que se generen falsos negativos, decir que ahí no se encuentra ningún peatón cuando en el frame anterior, 30 milisegundos aproximadamente antes, efectivamente estaba ahí.

Para evitar este tipo de problema se utiliza la técnica de seguimiento, la cual consta en establecer un tiempo de vida a cada uno de los BB que quedaron como resultado de la supresión no máxima. Quedándonos por simpleza con un solo BB, si en el frame siguiente aparece uno nuevo que se superpone con este (con un IOU mayor a un umbral pre establecido), entonces se reemplaza por el BB nuevo y se resetea su tiempo de vida al inicial. Caso contrario, el tiempo de vida sigue decrementando, al llegar a 0 este desaparece definitivamente. De esta manera, aún cuando no se haya detectado ningún peatón en un frame, conservamos los BB de frames anteriores que, muy probablemente, no tengas diferencias significativas en cuanto a los peatones que en él se encuentran.

2.5.6. Hard Negative Mining

El procedimiento conocido como Minería Negativa Dura o Hard Negative Mining consiste en separar los falsos positivos que arrojó nuestro modelo durante la etapa de evaluación (es decir, las imágenes que predijo como peatones y efectivamente, no lo eran) y volver a entrenarlo pero ahora con esos caso etiquetados (le agregamos la clase “No peatón”) para que los clasifique como lo debería haber hecho en una primera instancia.

2.6. Métricas

Hay varias métricas bien definidas que permiten evaluar que tan bien fue llevada a cabo la tarea de clasificación. Si bien son numerosas, en esta tesina utilizaremos la precisión, la exhaustividad y la medida-F. La primera se define como aquellas imágenes que fueron servidas al modelo para clasificar que fueron recuperados (es decir, dieron positivo en la clasificación: es peatón) y son relevantes (son efectivamente positivos); mientras que la exhaustividad se define por aquellos elementos relevantes que fueron recuperados.

En palabras simples, la precisión se mide a partir de cuantas imágenes fueron clasificadas como peatones y eran realmente peatones, y la exhaustividad por cuantas imágenes que eran realmente peatones y el modelo clasificó efectivamente como tales. En la figura 2.17 se puede distinguir gráficamente los elementos (imágenes) relevantes, y los verdaderos y falsos positivos y negativos.

Las dos métricas se definen como:

Precisión:

$$\frac{\text{Verdaderos positivos}}{\text{Verdaderos positivos + Falsos positivos}}$$

Exhaustividad:

$$\frac{\text{Verdaderos positivos}}{\text{Verdaderos positivos + Falsos negativos}}$$

Se obtendrá una precisión perfecta cuando el clasificador no haya tenido falsos positivos, y una exhaustividad perfecta cuando no tenga falsos positivos.

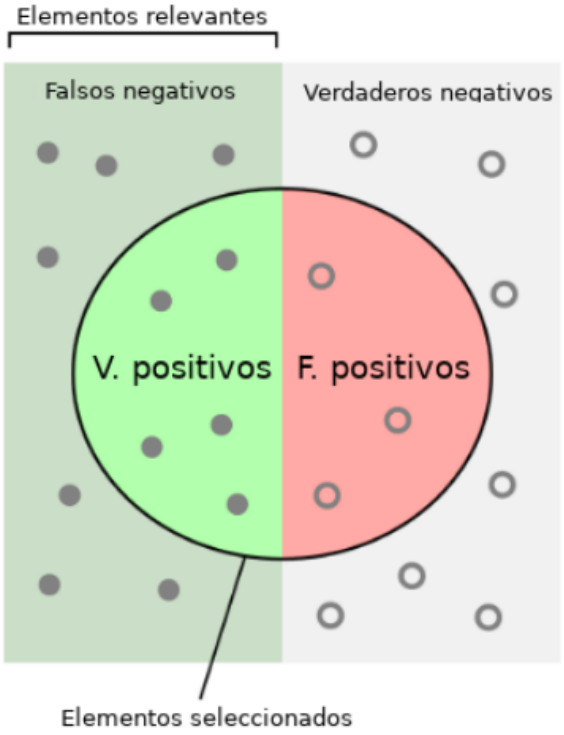


Figura 2.17 Representación gráfica de los elementos y las clasificaciones hechas por el modelo.

En cuanto a la medida-f (F-Score), es la media armónica de la precisión y la exhaustividad. Cuanto más cercano sea su valor a 1 mejor será la performance del clasificador, siendo 1 un resultado de precisión y exhaustividad perfectos. Por el contrario, una medida-F $= 0$ reflejará la peor precisión y exhaustividad posible. La fórmula de la medida-F es la siguiente:

$$2 * \frac{\text{precisión} * \text{exhaustividad}}{\text{precisión} + \text{exhaustividad}}$$

Capítulo 3

Deep Learning

En esta tesina, además de utilizar SVM como clasificador, también se hace uso de las denominadas redes neuronales artificiales. Estas fueron diseñadas basándose en la estructura neuronal biológica ya que buscan imitar su capacidad de aprender de situaciones (ejemplos de entrenamiento) para desarrollar una nueva respuesta. Por este motivo, es de gran necesidad explicar los conceptos básicos de una red neuronal biológica.

3.1. Redes neuronales biológicas

Las redes neuronales artificiales fueron inspiradas en la compleja estructura alojada en nuestro cerebro. En neurociencia, una red neuronal está compuesta de un conjunto de neuronas con la siguiente estructura:

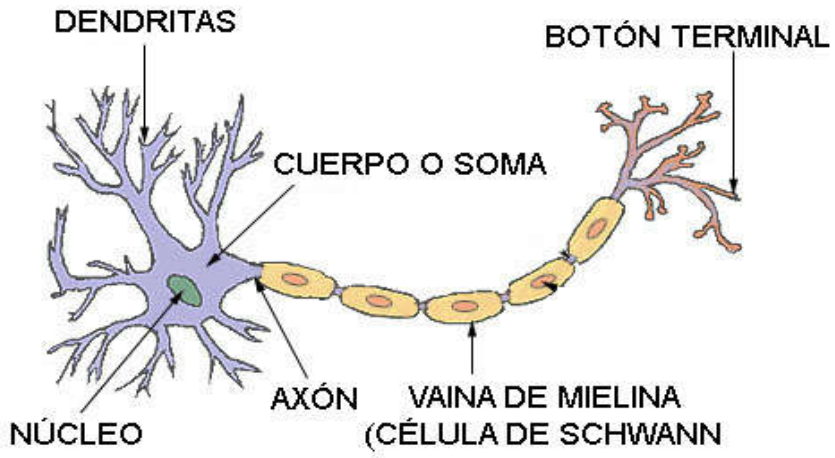


Figura 3.1 Estructura de una neurona biológica.

Cada una de estas neuronas se conecta a través de sus dendritas con el axón de sus pares, proceso conocido como sinapsis, para recibir, procesar y enviar información a partir de procesos químicos y eléctricos. El estímulo eléctrico que recibe cada una de las neuronas es acumulado en el cuerpo de la misma hasta que supera cierto umbral de voltaje eléctrico denominado voltaje de membrana (mV), al superarlo (es decir, que alcanza valores entre 55mV y 65 mV) genera un nuevo impulso eléctrico que se distribuye a través de su axón hacia las neuronas a las que está conectada. Cuando esto ocurre es que se dice que la célula neuronal se encuentra activada, caso contrario se dice que se encuentra inactiva y su voltaje varía entre valores negativos.

3.2. Redes neuronales artificiales

Las redes neuronales artificiales tienen la ventaja de poder aproximarse a cualquier función, son relativamente fáciles de entender, pueden realizar tareas tanto descriptivas como clasificatorias, y han demostrado tener excelentes resultados para tareas arduas frente a otros modelos predictivos (como SVM, Árboles de Decisión, entre otras).

La representación visual básica de una red neuronal artificial se puede observar en la figura 3.2. Gracias a esta figura es que podemos apreciar varias similitudes con respecto a una red neuronal biológica: haciendo foco solo en la capa oculta a modo de ejemplo, cada nodo puede verse como una neurona, esta posee múltiples entradas (dendritas) que reciben la información de las salidas de las neuronas de la capa anterior (que nos dan su valor de salida a través de su axón), recibido estos valores, cada neurona artificial opera con los valores recibidos y aplica una función de activación (análogo al voltaje de membrana) que será los valores de salida que serán transmitidos a todas las neuronas pertenecientes a la siguiente capa de la red neuronal.

Una red neuronal artificial se define como una tupla ordenada (N, V, w) donde N es el conjunto de neuronas, V es un conjunto $(i, j)|i, j \in N$ cuyos elementos son las conexiones entre la neurona i y la neurona j. w_{ij} es el peso de la conexión (es decir, el valor que pondera la entrada proveniente de la neurona i para con la neurona j) entre la neurona i y la neurona j.

Las neuronas en la capa de entrada tendrán los valores iniciales que servirán de peso para que las neuronas de las capas continuas realicen sus cálculos. Por último, en la capa de salida las neuronas servirán para determinar a qué clase pertenecen los valores de resultantes de la capa oculta.

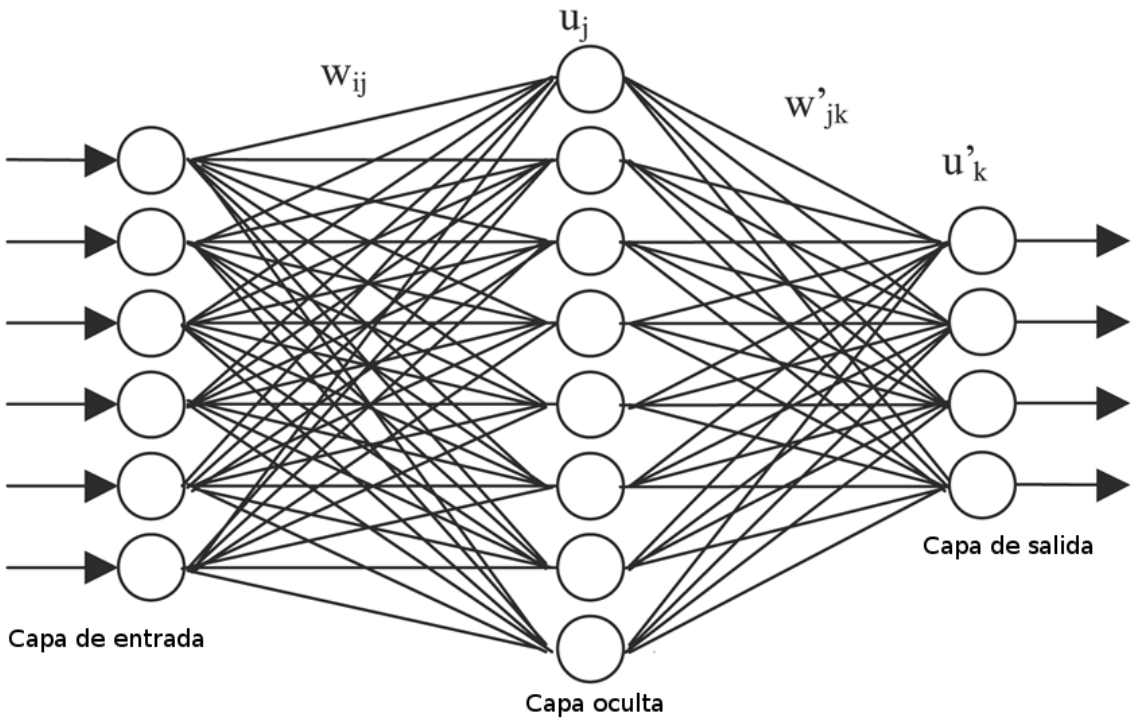


Figura 3.2 Representación gráfica de una red neuronal.

3.3. Perceptrón

El perceptrón fue desarrollado en las décadas del 1950 y 1960. Consiste en una única neurona con múltiples entradas x_i , pesos w_i para cada entrada x_i, que produce una única salida binaria (figura 3.3).

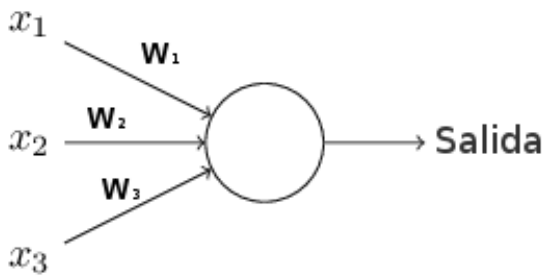


Figura 3.3 El perceptrón con sus valores de entrada y su salida.

La salida de este perceptrón estará regida por la sumatoria del producto entre las entradas y sus pesos, al igual que las neuronas biológicas la salida dependerá de si dicho valor sobrepasa o no un umbral establecido, siendo 0 si dicha sumatoria se encuentra por debajo

del mismo o 1 en el caso contrario. Por lo tanto, se define la función de activación de la siguiente manera:

$$f = \begin{cases} 0 & \text{si } \sum_j w_i x_j \leq \text{ umbral} \\ 1 & \text{si } \sum_j w_i x_j > \text{ umbral} \end{cases}$$

Pero ¿cómo podría una estructura así actuar como clasificador? Supóngase que, afrontando el problema de los peatones, quisiera clasificar, dado un conjunto de datos x_j que tomarán el valor 1 si se cumple y 0 en el caso contrario. Por poner un ejemplo:

- x1 = el objeto es una persona,
- x2 = el objeto está en movimiento y
- x3 = el objeto se encuentra en un ambiente urbano.

