Universidad Zaragoza

Trabajo Fin de Máster

Localización de objetos en tiempo real en imágenes para entornos domésticos

Autora

Iris Sesma Gracia

Director

Carlos Orrite Uruñuela

Escuela de Ingeniería y Arquitectura

Septiembre 2016

TRABAJOS DE FIN DE GRADO / FIN DE MÁSTER

Escuela de Ingeniería y Arquitectura Universidad Zaragoza

DECLARACIÓN DE AUTORÍA Y ORIGINALIDAD

(Este documento debe acompañar al Trabajo Fin de Grado (TFG)/Trabajo Fin de Máster (TFM) cuando sea depositado para su evaluación).

D./Dª. IRIS SESMA GRACIA,

con nº de DNI 72816582-P en aplicación de lo dispuesto en el art. 14 (Derechos de autor) del Acuerdo de 11 de septiembre de 2014, del Consejo de Gobierno, por el que se aprueba el Reglamento de los TFG y TFM de la Universidad de Zaragoza,

Declaro que el presente Trabajo de Fin de (Grado/Máster) MÁSTER, (Título del Trabajo) LOCALIZACIÓN DE OBJETOS EN TIEMPO REAL EN IMÁGENES PARA ENTORNOS DOMÉSTICOS,

es de mi autoría y es original, no habiéndose utilizado fuente sin ser citada debidamente.

Zaragoza, 21 de Septiembre de 2016

Fdo: Iris

Localización de objetos en tiempo real en imágenes para entornos domésticos

Resumen

Este trabajo se enmarca dentro del proyecto *Memory Lane* cuyo objetivo principal consiste en el desarrollo de un asistente para personas que sufren pérdidas de memoria. Este asistente será capaz de recordar desde dónde han dejado algún objeto hasta una receta de cocina.

En este trabajo se busca diseñar un sistema que sea capaz de aprender a reconocer objetos. De tal manera que el sistema, una vez entrenado, esté preparado para de responder en el menor tiempo posible que objetos contiene una cierta imagen de entrada. El asistente de *Memory Lane,* deberá tener la información disponible lo más rápido posible. Cabe indicar que los objetos a reconocer pueden estar a distintas distancias de la cámara y en distintos ángulos, es decir, los objetos pueden aparece en cualquier punto de la imagen y con cualquier tamaño y orientación.

Para ello se emplea la red neuronal convolucional (CNN) ConvNet como extractor de características. Tras ello, se reducen las características eligiendo máximos y empleando *Linear Discriminat Analysis* (LDA). Se entrena un perceptrón multicapa (MLP) como clasificador, al que luego se le pasará una imagen en varios subconjuntos y escalas. Combinando la información de salida del MLP se construirá un mapa de calor que permitirá detectar los límites de los distintos objetos presentes en dicha imagen.

Abstract

This thesis is part of Memory Lane project that seeks to be able to generate an assistant for people suffering memory loss; this assistant will be able to remember where you have left some object or a prescription.

This thesis seeks to design a system that is able to learn to recognize objects. It is intended that the system, once trained, is able to respond in the shortest time possible what objects are contained in a certain input image, since the assistant Memory Lane should have the information available as quickly as possible. The objects to recognize can be at different distances from the camera and at different angles, that is, the objects can appear anywhere in the image in any size and orientation.

To do so, the convolutional neural network (CNN) ConvNet is used as feature extractor. After that, the features are reduced by choosing maximums and using Linear Discriminat Analysis (LDA). A Multi Layer Perceptron (MLP) is trained as a classifier and an image with several subsets and scales will be passed to it. Combining the output of the MLP a heat map will be built that will identify the limits of the various objects present in said image.

Índice de contenido

Índice de Figuras

Índice de tablas

1 Introducción

1.1 Contextualización

El antecedente más destacado es el proyecto aprobado por el Ministerio de Economía y Competitividad dentro del Plan Estatal de Investigación Científica y Técnica y de Innovación 2013 -2016, denominado *Memory Lane*. En la Figura 1 se muestra un esquema de como se puede subdividir el proyecto. Este trabajo tocaría los puntos de "*semantic interpretation*" en la parte de aprendizaje supervisado y la de reconocimiento de objetos en sensores audiovisuales (*audiovisual sensing*).

Este proyecto plantea un relevante avance en el estado del arte en entornos inteligentes para dar soporte a la autonomía de personas con necesidades especiales. De un modo totalmente no intrusivo ni alienante, propone un sistema que permita adquirir cierta consciencia del contexto de una persona y grabarlo del mismo modo que las personas construimos nuestros recuerdos: a través de información audiovisual aumentada con información contextual sobre dónde, cuándo, con quién o cómo sucedieron las cosas.

Este proyecto estaría relacionado con la parte de localización de los objetos en las imágenes. Se pretende lograr un algoritmo que identifique los objetos de las imágenes en tiempo real, lo cual no significa tiempo cero, sino un tiempo en el cual los humanos no sean capaces de detectar, por lo que debe rondar valores inferiores a 100 mili-segundos.

1.2 Objetivos

El objetivo final que se intenta alcanzar es localizar y reconocer los objetos en un vídeo. Como un vídeo no es más que un conjunto de imágenes consecutivas, partiremos de la versión sencilla de localizar los objetos en imágenes y procuráremos optimizar los tiempos para que el algoritmo pueda emplearse en alguna solución de vídeo.

Para que los objetivos sean alcanzables se partirá de reconocer objetos. Tras ello, se buscará localizarlos. Durante ese proceso, se procurará minimizar tiempo invertido en reconocer y localizar los objetos en una imagen, para lograr que sea viable el procesado en tiempo real.

Finalmente, se espera que este sistema sea capaz de reconocer objetos de entornos domésticos. Aunque no existen bases de datos adecuadas, por lo que nos centraremos en las imágenes de entornos domésticos disponibles para cumplir con nuestra idea de integrarlo dentro del proyecto *Memory Lane.*

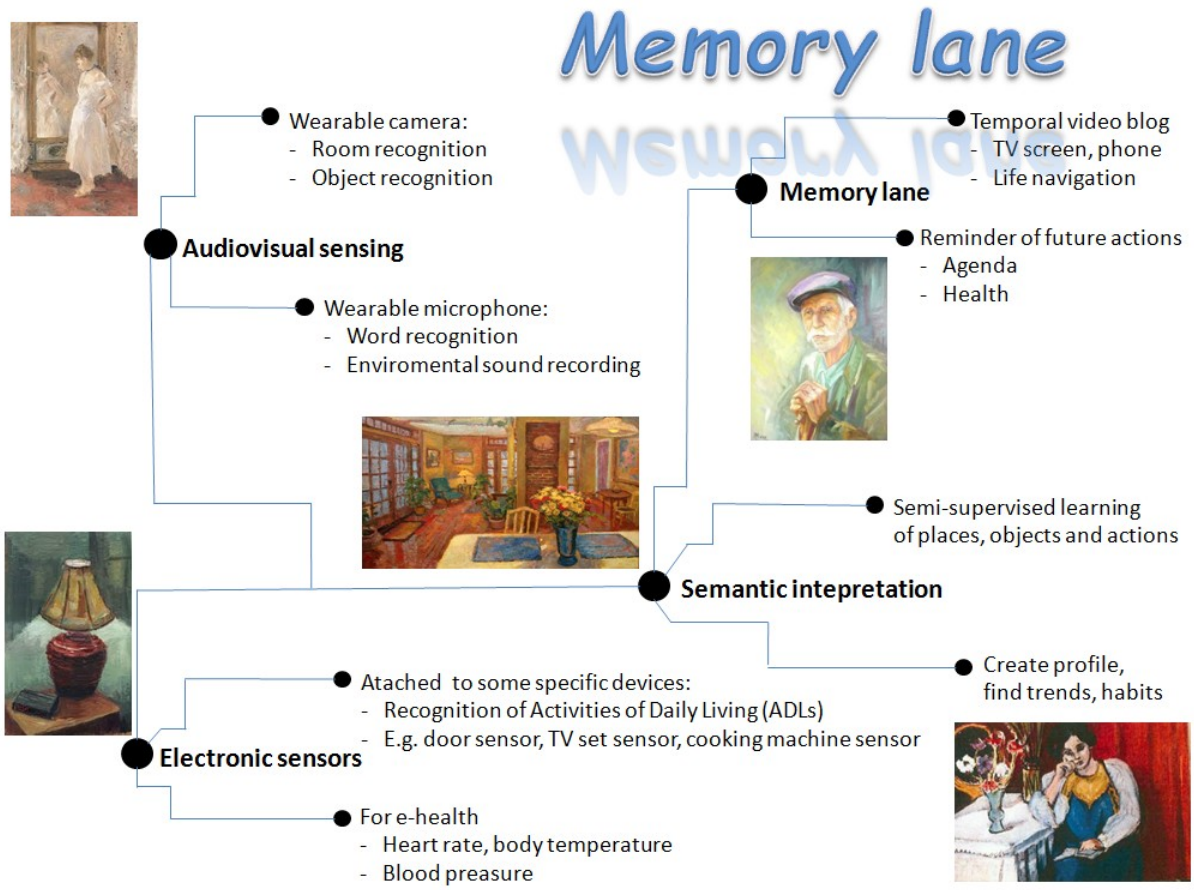


Figura 1: Esquema de Memory lane

1.3 Sobre este documento

A lo largo de este documento se va a recopilar el trabajo realizado. Para ello, se comenzará con conceptos teóricos de las herramientas que se utilizan para el desarrollo de éste, como son las redes neuronales. A continuación, en el estado del arte se agrupan los conceptos básicos y los trabajos previos en los que se apoya el desarrollo del mismo. También, se hará un repaso de otras soluciones propuestas para resolver problemas similares.

Tras conocer todas las herramientas de base para el desarrollo, se describirá el trabajo realizado y las motivaciones de las decisiones tomadas. A continuación, se presentarán y analizarán los resultados obtenidos.

Para finalizar, se extraerán conclusiones de todo lo visto previamente y se plantearán las posibles líneas futuras de investigación.

Para facilitar la lectura, se ha incluido un apéndice con las abreviaturas y los resultados menos relevantes están añadidos como anexo.

1.4 Bases de datos empleadas

Se van a emplear principalmente los datos de la BBDD de Pascal VOC2007[1] y VOC2012[2]. Cabe comentar, que la información de la primera está contenida en la segunda, pero al tener menos datos permite probar el funcionamiento del algoritmo de forma más rápida.

Ambas BBDD constan de 20 categorías como se muestra en la tabla 1 [Everingham2015] que se reparten en 4 grupos. Se va trabajar con la BBDD completa, pero en el fondo tenemos más interés por el grupo de los objetos domésticos. Aunque también de los otros grupos podría ser interesante el reconocimiento de animales domésticos o personas.

Tabla 1: Tipos de categorías (traducido de [Everingham2015]).

Vehículos	Domésticos	Animales	Otros
Avión	Botella	Pájaro	Persona
Bicicleta	Silla	Gato	
Barco	Mesa comedor	Vaca	
Autobús	Maceta	Perro	
Coche	Sofá	Caballo	
Moto	TV/Monitor	Oveja	
Tren			

Las imágenes de estas BBDD están etiquetas para distintos tipos de análisis: clasificación, detección, segmentación y clasificación de acciones. En el caso de VOC2012, además, están etiquetadas: cabezas, manos y pies humanos.

Para este trabajo sólo nos interesa la información de cuales son los objetos contenidos en una imagen y su ubicación, por lo que el resto de las etiquetas no se tendrán en cuenta.

Como no se pueden emplear las mismas imágenes para entrenar el sistema que para probarlo, puesto que el objetivo de estos sistemas es que sean capaces de generalizar a partir de lo aprendido, se proporcionan 3 grupos "*train*", "*val*" y "*test*" (VOC2007 tan sólo tiene las 2 primeras).

VOC2007 tiene unas 5000 imágenes y VOC2012 algo más de 17000, por ello consideraremos más fiables los resultados con más imágenes, las otras se emplearán para probar el código.

Aunque no se emplea en este trabajo directamente, cabe mencionar también la base de datos de ImagiNet[3]. La CNN que se va a emplear como extractor de características se entrenó con ella. Dicha base de datos contiene 1000 categorías y 1,2 millones de imágenes.

1 http://host.robots.ox.ac.uk/pascal/VOC/voc2007/index.html
2 http://host.robots.ox.ac.uk/pascal/VOC/voc2012/index.html
3 http://www.image-net.org/challenges/LSVRC/2012/

2 Estado del Arte

En este capítulo se van a repasar los conceptos relevantes de reconocimiento y localización de objetos. Además de la descripción del funcionamiento básico de clasificador como introducción a la temática.

2.1 Introducción al reconocimiento de patrones

Lograr que las máquinas interpreten la realidad por nosotros no es un objetivo nuevo. De hecho, hace años que nos corrigen los textos, hace unos pocos menos se logro el hito de que interpretasen nuestras palabras y las convirtiesen en texto, pero se seguía escapando la información visual en la cual se basa gran parte de nuestro conocimiento. Aunque se lleva trabajando en ello desde hace mucho como demuestra el artículo de Fu y Rosenfeld de 1976 sobre reconocimiento de patrones y procesado de imágenes [Fu1976].

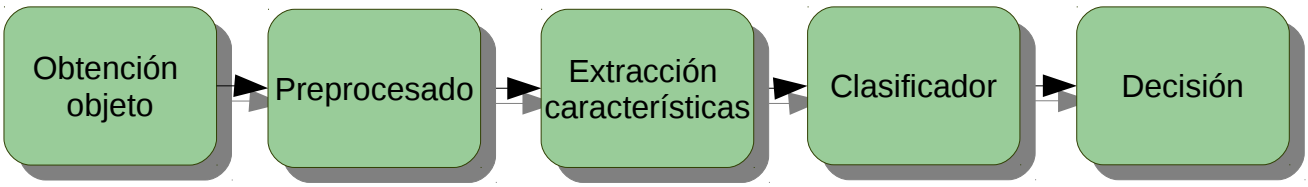


Figura 2: Esquema general reconocimiento de patrones.

El proceso para lograr que las máquinas aprendan sigue una estructura aunque puedan diferir las técnicas empleadas en cada uno de los pasos. Se puede interpretar como esquema general la Figura 2. Estos pasos consisten en: obtener el objeto de interés como imágenes, señales biológicas, voz, etc. Habitualmente se somete a un preprocesado para redimensionar la imagen, filtrar el ruido de las señales biológicas o la voz, etc. Tras ello, se extraen las características colores y formas de la imagen, frecuencia y forma de las señales, etc. El siguiente paso es el clasificador, en el cual se bifurca el proceso según estemos en la fase de “entrenamiento” o de “operación”, puesto que el clasificador se adaptará al problema que se quiera resolver, es necesario prepararlo para ello, cuando el sistema se dé por válido se podrá pasar a emplearlo. Por último, se toma la decisión de si está bien clasificado o no, esto tiene especial relevancia en la fase de entrenamiento, en la cual, si el resultado no es el esperado se puede corregir el clasificador.

Sobre la “obtención del objeto”, señalar que las imágenes que se van a emplear son de una base de datos con imágenes de la vida cotidiana, ampliamente empleada en el campo de estudio [Girshick2014;Sermanet2013;Donahue2013;Hoiem2012]. Como no se ha participado en la obtención no existe control sobre su calidad, su tamaño o cualquier otro parámetro que pueda influir posteriormente en el resultado.

Para facilitar el trabajo del clasificador es necesario comenzar por extraer unas características capaces de discernir entre las distintas categorías. En gran medida el éxito o el fracaso del método va a depender de que se puedan separar fácilmente las clases por tener unas buenas características. Estas no siempre se pueden conseguir de forma sencilla, en muchas ocasiones requieren un preprocesado previo de los datos antes de extraer dichas características o una transformación posterior del espacio de características como PCA (*Principal Componet Analysis*) o LDA (*Linear Discriminant Analysis*). Además, se debe lograr optimizar el número de éstas porque un exceso de ellas podría añadir únicamente información redundante y un mayor coste computacional [Guyon2008].

PCA permite extraer las características principales de un conjunto de datos, transforma el conjunto de datos de entrada en otro proyectado de forma que permite ser ordenado de más a menos relevancia, lo que permite eliminar la información poco relevante. De forma más abstracta, se puede ver el conjunto de datos como un conjunto vectorial en el cual proyectamos sus bases para que sigan las variaciones máximas del conjunto. A esas nuevas bases les llama vectores propios y el valor que tienen asociado es el que sirve para medir su peso en el conjunto. Desde el punto de vista de este trabajo se emplearán para quedarnos con las mejores características.

Por otro lado, LDA es similar a PCA aunque busca que las componentes de la proyección sean linealmente independientes en vez de en maximizar las componentes. Se centra en discriminar lo mejor posible las clases. Otra de las peculiaridades respecto a PCA es que reduce las dimensiones del conjunto de datos, por lo que siempre se fuerza la reducción del espacio vectorial. PCA y LDA pueden ser complementarias, en muchas ocasiones se emplea en primer lugar PCA para encontrar las características más representativas, y tras ello se aplica LDA para optimizar la separación entre ellas. En el anexo 8.2 Métodos de reducción de características, se profundiza más en LDA y PCA.

Otra de las herramientas que se va a emplear son las redes neuronales artificiales, en concreto, MLP (*Multi Layer Percepton*) para la clasificación y CNN (*Convolutional Neural Networks*) en la extracción. La idea del procesado de información con redes neuronales artificiales surge de intentar imitar el funcionamiento del cerebro [Martín2006], puesto que muchos procesadores con muy poca velocidad y capacidad de procesamiento tienen una gran potencia de cálculo y aprendizaje.

Los clasificadores se emplean en dos fases distintas, por un lado hay que entrenarlos para que sepan resolver el problema que nos interesa, en muchas ocasiones, como en nuestro caso, se emplea un sistema supervisado, en el cual se le indica el resultado al que queremos que llegue. Es decir, si se quiere que un sistema aprenda a distinguir enfermedades según los síntomas, se le facilitarán las características (los síntomas) y el resultado esperado (la enfermedad). Tras entrenar el sistema de clasificación, se podrá emplear ese clasificador entrenado para que dados unos síntomas se determine que enfermedades son las más probables.

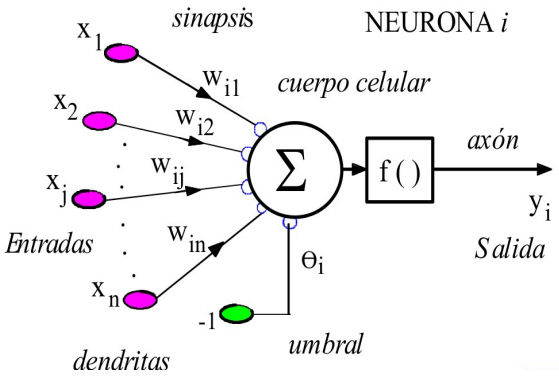
Tras la clasificación llega la fase de decisión, en la cual hay que escoger entre los resultados obtenidos cual es el válido. Dependiendo del sistema de clasificación empleado, la salida se puede representar de distintas formas, puede ocurrir que haya varios datos correspondiendo con las distintas clases y que haya que tomar una decisión. Uno de los métodos más simples es escoger el máximo. En nuestro caso, como se expondrá en el capitulo 3, antes de tomar la decisión sobre dónde está un objeto en una imagen se determinará qué ha clasificado mirándola a distintas escalas.

2.2 Redes Neuronales

Sobre el tema de las redes neuronales se ha escrito mucho (se ha empleado como referencia [Martín2006]), en este documento se expondrán únicamente algunas nociones básicas. Con ese fin, en este apartado y en el anexo 8.1 se dará una visión general para poder comprender los dos tipos de redes que se emplean más habitualmente en clasificación MLP (*Multi Layer Perceptron*) y SVM (*Support Vector Machine*).

Las redes neuronales artificiales parten de la idea del funcionamiento de las neuronas biológicas, en las cuales hay muchos procesadores con muy poca capacidad y lentos en comparación con los procesadores de los ordenadores actuales que son pocos y muy potentes, y a pesar de ello tienen una capacidad mayor de aprendizaje.

Cada una de las neuronas tiene su función que la modela, se pueden definir en general como:



$$y_i = f_i\left(\sum_{j=1}^{n} w_{ij} x_j - \theta_i\right)$$

$f(.)$ *función de activación*
w_{ij} *pesos sinápticos*
θ_i *bias o threshold (umbral)*

Figura 3: Modelo neurona artificial [Martín2006].

La función genérica que está definida arriba se puede particularizar para distintos casos como los que se pueden observar en la Figura 4.

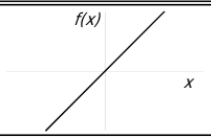
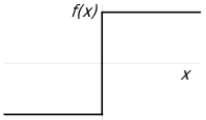
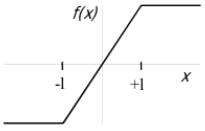
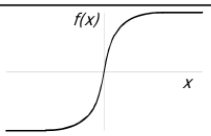
	Función	Gráfica
Identidad	$y = x$	
Escalón	$y = sign(x)$ $y = \mathrm{H}(x)$	
Lineal a tramos	$y = \begin{cases} -1, & si\ x < -l \\ x, & si\ +l \le x \le -l \\ +1, & si\ x > +l \end{cases}$	
Sigmoidea	$y = \frac{1}{1+e^{-x}}$ $y = \tanh(x) = \frac{e^{x} - e^{-x}}{e^{x} + e^{-x}}$	

Figura 4: Tabla con funciones típicas empleadas en neuronas [Martín2006].

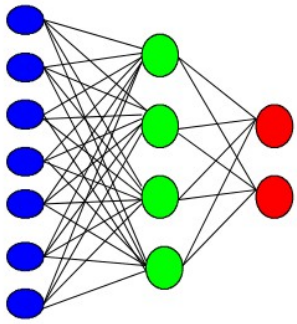


Figura 5: Ejemplo conexión entre capas de neuronas.

Las neuronas no trabajan solas, se organizan por capas, donde las neuronas pertenecientes a una misma capa no están conectadas entre ellas. Lo que tienen en común es que reciben información de todas las neuronas de la capa previa (o las entradas en caso de ser la primera capa). A su vez, las salidas se conectan con todas las neuronas de la capa siguiente, como se puede observar en la Figura 5.

Los pesos que hemos visto w_{ij} son los que contienen la información de lo que han aprendido las neuronas. Estos pesos inicialmente son aleatorios, después, durante el proceso de aprendizaje, esos valores varían para ir aprendiendo el camino adecuado. Esto produce que las primeras iteraciones den salidas erróneas, pero éstas se comparan con las salidas teóricas, y los pesos se modifican para que la próxima salida sea más parecida a la teórica. Esto se repite hasta que la red este entrenada.

Contra más complejo sea un problema más neuronas va a necesitar para ser modelado. Una de las estructuras típicas son las redes con 3 capas, de las cuales no se pueden variar el número de neuronas de entrada ni salida, puesto que vienen impuestos por el problema. Eso quiere decir, que cuanto más complejo sea el problema, más neuronas harán falta en la intermedia, la cual se conoce también como capa intermedia.

Existen redes neuronales que realimentan la entrada con información de las salidas, se conocen como *"feedback"*, pero debido a su complejidad no son muy populares. Las que se van a emplear son las redes unidireccionales o *"feed forward"*, en las que las neuronas solo contienen como entradas las capas anteriores.

Como ya se ha comentado, para más información sobre las redes neuronales se puede acudir al anexo 8.1 de este trabajo.

2.3 Deep Learning

Se llama *deep learning* o aprendizaje profundo a las estructuras de redes neuronales con varias capas ocultas conectadas en cascada. Éstas se caracterizan por la capacidad de aprendizaje automático. Gracias a las herramientas que proporcionan se puede dejar que la red aprenda "sola", con un mínimo de interacción humana. De esa forma puede lograr aprender obteniendo las mejores características para distinguir objetos en una imagen, por ejemplo, sin necesidad de que se definan esas características a priori. El proceso de aprendizaje humano de cómo llegamos a saber que un determinado objeto es una silla, si queremos transmitir esa información a un algoritmo es complejo, puesto que algunos detalles se nos escapan, lo sabemos pero no somos consciente de ellos. De esa forma, los sistemas de aprendizaje automático proporcionan herramientas para que los sistemas se fijen en los detalles que pueden ayudarles a distinguir lo que se les intente enseñar. Una de las desventajas de estos sistemas es el altísimo número de ejemplos que necesitan para aprender correctamente, puesto que son redes muy complejas.

Se va a estudiar en más profundidad la red neuronal convolucional (CNN por las siglas en inglés), que se empleará más adelante como extractor de características de un sistema. Esta red fue entrenada con la base de datos de Imaginet[4] la cual contiene algo más de 14 millones de imágenes en 27 categorías principales.

2.3.1 Convolutional Neural Network

Más concretamente, se va a estudiar la arquitectura de la red de Krizhevsky [Krizhevsky2012] consta de ocho capas con pesos distintos: Cinco capas convolucionales y tres capas totalmente conectadas.

Los autores encontraron que prescindir de cualquier capa convolucional resultaba en un rendimiento

4 Imaginet: http://image-net.org/

inferior. La salida de la última capa está conectada a una capa neuronal *softmax*[5] de 1000 entradas que produce una distribución sobre 1000 etiquetas de clase.