Ahora, cada una de estas entradas, quizás, no tengan el mismo "peso" a la hora de clasificar un objeto como peatón o no peatón. Por ende, seteamos los w_j, que a mayor valor, más peso posee la entrada en la clasificación final:

- w1 = 6,
- w2 = 3 y
- w3 = 1

Por último, seteamos el umbral a un valor fijo como podría ser, 5. Entonces, haciendo las cuentas nuestro clasificador tomará como peatón (es decir, salida $= 1$) cuando el objeto sea una persona, si es una persona y está en movimiento ($6+3=9$), o si todas las condiciones se cumplen en conjunto ($6+3+1=10$) ya que la suma de sus valores supera el umbral establecido. El caso de que el elemento esté en movimiento y en un ambiente urbano ($3+1=4$) no será condición suficiente para nuestro clasificador para tildar al objeto como un peatón ya que queda por debajo del umbral.

Ahora, puede que queramos también limitar la activación del perceptrón para que no se dispare tan fácil, para ello, podemos redefinir la función de activación cambiando de lado el umbral y dar lugar a lo que se conoce como bias (su notación es b), siendo este $b = -$umbral:

$$salida = \begin{cases} 0 & \text{si } w \cdot x + b \leq 0 \\ 1 & \text{si } w \cdot x + b > 0 \end{cases}$$

Se puede pensar en el bias como una medida de lo fácil que es para el perceptrón obtener un 1. O para ponerlo en términos más biológicos, el bias es una medida de lo fácil que es activar la neurona. Para un perceptrón con un bias realmente grande, es extremadamente fácil emitir un 1. Pero si el bias es muy negativo, entonces será difícil alcanzar ese valor. En [41] se explica todo el procedimiento con mucho más detalle.

Para todos los modelos que veremos más adelante utilizaremos siempre el bias y dejaremos de lado el umbral. Práctica ampliamente aceptada y utilizada en todo el mundo para diferentes estructuras de redes neuronales y propósitos [51] [25] [45] [58] [49].

3.4. Neuronas Sigmoideas

Como se mostró en la introducción de este capítulo, las estructuras neuronales que se utilizan resultan más complejas que un simple perceptrón. Así mismo, la función de activación de este último resulta limitada en la práctica real.

Las neuronas sigmoideas tienen como salida un valor entre 0 y 1, ya que poseen como función de activación la función sigmoidea, que consiste en:

$$\sigma(z) = \frac{1}{1+e^{-z}}$$

Siendo $z = w \cdot x + b$. Dejando una función de la forma:

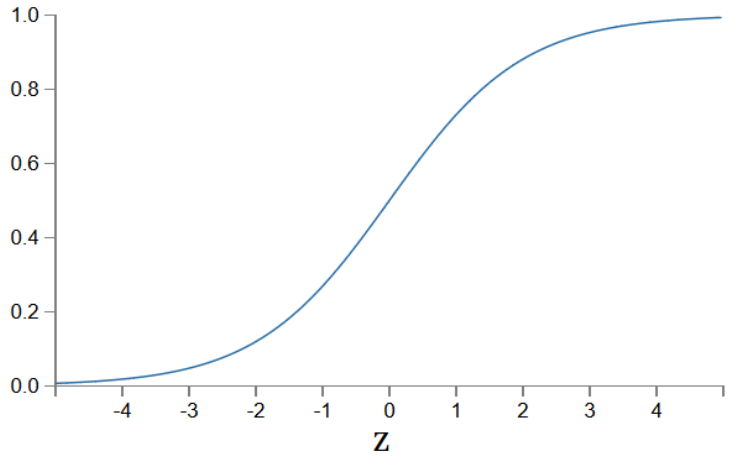


Figura 3.4 Función Sigmoidea.

Entonces, cuando z es un número positivo muy alto, entonces $e^{-z} \approx 0$ entonces $\sigma(z) \approx 1$. Para el caso contrario, cuando z es un número muy negativo $e^{-z} \to \infty$ por lo que $\sigma(z) \approx 0$. Solo cuando $w \cdot x + b$ tienen un tamaño modesto es que la función presenta mucha desviación.

Pero, ¿qué ventaja posee este tipo de neurona artificial frente al anteriormente explicado Perceptrón?. Gracias a la forma de esta función no solo se obtiene una salida que no es simplemente binaria, sino además que su linealidad permite percibir y manejar los pequeños cambios en sus pesos y biases de modo tal que se pueda transferir esas variaciones al output final de la neurona, en breve se detallará cómo estos cambios permiten entrenar un modelo clasificador usando redes neuronales compuestas por este tipo de neuronas artificiales.

3.5. Redes Feedforward

Como se describió en la sección *Redes neuronales artificiales* de este capítulo, la arquitectura de una red neuronal artificial consta de varias capas: la de entrada en el extremo izquierdo, la de salida en el derecho y las denominadas capas ocultas entre las dos capas anteriores. Cuando se habla de Deep Learning (en español, Aprendizaje profundo) se hace referencia a una técnica de Machine Learning que aborda redes neuronales con más de una capa oculta a fin extraer información de alto nivel de los conjuntos de datos que se utilicen. Cuantas más capas ocultas posea la red, mayor es la cantidad de descriptores que esta podrá extraer a fin de realizar la tarea para la que fue construida.

Esta estructura, en la práctica real, resulta mucho más compleja, estando conformada por más de una capa oculta con muchos más nodos. Esta red, donde la salida de cada capa es la entrada de la siguiente es llamada Red Feedforward. En ella, cada resultado de la función de activación (en nuestro caso la función Sigmoidea) conforma la salida que será consumida por cada una de las neuronas de la siguiente capa, así hasta llegar a la salida, que en nuestro caso son solo dos neuronas, una para cada clase que se quiere identificar: “es peatón” o “no es peatón” (figura 3.5). La neurona de salida con más peso será la que nos permita identificar a qué clase pertenece. Sin embargo, más adelante se especifica qué valores poseerá cada una de estas neuronas, ya que hay múltiples alternativas que varían dependiendo la función de activación de las mismas.

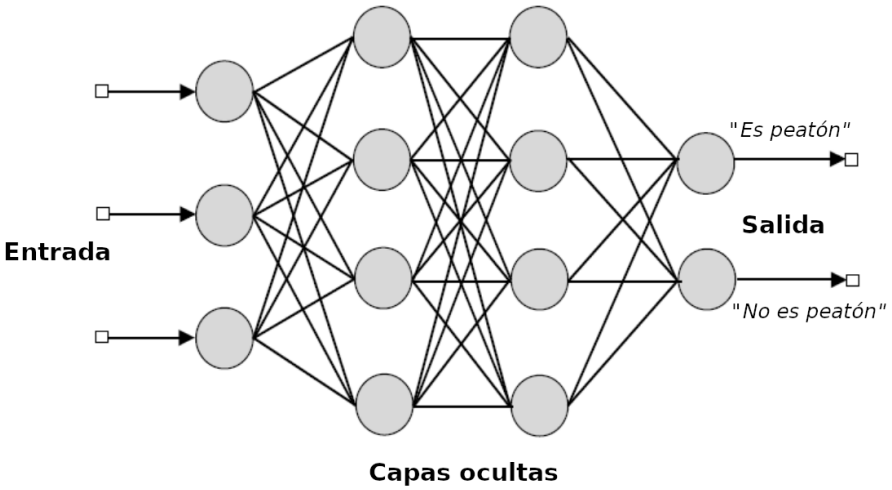


Figura 3.5 Estructura de la red, con su capa de entrada, dos capas ocultas y la capa de salida con una neurona por cada clase posible (en este caso dos: peatón o no peatón).

3.6. Backpropagation

El procedimiento hasta ahora descrito resulta aleatorio, ya que los pesos y biases de cada una de las neuronas de las capas ocultas que componen la red son inicializados de esa forma. Por lo tanto, ¿cómo se podría, cuando la salida del modelo no es la que corresponde, corregir dichos valores?, aquí es donde entra en juego el proceso denominado backpropagation.

3.6.1. Función de error

Al obtener una clase final desde el proceso de feedforward se puede cuantificar que tan bien ejecutó la tarea nuestra red neuronal, calcular el error de dicha predicción nos ayuda en el proceso. Para ello, utilizamos una función de error como ser el error cuadrático, distancia de Hellinger, entropía cruzada, entre muchas otras. para la clasificación de nuestra red utilizamos la función de entropía cruzada (Cross Entropy) definida como:

$$C = -\frac{1}{n}\sum_x [ylna + (1-y)ln(1-a)]$$

Donde n es el número total de datos de entrenamiento, la sumatoria es sobre cada sample del conjunto de entrenamiento x, y es el resultado deseado y $a = \sigma(z)$ donde z, recordemos, es la sumatoria de todos los pesos y biases del input de la neurona.

Esta función de costo es cercana a 0 cuando el resultado es cercano al deseado, para ver esto podría tomarse por ejemplo, si $y = 0$ ("no peatón") y la función de activación también es próxima a $0 (a \approx 0)$ para algún x cualquiera en la neurona de salida correspondiente a la clase *peatón* veremos que el primer término se cancela ya que $y = 0$ y el segundo término queda $-ln(1-a) \approx 0$. Se obtiene un resultado similar cuando $y = 1$ y $a \approx 1$ para la neurona de peatón. Por lo tanto, el costo será bajo siempre que la respuesta del modelo esté cerca de la salida deseada: en el caso del peatón la neurona de *peatón* cercana a 1 y la de *no peatón* cercana a 0, caso inverso en que la entrada corresponda a un no peatón.

3.6.2. Gradiente descendente

Como el objetivo de una red neuronal es poder clasificar los ejemplos cometiendo el menor número de errores, resulta de interés minimizar este error para cada uno de los ejemplos que utiliza para el entrenamiento dicha red neuronal (así como se busca el margen óptimo o vector de soporte como se explica en sección 2.3.1 de máquinas de vectores de soporte).

Para ello, se utiliza el algoritmo de gradiente descendente, cuya finalidad es encontrar valores mínimos de funciones convexas y diferenciales en todo su dominio. En este caso, el mínimo para la función de costo de entropía cruzada $C(w,b)$. Este algoritmo iterativo comienza desde un punto del dominio aleatorio y con cada ejemplo presentado ajusta los pesos en una proporción α del error cometido (que es lo que se conoce como **learning rate**, parámetro que será ajustado durante la implementación más adelante). De esta forma, se avanza en la función de costo hasta encontrar los mejores valores para w y b en los cuales dicha función resulta mínima. La matemática detrás de este algoritmo posee una complejidad que escapa al dominio de la tesis. La figura 3.6 muestra el proceso con carácter ilustrativo.

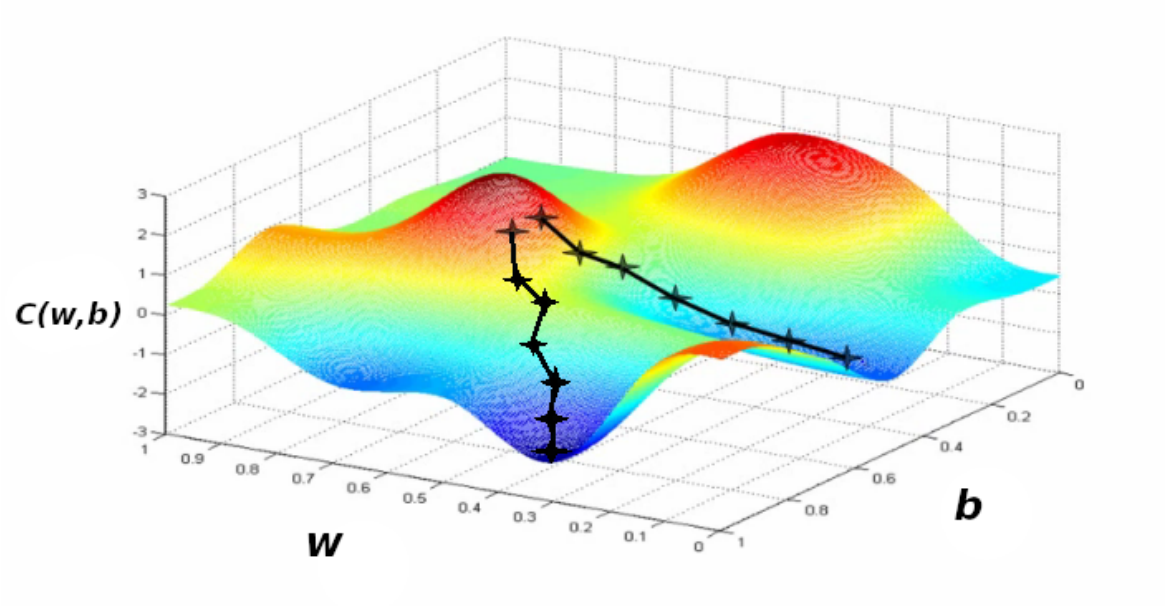
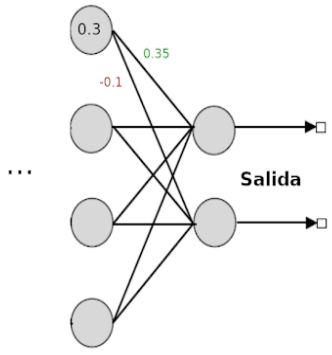


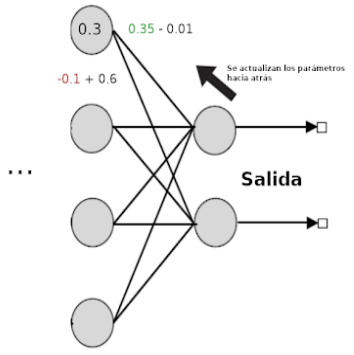
Figura 3.6 Representación sobre un plano 3D del procedimiento de gradiente descendente.