Para modelar la salida de la neurona f como una función de su entrada x, usaron la no-linealidad no-saturante $f(x)=max(0,x)$, refiriéndose a las neuronas como Unidades Lineales Rectificadas (ReLUs). Las ReLUs permiten entrenar la CNN muy rápidamente y tienen la propiedad de no necesitar normalización de entrada para prevenir la saturación. De todas formas se usa un esquema/programa de normalización local para ayudar a generalizar. Para reducir el sobreentrenamiento (se aclara este concepto en el apartado 8.1.3), usaron una técnica de *max pooling* [Nagi2011], que resume las salidas de grupos de neuronas vecinas en el mismo mapa de kernel.

En detalle, en la Figura 6 la primera capa convolucional filtra la imagen de entrada de tamaño 224x224x3 con 96 filtros kernel de tamaño 11x11x3 con un paso de 4 píxeles. La segunda capa convolucional toma como entrada la salida de la primera capa convolucional y la filtra con 256 filtros *kernels* de tamaño 5x5x48. Las últimas tres capas convolucionales están conectadas a otra sin ningún *pooling* ni normalización. La tercera capa convolucional tiene 384 filtros kernel de tamaño 3x3x256 conectados a las salidas de la segunda capa convolucional. La cuarta capa convolucional tiene 384 filtros *kernels* de tamaño 3x3x192, y la quinta capa convolucional tiene 256 filtros *kernels* de tamaño 3x3x192. Las capas totalmente conectadas tienen 4096 neuronas cada una.

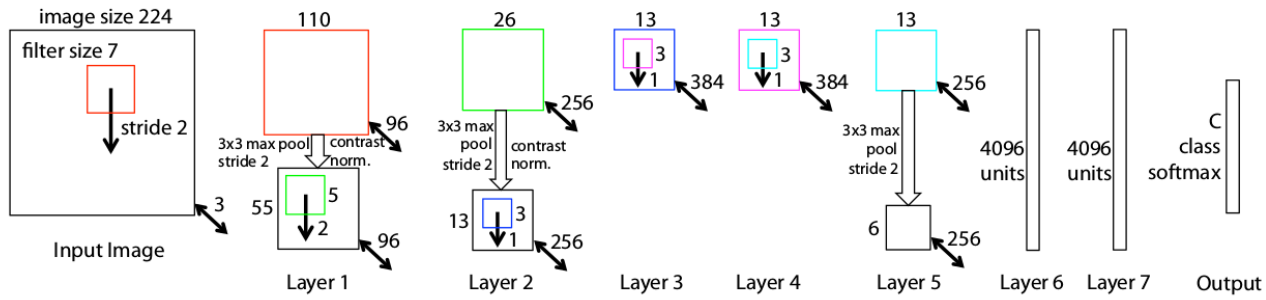


Figura 6: Arquitectura de 8 capas del modelos ConvNet. [Zeiler2014]

Para el interés de este trabajo se emplearán únicamente las capas convolucionales, es decir, la capa 5, sería la última. Cabe destacar que todas las capas son cuadradas.

La descripción de qué ésta sucediendo a través de las distintas capas de las redes convolucionales, puede resultar muy abstracta, por ello en la Figura 7 se muestra que hay a la salida de cada una de las capas. Se puede observar como la red es capaz de identificar y extraer la información más relevante de las imágenes.

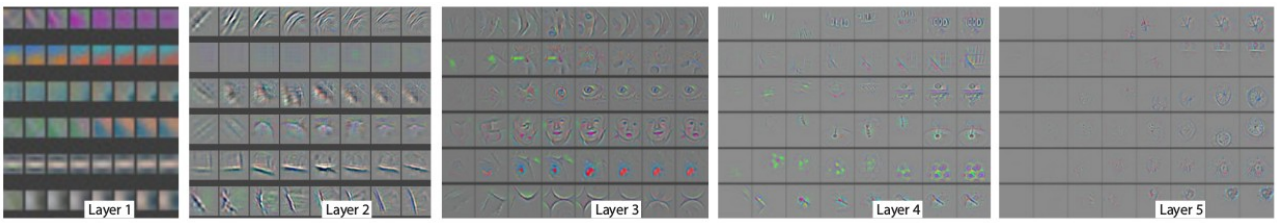


Figura 7: Evolución del entrenamiento de un modelo de características elegidas aleatoriamente a lo largo de 5 capas del modelo de la CNN. [Zeiler2014]

5 Softmax también es conocida como normalización exponencial [Fancourt2001].

2.4 Localización de objetos

Últimamente se ha escrito mucho sobre este tema, algunos como el proyecto DeCAF [Donahue2013] o R-CNN [Girshick2014], al igual que nosotros, emplean la CNN de Krizhevsky hasta la capa 5 como extractor de característica [Krizhevsky2012]. Tras la extracción de características se encuentran un abanico de distintos enfoques para intentar lograr la mejor localización posible.

Aunque no todos los artículos sobre este tema buscan proponer un nuevo diseño mejor que los anteriores, en algunos casos [Hoiem2012] analizan los errores que se producen en detección de la Bounding Box (BB)[6] de los objetos. Entre los errores de falsos positivos (FP), se encuentran el detectar un objeto en un lugar cuando en realidad no esta allí, estos autores destacan que se producen por lo general por dos motivo fundamentalmente. Por un lado, se puede confundir por tener un entorno similar como un pájaro o un avión, y por otro, por el parecido de los objetos en sí mismos, como por ejemplo un caballo y una vaca. Otro posible error si se busca la perfección del encuadre entre el BB encontrado y el etiquetado previamente en la imagen (esto suele ser parte de la información de la base de datos), es que haya mucha diferencia entre el encuadre que se ha encontrado y el que existía, como sería encontrar únicamente la cara de un gato y en vez de el gato completo. También analizan los falsos negativos (FN) aquellos objetos que están etiquetados, pero el sistema que se está probando no es capaz de encontrar. Sobre ello concluyen que los objetos pequeños o parcialmente ocultos son los más difíciles de localizar por lo general, aunque con excepciones como que si únicamente se ve la cara de un gato o una rueda de una bici, que resultan fáciles. Otro factor que puede dificultar el reconocimiento puede ser un punto de vista inusual del objeto, como podría ser un coche visto desde abajo.

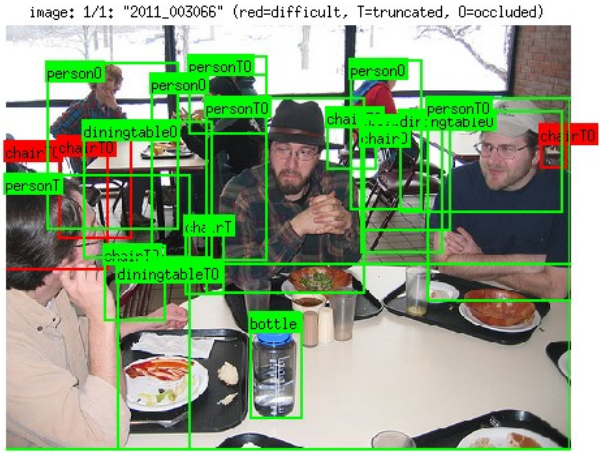


Figura 8: Ejemplo imagen etiquetada con BB.

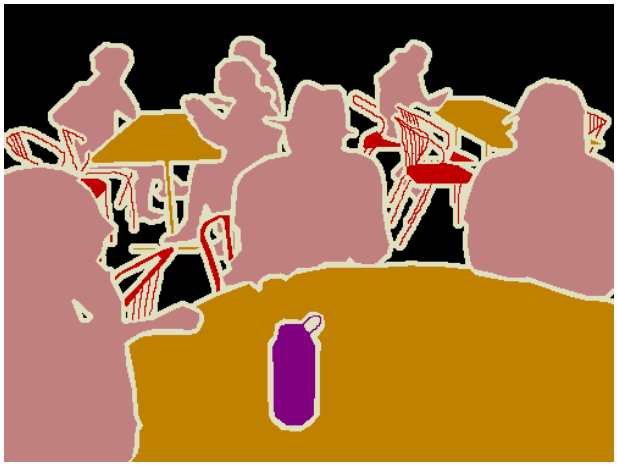


Figura 9: Ejemplo imagen etiquetada con "semantic segmentation".

Por lo general los sistemas buscan el BB de los objetos, pero esta información puede quedarse muy basta, pues no tiene en cuenta la forma real de los objetos, por ello se empieza a abrir un nuevo campo conocido como "*semantic segmentation*". En ella se busca más precisión sobre los objetos, esto aunque puede ser interesante tiene como desventaja que existen muy pocas imágenes etiquetadas con tanta precisión en las BBDD de entrenamiento, por lo que no se puede lograr tanta precisión. En la Figura 8 se muestra un ejemplo de un etiquetado de imagen con BB y otra con etiquetado de *semantic segmentation* en la Figura 9. En esta línea están desarrollados trabajos como el de Long, Shelhamer y Darrell [Long2015] que emplean en redes convolucionales para encontrar las posiciones y clases de los objetos.

6 Bounding Box: es el recuadro que enmarca a un objeto que se quiera identificar en una imagen. Se identifica con 4 valores, de los cuales 2 son el punto x_o, y_o con respecto al origen de la esquina superior izquierda de la imagen, y los otros 2 pueden ser o el ancho y el alto con respecto a ese punto, o bien, el punto inferior derecho con respecto al origen.

Un método que nos ha llamado la atención es R-CNN *(Regions with Convolutional Neural Networks features*) [Girshick2014], no sólo logra buenos resultados sino que en su segunda versión Fast R-CNN [Girshick2015] logra tiempos de análisis de cada imagen muy bajos. Los bloques generales que emplean son el extractor de características como el que hemos visto en el apartado 2.3.1 y SVM como red de clasificación. Para empezar, se recortan las regiones de las imágenes, después deforman dichas regiones a cuadradas, posteriormente se extraen las características de las distintas regiones. En la versión rápida lo que generan es un SVM truncado como detector. Este es el elemento clave con el que logran tiempos inferiores a los 300ms.

Otro de los artículos que resuelve el mismo problema es del diseño de OverFeat [Sermanet2013], este tiene como particularidades la forma de mezclar los predictores, puesto que entrenan un sistema que busca el mejor predictor comparándolo con la salida, y el hecho de reentrenar con los no-objetos, haciendo un nuevo entrenamiento con aquellos objetos que daban falsos positivos en los primeros resultados.

En algunos sistemas, no utilizan directamente la red ConvNet sino que la reentrenar con su propia base de datos antes de emplearla [Oquab2014]. En concreto el equipo de Oquab tras reentrenar ConvNet con sus 7 capas, introdujo al final otras 2 capas completamente nuevas. Con todo ello volvieron a entrenar, y lo emplearon para obtener mapas de calor que indicasen la probabilidad de que hubiese un determinado objeto en un punto de la imagen de entrada.

Por otro lado los esfuerzos de DeCAF (Deep Convolutional Activation Feature) [Donahue2013] se centran en estudiar las características para hacer que sean lo más separables posible. Lo que quieren comprobar es si la información de las características obtenidas de la CNN de ConvNet es generalizable para otras bases de datos. La conclusión es que sí que tienen capacidad de generalización y se pueden emplear en entornos distintos.

3 Desarrollo propio

Se ha trabajado sobre la herramienta MatLab, en varias versiones del programa. Lo que generó problemas al emplear algunas de las herramientas como el Toolbox de redes neuronales, en las cuales cambiaban parámetros por defecto en su generación entre versiones y provocaba que el sistema diseñado no funcionase en las versiones más nuevas.

Las funciones empleadas para la extracción de características fueron desarrolladas anteriormente dentro del proyecto *Memory Lane*.

A continuación, se va a describir el trabajo realizado siguiendo el esquema por bloques de funcionalidad del sistema descrito en la Figura 2.

3.1 Objetos

Como se ha visto hasta ahora se va a partir de la BBDD de Pascal VOC2007 y VOC2012. En VOC2007 todas las imágenes están etiquetadas y en VOC2012 hay dos grupos etiquetados "*training*" y "*validation*", pero aún quedan imágenes que se pueden incluir en un grupo de "*testing".* Este grupo lo dejamos para el final para poder comprobar de forma supervisada los resultados finales.

Cada imagen tiene anotaciones sobre que objetos contiene y la localización de los mismos. Esta información se concentra en los *Bouning Box (BB),* que son 4 valores que indican la esquina superior izquierda de la imagen y la inferior derecha. Se va a emplear esta información tanto en entrenamiento como en la posterior validación.

La extracción de características de ConvNet reduce drásticamente la dimensionalidad de ancho y alto de las imágenes. Para ello se emplean las primeras 5 capas de ConvNet que como se ve en la Figura 6 reduce de 224 a 6 la dimensión, lo cual en imágenes de baja resolución como las proporcionadas por Pascal imposibilita extraer información de objetos pequeños, como se verá en los resultados del apartado 4.2. Como primera medida para luchar contra este problema a falta de imágenes de alta calidad, se propone emplear las imágenes ampliadas. Otra de las opciones es cambiar el tamaño base de los objetos de 6x6x512 a otro menor que permita tener más bloques de características por cada objeto e imagen. La versión que se está empelando de ConvNet tiene profundidad de 512 en vez de 256 a la salida. Por el resto, el funcionamiento es idéntico al descrito en el apartado 2.3.1.

3.2 Entrenamiento

Se van a incluir los datos de las imágenes clasificados en Pascal como "*training*" y "*validation*". Se va trabajar sobre los objetos de esas imágenes para obtener un modelo de procesado de características y clasificador que se empleará posteriormente para las pruebas de validación con el conjunto de datos de *"test"*.

El proceso de entrenamiento va a requerir mucho tiempo, pero una vez entrenada la red ya no afecta, por ello no se le da importancia. Una medida de rendimiento que sí que va a tener relevancia por las limitaciones que supone es la ocupación en memoria de las operaciones, puesto que pueden provocar como mínimo tiempos de ejecución muy altos y en el peor de los casos que la máquina se quede sin memoria y no termine la ejecución, también se conoce como "*out of memory"*. En muchas ocasiones nos referiremos al concepto de "alto coste computacional" que aglutina los conceptos de que requiere muchos recursos de la máquina para ser procesado, como poner los procesadores al 100% o acabar con la memoria RAM disponible.

3.2.1 Preprocesado

Antes de empezar a trabajar hay que preparar los datos para facilitar la extracción de características. Al emplear como extractor de características la CNN de ConvNet [Krizhevsky2012] sabemos que es necesario tener imágenes de objetos cuadrados, como se ha visto en la sección 2.3, puesto que si no el sistema los deforma.

Extraer las características mediante la CNN tiene un alto coste computacional, por lo que se va a procurar que se ejecute el número mínimo de veces. Nuestro interés en el entrenamiento son los objetos contenidos en las imágenes, no las imágenes en sí mismas, pero por ahorro computacional se extraerán las características de la imagen completa y posteriormente se recortarán los objetos de las imágenes. Como las referencias de lo BB van a ser necesarias en el dominio de las características, nos preocupamos de transformarlas.

Para que los objetos sean cuadrados en el dominio de las características, pasamos los BB de rectangulares a cuadrados. En algunas imágenes eso supone que se va a superar el limite de las mismas, por ello se opta por incluir franjas de ruido blanco del tamaño necesario, como se muestra en la Figura 10. Se probó también a rellenar los bordes con negro, pero provocaba mal aprendizaje en la red neuronal puesto que recordaba las franjas negras como información útil y no lo era. Otra de las opciones además de rellenar, era deformar la imagen estirándola o contrayéndola para que fuese cuadrada, esto que podría funcionar para casos de clasificación, pero no sirven en localización. Por poner un ejemplo que sirvió de clave para la decisión del método: imágenes de cuchillos[7], si se deforman siempre de la misma forma tanto las de entrenamiento como las de test puede funcionar, pero si se busca un cuchillo deformado a cuadrado en una imagen sin deformar es poco probable que encontrarla.

Figura 10: Ejemplo imagen con ruido blanco en los bordes.

El método escogido para hacer cuadrado el objeto rectangular, fuerza a incluir información del contexto. Contra mayor sea la diferencia entre el ancho y el alto más información extra que la BB del objeto, puesto que sólo será ruido blanco si supera los limites de la imagen. Esta información extra supone cierto peligro, puesto que pueden distorsionar el entrenamiento. Se

7 A pesar de que no existe la categoría "chuchillos" en la base de datos de Pascal, si que lo hacía en la creada en un proyecto previo dentro del marco de *Memory Lane*.

valoró emplear un sistema que incluyese en los alrededores del BB rectangular información inventada (como ruido blanco), pero finalmente se optó porque resultaba más apropiado entrenar con información de contexto de un objeto que con información "inventada", puesto que la realidad implica que los objetos no están aislados.

Como se ha mencionado antes, la baja resolución de las imágenes de la BBDD de Pascal implican limitaciones para los objetos pequeños. Para intentar obtener mayor resolución en el dominio de las características se hacen varios intentos de entrenar el sistema con las imágenes ampliadas al doble de su tamaño original, aunque finalmente se descarta la idea por problemas de memoria RAM y cierres incontrolados de MatLab.

3.2.2 Extracción de las características

Para extraer características se ha empleado la red neuronal convolucional propuesta en Kizhevsky en 2012 [Krizhevsky2012], como se ha ido viendo hasta hay que tener en cuenta sus particularidades a la hora de tratar las imágenes. A pesar de ello, siguiendo la filosofía de Donahue [Donahue2013], tomamos la red como una caja negra, nos preocupamos únicamente de las entradas y salidas de la misma.

La entrada serán las imágenes que hemos preparado en el preprocesado con ruido blanco en los bordes en algunos casos ($NxMx3$) y la salida una versión proporcional de las mismas en una escala $nxmx512$, donde $n=N\cdot 6/224$ y $m=M\cdot 6/224$. En realidad no empleamos la red completa, únicamente hasta la quinta capa, indicada en la Figura 6, con profundidad de 512.

3.2.3 Procesado de las características

Tras extraer las características es necesario procesarlas antes de entrenar la red neuronal MLP que se quiere entrenar como clasificador. La red MLP se va a entrenar con los objetos, no con las imágenes completas, para ello primero hay que recortar los objetos contenidos en las imágenes. Para el entrenamiento de la red es necesario tener las mismas dimensiones en los datos. Por ello, cada uno de los objetos se reescala al mismo tamaño. El tamaño escogido fue 6x6, coincidiendo con el caso original [Krizhevsky2012]. Se probó a escoger tamaños menores con idea de que mejorase la detección de objetos pequeños, pero la mejora no existía, por ello se opto por emplear los valores originales.

Se emplea una técnica para aumentar el número de imágenes de cada clase y la capacidad de detección de los objetos a distintas escalas. Ésta consiste en recorrer cada uno de los objetos (con BB rectangular) en el dominio de las características con bloques del tamaño escogido (6x6x512), y quedarnos con esos datos como un objeto más. La técnica empleada para aumentar el número de ejemplos, no es exactamente la misma que empleada en ConvNet [Krizhevsky2012], aunque persigue los mismos objetivos de mejorar el entrenamiento de la red. Como se ha comentado previamente, puede haber otros objetos dentro del BB de un objeto mayor, o simplemente solapados, por ello los casos de objetos que coincida con el BB de otro no se incluyen.

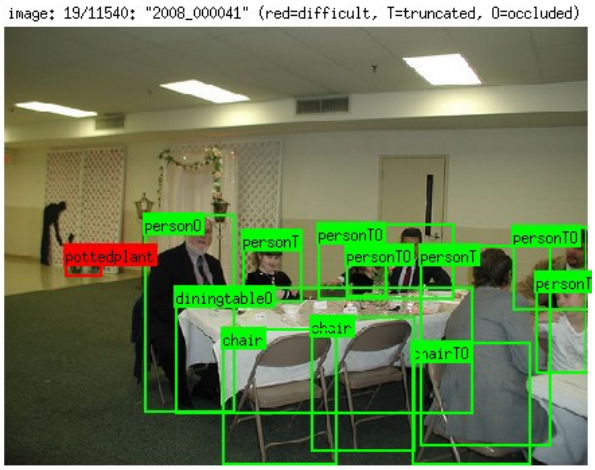


Figura 11: Ejemplo de escena con muchas etiquetas que se solapan.

Además de los objetos “extras” extraídos del objeto original, se reescala el objeto

original al tamaño de 6x6. Así de los objetos de mayor tamaño se obtendrán ejemplos de como es parcialmente, como podría ser la pata o el respaldo de una silla en vez de la silla completa, pero si esa silla coincide en espacio con una persona se ignora esa parte parcial del objeto.

Para recortar, hay que tener en cuenta que algunas imágenes tienen objetos identificados con otros encima, como se muestra en la Figura 11. En algunos casos es inevitable introducir información de otros objetos, como que una mesa tiene sillas alrededor o las sillas pueden tener personas encima. Como se ha comentado previamente no nos importa demasiado introducir contexto a los objetos, puesto que existirá en el mundo real. En los casos que haya solapamientos, se evita introducir los objetos "extra" de los que hablábamos en el párrafo anterior, puesto que se podría introducir una parte de una silla etiquetada como mesa, eso introduciría errores en la BBDD que se emplea para el entrenamiento.

Agrupar la información de los bloques de 6x6x512 resulta muy poco práctico, por lo que se opta por transformar cada uno de los bloques en un vector, y agrupar todos los vectores en una matriz. Tras ello, se quieren obtener las características principales. Para ello, se aplica PCA y LDA. Como se ha visto antes, LDA reduce el número de características al número de clases menos 1 [Welling2005]. Por lo que, las 18.432 características pasan a ser 19. Cómo la memoria física no permitía cargar en memoria una matriz con todos los ejemplos (unos 40.000) cada uno con sus 18.432 características se ideó un sistema por el cual se hacían bloques con las características y de ellas se obtenían las más relevantes, descartando así las que contenían menos información. Se agrupaban las características relevantes en una nueva matriz y se aplicaba PCA y LDA. Con ello se lograban unos resultados en reconocimiento entre el 98-99%. El problema de este sistema era el elevado coste computacional, esto provocaba errores de "*out off memory*" en algunos casos.

Por ello se planteó un nuevo sistema, el cual partía de quedarse únicamente con el máximo de cada uno de los bloques de 6x6, lo cual reduce el espacio de características de 18.432 a 512 de forma muy rápida y con un coste computacional muy bajo. Esta idea se probó con y sin emplear un segundo reductor de características tras esa fase inicial (LDA). Los resultados en la fase de reconocimiento empleando LDA eran bastante prometedores, 97-98% de aciertos como global. La gran ventaja de esta técnica era la ganancia en tiempo la fase de reducción de características que se había reducido de 4 horas a poco más de 10 minutos, por lo que se optó por emplear este método para las siguientes fases.

3.2.4 Clasificador

El clasificador escogido es una red MLP de 19 entradas, que corresponden al número de características tras la reducción LDA, y 20 salidas, que corresponden al número de clases en las que se quiere clasificar. Este tipo de redes cuentan con una capa oculta, a la cual originalmente se le asignaron 20 neuronas, pero como no lograba aprender toda la complejidad de la red se terminó aumentando. Tras varías pruebas se fijó finalmente en 50. Se puede ver la Figura 12 la red neuronal empleada. Existen distintos algoritmos de entrenamiento para la red, clasificados a grandes rasgos como lineales y de segundo orden, los de este orden pueden llegar a dar mejores resultados, aunque requiriendo costes computacionales altísimos que la máquina empleada no era capaz de soportar. Finalmente, el algoritmo empleado para entrenar la red fue RPROP[8], el cual fue presentado en 1993 por Riedmiller y Braun [Riedmiller1993] y que actualmente está considerado el más rápido en MatLab.

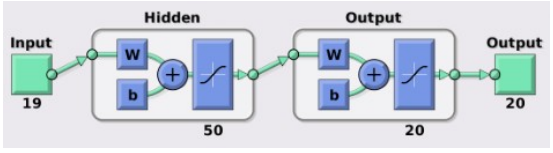


Figura 12: Estructura red neuronal empleada.

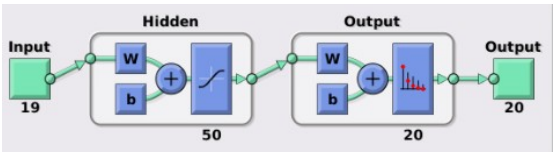


Figura 13: Estructura red neuronal diseñada por MatLab 2014a.

MalLab tiene herramientas internas para tratar con estas redes neuronales, lo cual es una ventaja y desventaja

simultáneamente, por un lado facilita mucho el diseño de la red pero por otro cambian el diseño por defecto del tipo de neuronas de salida. Por ejemplo, la versión de Matlab 2012b tiene neuronas sigmoideas en la capa intermedia y en la de salida, pero en la versión 2014a estas neuronas son de votación de apariciones como se muestra en la Figura 13. El tipo asignado por defecto no es fácil de controlar ni modificar. Por ello, para poder hacer los cálculos en una máquina más potente que sólo tenía la versión 2014a hubo que diseñar y guardar la red en una versión previa en otra máquina, guardar la red como variable y cargarla para entrenar en la máquina potente con la versión moderna de MatLab.

Otro tema delicado es el tema del número de ejemplos de cada clase, si la diferencia entre ellos es muy grande; se puede sobreaprender las clases muy numerosas y relegar a nada las poco representadas. Esto nos lleva a reducir el número de ejemplos de cada clase para que coincida con el mínimo. En este proceso se eligen primero los ejemplos de los objetos completos y no los parciales que comentábamos previamente. La desventaja de esto es que para que la red aprenda bien contra más ejemplos tenga mejor, por ello, se plantea una solución de compromiso, por la cual se permite que el número de ejemplos de cada clase pueda ser el doble que el de la que tenga con menos ejemplos. Esta solución de compromiso permite aumentar el número de ejemplos disponibles evitando que una ellas quede muy poco representada.