3.6.3. Algoritmo de backpropagation

El procedimiento de entrenamiento de una red neuronal consiste en presentarle un ejemplo, obtener la salida para dicho ejemplo (figura 3.7a), evaluar el error como se explica en la sección 3.6.1 y mediante la técnica de gradiente descendente (vista en la sección 3.6.2) se actualizan los pesos y biases de la red (figura 3.7b). Este procedimiento se repite n veces para cada uno de los ejemplos presentados, donde n es un parámetro que se establece antes de comenzar el entrenamiento, cuanto mayor sea n más tiempo demandará entrenar al modelo.



(a) Red neuronal con los biases y pesos ya cargados.



(b) Proceso de actualización de los pesos de las neuronas.

Figura 3.7 Representación del proceso de *backpropagation*.

3.6.4. Sobreajuste

El sobreajuste, en inglés *overfitting* se da cuando el modelo ajusta tanto sus valores a los ejemplos de entrenamiento que ya no es capaz de generalizar bien los datos que en él se evalúan. Cuando hay sobreajuste el modelo ejecuta una performance sobresaliente frente al mismo dataset de entrenamiento, pero obtiene muy malos resultados contra una base de datos de evaluación. Hay varias maneras de evitar el sobreajuste, entre ellas conseguir más ejemplos de entrenamiento a fin de aportar más variedad a los valores que calcula nuestra red, filtrar algunos atributos, o entre otras, aplicar alguna técnica de regularización. Esta última consiste en suavizar el entrenamiento para que el ajuste se aplique de manera más gradual y no como lo haría normalmente, reduciendo así el error y la probabilidad de un sobreajuste.

3.7. Redes neuronales convolucionales

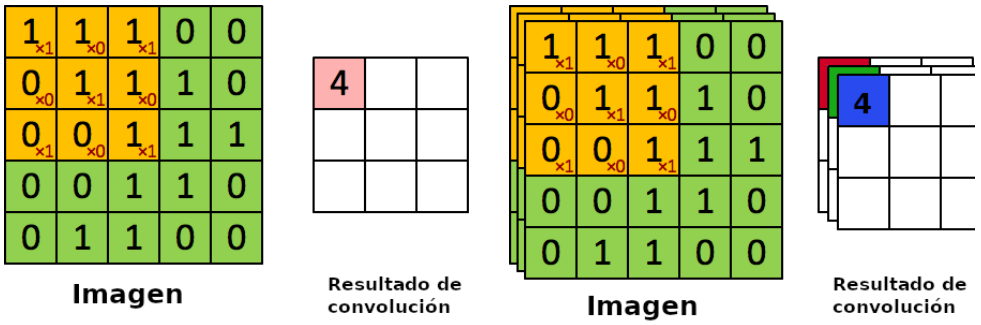
Al igual que con el SVM, las estructuras de las redes neuronales también podrían valerse de los descriptores HOG y LBP para realizar su tarea de clasificación. Pero esto acarrea el problema de tener que limitarse al uso de información únicamente relacionada con los bordes (HOG) y la textura (LBP). Tener a disposición otro tipo de descriptores podría resultar beneficioso para la performance de la clasificación final. Con el fin de sobrepasar esta limitación es que surgieron las denominadas redes neuronales convolucionales. Estas son un tipo especial de redes neuronales que consisten en múltiples capas de los llamados filtros convolucionales, estos se aplican a la matriz que representa la imagen. Un filtro convolucional (también llamado simplemente *filtro* o *kernel*) no es más que una matriz de valores que se aplican a los píxeles de una imagen para socavar información, hay varios filtros ”populares” como el de la figura 3.8 que sirve para extraer datos sobre los bordes de una imagen. Gracias a estos filtros se pueden obtener distintos descriptores de la imagen como profundidad, texturas, color, movimiento, forma, entre muchos otros. En la década del 80 o 90 los investigadores tenían que armar estos filtros manualmente, tarea que resultaba costosa, sobre todo en casos donde el filtro podía alcanzar tamaños grandes (como por ejemplo, de $7 * 7$). Una capa convolucional está conformada por estos filtros, donde cada elemento de los mismos es un peso de la propia capa, dándole la posibilidad de ajustar los valores (y por consiguiente, hallar el filtro convolucional) durante la etapa de entrenamiento.

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

Figura 3.8 Filtro para la obtención de información de los bordes de una imagen.

Este proceso de extracción de información a partir de un filtro se llama **convolución** (figura 3.9a). Cada filtro no es más que una matriz de $N * M$ que recorre la imagen y para cada posición de dicho recorrido genera un solo valor de salida que se da por la suma de los productos de cada píxel del filtro por su píxel correspondiente en la imagen, por lo que el resultado de la convolución es una matriz de menor tamaño llamada **Convolve Feature** ó **Feature Map** ("Mapa de características" en español). La cantidad de píxeles que se desplaza el filtro entre cada porción analizada se conoce como stride en caso de que se procese una matriz y se desplace un píxel en X y otro en Y se habla de un stride de $1 * 1$. En la figura 3.10 se puede observar el resultado final de aplicar un filtro a una imagen en escala de grises.

En el caso de una imagen a color la convolución se aplica a cada una de las 3 matrices que componen la imagen original. En la figura 3.9b se muestra el proceso para dichas imágenes RGB.



(a) Proceso de convolución en una imagen en escala de grises.

(b) Proceso de convolución en una imagen RGB.

Figura 3.9 Proceso de convolución.

El resultado de la extracción de los descriptores generados en el proceso de convolución serán utilizados más adelante como materia prima para el clasificador, a fin de que disponga de mucha más información para distinguir las clases a la que pertenece cada ejemplo de entrenamiento y evaluación.

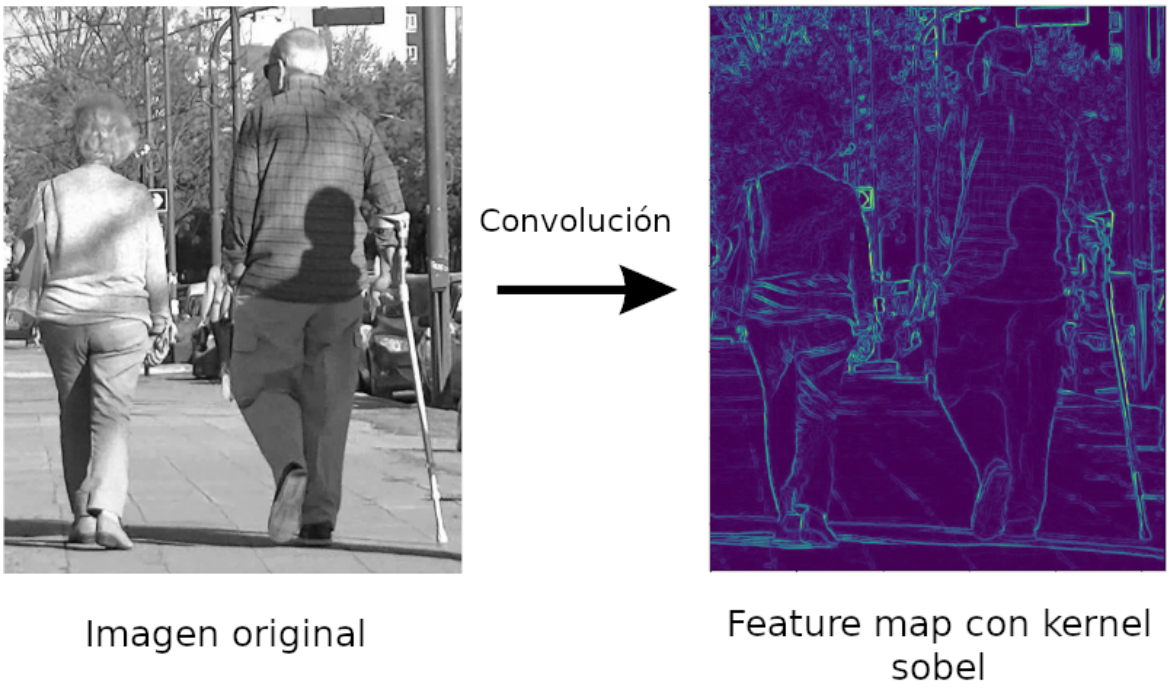


Figura 3.10 Resultado de convolución a partir de un filtro/kernel llamado *Sobel* que aporta información respecto a los bordes de la imagen. Está dado por [{-1, 0, 1}, {-2, 0, 2}, {-1, 0, 1}].

3.7.1. Capa Max Pooling en 2D

Una capa Max Pooling 2D es una capa que habitualmente se utiliza inmediatamente después de una capa convolucional y su objetivo es simplificar la información del Feature Map a fin de optimizar el cómputo de dicha matriz al reducir considerablemente su tamaño. Esta capa toma cada Feature Map generado por la capa convolucional y genera otro Feature Map más condensado. Para ello, utiliza un nuevo filtro que resume la información de la matriz que se se está procesando.

Una de las técnicas más utilizadas actualmente (y de la que se va a hacer uso a lo largo de lo que queda de la tesina) es la técnica llamada Max-pooling que toma el valor máximo entre la sub matriz $N * M$ de la imagen para formar el Feature Map resultante. La figura 3.11 de abajo muestra el procedimiento:

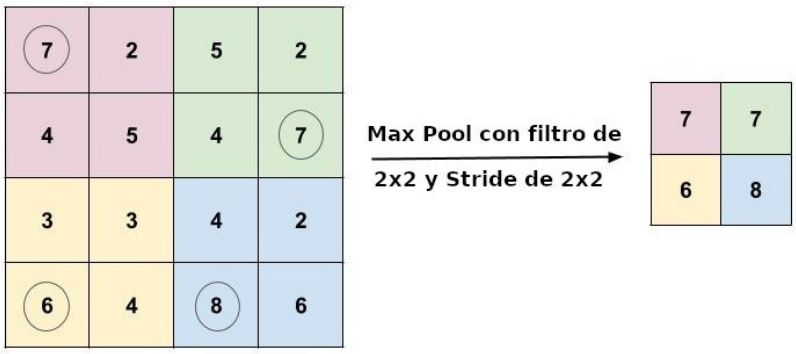


Figura 3.11 Proceso de Max Pooling, el feature map resultante es más reducido.

3.7.2. Capa Dropout

Como mencionamos en el apartado de sobreajuste, existen técnicas para evitar este tipo de inconvenientes. Una de ellas eran las técnicas de regularización. En nuestro caso, esto se traduce a una nueva capa que vamos a integrar al modelo: las capas *Dropout*. Esta capa se encarga de descartar los valores que recibe de algunas neuronas de la capa anterior elegidas aleatoriamente en cada recorrido con el propósito de, como ya se dijo, suavizar los ajustes que produce nuestro modelo. En la práctica, si se elige una capa Dropout con un factor de $0,8$ significa que el $80\,\%$ de los valores que reciba serán despreciados y no considerados en la siguiente capa de la red. Las capas Dropout son muy utilizadas en la práctica ya que alcanza mejoras en cuanto a la eficiencia y eficacia de los modelos vigentes [50] [10].

3.8. Funciones de activación

Como se comentó anteriormente, las neuronas posee una función de activación para producir su salida. Si bien la función Sigmoidea sigue siendo ampliamente utilizada a la fecha, en la implementación que se mostrará más adelante se utiliza una función llamada **ReLU** (Rectified Linear Unit) o Unidad linear rectificada que se define como:

$$f(x) = max(0, x)$$

La forma de la función esta dada por la figura 3.12. Siendo x la entrada de la neurona. Esta función resulta mucho más performante que la Sigmoidea y obtuvo excelentes resultados en la práctica, no limitándose exclusivamente a la clasificación o detección de peatones: [23] [39] [10].

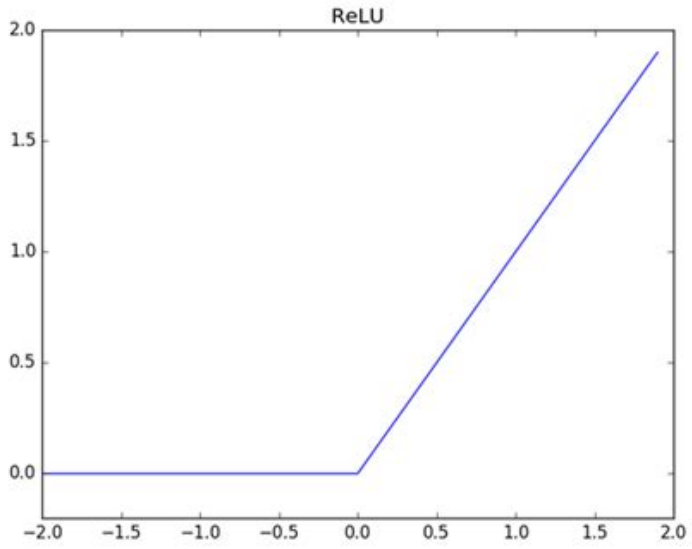


Figura 3.12 Función ReLU.

Otra función que usaremos para la última capa de la red es la función **Softmax** la cual se define como:

$$f(x) = \frac{1}{1+exp(-\theta^t x)}$$

Donde θ representa el vector de pesos y x es el vector de valores de entrada. Se utiliza para aproximar los resultados de la capa a $y \in 0,1$. Es decir, produce un escalar $f(x) \in \mathbb{R}, 0 < f(x) < 1$. Gracias a esto, los valores de salida de cada una de las neuronas de la última capa de la red será una valor entre 0 y 1, siendo la suma de toda las salidas igual a 1. Esta característica, en la práctica, permite interpretar la salida de la red desde un aspecto probabilístico, de esta manera, se deja de lado la salida binaria (*peatón* o *no peatón*) para dar lugar a una interpretación que determina la probabilidad de cada una de las posibles respuestas.

Capítulo 4

Transferencia de aprendizaje

Al momento de entrenar un modelo clasificador, ya sea un SVM, una red neuronal u otro, es indispensable contar con un conjunto de entrenamiento que permita ajustar los parámetros propios de cada modelo a fin de que su performance mejore. Dichos datos de entrenamiento (y evaluación) pueden provenir de diferentes fuentes, como COCO [36] base de datos creada por Microsoft que consta de más de 2,5 millones de objetos reconocibles correspondientes a 91 categorías distintas de objetos (auto, perro, persona, mesa, etc); ImageNet [46] [16] [15] con más de 14 millones de imágenes de mas de 20 mil categorías; entre muchos otros [33] [2] [40] [54]. Por este motivo, resulta vital conocer en detalle con qué datos se está entrenando el clasificador, por qué conviene más una base de datos que otra, qué ocurre cuando se combinan más de un conjunto de datos, etc. Se denomina transferencia de aprendizaje al proceso de comparar que tan efectivo es el aprendizaje adquirido por el clasificador a partir de una base de datos, cuando se enfrente con otro conjunto de datos que no proviene de la misma fuente que el primero. Dicho proceso responde a algunas de las preguntas anteriormente planteadas: ¿Qué ocurre la performance del modelo clasificatorio al combinar conjuntos de datos, y testearlo contra otro totalmente distinto? ¿Qué consecuencias y ventajas se obtienen, y cuál es más propicia para la tarea de la clasificación de peatones?

La transferencia de aprendizaje no es nueva y ha habido múltiples artículos que abordan el tema utilizando diferentes bases de datos o para distintos fines. Por ejemplo, en [7] se mide la transferencia de aprendizaje para la detección de peatones utilizando dos bases de datos distintas a las del artículo que se explicará en esta tesina, y se proponen distintas técnicas para mejorar la transferencia final: la primera es tomar datos de entrenamientos que puedan ser parecidos a los datos de evaluación y moverlos a este último conjunto, la segunda consiste en un modelo nuevo basado en el aprendizaje por transferencia, técnica en la que se hace uso solo del 90 % del conjunto de entrenamiento. En comparación con los métodos tradicionales de detección de peatones, ese algoritmo propuesto puede adaptarse a diferentes escenas y

obtener un mejor rendimiento. Por poner otro ejemplo, en [32] se mide la correlación entre la performance de un modelo llamado y su transferencia de aprendizaje para la clasificación de objetos en general, demostrando que un buen clasificador no siempre ofrece una buena transferencia de lo aprendido.

Durante el desarrollo de la tesina y con los resultados conseguidos se publicó un artículo académico [6] que midió la trasferencia de aprendizaje a partir de tres bases de datos conocidas en el estado del arte. A continuación se da una introducción a dichas base de datos, el preprocesamiento de las imágenes y la métrica de evaluación utilizada.

4.1. Datasets

Los datasets o conjuntos de datos (imágenes) de los que haremos uso son tres, y todos se encuentran accesibles a través de internet.

4.1.1. INRIA

Creado para un trabajo de investigación [12] y una tesis doctoral [11], disponible para descargar en *http://pascal.inrialpes.fr/data/human/*. Consiste en imágenes en RGB de **personas** (nótese que no son necesariamente peatones) divididos en dos formatos: las imágenes originales con la información de los BB de las personas, y las imágenes positivas escalas a $64 * 128$ píxeles de ancho y alto en conjunto con los ejemplos negativos sin re-escalar (ejemplos positivos y negativos en la figura 4.1). Los elementos se encuentran divididos en carpetas separadas de entrenamiento y evaluación. Ambas carpetas poseen sus subcarpetas “pos” y “neg”, haciendo referencia a los ejemplos positivos y negativos respectivamente.

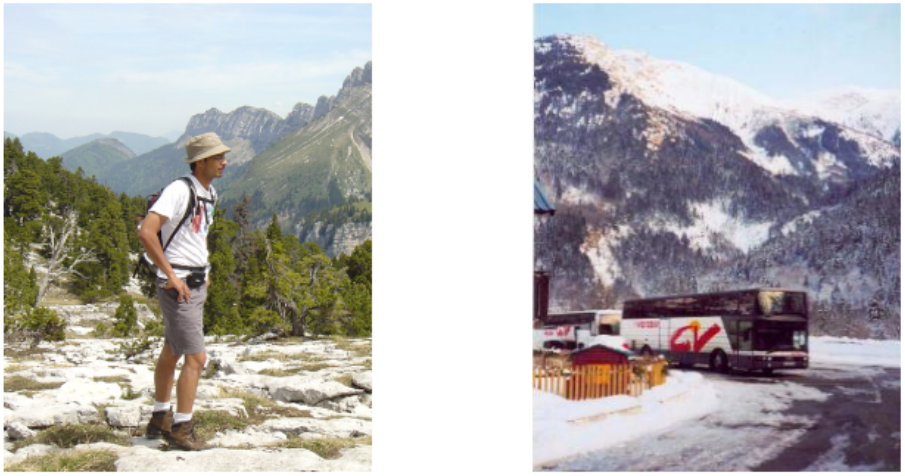


Figura 4.1 De la base de datos INRIA: ejemplo positivo a la izquierda, ejemplo negativo a la derecha.

4.1.2. Daimler

Compone una base de datos mucho más amplia que INRIA, conformada también por datos de entrenamiento y evaluación. Se puede obtener desde su página oficial (*http://www.gavrila.net/Datasets/Daimler_Pedestrian_Benchmark_D/Pedestrian_Path_Predict_GCPR_1/pedestrian_path_predict_gcpr_1.html*) y ofrece 15560 ejemplos positivos de entrenamiento en la escala de $48 * 96$ píxeles y 6744 negativas de las que se generará más ejemplos para entrenar (el procedimiento se explica más adelante). En cuanto a los casos de evaluación, aporta 21790 imágenes con 56492 BB definidos de peatones completos o parcialmente visibles que es el conjunto de cuadros de un video de 27 minutos filmados desde un vehículo en movimiento. Dicho conjunto fue generado originalmente para un estudio sobre predicción de recorridos en un periodo corto de tiempo (< 2 segundos) utilizando filtros Bayesianos [48]. Algunos ejemplos pueden apreciarse en la figura 4.2.

Figura 4.2 De la base de datos Daimler: peatones en algunos fotogramas que conforman el conjunto de datos.

4.1.3. TUD-Brussels

Este conjunto es mucho más reducido, contando con 1092 ejemplos de entrenamiento junto con la definición de los BB de 1776 peatones. En cuanto al dataset de evaluación, 508 imágenes con 1326 peatones definidos (en la figura 4.3 se muestran algunas imágenes pertenecientes a esta base de datos). Al igual que Daimler, este dataset también fue generado para un artículo que cubría técnicas de seguimiento y predicción de caminos con el fin de mejorar la detección de peatones [53]. Disponible para su descarga en *https://www.mpi-inf.mpg.de/departments/computer-vision-and-multimodal-computing/research/people-detection-pose-estimation-and-tracking/multi-cue-onboard-pedestrian-detection/*.

Figura 4.3 Conjunto de datos de TUD-Brussels con los BB de peatones definidos.

4.1.4. Daimler Mono Pedestrian Detection Benchmark Dataset

Generado para [17], consta de **1560** ejemplos **positivos** y **2000** ejemplos **negativos**. Este último dataset es ajeno al conjunto Daimler presentado anteriormente y no debe confundirse a pesar de la similitud del nombre. Disponible para descargar en *http://www.gavrila.net/Datasets/Daimler_Pedestrian_Benchmark_D/Daimler_Mono_Ped__Detection_Be/daimler_mono_ped__detection_be.html*. En la figura 4.4 se pueden observar algunos ejemplos positivos y negativos que conforman dicha base de datos.

Figura 4.4 De la base de datos Daimler Mono Pedestrian Detection Benchmark Dataset: ejemplos positivos arriba, negativos abajo.

4.1.5. Generación de ejemplos extra

Si sabemos que los conjuntos negativos que nos ofrecen las bases de datos no contienen peatones en lo absoluto, existe una técnica para generar muchos más samples de entrenamiento. La misma consiste en recorrer con una ventana deslizante cada uno de los ejemplos negativos, guardando esa subimagen como sample nuevo. Utilizando este método es que se ampliaron los ejemplos negativos de entrenamiento y evaluación, llegando a más de 30 mil samples en el caso de INRIA y Daimler. En cuanto a los ejemplos positivos, existen técnicas sencillas para la generación de nuevos ejemplos: podría tomarse la imagen de cualquier peatón y generar ejemplos extras de la misma con leves modificaciones, como una réplica espejada o inclinada en diferentes ángulos, ya que un peatón al derecho y al revés, seguirá siendo un peatón, por lo que el ejemplo será válido para la etapa de entrenamiento. La generación de ejemplos extra nos permite contar con un conjunto de datos más variado, hecho que hace que nuestro clasificador posea más información para discriminar correctamente ambas clases. Cabe aclarar que para esta tesina solo se utilizó la generación de ejemplos negativos, y en cuanto a los positivos, solo se utilizaron los aportados por las bases de datos, ya que se contaba con suficientes ejemplos de estos últimos para el proceso de entrenamiento.

Capítulo 5

Experimentación

En este capítulo se muestra la experimentación realizada en el marco de esta tesis para la clasificación de peatones. En la sección 5.1 se realiza una comparativa entre dos algoritmos de aprendizaje automático para las tareas de clasificación: un SVM alimentado por los descriptores HOG y LBP, y una estructura de red neuronal de varias capas. La sección 5.2 consiste en la medición de la transferencia de aprendizaje de un SVM para las tres bases de datos de entrenamiento utilizadas.

Todos los experimentos se realizan sobre una computadora de 8 *gigabytes* de memoria RAM con un CPU Intel i7 6500U de 2 núcleos que procesan a $\approx$ 2.50GHz cada uno.

5.1. Comparación de algoritmos de machine learning

En esta sección se detallan los experimentos realizados con varios clasificadores: un SVM de núcleo lineal y dos redes neuronales con diferentes estructuras a fin de comparar la performance de los tres con respecto a la clasificación de peatones.

Para entrenar todos los modelos se utilizan diferentes ejemplos correspondientes a las tres bases de datos descritas en el capítulo 4 de transferencia de aprendizaje: INRIA, Daimler y TUD-Brussels. El dataset de evaluación es siempre el mismo y corresponde a Daimler Mono Pedestrian Detection Benchmark Dataset.

5.1.1. Preprocesamiento y parámetros del SVM

El SVM lineal entrenado se vale de imágenes de peatones y no peatones provistas por los tres datasets escaladas a 48 píxeles de ancho por 96 de alto (debido a que los ejemplos de todos los datasets respetan la relación $1:2$ es que se puede aplicar un reescalamiento

sin perder su aspecto original). Todas son convertidas a escala de grises y sus píxeles son normalizados en el intervalo $[0-1]$.

Para la implementación de los scripts utilizados en los experimentos se utiliza la librería scikit-image [4] para la manipulación de imágenes y scikit-learn [28] para instanciar el clasificador, cuyo único parámetro modificado es el C, que define ”cuánto error” está dispuesto a afrontar nuestro modelo. Si el parámetro C es muy grande, el SVM generará un margen pequeño que se ajuste a la mayor cantidad de ejemplos de entrenamiento posible. Por el contrario, si el valor es chico, el margen será mayor, dando lugar a que varios ejemplos sean mal clasificados, pero realizará una generalización más amplia de los casos de entrenamiento. Para este trabajo se configura un $C = 0.1$, igual peso para las dos clases y regularización *L2*. El entrenamiento del modelo se realiza mediante el método SMO de la librería Liblinear [35] con la formulación dual, verificando que el modelo converge y no sobre ajusta. El descriptor HOG es extraído en bloques de $2*2$ celdas de $8*8$ píxeles cada uno. Por último, el LBP se extrae con radio $R = 2$ y $2*R$ vecinos al píxel central. Todos estos parámetros son seleccionados con estos valores debido a que son los mismos que se utilizan en [55] y [22].

Cabe aclarar que no se utilizan todos los ejemplos disponibles de los tres conjuntos de datos de entrenamiento por limitaciones de hardware, haciendo imposible utilizar los descriptores de alta dimensionalidad de más de 50 mil imágenes. Se opta, entonces, por realizar las pruebas con subconjuntos de las bases de datos.

5.1.2. Resultados con SVM

En esta sección se presentan los resultados de utilizar los diferentes descriptores como datos de entrenamiento de un SVM. Se realizaron tres ensayos, entrenando solo con los descriptores HOG (tabla 5.1), solo con los descriptores LBP (tabla 5.2) y usando una combinación de ambos (tabla 5.3). Los resultados presentan la precisión, exhaustividad y medida-f obtenidas al utilizar los mismos datos con los que realiza el entrenamiento y los que se usan en la etapa de evaluación (Daimler Mono Detection). Este último dato no carece de importancia ya que sirve de referencia para saber si el modelo está sobre ajustando, ya que si los resultados para el mismo conjunto de entrenamiento resultan óptimos, pero al evaluar los datos de Daimler Mono Detection presenta una performance muy pobre estaría indicando que el clasificador no está generalizando lo suficiente como para clasificar eficazmente nuevas imágenes. En este caso, se debe replantear la estructura del modelo de manera tal que los resultados obtenidos no presenten overfitting.

Tabla 5.1 Resultados del SVM utilizando **HOG**. Columna ”Datasets”: combinación de las bases de datos INRIA (I), Daimler (D) y TUD-Brussels (B) utilizadas para el entrenamiento. Columna ”Entrenamiento”: métricas obtenidas evaluando el mismo conjunto de datos con los que se entrena. Columna ”Evaluación”: métricas obtenidas al evaluar Daimler Mono Detection. En color verde los mejores resultados obtenidos.