3.3 Test

En el test, el objetivo es otro, por lo que la forma de proceder va a ser algo distinta. Como en la fase de entrenamiento no se ha logrado que funcionen realmente ninguna de las técnicas para detección de objetos pequeños, se van a introducir las imágenes de test en 2 escalas distintas: una en la original y otra ampliada al doble. En este caso ya no sabemos donde están los objetos, por lo que se trabaja sobre las imágenes completas. En realidad si que están indicadas los BB de cada uno de los objetos, pero no vamos a emplear esa información hasta la fase de validación de los datos.

3.3.1 Procesado de las características

Se emplea nuevamente la CNN como caja negra, con la diferencia de que en este caso la empleamos 2 veces para la imagen en tamaño original y para la imagen con el doble del tamaño original. Por ahorrar el coste computacional que supone la segunda CNN de tamaño mayor se podría plantear aumentar la imagen en el dominio de las características, pero es un dominio poco conocido y no resulta muy fiable interpolar información en él, por ello se opta por aumentar la imagen a priori.

Tras obtener la imagen en el dominio de las características, se recortan objetos de tamaños 6x6 para poder pasarlos por la MLP.

Esto se hace para 3 escalas en el caso de la imagen del tamaño original y para 2 en el caso del tamaño doble. Una de las escalas en ambos casos es la que se ha obtenido a la salida de la CNN, la cual simplemente se recorre entera recortando los bloques de 6x6x512. La obtenida a partir de la imagen original se diezma de forma que su lado más estrecho es se diezma a 2 escalas. Una de ellas se diezma para obtener la imagen cuyo lado más corto sea 6, y se recorre en el otro eje para lograr los bloques de 6x6x512. El otro diezmado usa una escala intermedia entre el tamaño máximo de las imágenes en cada escala y el mínimo que es 6, varía en cada caso. Tras el diezmado se recorre para obtener los distintos bloques de 6x6x512.

Una vez se tienen todos los bloques se usa el mismo procesado de elegir el máximo de cada uno de los planos de 6x6, para reducir las características a 512 para cada caso. Tras ello se usa la matriz LDA obtenida en el entrenamiento, de forma que reducimos la dimensionalidad del espacio de características a 19.

3.3.2 Clasificador

Empleamos la red MLP que hemos entrenado. Para cada uno de los bloques nos devuelve un vector con 20 valores, cada uno de ellos corresponde a la probabilidad de que pertenezca una de las clases con valores comprendidos entre 0 y 1.

Los resultados para cada uno de los objetos la tenemos en forma de matriz de tamaño número de objetos por el número de clases, de ella queda excluida la información espacial de la ubicación y tamaño de cada uno de los objetos.

Esa información ha quedado guardada en otra variable, por lo que a continuación se va a proceder a construir un mapa de calor de forma que se agrupan ambas informaciones y sean visualizables de forma intuitiva.

3.3.3 Construcción de mapas de probabilidad

Se van a construir 20 mapas, uno para cada clase. Aunque en un primer momento se van a reconstruir 5 mapas para cada clase, uno por cada escala, después se redimensionan al tamaño original de la imagen, tras ello se combinan para obtener el mapa de calor de esa clase. Todo ello se explicará a continuación con más detalle.

Comenzando con los aspectos que hay que tener en consideración a la hora de reconstruir: todos los bloques son cuadrados y únicamente tenemos un valor para cada uno de ellos, por lo que reescalados a su tamaño original tendríamos planos monocromáticos. Intuitivamente se puede llegar a la conclusión de que sería más conveniente que hubiese un máximo de probabilidad de aparición del objeto en el centro del mismo y que ésta fuese bajando según se alcanzan los extremos. Esto ayuda a que exista un gradiente que permita distinguir objetos cercanos de la misma clase. Se puede plantear que esto provocaría que se generasen múltiples objetos pequeños donde solamente hay uno, en la realidad esto no llega a ocurrir por el alto solapamiento entre los objetos y porque se le da más peso a los valores máximos que a los mínimos encontrados. Ello provoca que si el objeto es muy grande, aunque en el extremo de algunos de los bloques estemos reduciendo su aportación, en alguno de los bloques cercanos esos píxeles estarán centrados, a ellos se les dará más peso al quedarnos con el máximo. En el caso de los objetos pequeños el hecho de quedarnos con los máximos les da la oportunidad de llegar a aparecer.

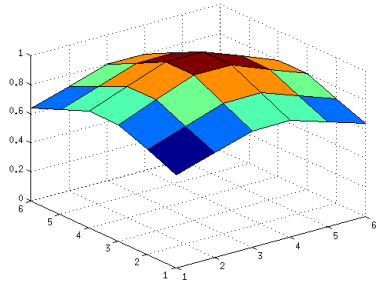


Figura 14: Matriz piramidal empleada como máscara para dar mayor peso al centro del bloque que a los extremos.

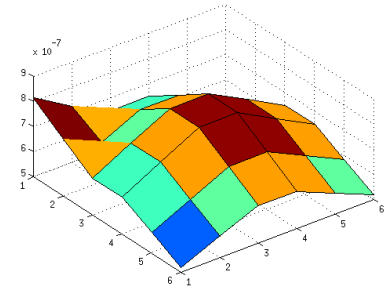


Figura 15: Ejemplo construcción mapa calor, mapa parcial.

Teniendo en cuenta las consideraciones vistas, se comienza a construir cada uno de los planos para cada escala, se reconstruyen en la escala en la que están de 6x6 y posteriormente se reescalan al tamaño de la imagen original. Esto ahorra conflictos de redondeos de píxeles, en caso de intentar reescalar primero cada uno de los bloques dan valores decimales difíciles de indicar como limites de píxeles, que provocan la necesidad de redondeo. El error de redondeo va a existir en todos los casos puesto que la diferencia entre escalas es de 224:6, pero contra menos veces se realice dicho redondeo menor, será su efecto.

A la salida del clasificador existe una matriz que tiene indicados los valores que corresponden a cada una de las posiciones y de la probabilidad entre 0 y 1. Cada uno de esos valores tenía un tamaño

de 6x6 en la entrada, y como hemos visto; se quiere reconstruir a esa escala. Por tanto se multiplican esos valores escalares por una matriz 6x6 que se pueda representar en un plano. Esta matriz no va a ser cuadrada y de valores constantes, puesto que sería poco realista, se va a construir de forma piramidal como la que se muestra en la Figura 14. La pirámide vale 1 en el máximo y 0,64 en la base. Se van recorriendo las posiciones y asignando los valores, aunque no directamente, puesto que existía superposición entre los valores, por ello, antes de asignar los valores de la matriz triangular (multiplicada por el valor de salida de MLP) se comparan píxel a píxel con los valores previos asignados (Figura 15), quedándonos así con los máximos en cada caso. Así se construye y obtiene un mapa como el que se muestra en la Figura 16.

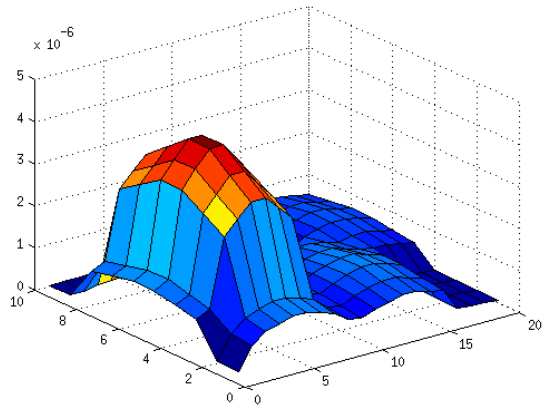


Figura 16: Ejemplo construcción mapa calor, mapa completo.

Las pirámides tiene como objetivos crear gradientes, para que un bloque con un valor muy alto no pueda eclipsar al resto, de esta forma si el objeto es más grande; los siguientes bloques también lo demostraran, y en caso de que no siga no será más grande de lo que corresponde.

Una vez construidos los 5 mapas en distintas escalas, los reescalamos al tamaño original de la imagen deshaciendo así el reescalado sufrido al pasar por la CNN y el de las escalas.

A la hora de combinar la información de las distintas escalas surgen problemas, alguna de ellas detectan información de objetos grandes, otras de objetos pequeños, nos interesa encontrar el máximo posible de ellos, pero las escalas que localizan objetos pequeños son más propensas a incluir falsos positivos (encontrar objetos que realmente no están ahí). La primera técnica que se probó fue emplear la información de los máximos, esto generaba muchísimos falsos positivos, porque en las escalas con bloques pequeños se encuentran con facilidad partes parciales de objetos. Empleando únicamente la información de la media los objetos pequeños que se encuentran sólo para una de las escalas, por ejemplo, quedan reducidas a nada, por lo que se perderían en la detección. Nuevamente se plantea una solución de compromiso, que incluya parte de ambas informaciones. Por ello se opta por obtener el máximo de todas las escalas y la media, y realizar otra media para combina dicha información. Como se ve en la siguientes expresión:

$$mapaCalorFinal(:,:) = \frac{max(mapaCalor,3) + media(mapaCalor,3)}{2}$$

Donde lo $mapaCalor$ es un tensor de $NxMx5$, y $mapaCalorFinal$ es una matriz de NxM, la cual corresponde con el tamaño de la imagen de entrada, las 5 capas de $mapaCalor$ corresponden a 5 mapas construidos para cada una de las escalas. De esa forma combinamos el máximo y la media de los mapas obtenidos para las distintas escalas.

3.3.4 Decisión

Una vez tenemos un mapa con valores entre 0 y 1 para cada uno de los píxeles, se puede representar para ver si se están detectando objetos en esa clase, como se ve en los mapas de colores de la Figura 18, Figura 20 y Figura 22. A partir de aquí hay que elegir un umbral de detección, es decir, a partir de que valor consideramos que lo que se ha encontrado es realmente un objeto. Se prueba con distintos valores entre 0,5 y 0,95, los resultados se van a mostrar para distintos umbrales.

Aplicado el umbral se emplea una función proporcionada por MatLab que permite localizar el BB de

cada una de las regiones contenidas. Se aplica a cada uno de los mapas de color para obtener los objetos de cada clase si es que existe alguno.

Con ello ya tenemos resultados, sabemos donde están los objetos localizados por nuestro sistema.

3.4 Validación resultados

Aunque para el sistema una vez en funcionamiento se acabaría en el apartado anterior, para esta fase todavía queda evaluar si los resultados obtenidos son fiables.

Para ello empleamos la información que se proporcionaba con las imágenes originalmente, y se va a buscar cual es la coincidencia entre los BB encontrados y los originales. Así se podrá evaluar como de bueno es el sistema diseñado.

Poniendo el caso de que en la imagen solamente hay un objeto de una cierta clase y se ha localizado, la validación es sencilla, se ve el área de solapamiento y si supera el 50% se da por valida la localización [Everingham2015]. En caso de encontrar un objeto de una clase que no está indicado entre los ejemplos también resulta sencillo clasificarlo como falso positivo, viceversa en caso de un objeto que existe pero no se encuentra ningún otro de esa clase, se trata de un falso negativo.

La dificultad llega cuando se encuentran múltiples objetos de una clase y también hay múltiples objetos de esa misma clase en la imagen original, hay que decidir si están bien o mal emparejados.

Finalmente se opto por evaluar si se encontraba o no el objeto etiquetado en el punto en el que el sistema lo encontraba. Empleando la siguiente relación:

$$resultado = max\left(\frac{areaObjEncontrado \cap areaObjEtiquetado}{areaObjEncontrado}, \frac{areaObjEncontrado \cap areaObjEtiquetado}{areaObjEtiquetado}\right)$$

Si el resultado de la misma es mayor de 0.5 se dará por buena dicha localización para el objeto.

Existen otros métodos de validar la localización de asignar un único objeto a una única localización, pero estos nos daban muchos problemas puesto que al no ser capaces de separar realmente los objetos de la misma clase se que están superpuestos, como se muestra en la Figura 17. Los resultados proporcionados por el otro sistema de validación nos daba resultados poco realistas sobre la bondad real del sistema. Se obtendría así:

$$resultado = \frac{areaObjEncontrado \cap areaObjEtiquetado}{areaObjEncontrado \cup areaObjEtiquetado}$$

Lograr resultados buenos con este sistema requeriría un nuevo rediseño del sistema completo de localización de los BB. Por ello se descartó dicho sistema por el momento, se plantea como trabajo futuro, algunas alternativas que mejorarían el sistema de localización el BB.