Datasets	**Entrenamiento**	**Evaluación**
I + D	Prec.: 0.958 Exh.: 0.929 Med-F: 0.943	Prec.: 0.989 Exh.: 0.969 Med-F: 0.978
I + B	Prec.: 0.920 Exh.: 0.808 Med-F: 0.860	Prec.: 0.980 Exh.: 0.533 Med-F: 0.690
D + B	Prec.: 0.987 Exh.: 0.983 Med-F: 0.984	Prec.: 0.995 Exh.: 0.978 Med-F: 0.986

Tabla 5.2 Resultados del SVM utilizando **LBP**. Columna ”Datasets”: combinación de las bases de datos INRIA (I), Daimler (D) y TUD-Brussels (B) utilizadas para el entrenamiento. Columna ”Entrenamiento”: métricas obtenidas evaluando el mismo conjunto de datos con los que se entrena. Columna ”Evaluación”: métricas obtenidas al evaluar Daimler Mono Detection. En color verde los mejores resultados obtenidos.

Datasets	**Entrenamiento**	**Evaluación**
I + D	Prec.: 0.837 Exh.: 0.553 Med-F: 0.651	Prec.: 0.888 Exh.: 0.483 Med-F: 0.625
I + B	Prec.: 0.605 Exh.: 0.703 Med-F: 0.650	Prec.: 0.550 Exh.: 0.246 Med-F: 0.339
D + B	Prec.: 0.746 Exh.: 0.845 Med-F: 0.792	Prec.: 0.855 Exh.: 0.766 Med-F: 0.808

Tabla 5.3 Resultados del SVM utilizando **HOG** y **LBP**. Columna ”Datasets”: combinación de las bases de datos INRIA (I), Daimler (D) y TUD-Brussels (B) utilizadas para el entrenamiento. Columna ”Entrenamiento”: métricas obtenidas evaluando el mismo conjunto de datos con los que se entrena. Columna ”Evaluación”: métricas obtenidas al evaluar Daimler Mono Detection. En color verde los mejores resultados obtenidos.

Datasets	**Entrenamiento**	**Evaluación**
I + D	Prec.: 0.987 Exh.: 0.989 Med-F: 0.987	Prec.: 0.964 Exh.: 0.921 Med-F: 0.942
I + B	Prec.: 0.988 Exh.: 0.981 Med-F: 0.984	Prec.: 0.855 Exh.: 0.459 Med-F: 0.597
D + B	Prec.: 1.000 Exh.: 0.999 Med-F: 0.999	Prec.: 0.973 Exh.: 0.920 Med-F: 0.945

Conclusión

Puede observarse que, como se puede apreciar en las tablas, al usar los mismos datos de entrenamiento los resultados son, en su mayoría, superiores. Incluso perfectos como es el caso de Daimler y Brussels al usar ambos descriptores.

En cuanto a qué descriptor logra una mejor generalización, LBP no resulta ser muy eficaz durante la clasificación, logrando en algunos casos una exhaustividad tan baja como 0.246. Sin embargo, al combinar ambos, se logra una mejora muy significativa, alcanzando valores 0.921 para las mismas bases de datos que fracasaron en la tarea con el patrón binario local.

Desde la perspectiva de datasets de entrenamiento, al igual que en la sección 5.1.2 donde se explayan los resultados de la comparación de los diferentes algoritmos, Daimler resultó ser la más propicia para la tarea ya que cada vez que esta es incluida, los resultados son mejores que cuando se ausenta. Siendo INRIA el caso contrario, logrando en conjunto con Brussels el peor resultado para todas las combinaciones de descriptores evaluadas. Incluso los resultados obtenidos con dicho dataset, al combinarse con Brussels (tabla 5.3) presenta una gran diferencia de exhaustividad (98 % para los mismos datos de entrenamiento y menos del 46 % para los datos de evaluación) dejando en evidencia un caso de sobreajuste, hecho que amerita analizar el modelo para evitar dicha situación.

5.1.3. Resultados con Deep Learning

Para los siguientes experimentos se utiliza la misma computadora que con el SVM. La implementación de las redes neuronales evaluadas en las siguientes secciones se realiza con una librería de lenguaje Python llamada Tensorflow [24] desarrollada por Google para la estructuración, entrenamiento y evaluación de modelos de Machine Learning que no se limita exclusivamente a redes neuronales. El entrenamiento de los dos modelos que se presentan se realiza a partir de 14100 ejemplos positivos y 30000 ejemplos negativos correspondientes únicamente a la base de datos Daimler. Se recuerda que los datos de evaluación corresponden a la base de datos Daimler Mono Detection al igual que los experimentos realizados con el SVM en la sección 5.1.2.

A diferencia del entrenamiento del SVM, con las redes neuronales sí fue posible utilizar el dataset completo que ofrece la base de datos Daimler debido a que las implementaciones permiten un entrenamiento en pequeño batches, evitando así el colapso de la memoria RAM al utilizar gran cantidad de información.

Estructuras y experimentación

Una vez generados todos los descriptores a partir de los filtros que se aplican a la imagen en el proceso de convolución, estos sirven de entrada a una red neuronal profunda como las que se describió anteriormente en la sección 3.2 del capítulo 3. Se hicieron dos experimentos con diferentes arquitecturas a los cuales serán referenciados como *Modelo A* y *Modelo B*.

Modelo A: consta en una estructura conformada por una capa convolucional con 32 filtros de tamaño $2 * 2$ y un stride $= 1$, con ReLU como función de activación; una capa MaxPool de $2 * 2$, una capa oculta de 512 neuronas, nuevamente, con activación ReLU; una capa de Dropout con un factor de $0,2$ y una última capa Softmax cuya salida consiste en dos valores que son la probabilidad de que la imagen se trate de un peatón o no: ambos valores entre 0 y 1 cuya suma es igual a 1. El modelo tiene configurado un learning rate $= 0.0005$ y la función de pérdida es la entropía cruzada, explicada en el capítulo de 3 de Deep Learning, sección 3.6.1. En la figura 5.1 se aprecia de manera sencilla la estructura descrita.

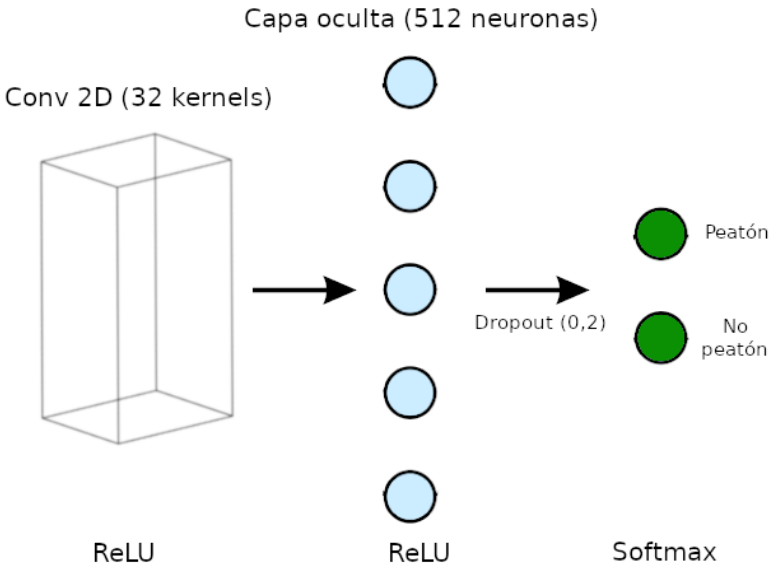


Figura 5.1 Representación gráfica de las capas que conforman el modelo A.

Modelo B: se ajustan los parámetros y cantidad de capas de forma empírica, resultando en una estructura formada por: una capa convolucional de 32 filtros de $3 * 3$ con stride $= 1$; una capa MaxPool de $2 * 2$; otra capa convolucional de 64 filtros de $1 * 1$, también con stride $= 1$; una capa MaxPool de $2 * 2$; seguida de una red profunda igual al *modelo 1* con una capa oculta de 512 neuronas con activación ReLU; una capa de Dropout con un factor de $0,2$ y una última capa Softmax. Ambas capas convolucionales tienen función de activación ReLU y el learning rate es exactamente igual. En la figura 5.2 se resume la composición del modelo.

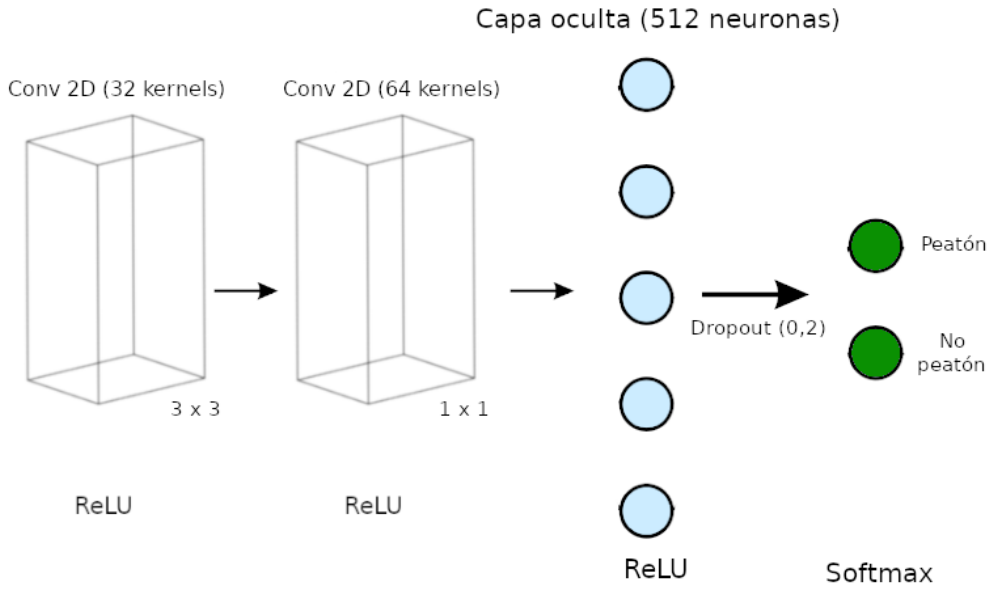


Figura 5.2 Representación gráfica de las capas que conforman el modelo B.

Los resultados que arroja el *Modelo A* entrenando solo con un único dataset mejoran respecto a la exhaustividad e iguala en cuanto a la precisión frente al mejor resultado obtenido con el SVM (0.978 y 0.995 respectivamente). El dataset de evaluación es exactamente el mismo con el que se evaluó en los experimentos anteriores: Daimler Mono Detection.

Las métricas obtenidas por el *Modelo B* resultan mejores a los ya de por sí, excelentes resultados anteriores, consiguiendo una precisión, exhaustividad y medida-f del 99 %. Se presentan los resultados obtenidos por ambos modelos en la tabla 5.4 para un visión más general y detallada.

Tabla 5.4 Resultados obtenidos por los modelos A y B para los datos de evaluación (Daimler Mono Detection) y los datos de entrenamiento (Daimler).

Modelo	Evaluación	Entrenamiento
Modelo A	Prec.: 0.988 Exh.: 0.995 Med-F: 0.991	Prec.: 0.989 Exh.: 0.999 Med-F: 0.993
Modelo B	Prec.: 0.998 Exh.: 0.998 Med-F: 0.998	Prec.: 0.999 Exh.: 0.998 Med-F: 0.998

Conclusión

La performance final de un modelo de aprendizaje profundo resulta superior al enfrentarse al mismo dataset de evaluación que para un SVM. No es menor destacar, que el modelo es entrenado con un único dataset y con una estructura relativamente sencilla. No hay preprocesamiento de las imágenes aparte de un escalado para normalizar el tamaño. En dominios más complejos estas redes poseen muchas más capas, de diferentes tamaños, funciones de activación y parámetros, lo que les permite ajustarse cada día mejor a la enorme variación que presenta un entorno real.

En cuanto a eficiencia se refiere, las estructuras de Deep Learning resultan mucho más lentas tanto para ejecutar la clasificación como para entrenarlas, lo que se presta a diferentes análisis para establecer qué tanto amerita una estructura más lenta y compleja cuando la diferencia de performance final es prácticamente nula. Dicho esto, sin embargo, la gran variedad de librerías que se pueden encontrar hoy en día ofrecen no solo performance a nivel de instrucción, sino también, soporte para GPU que reduce notoriamente el tiempo de ejecución necesario para la realización de las tareas en redes neuronales de gran tamaño. En la tabla 5.5 se expresan los tiempos de ejecución de los algoritmos de entrenamiento y evaluación (en segundos) del SVM y de los dos modelos de redes neuronales (*Modelo A* y *Modelo B*). En dicha tabla figura, además, la desviación estándar para el SVM, ya que los tiempos varían mucho cuando el entrenamiento y evaluación se realizan con un único descriptor o con la unificación de ambos. En el caso del SVM el valor se calcula a partir del promedio de los tiempos que toma entrenar con cada una de las combinaciones posibles: INRIA y Daimler, INRIA y Brussels, y Daimler y Brussels; utilizando los descriptores HOG, LBP y HOG + LBP. Los tiempos de las redes neuronales son producto del único entrenamiento que se realiza a partir de la base de datos Daimler. Queda en evidencia que los algoritmos de aprendizaje profundo resultan muy lentos en comparación con el SVM tanto

para el entrenamiento (≈ 57 % más lento para el peor caso) como para la evaluación (≈ 98 % más lento para el peor caso). Sin embargo, no está de más advertir la desviación estándar que presentan los tiempos de entrenamiento para el SVM, siendo que esto se debe a la gran diferencia en los tiempos que conlleva entrenar con un único descriptor frente a hacerlo con la combinación de ambos, siendo esta última muy costosa.

Estas disimilitudes en tiempo de entrenamiento y ejecución no quedan ajenas al análisis para decidir cual modelo es más conveniente para las tareas de clasificación y detección, dependerá del balance entre las métricas obtenidas, costos de hardware y la eficiencia en tiempos de ejecución que se crea más propicio, para decidir.

Tabla 5.5 Tiempo en segundos que conlleva cada uno de los modelos evaluados al realizar el entrenamiento y la evaluación. En verde los mejores tiempos, en rojo los peores.

Algoritmo	**Tiempo de entrenamiento**	**Tiempo de evaluación**
SVM	≈ 158 seg (±121)	≈ 0,19 seg (±0.07)
Modelo A	≈ 370 seg	≈ 3 seg
Modelo B	≈ 270 seg	≈ 4 seg

Por último, en la figura 5.3 se presenta una comparación general de los mejores resultados obtenidos de cada uno de los modelos y descriptores evaluados a fin de ofrecer una visión global de la performance que ofrece cada uno de los algoritmos vistos.

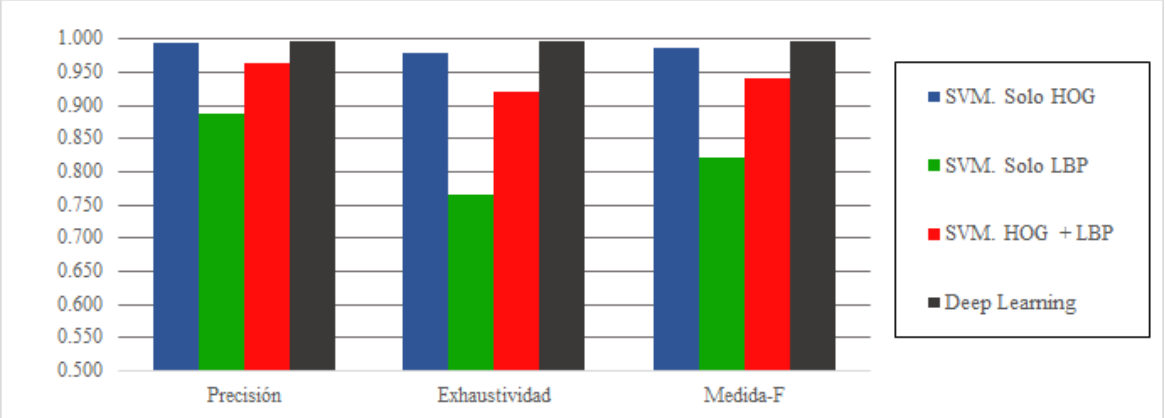


Figura 5.3 Comparación de la precisión, exhaustividad y medida-F obtenidas por el SVM para cada una de las 3 combinaciones posibles de descriptores, y por las redes neuronales, obtenidos en la etapa de evaluación.

5.2. Experimentos de transferencia

En primer lugar, se entrena un SVM de núcleo lineal distinto para cada una de las tres bases de datos, utilizando su conjunto de entrenamiento respectivo. Luego, por cada modelo, se realiza la evaluación del mismo con los tres conjuntos de evaluación de las bases de datos. Como métricas de desempeño se utilizan en un principio la precisión y exhaustividad, para luego medir la medida-F con el fin de combinar las métricas anteriores en una sola.

Como se explica en la sección 5.1.3, por cuestiones de eficiencia es que se entrena únicamente un SVM y no se hace uso de algoritmos de Deep Learning para realizar la medición de la transferencia. Entrenar grandes volúmenes de datos para repetidos experimentos abarcaría mucho tiempo, hecho que no ocurre con un SVM lineal.

De este modo, se obtiene una matriz de transferencia D, de tamaño $3 * 3$ (presentada en las tablas de la sección 5.2.1), entre las tres bases de datos, donde $D_{i,j}$ nos indica la medida-F del modelo al ser entrenado con la base de datos i y evaluado con la base de datos j. De este modo, se puede observar la similitud que hay entre las características de las imágenes de las diferentes bases de datos para cada descriptor.

Las imágenes a color de INRIA y TUD-Brussels son convertidas a escala de grises de modo que el cálculo del HOG y los LBPs sea el mismo para los distintos conjuntos de datos. Además, se realiza una normalización del rango de las imágenes al intervalo $[0 \dots 1]$.

No se aplica ninguna ecualización del histograma ni ningún otro tipo de preprocesamiento adicional a las imágenes para preservar las distribuciones originales de los datos.

Se calculan los HOGs sobre la imagen resultante con un tamaño de celda de $8 * 8$ píxeles, con bloques de $2 * 2$ celdas. Se aplica normalización *L2* a los descriptores obtenidos. Los LBPs se calculan con 8 puntos de muestreo en un radio de 1 píxel.

Como modelo de clasificación se utiliza un SVM ya que es el modelo más usado en la literatura, al mismo tiempo de ser simple y muy eficiente computacionalmente. Los parámetros son los mismos que los utilizados durante la comparación de los diferentes algoritmos de aprendizaje automático detallada en la sección 5.1 .

5.2.1. Resultados de transferencia

Se desarrollan experimentos midiendo la precisión, exhaustividad y medida-f que se muestran en las matrices de transferencia en las tablas 5.6, 5.7 y 5.8 utilizando las bases de datos mencionadas para los descriptores HOG, LBP y la combinación de ambos, respectivamente. Claramente, la transferencia de aprendizaje no es trivial. Esto queda reflejado al ver que la diagonal principal de cada matriz, que contiene los resultados de entrenar y evaluar un modelo con la misma base de datos tiene una medida-F muy superior al resto de las entradas.

Tabla 5.6 Matrices de transferencia entre las tres bases de datos utilizadas. Las entradas muestran la *precisión*, *exhaustividad* y *medida-f* usando **HOG**. Filas: Entrenamiento. Columnas: Evaluación. Abreviaciones: I = INRIA, D = Daimler, B = TUD-Brussels. En verde los resultados más altos.

	I	**D**	**B**
I	Prec.: 0.997 Exh.: 0.449 Med-F: 0.619	Prec.: 0.955 Exh.: 0.176 Med-F: 0.297	Prec.: 0.333 Exh.: 0.036 Med-F: 0.064
D	Prec.: 0.888 Exh.: 0.800 Med-F: 0.841	Prec.: 0.990 Exh.: 0.980 Med-F: 0.984	Prec.: 0.538 Exh.: 0.754 Med-F: 0.627
B	Prec.: 0.945 Exh.: 0.204 Med-F: 0.335	Prec.: 0.989 Exh.: 0.248 Med-F: 0.396	Prec.: 1.000 Exh.: 0.672 Med-F: 0.803

Tabla 5.7 Matrices de transferencia entre las tres bases de datos utilizadas. Las entradas muestran la *precisión*, *exhaustividad* y *medida-f* usando **LBP**. Filas: Entrenamiento. Columnas: Evaluación. Abreviaciones: I = INRIA, D = Daimler, B = TUD-Brussels. En verde los resultados más altos.

	I	**D**	**B**
I	Prec.: 0.803 Exh.: 0.099 Med-F: 0.176	Prec.: 0.563 Exh.: 0.042 Med-F: 0.078	Prec.: 0.121 Exh.: 0.081 Med-F: 0.097
D	Prec.: 0.934 Exh.: 0.552 Med-F: 0.693	Prec.: 0.929 Exh.: 0.850 Med-F: 0.887	Prec.: 0.229 Exh.: 0.354 Med-F: 0.278
B	Prec.: 0.834 Exh.: 0.093 Med-F: 0.167	Prec.: 0.705 Exh.: 0.085 Med-F: 0.151	Prec.: 0.277 Exh.: 0.181 Med-F: 0.218

Tabla 5.8 Matrices de transferencia entre las tres bases de datos utilizadas. Las entradas muestran la *precisión*, *exhaustividad* y *medida-f* usando **HOG** y **LBP**. Filas: Entrenamiento. Columnas: Evaluación. Abreviaciones: I = INRIA, D = Daimler, B = TUD-Brussels. En verde los resultados más altos.

	I	**D**	**B**
I	Prec.: 0.958 Exh.: 0.449 Med-F: 0.611	Prec.: 0.815 Exh.: 0.244 Med-F: 0.375	Prec.: 0.222 Exh.: 0.200 Med-F: 0.210
D	Prec.: 0.885 Exh.: 0.795 Med-F: 0.837	Prec.: 0.981 Exh.: 0.978 Med-F: 0.979	Prec.: 0.447 Exh.: 0.736 Med-F: 0.556
B	Prec.: 0.875 Exh.: 0.240 Med-F: 0.376	Prec.: 0.937 Exh.: 0.230 Med-F: 0.369	Prec.: 0.858 Exh.: 0.663 Med-F: 0.748

Daimler parece tener mejor capacidad de transferencia, es posible que se deba a la mayor cantidad de ejemplos de esa base de datos y su normalización de escala: en la mayoría de los ejemplos los peatones que figuran en ellas son del mismo tamaño. Por otro lado, se puede observar que el mejor resultado obtenido de manera general es utilizando el descriptor HOG. El agregado del descriptor LBP como parte representativa de una imagen no parece generar gran impacto sobre la precisión del modelo, considerando el aumento significativo en la dimensión del descriptor final. No obstante, este agregado demuestra mejorar significativamente la transferencia de aprendizaje para algunos casos como por ejemplo al entrenar con la base de datos INRIA y evaluar con las otras dos. Esto indica que la transferencia es muy dependiente del descriptor utilizado. Es posible que esto se deba también al hecho de que INRIA es una base de datos de personas y no de peatones exclusivamente. En la figura 5.4 se puede ver la diferencia de diversidad de los ejemplos positivos aportados por INRIA y por Daimler.

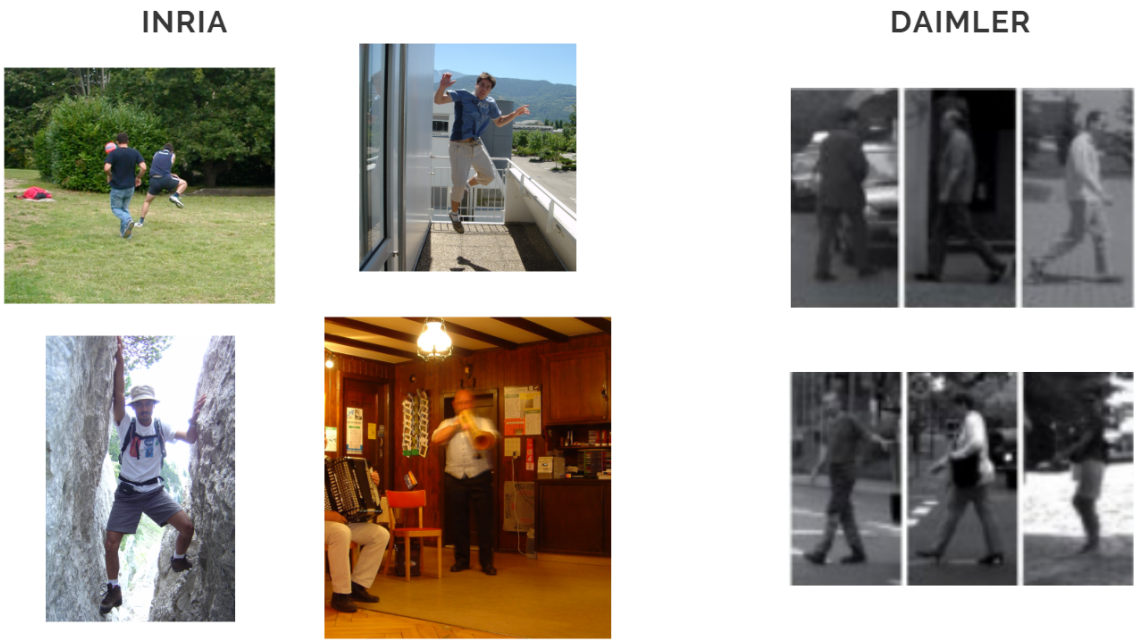


Figura 5.4 Algunos ejemplos positivos de INRIA (izquierda) y Daimler (derecha).

Una extensión del experimento anterior consiste en unir los conjuntos de entrenamiento de dos bases de datos para generar el modelo, y realizar la evaluación de igual forma que en el experimento anterior. De esta forma se puede evaluar si el agregado de datos diversos contribuye a la mejora de la transferencia. Es decir, analiza la complementariedad de las bases de datos. En las tablas 5.9, 5.10 y 5.11 se muestran la precisión, exhaustividad y medida-f conseguidas con las combinaciones de mas de una base de datos como conjunto de entrenamiento.

Tabla 5.9 Matrices de transferencia entre las tres bases de datos utilizadas al entrenar con múltiples bases de datos. Las entradas muestran la *precisión*, *exhaustividad* y *medida-f* usando **HOG**. Filas: Entrenamiento. Columnas: Evaluación. Abreviaciones: I = INRIA, D = Daimler, B = TUD-Brussels. En verde los resultados más altos.

	I	**D**	**B**
I + D	Prec.: 0.985 Exh.: 0.728 Med-F: 0.837	Prec.: 0.993 Exh.: 0.930 Med-F: 0.960	Prec.: 0.726 Exh.: 0.481 Med-F: 0.578
I + B	Prec.: 0.995 Exh.: 0.798 Med-F: 0.747	Prec.: 0.985 Exh.: 0.480 Med-F: 0.645	Prec.: 0.964 Exh.: 0.500 Med-F: 0.658
D + B	Prec.: 0.939 Exh.: 0.723 Med-F: 0.816	Prec.: 0.997 Exh.: 0.957 Med-F: 0.976	Prec.: 0.934 Exh.: 0.781 Med-F: 0.850

Tabla 5.10 Matrices de transferencia entre las tres bases de datos utilizadas al entrenar con múltiples bases de datos. Las entradas muestran la *precisión*, *exhaustividad* y *medida-f* usando **LBP**. Filas: Entrenamiento. Columnas: Evaluación. Abreviaciones: I = INRIA, D = Daimler, B = TUD-Brussels. En verde los resultados más altos.