Las estadísticas que se van a emplear para la validación de los datos son la sensibilidad y la precisión o valor predictivo positivo [Sokolova2006]. Existen muchas otras formas de validar las relaciones entre aciertos y fallos de los sistemas de clasificación, pero los demás necesitan una casuística que no tenemos que son los Verdaderos Negativos, esos casos sería los “no objetos/no clase” que no tienen cabida en nuestro sistema. Por ello contamos con casos de:

- Verdaderos Positivos (VP): objetos etiquetados y encontrados.
- Falsos Positivos (FP): objetos encontrados pero no etiquetados.
- Falsos Negativos (FN): objetos etiquetados pero no encontrados.

$$senibilidad = \frac{VP}{VP + FN}$$

$$precisión = \frac{VP}{VP + FP}$$

No es sencillo interpretar ambas métricas conjuntamente, por ello se va a emplear también el valor F (*F-score*) [Sokolova2006], el cual nos proporciona esta información conjunta de la siguiente forma:

$$F = 2 \cdot \frac{precisión \cdot sensibilidad}{precisión + sensibilidad}$$

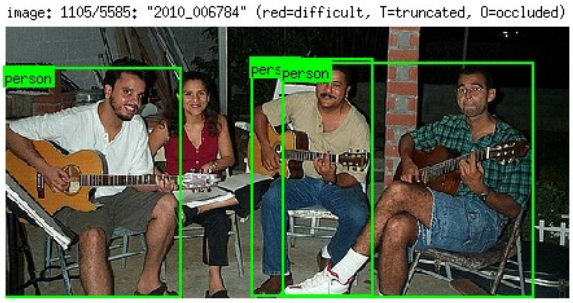


Figura 17: Ejemplo imagen con etiquetas de objetos solapadas.

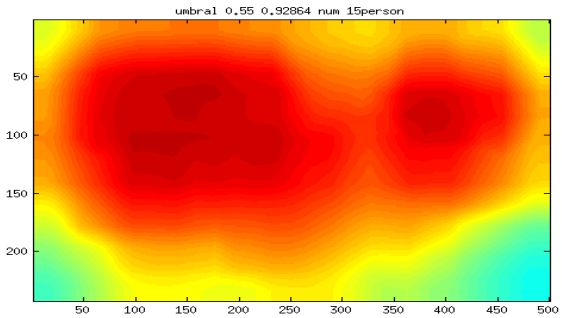


Figura 18: Ejemplo mapa calor de categoría "persona".

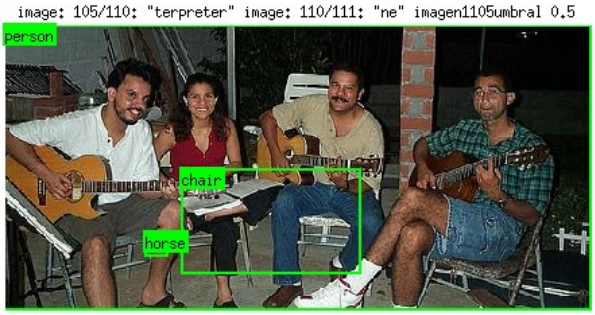


Figura 19: Ejemplo objetos encontrados con umbral 0,5.

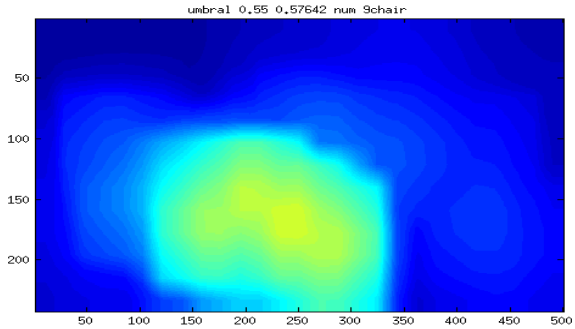


Figura 20: Ejemplo mapa calor de categoría "silla".

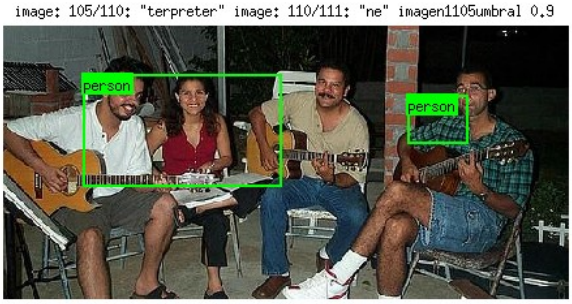


Figura 21: Ejemplo objetos encontrados con umbral 0,9.

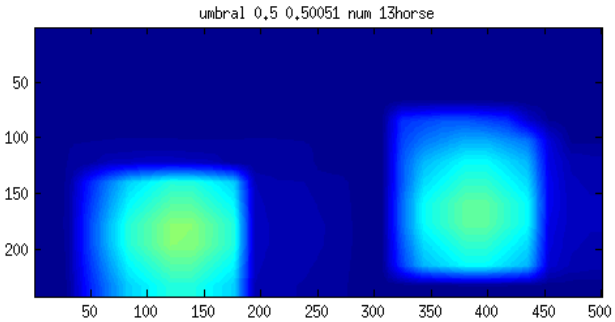


Figura 22: Ejemplo mapa calor de categoría "caballo".

4 Resultados propios y otros algoritmos

En este apartado se van a exponer los distintos resultados obtenidos, tanto tras el entrenamiento en clasificación, como tras pasar el test detección y tiempos de ejecución por imagen.

4.1 Clasificación del entrenamiento

Antes de lograr los resultados finales de localización de los objetos tenemos los resultados de cómo es la red neuronal. La red es la que tiene que determinar que objeto hay en la imagen, si es que hay alguno, en el caso de que no sea capaz. En la parte de clasificación es el modo fácil porque la imagen está centrada y es seguro que contiene algún objeto, si ni en esas condiciones óptimas se logran buenos resultados, va a ser prácticamente imposible lograrlos en la parte de localización.

4.1.1 Objetos de las imágenes de entrenamiento

En el caso del entrenamiento se han empleado objetos obtenidos de imágenes de entrenamiento y de validación, pero como se comenta en el apartado anterior, el número de imágenes de cada clase que se emplean no tienen porque estar balanceados. Lo recomendable para un buen entrenamiento es que para cada categoría el número de ejemplos sea similar, pero esto no siempre ocurre.

En la Tabla 2 tenemos la información de cuantos objetos participan en el entrenamiento de cada categoría. Como se ha comentado, la opción empleada es la de compromiso que emplea hasta 2 veces el doble del conjunto con menos ejemplos. Si no lo empleásemos correríamos el riesgo de que el conjunto no entrenase bien por causa de falta de ejemplos o por exceso en alguna de las categorías, ignorase otras.

4.1.2 Resultados de las imágenes de entrenamiento

Finalmente para el entrenamiento se han empleado objetos obtenidas de imágenes de entrenamiento y de validación, pero como se comenta en el apartado anterior el número de imágenes de cada clase que se emplean no tienen porque ser el mismo. Vemos distintos criterios de elección y cuales dan mejores resultados.

Tabla 2: Número de objetos por categoría para cada uno de los casos de selección de conjunto de entrenamiento.

Categorías	Nº categoría	Número de objetos		
		En el caso mínimo	Permitiendo hasta el doble del mínimo	Todos objetos
Avión	1	1256	2512	3401
Bicicleta	2	1256	2512	3723
Pájaro	3	1256	2512	4484
Barco	4	1256	1256	1775
Botella	5	1256	1256	1997
Autobús	6	1256	1256	1713
Coche	7	1256	2512	5195
Gato	8	1256	2512	13510
Silla	9	1256	2512	4347
Vaca	10	1256	1256	2074
Mesa comedor	11	1256	1256	1256
Perro	12	1256	2512	11152
Caballo	13	1256	2512	2755
Moto	14	1256	2512	3502
Persona	15	1256	2512	16394
Maceta	16	1256	2512	2673
Oveja	17	1256	1256	2190
Sofá	18	1256	1256	2172
Tren	19	1256	2512	2829
Monitor TV	20	1256	1256	1776
Total		25120	40192	88918

En la Figura 23 tenemos la matriz de confusión[9] de la salida del clasificador MLP, nos dice como de bien clasifica cada una de las categorías y a que clase van a parar los errores. La matriz tiene tamaño 21x21, donde las primeras 20 columnas corresponden a cada una de las clases esperadas, y las filas a las clases obtenidas. La diagonal muestra para cada clase que número de valores se ha clasificado correctamente y el porcentaje que corresponde sobre el total de casos. Los valores que la diagonal deja en la parte superior, corresponden a Falsos Negativos, valores que etiquetados originalmente no se han encontrado. Mientras que, la parte inferior de la diagonal corresponde a los Falsos Positivos, valores que se han encontrado y no deberían estar ahí. La primera y la última columna corresponden a valores resumen de porcentaje de aciertos y fallos para cada una de las clases. Así en la última columna, los porcentajes están calculados sobre total de objetos encontrados, mientras que, en la última fila los porcentajes se calculan sobre el total de objetos etiquetados realmente. Así el último valor de la tabla corresponde a la estadística global de los valores que han sido correctamente predichos.

Fijándonos en la última fila de la Figura 23, tenemos el conjunto de aciertos según la matriz de confusión es del 82,4%, eso quiere decir que casi el 18% de las veces fallamos con la clase que corresponde un objetos. Por categorías la que más errores tiene es la categoría 18, “sofá”, con casi un 40% de errores. De

9 http://es.mathworks.com/help/nnet/ref/plotconfusion.html

los cuales la mayor parte se están confundiendo con “silla” y con “persona”. La confusión es con objetos que fácilmente pueden estar en el contexto de la imagen [Hoiem2012], por lo que puede que el problema sea que al introducir la información extra que se añade por el hecho de que la CNN necesita objetos cuadrados y las botellas tienden a tener un BB muy rectangular.

El que mayor tasa de aciertos tiene es avión con un 93,4%. La mayoría de los errores se reparten en otro vehículos como trenes, autobuses o barcos, en especial los barcos, por lo que puede ser también por confusión del fondo azul del mar y del cielo. Por supuesto, algunos errores también se deben a confusiones con pájaros.

Tras muchas combinaciones estos fueron los mejores resultados que fuimos capaces de lograr, por lo tanto se siguió adelante a partir de ahí. Obviamente, lo ideal sería un acierto del 100% en estos casos para poder llegar a la fase de localización con mayor fiabilidad.

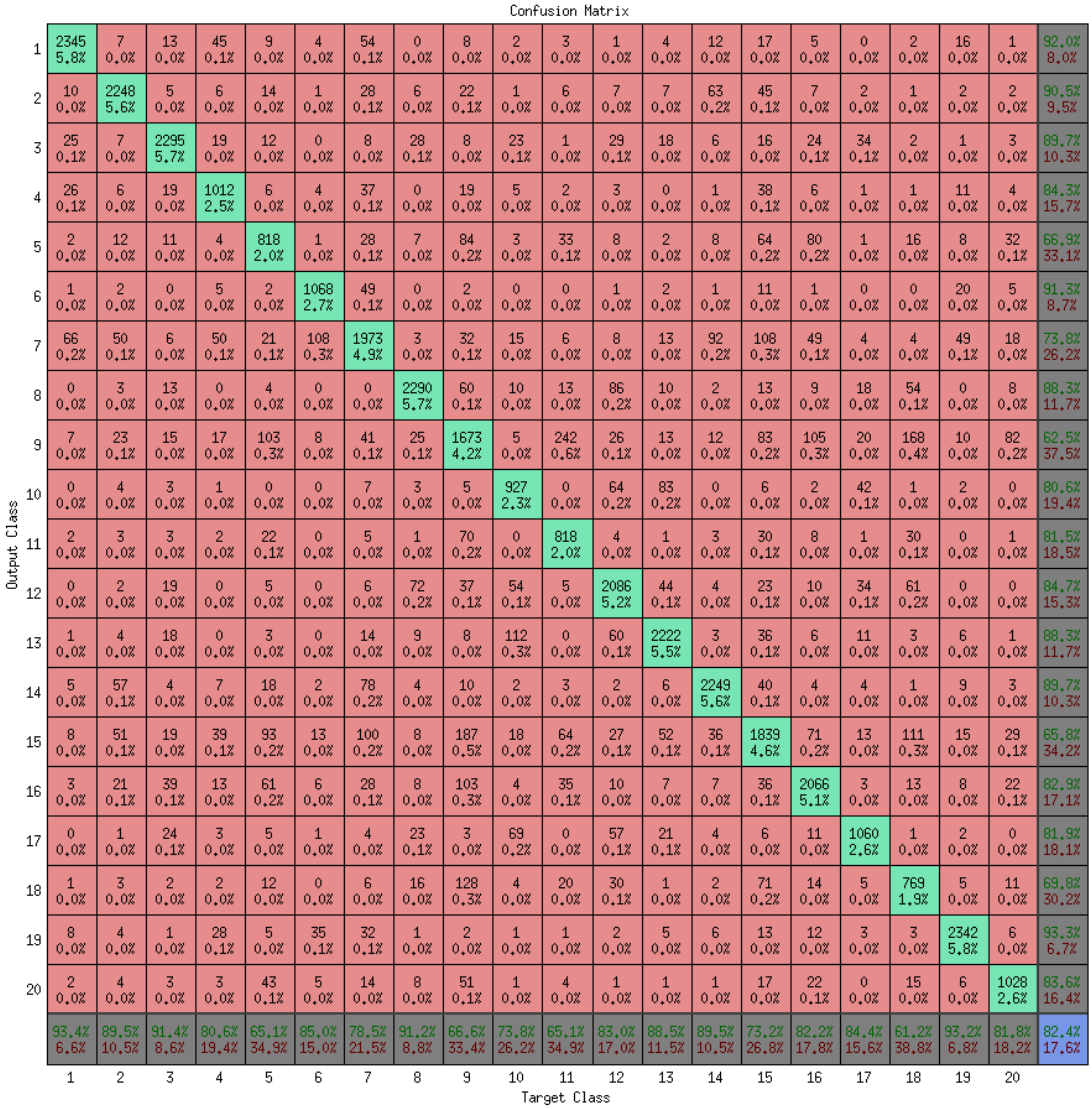


Figura 23: Matriz confusión empleando la técnica de objetos extra.