	I	**D**	**B**
I + D	Prec.: 0.859 Exh.: 0.304 Med-F: 0.449	Prec.: 0.900 Exh.: 0.586 Med-F: 0.709	Prec.: 0.179 Exh.: 0.272 Med-F: 0.215
I + B	Prec.: 0.874 Exh.: 0.162 Med-F: 0.273	Prec.: 0.691 Exh.: 0.084 Med-F: 0.149	Prec.: 0.193 Exh.: 0.163 Med-F: 0.176
D + B	Prec.: 0.937 Exh.: 0.422 Med-F: 0.581	Prec.: 0.936 Exh.: 0.745 Med-F: 0.829	Prec.: 0.291 Exh.: 0.318 Med-F: 0.303

Tabla 5.11 Matrices de transferencia entre las tres bases de datos utilizadas al entrenar con múltiples bases de datos. Las entradas muestran la *precisión*, *exhaustividad* y *medida-f* usando **HOG** y **LBP**. Filas: Entrenamiento. Columnas: Evaluación. Abreviaciones: I = INRIA, D = Daimler, B = TUD-Brussels. En verde los resultados más altos.

	I	**D**	**B**
I + D	Prec.: 0.972 Exh.: 0.746 Med-F: 0.844	Prec.: 0.975 Exh.: 0.920 Med-F: 0.946	Prec.: 0.457 Exh.: 0.490 Med-F: 0.472
I + B	Prec.: 0.967 Exh.: 0.546 Med-F: 0.697	Prec.: 0.923 Exh.: 0.404 Med-F: 0.562	Prec.: 0.525 Exh.: 0.472 Med-F: 0.497
D + B	Prec.: 0.924 Exh.: 0.689 Med-F: 0.789	Prec.: 0.991 Exh.: 0.941 Med-F: 0.965	Prec.: 0.770 Exh.: 0.763 Med-F: 0.766

Se puede observar que al entrenar con varias bases de datos aumenta la medida-F en casi todos los casos con respecto a usar una sola base de datos. No obstante, se debe considerar un efecto de tamaño por la unión de los conjuntos de entrenamiento. Por ejemplo, TUD-Brussels no aumenta de forma considerable los ejemplos al sumarlos a los de INRIA y sin embargo aumenta significativamente la medida-F de su conjunto de evaluación.

Se resume en las figuras 5.5, 5.6 y 5.7 los resultados obtenidos en todos los experimentos utilizando el descriptor HOG que fue el que mejor se desempeñó, siendo la medida-f superior tanto al usar una única base de datos de entrenamiento como al utilizar combinaciones de ellas frente a los resultados logrados con LBP y la combinación HOG-LBP.

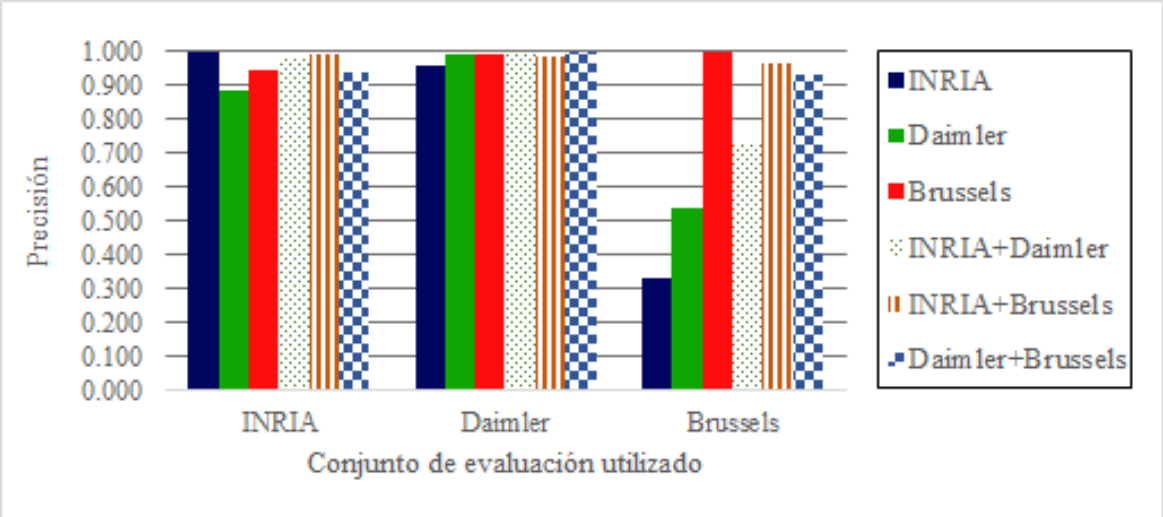


Figura 5.5 *Precisión* para los diferentes experimentos realizados de transferencia de aprendizaje. Las diferentes series muestran las distintas configuraciones de entrenamiento, mientras que las tres columnas principales representan las tres bases de datos de evaluación.

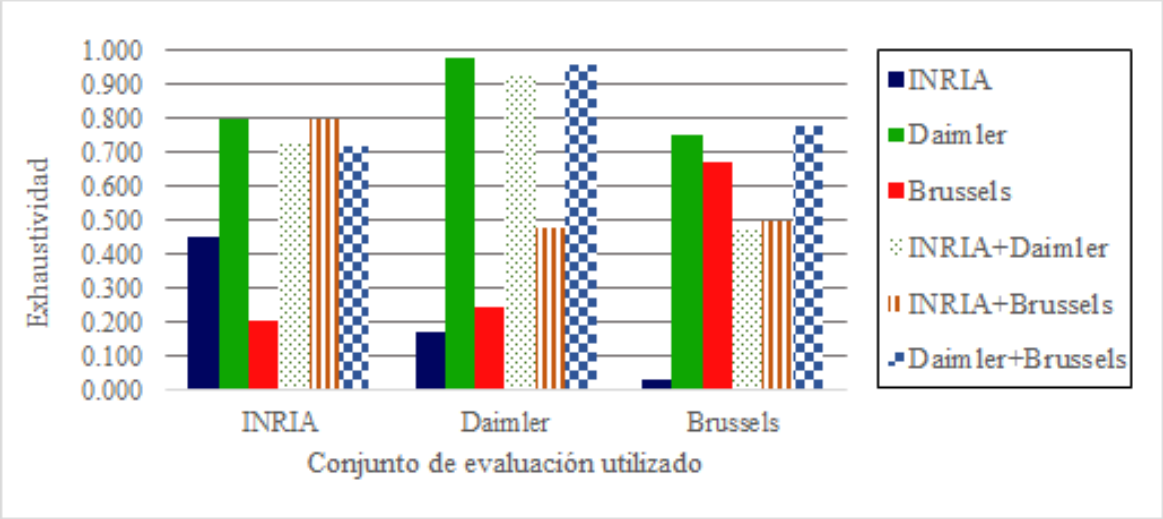


Figura 5.6 *Exhaustividad* para los diferentes experimentos realizados de transferencia de aprendizaje. Las diferentes series muestran las distintas configuraciones de entrenamiento, mientras que las tres columnas principales representan las tres bases de datos de evaluación.

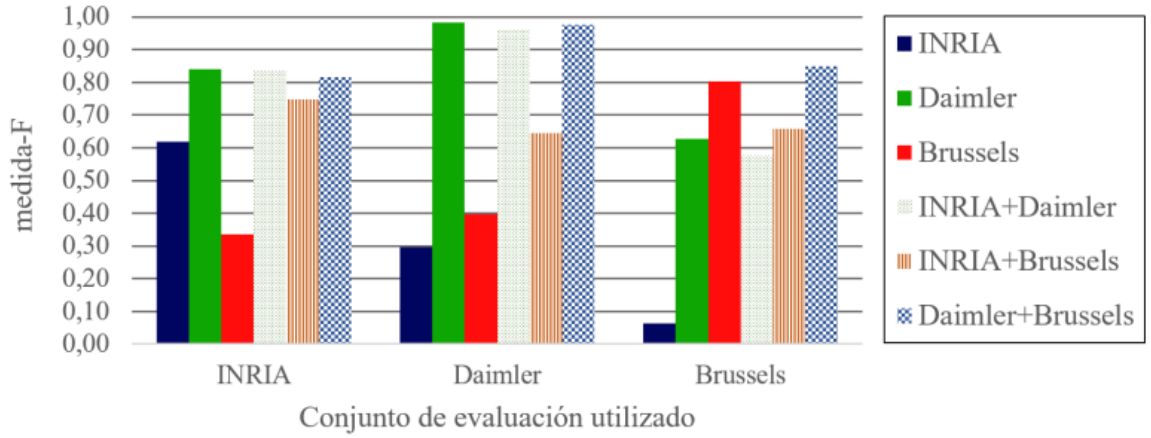


Figura 5.7 *medida-f* para los diferentes experimentos realizados de transferencia de aprendizaje. Las diferentes series muestran las distintas configuraciones de entrenamiento, mientras que las tres columnas principales representan las tres bases de datos de evaluación.

5.2.2. Conclusión de transferencia

Si bien los conjuntos de datos más ampliamente utilizados para la detección de peatones tienen la misma finalidad, los resultados presentan diferencias significativas que hacen no trivial la utilización de un modelo entrenado en un entorno real, con variaciones en aspectos de luminosidad, condiciones de escena, escalas, entre otros. Se puede observar también, que al entrenar con múltiples bases de datos, la combinación Daimler-Brussels utilizando LBP logra un muy buen desempeño frente al resto de las combinaciones de las bases de datos que utilizan el mismo descriptor, estando estas últimas muy lejos de alcanzar la precisión y exhaustividad que obtienen los primeros.

Ninguna de las tres bases de datos resulta ideal como único modelo de entrenamiento para llevar a cabo un proceso de transferencia de aprendizaje, aunque, en base a los resultados, Daimler parece la más apta de las evaluadas.

En cuanto a descriptores refiere, los descriptores HOG resultaron ser la mejor solución para llevar a cabo la clasificación con el SVM, tanto al utilizarse solo, como al usarse en su conjunto con el descriptor LBP. Al igual que en la sección 5.1.2 LBP consigue los peores resultados, con la excepción de Daimler que obtiene una precisión y exhaustividad bastante cercana a la obtenida utilizando HOG como descriptor para la clasificación.

Capítulo 6

Conclusión y trabajo a futuro

En este capítulo se expone un resumen de los objetivos logrados durante el desarrollo de la tesina. Por último, se detallan las líneas de investigación futuras que proponen extender los temas estudiados en esta tesina.

6.1. Conclusiones generales

A lo largo de la tesis se estudiaron las diferentes técnicas para la clasificación de peatones en imágenes y como se pueden utilizar los modelos que llevan a cabo esta tarea.

En primer lugar se cubrieron los conceptos básicos de representación y procesamiento de imágenes por computadora y la información que se puede extraer de ellas. Se estudiaron dos *descriptores* muy utilizados en este tipo de tareas (HOG y LBP), que sirvieron para armar los ejemplos de entrenamiento de un SVM de núcleo lineal que actuaría como clasificador. Posteriormente se estudió, previa introducción a las redes neuronales biológicas, el funcionamiento y estructura de las redes neuronales artificiales y su uso como clasificadores.

Se expuso el concepto de transferencia de aprendizaje, el cual consiste en medir la eficacia de un modelo al ser entrenado con un conjunto de datos y evaluado con otro completamente distinto. Para la medición de este concepto se realizó una comparación de las diferentes bases de datos disponibles para el entrenamiento de los clasificadores y los experimentos pertinentes para concluir dicha comparación.

Ya explicados los SVM y las redes neuronales, se detalló una serie de experimentos con el motivo de medir la precisión, exhaustividad y medida-f de ambos para un mismo conjunto de datos de evaluación. Se obtuvieron y procesaron tres bases de datos para entrenar y una para evaluar muy utilizadas en el estado del arte llegando a la conclusión de que los modelos de Deep Learning representan una mejora prometedora en el campo no solo de la clasificación y detección de peatones, sino también en los desafíos que la inteligencia artificial tiene y tendrá

vigentes. Sin embargo, se dejó en evidencia que si bien se consiguen mejores resultados, las estructuras de aprendizaje profundo acarrean problemas de eficiencia computacional, siendo $\approx 57\,\%$ más lento para llevar a cabo el entrenamiento y $\approx 98\,\%$ más lento para desempeñar la tarea de clasificación que un SVM lineal. Por este motivo, soluciones con estructuras de Deep Learning más complejas conllevaría costos en materia de hardware que hay que afrontar si se busca una solución que presente tiempos de respuesta aceptables para entornos más desafiantes.

Finalizados los experimentos comparativos de los clasificadores se midieron las tres métricas utilizadas en los mismos para medir la transferencia de aprendizaje que ofrece el SVM al entrenar con una base de datos y evaluar el mismo conjunto de entrenamiento y las otras dos restantes, utilizando HOG, LBP y la combinación de los dos como descriptores a utilizar. Luego se realizó una extensión de dicho experimento, entrenando con combinaciones de a pares de los tres datasets y evaluando con los restantes.

Los resultados arrojados confirman que la transferencia no siempre es aceptable, siendo LBP e INRIA el descriptor y la base de datos que peores resultados presentan. Por el contrario, Daimler logra los mejores resultados, siendo HOG el descriptor con el que mejor performance consigue. La combinación de Daimler con Brussels alcanza una mejor transferencia para Brussels como dataset de evaluación utilizando LBP. Hablando únicamente de combinaciones de descriptores, solo se consigue una mejora en algunos casos aislados como al entrenar con INRIA y Daimler y evaluar para la primera, consiguiendo una ínfima diferencia; en el caso de entrenar con una sola base de datos, la combinación HOG-LBP no logra superar ninguna medida-f obtenida al hacer uso únicamente de HOG, hecho que, considerando el incremento de dimensionalidad que significa unir ambos descriptores, no vale la pena considerar la combinación para transferir conocimiento de un modelo a conjuntos de datos de diferentes variaciones.