4.2 Salida test: localización

4.2.1 Objetos de las imágenes de test

Cabe destacar entre los objetos contenidos en las imágenes de test, que la mayoría de las etiquetas corresponden a la categoría “persona”. Además, con una diferencia muy significativa puesto que contiene 7272 personas de 8577 ejemplos totales, y la siguiente categoría más numerosa, es “silla” con 150. Como se puede observar en la Tabla 3, en la cual además se muestran los resultados de los objetos localizados y asignados correctamente.

Tabla 3: Objetos encontrados y asignados correctamente (VP) con distintos umbrales.

Categorías	Reales	Umbrales empleados									
		0.5	0.55	0.6	0.65	0.7	0.75	0.8	0.85	0.9	0.95
Avión	48	47	47	47	46	43	42	40	38	32	22
Bicicleta	47	41	41	41	40	36	36	35	32	24	15
Pájaro	50	46	46	45	44	43	42	38	37	34	27
Barco	60	56	56	54	50	50	48	44	38	26	16
Botella	79	45	43	37	31	24	20	13	5	1	0
Autobús	48	35	34	32	30	27	25	23	21	20	8
Coche	128	88	86	81	76	62	51	38	26	14	9
Gato	50	47	47	45	45	45	44	42	42	40	32
Silla	150	81	74	54	44	38	27	16	10	7	2
Vaca	69	53	51	42	40	32	28	19	16	9	1
Mesa comedor	53	33	29	28	27	23	22	21	15	11	9
Perro	57	57	51	47	42	40	38	33	28	26	20
Caballo	53	46	46	44	44	42	39	35	31	26	24
Moto	50	50	48	49	48	45	42	41	38	37	28
Persona	7272	5329	4969	4571	4133	3578	2892	2166	1436	650	101
Maceta	103	88	86	81	69	62	58	48	46	37	27
Oveja	90	80	78	78	75	72	68	64	58	48	38
Sofá	55	37	34	31	27	22	20	14	9	5	1
Tren	48	48	47	45	44	44	44	43	41	36	34
Monitor TV	67	60	59	53	47	46	44	40	36	32	26
Total	8577	6367	5972	5505	5002	4374	3630	2813	2003	1115	440

4.2.2 Resultados de localización

En la tabla anterior los resultados son absolutos, lo cual los hace precisos, pero difíciles de interpretar. Por ello, como se expuso en el apartado 3.4, se van a obtener otras medidas como la sensibilidad, la precisión y el valor-F que permitan evaluar mejor los resultados obtenidos.

La capacidad del sistema para encontrar objetos disminuye conforme se aumenta el umbral, estos datos, que se muestran en la Figura 24 en forma de la sensibilidad, se pueden interpretar como que al aumentar el umbral necesario para decidir que existe un objeto en esa posición; encontramos menos objetos y por tanto hay menos coincidencias. También hay que tener cuidado a la hora de interpretar los resultados puesto que la única clase con suficientes ejemplos es “persona”. Todas las demás tiene muy pocos ejemplo etiquetados. Como se observará, se han eliminado parte de los umbrales en la representación, esto se debe a que dificultaban mucho la interpretación de la gráfica por exceso de datos. Indicar que existe una tabla con los datos detallados en la Tabla 9 del anexo.

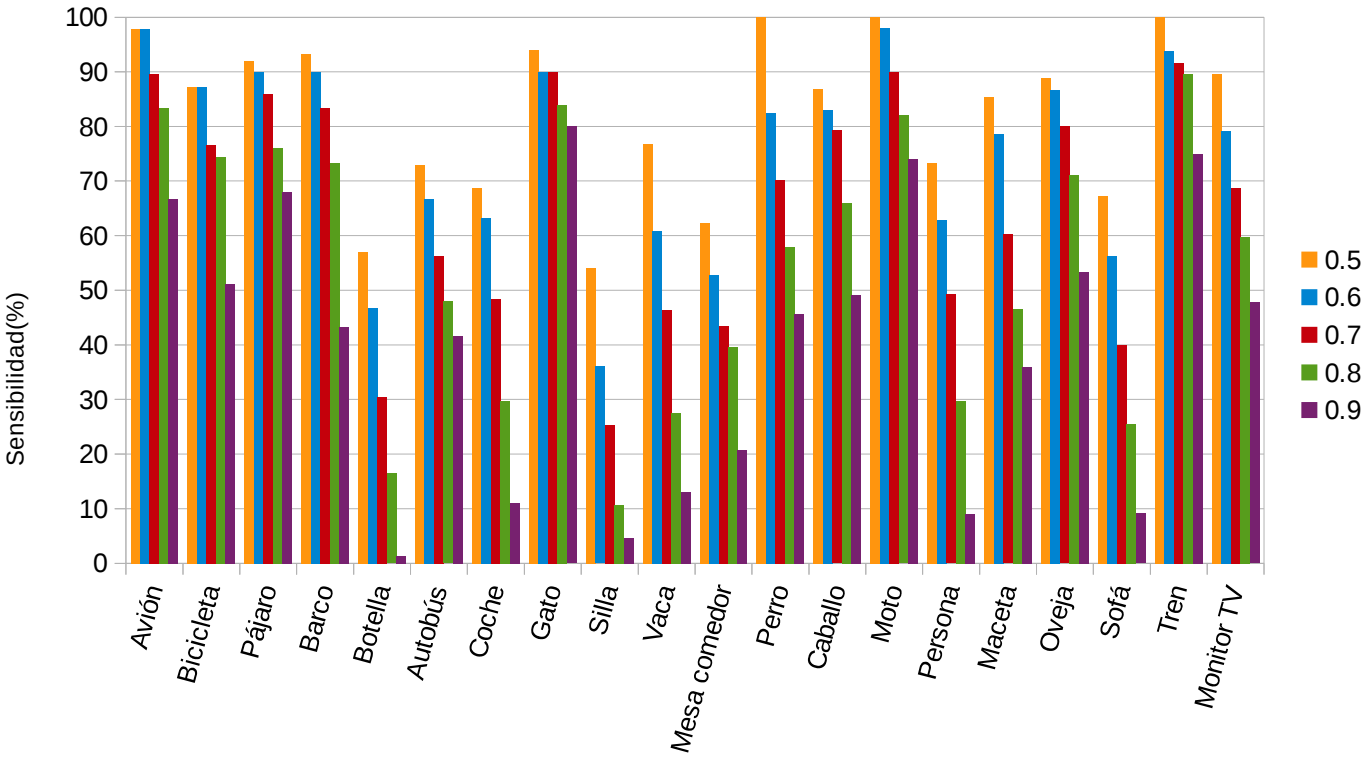


Figura 24: Gráfica de la sensibilidad de cada una de las clases para distintos umbrales.

Por ello, la sensibilidad nos dará información de la capacidad del sistema para encontrar los objetos etiquetados. Mientras que la precisión mostrará cuantos objetos estamos encontrando cuando no deberíamos por no estar etiquetados.

El hecho de encontrar más objetos también tiene su parte negativa y es que encontramos objetos que no están etiquetados, es decir, falsos positivos. Aunque posteriormente veremos casos en los que esto no termina de ser cierto, puesto que hay objetos no etiquetados que se están encontrando porque sí que están la imagen. Ciertamente, no es todos los casos de FP, y no se han observado con atención las 5585 imágenes que se están empleando en el test, únicamente poco más de cien.

El efecto negativo del exceso de falsos positivos se puede observar en la gráfica de la Figura 25, en la cual la precisión baja en los umbrales bajos, puesto que se están encontrando muchos objetos y la mayoría no corresponden con ninguno etiquetado.

Para poder aunar la información proporcionada por la sensibilidad y la precisión, que permite evaluar el efecto global tenemos en la Figura 26 el valor-F, esta medida nos permite apreciar que en conjunto un umbral más alto nos da mejores resultados.

Aunque esto no ocurre para todos los casos, en el caso de la categoría “persona” el valor-F disminuye conforme aumenta el umbral. Hay que tener en cuenta que la perdida de calidad es bastante menor que el aumento que sufren el resto de las categorías, por lo que en media siguen ganando los umbrales altos. Otro detalle es el hecho de que existan muchos más ejemplos de esa clase, como se ha mencionado previamente, lo cual hace que sea el valor más fiable estadísticamente y a la vez el único que proporciona un resultado que no concuerda con el resto.

En algunos de los casos con pocos ejemplos y muchos falsos positivos, cabe la posibilidad de que se deban a que están encontrando objetos que están allí pero no están etiquetados, como en el ejemplo la “silla” que encuentra en el ejemplo de la Figura 19 sin que halla una etiqueta como se ve en la Figura 17. Aunque como se muestra en la misma imagen, también se está etiquetando un caballo que no existe.

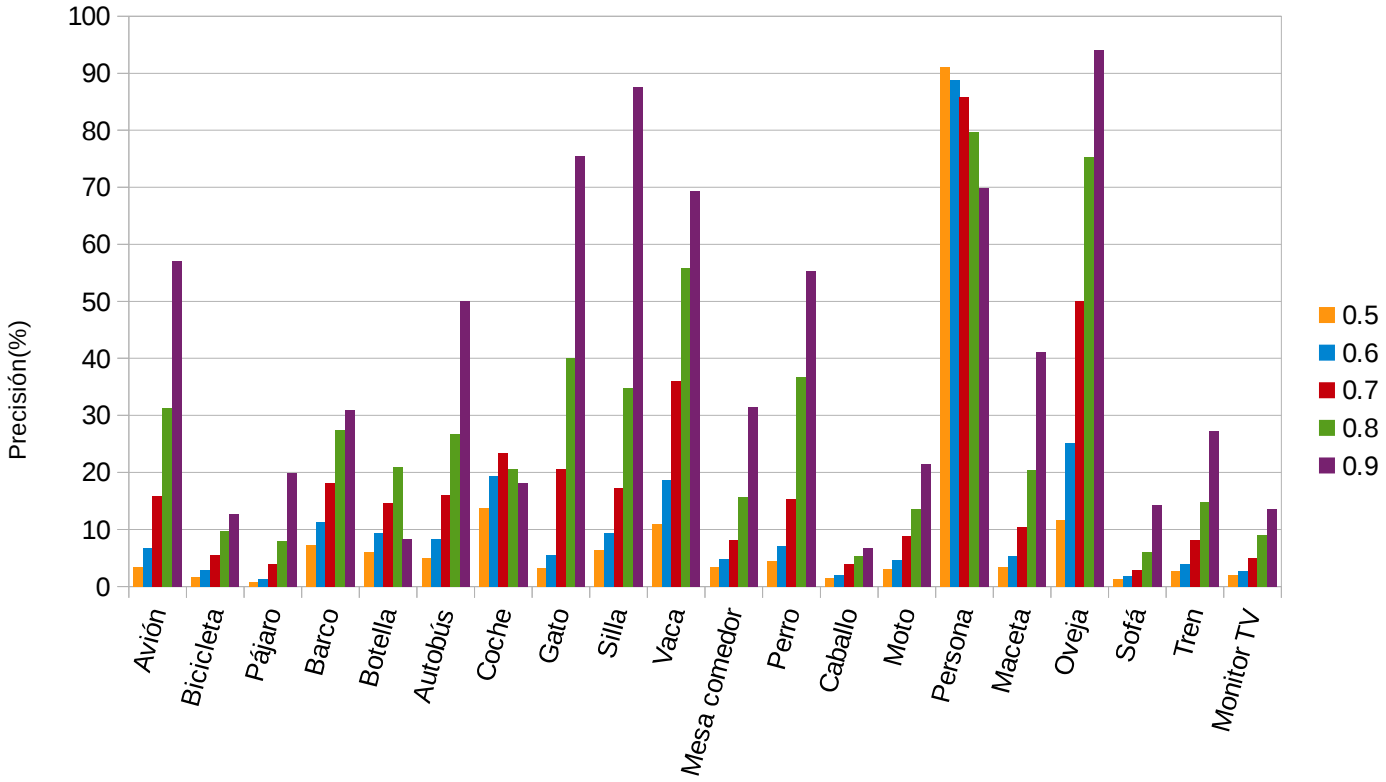


Figura 25: Gráfica de la precisión de cada una de las clases para distintos umbrales.