6.2. Líneas de trabajo futuras

Si bien se determinó un procedimiento que permitió obtener métricas como la precisión, exhaustividad y medida-f durante el proceso de clasificación, en la detección este tipo de mediciones poseen una mayor complejidad ya que se deben definir con anterioridad todos los BB de los peatones que figuran en cada una de la imágenes (o cuadros del video), luego llevar a cabo la tarea como fue descrita durante el trabajo: con una ventana deslizante que va recorriendo la imagen y marcando en ésta los BB correspondientes a los peatones que el clasificador detecte. Por cada uno de los BB generados durante la detección se debe realizar una evaluación del IOU con respecto a cada uno de los BB definidos con anterioridad, si

dicho valor es mayor que un umbral determinado, entonces se puede considerar como un verdadero positivo.

Además de los pasos recién definidos, que dificulta la preparación de material para la evaluación al tener que especificar frame por frame la posición de cada uno de los peatones que en ellos aparecen, también hay que considerar que los BB se presenta, naturalmente, de distintos tamaños, lo que dificulta a la hora de definir un tamaño de ventana deslizante que generalice, en su defecto, la mayoría de los casos.

Otra mejora que se podría aplicar a los algoritmos vistos es la utilización de las bases de datos completas. Las limitaciones de memoria RAM para entrenar un SVM con los descriptores propuestos seguirán existiendo, pero las redes neuronales, que durante la tesina fueron entrenadas solo utilizando el conjunto de datos proveniente de Daimler, podrán hacer uso de los datasets completos de entrenamiento correspondientes a INRIA y Brussels a fin de analizar si los resultados finales mejoran. Otra alternativa a considerar es probar distintos subconjuntos de las bases de datos a fin de observar si la performance final difiere positivamente.

En resume, la tarea de detección se presta para futuros desarrollos que abarquen múltiples ventanas deslizantes con los desafíos de eficiencia que eso significa, o estructuras especiales que afrontan el problema con gran eficacia como el sistema *YOLO* [44] (a la fecha en su versión *V3*) que fue diseñado para detectar objetos a múltiples escalas y evaluando solo una única vez la imagen completa, es decir, las ventanas deslizantes se hacen completamente prescindibles. Además, aplicar soluciones que hagan uso de paralelismo a nivel de instrucción, con el análisis de diferencias en cuanto a performance también resulta interesante de medir en la próxima etapa de investigación sobre la detección de peatones, ya que implementaciones como las citadas en el apartado de Deep Learning resultan costosas computacionalmente y no fueron evaluadas sistemáticamente durante la tesis.

Una vez detectado un peatón en una imagen, la investigación se presta a reconocer nuevos patrones en las imágenes como podrían ser situaciones de peligro (personas armadas, peleas, etc), establecer estructuras que realicen la detección en un tiempo de respuesta menor, análisis de situaciones (qué es lo que está haciendo el peatón detectado), seguimiento [14], estimar la ubicación de la persona aún cuando estos están ocultos detrás de objetos [59], entre otros muchos problemas que día a día se intentan resolver [26].

Bibliografía

[1] Angelova, A., Krizhevsky, A., Vanhoucke, V., Ogale, A., and Ferguson, D. (2015). Real-time pedestrian detection with deep network cascades. In *Proceedings of BMVC 2015*.

[2] Antol, S., Agrawal, A., Lu, J., Mitchell, M., Batra, D., Zitnick, C. L., and Parikh, D. (2015). VQA: Visual Question Answering. In *International Conference on Computer Vision (ICCV)*.

[3] Benenson, R., Omran, M., Hosang, J., and Schiele, B. (2014). Ten years of pedestrian detection, what have we learned? volume 8926.

[4] Bootstrap and Sphinx (2019). Scikit-image. https://scikit-image.org/.

[5] Boser, B. E., Guyon, I. M., and Vapnik, V. N. (1992). A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, COLT '92, pages 144–152, New York, NY, USA. ACM.

[6] Camele, G., Quiroga, F., Ronchetti, F., Hasperué, W., and Lanzarini, L. (2018). Transferencia de aprendizaje para clasificación de peatones. *CACIC*.

[7] Cao, X., Wang, Z., Yan, P., and Li, X. (2013a). Transfer learning for pedestrian detection. *Neurocomputing*, 100:51 – 57. Special issue: Behaviours in video.

[8] Cao, X., Zhong, W., Yan, P., and Liu, W. (2013b). Transfer learning for pedestrian detection. *Neurocomputing*, 100:51–57.

[9] Chen, X., Wei, P., Ke, W., Ye, Q., and Jiao, J. (2015). Pedestrian detection with deep convolutional neural network. In Jawahar, C. and Shan, S., editors, *Computer Vision - ACCV 2014 Workshops*, pages 354–365, Cham. Springer International Publishing.

[10] Dahl, G. E., Sainath, T. N., and Hinton, G. E. (2013). Improving deep neural networks for lvcsr using rectified linear units and dropout. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 8609–8613.

[11] Dalal, N. (2006). *Finding People in Images and Videos*. Theses, Institut National Polytechnique de Grenoble - INPG.

[12] Dalal, N. and Triggs, B. (2005). Histograms of oriented gradients for human detection. In *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volume 1, pages 886–893 vol. 1.

[13] Dalal, N., Triggs, B., and Schmid, C. (2006). Human detection using oriented histograms of flow and appearance. volume 3952, pages 428–441.

[14] Dehghan, A., Idrees, H., Zamir, A. R., and Shah, M. (2014). Automatic detection and tracking of pedestrians in videos with various crowd densities. In Weidmann, U., Kirsch, U., and Schreckenberg, M., editors, *Pedestrian and Evacuation Dynamics 2012*, pages 3–19, Cham. Springer International Publishing.

[15] Deng, J., Berg, A. C., Li, K., and Fei-Fei, L. (2010). What does classifying more than 10,000 image categories tell us? In Daniilidis, K., Maragos, P., and Paragios, N., editors, *Computer Vision – ECCV 2010*, pages 71–84, Berlin, Heidelberg. Springer Berlin Heidelberg.

[16] Deng, J., Russakovsky, O., Krause, J., Bernstein, M., Berg, A. C., and Fei-Fei, L. (2014). Scalable multi-label annotation. In *ACM Conference on Human Factors in Computing Systems (CHI)*.

[17] Enzweiler, M. and Gavrila, D. (2009). Monocular pedestrian detection: Survey and experiments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 31:2179–2195.

[18] Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J., and Zisserman, A. (2010). The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, 88(2):303–338.

[19] Felzenszwalb, P. F., Girshick, R. B., McAllester, D., and Ramanan, D. (2010). Object detection with discriminatively trained part-based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1627–1645.

[20] Freund, Y. and Eschapire, R. (1997). A decision-theoretic generalization of on-line learning and an application to boosting. *Journal of Computer and System Sciences*, 55(1):119 – 139.

[21] G. Lowe, D. (2004). Distinctive image features from scale-invariant keypoints. *International Journal of Computer Vision*, 60:91–110.

[22] Gan, G. and Cheng, J. (2011). Pedestrian detection based on hog-lbp feature. In *2011 Seventh International Conference on Computational Intelligence and Security*, pages 1184–1187.

[23] Glorot, X., Bordes, A., and Bengio, Y. (2011). Deep sparse rectifier neural networks. In Gordon, G., Dunson, D., and Dudík, M., editors, *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, volume 15 of *Proceedings of Machine Learning Research*, pages 315–323, Fort Lauderdale, FL, USA. PMLR.

[24] Google (2019). Tensorflow. https://www.tensorflow.org/.

[25] Gregor, K., Danihelka, I., Graves, A., and Wierstra, D. (2015). DRAW: A recurrent neural network for image generation. *CoRR*, abs/1502.04623.

[26] Güler, R. A., Neverova, N., and Kokkinos, I. (2018). Densepose: Dense human pose estimation in the wild. *CoRR*, abs/1802.00434.

[27] Haythem, A., Helali, A., Nasri, M., Maaref, H., and Youssef, A. (2014). Improved feature extraction method based on histogram of oriented gradients for pedestrian detection. pages 1–5.

[28] INRIA (2019). Scikit-learn. https://scikit-learn.org/stable/.

[29] Jamshed, M., Parvin, S., and Akter, S. (2015). Significant hog-histogram of oriented gradient feature selection for human detection. *International Journal of Computer Applications*, 132:20–24.

[30] Jiang, N., Xu, J., Yu, W., and Goto, S. (2013). Gradient local binary patterns for human detection. In *2013 IEEE International Symposium on Circuits and Systems (ISCAS2013)*, pages 978–981.

[31] Kobayashi, T., Hidaka, A., and Kurita, T. (2007). Selection of histograms of oriented gradients features for pedestrian detection. pages 598–607.

[32] Kornblith, S., Shlens, J., and Le, Q. V. (2018). Do better imagenet models transfer better? *CoRR*, abs/1805.08974.

[33] Kuznetsova, A., Rom, H., Alldrin, N., Uijlings, J., Krasin, I., Pont-Tuset, J., Kamali, S., Popov, S., Malloci, M., Duerig, T., and Ferrari, V. (2018). The open images dataset v4: Unified image classification, object detection, and visual relationship detection at scale. *arXiv:1811.00982*.

[34] Li, H., Wu, Z., and Zhang, J. (2016). Pedestrian detection based on deep learning model. In *2016 9th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI)*, pages 796–800.

[35] Lin, C.-J. (2019). Liblinear. https://www.csie.ntu.edu.tw/~cjlin/liblinear/.

[36] Lin, T., Maire, M., Belongie, S. J., Bourdev, L. D., Girshick, R. B., Hays, J., Perona, P., Ramanan, D., Dollár, P., and Zitnick, C. L. (2014). Microsoft COCO: common objects in context. *CoRR*, abs/1405.0312.

[37] Moreno, J. D. A. (2012). Análisis de textura en imágenes a escala de grises, utilizando patrones locales binarios (lbp).

[38] Murphy, K. P. (2012). *Machine Learning: A Probabilistic Perspective (Adaptive Computation and Machine Learning series)*.

[39] Nair, V. and Hinton, G. E. (2010). Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th International Conference on International Conference on Machine Learning*, ICML'10, pages 807–814, USA. Omnipress.

[40] Netzer, Y., Wang, T., Coates, A., Bissacco, A., Wu, B., and Ng, A. Y. (2011). Reading digits in natural images with unsupervised feature learning.

[41] Nielsen, M. (2015). *Neural Networks and Deep Learning*.

[42] Ouyang, W. and Wang, X. (2013). Single-pedestrian detection aided by multi-pedestrian detection. In *2013 IEEE Conference on Computer Vision and Pattern Recognition*, pages 3198–3205.

[43] Palm, G. (1986). Warren mcculloch and walter pitts: A logical calculus of the ideas immanent in nervous activity. In Palm, G. and Aertsen, A., editors, *Brain Theory*, pages 229–230, Berlin, Heidelberg. Springer Berlin Heidelberg.

[44] Redmon, J., Divvala, S. K., Girshick, R. B., and Farhadi, A. (2015). You only look once: Unified, real-time object detection. *CoRR*, abs/1506.02640.

[45] Ren, S., He, K., Girshick, R. B., and Sun, J. (2015). Faster R-CNN: towards real-time object detection with region proposal networks. *CoRR*, abs/1506.01497.

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

[47] Schmidhuber, J. (2015). Deep learning in neural networks: An overview. *Neural Networks*, 61:85 – 117.

[48] Schneider, N. and Gavrila, D. M. (2013). Pedestrian path prediction with recursive bayesian filters: A comparative study. In Weickert, J., Hein, M., and Schiele, B., editors, *Pattern Recognition*, pages 174–183, Berlin, Heidelberg. Springer Berlin Heidelberg.

[49] Shelhamer, E., Long, J., and Darrell, T. (2017). Fully convolutional networks for semantic segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 39(4):640–651.

[50] Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958.

[51] Srivastava, R. K., Greff, K., and Schmidhuber, J. (2015). Training very deep networks. *CoRR*, abs/1507.06228.

[52] Tomè, D., Monti, F., Baroffio, L., Bondi, L., Tagliasacchi, M., and Tubaro, S. (2016). Deep convolutional neural networks for pedestrian detection. *Signal Processing: Image Communication*, 47:482 – 489.

[53] Wojek, C., Walk, S., and Schiele, B. (2009). Multi-cue onboard pedestrian detection. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 794–801.

[54] Xiao, H., Rasul, K., and Vollgraf, R. (2017). Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms.

[55] Yang, S., Liao, X., and Borasy, U. (2012). A pedestrian detection method based on the hog-lbp feature and gentle adaboost. *International Journal of Advancements in Computing Technology*, 4:553–560.

[56] Yang, Y., Mingwu, R., and Jingyu, Y. (2014). Obstacles and pedestrian detection on a moving vehicle. *International Journal of Advanced Robotic Systems*, 11(4):53.

[57] Zhang, L., Lin, L., Liang, X., and He, K. (2016a). Is faster r-cnn doing well for pedestrian detection? In Leibe, B., Matas, J., Sebe, N., and Welling, M., editors, *Computer Vision – ECCV 2016*, pages 443–457, Cham. Springer International Publishing.

[58] Zhang, R., Isola, P., and Efros, A. A. (2016b). Colorful image colorization. *CoRR*, abs/1603.08511.

[59] Zhao, M., Li, T., Abu Alsheikh, M., Tian, Y., Zhao, H., Torralba, A., and Katabi, D. (2018). Through-wall human pose estimation using radio signals.