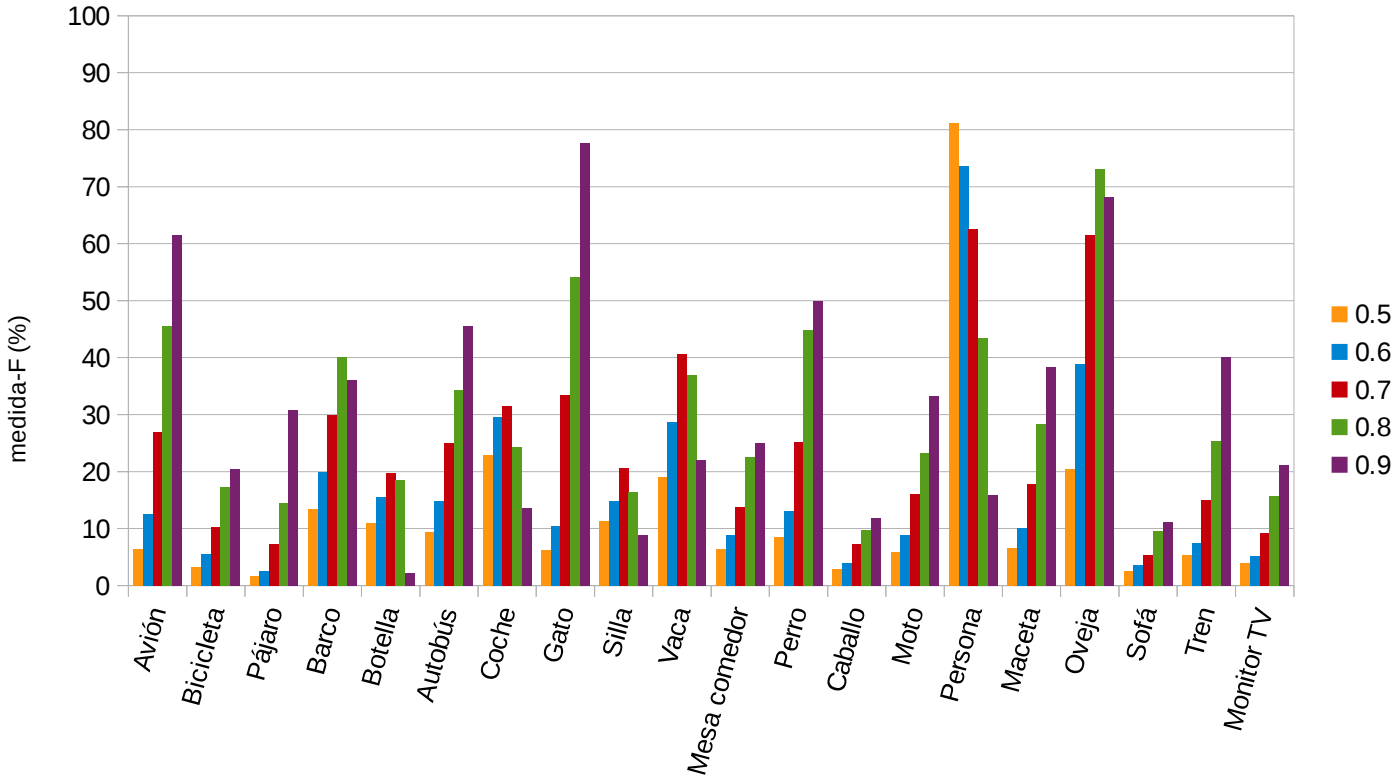


Figura 26: Gráfica de la medida-F de cada una de las clases para distintos umbrales.

4.2.3 Tiempos ejecución en test

El sistema tiene la capacidad de responder a qué objetos hay y dónde están en una imagen 7,72sg de media. Ciertamente esto queda lejos de lo que se puede considerar un sistema en tiempo real, el cual exige que se determine en tiempos inferiores a 1sg.

En la gráfica distribución del tiempo de ejecución para cada imagen de la Figura 27 se observa que la mayoría de la imágenes tardan entre 5 y 9 segundos.

Los tiempos no terminan de ser adecuados para lo que se está buscando, aunque bajo ciertas condiciones podrían serlo. Para empezar, los tiempos de ejecución mejorarían mucho si pasásemos el código de MatLab a C. MatLab resulta muy cómodo para prototipado, pero es muy lento. Además en C se podrían paralelizar fácilmente parte de los procesados lo cual también permitiría mejorar mucho los tiempos de ejecución por imagen. De hecho, para poder emplear el sistema en un entorno real, es un requisito indispensable la migración de lenguaje de programación. Una vez aplicados esos cambios seguramente se podrían alcanzar los tiempos de ejecución por imagen deseables.

Por comparar con otros sistemas, de los que se han mencionado en el apartado 2.4, R-CNN en su primera versión [Girshick2014], funcionado sobre CPU[10] lograba tiempos de ejecución por imagen de 53 segundos. Empleando la GPU[11], logran bajar los tiempos a 13sg/imagen. En la segunda versión, Fast R-CNN, logran lo que sería nuestro objetivo a lograr de tiempos entre 200 y 320ms/imagen dependiendo del tamaño de la imagen.

10 CPU: Central *Processing Unit*
11 GPU: *Graphics Processing Unit*

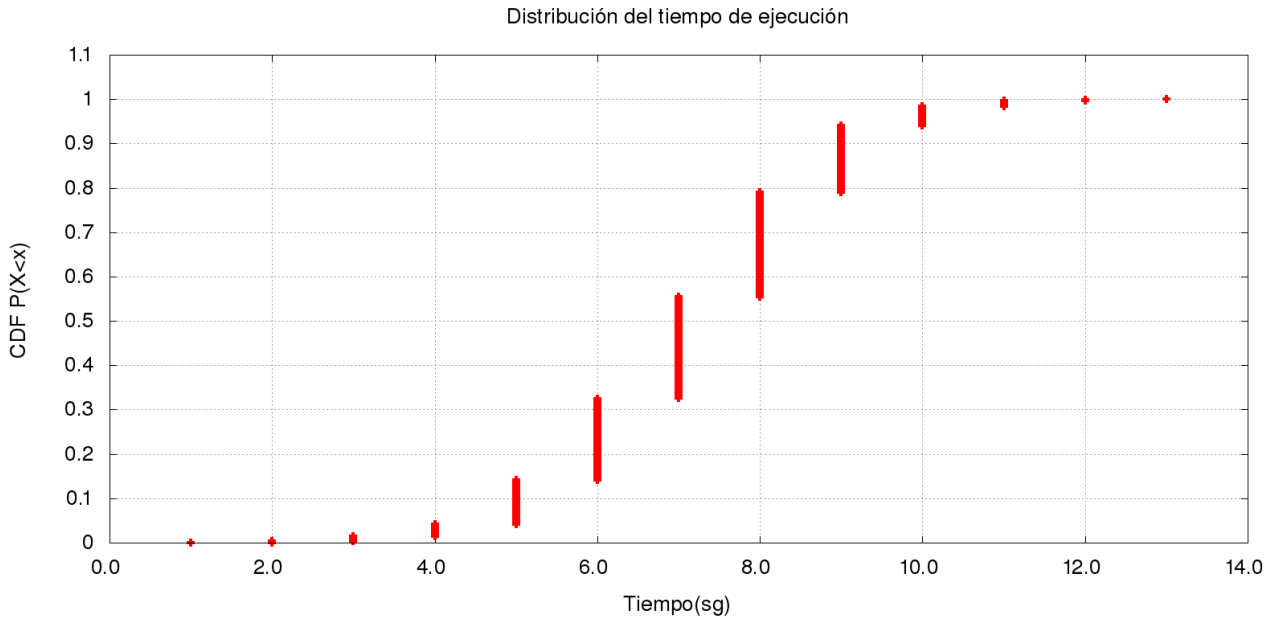


Figura 27: Distribución del tiempo de ejecución para cada imagen de test para un único umbral.

5 Conclusiones y líneas futuras

Este trabajo se puede enfocar desde dos puntos de vista, por un lado el del objetivo final con aplicaciones en el campo de la asistencia a domicilio, y por otro, la puramente técnica de sistemas de reconocimiento de objetos en imágenes. Por ello, las conclusiones pueden abordarse desde ambos.

Partiendo del punto de vista puramente técnico, para el caso con más representación de ejemplos en el test se han logrado unos resultados prometedores al identificar que objetos había. Aunque no se haya logrado encuadrar adecuadamente el *Bounding Box* (BB), lo cual podría mejorarse. En cuanto a los casos con menos representación de ejemplos habría que probarlos con una base de datos mayor para poder extraer conclusiones.

Una de las mejoras que podrían probarse, para determinar con mayor precisión BB, sería emplear una red neuronal entrenada con los mapas de calor obtenidos y teniendo como objetivos de salida ejemplos de BB esperadas. Puesto que la técnica que se ha empleado de determinar el BB, un umbral, encuentra demasiados resultados o muy pocos, sin que se pueda hacer mucho por optimizar la detección de los objetos correctos y reducir los falsos positivos.

Respecto al objetivo de los tiempos por imagen, a falta de implementar el sistema en un lenguaje de programación funcional, parece que nos hemos quedado cerca del objetivo. Aunque como el sistema aun requiere mejoras antes de poder implementarse, habría que procurar que las incorporaciones al algoritmo tuviesen el mínimo coste computacional posible.

Hay que tener en cuenta que el problema propuesto en este trabajo no está cerrado, a lo largo de su desarrollo se han tomado decisiones sobre al implementarlo que no son la única solución viable. En algunos casos, las opciones que se probaron quedaron completamente descartadas, en otras ocasiones, no hubo tiempo suficiente para valorar otros cambios y probar todas las alternativas viables.

Una de estas posibles variantes es cambiar la red neuronal única del clasificador, que necesita ser reentrenada cada vez que se quiera añadir una nueva clase, por varias redes neuronales más sencilla. Cada una de estas redes estaría dedicada exclusivamente a identificar una de las clases, por lo que tendría una única salida, simplificándola enormemente. La otra ventaja sería que facilitaría la escalabilidad en caso de que se quieran añadir a posteriori nuevas clases, puesto que no obligaría a reentrenar todo el sistema.

Desde el punto de vista asistencial, con este proyecto se ha dado un paso adelante, para que en un futuro, pueda existir un dispositivo con la capacidad de recordarnos dónde se quedaron las gafas o cómo solíamos hacer nuestra receta favorita, que hemos olvidado a causa del alzhéimer o la demencia.

6 Referencias

[Chatfield2014] Chatfield, K., Simonyan, K., Vedaldi, A., & Zisserman, A. (2014). Return of the devil in the details: Delving deep into convolutional nets. arXiv preprint arXiv:1405.3531.

[Cover1965] Cover, T. M. (1965). Geometrical and statistical properties of systems of linear inequalities with applications in pattern recognition. IEEE transactions on electronic computers, (3), 326-334.

[Cybenko1989] Cybenko, G. (1989). Approximation by superpositions of a sigmoidal function.Mathematics of control, signals and systems, 2(4), 303-314.

[Donahue2013] Donahue, J., Jia, Y., Vinyals, O., Hoffman, J., Zhang, N., Tzeng, E., & Darrell, T. (2013). Decaf: A deep convolutional activation feature for generic visual recognition. arXiv preprint arXiv:1310.1531.

[Everingham2015] Everingham, M., Eslami, S. A., Van Gool, L., Williams, C. K., Winn, J., & Zisserman, A. (2015). The pascal visual object classes challenge: A retrospective. International Journal of Computer Vision, 111(1), 98-136.

[Fancourt2001] Fancourt, C. L., & Principe, J. C. (2001). Optimization in companion search spaces: The case of cross-entropy and the Levenberg-Marquardt algorithm. InAcoustics, Speech, and Signal Processing, 2001. Proceedings.(ICASSP'01). 2001 IEEE International Conference on (Vol. 2, pp. 1317-1320). IEEE.

[Fu1976] Fu, K. S., & Rosenfeld, A. (1976). Pattern recognition and image processing.IEEE transactions on computers, 25(12), 1336-1346.

[Girshick2014] Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. InProceedings of the IEEE conference on computer vision and pattern recognition(pp. 580-587).

[Girshick2015] Girshick, R. (2015). Fast r-cnn. In Proceedings of the IEEE International Conference on Computer Vision (pp. 1440-1448).

[Guyon2008] Guyon, I., Gunn, S., Nikravesh, M., & Zadeh, L. A. (Eds.). (2008). Feature extraction: foundations and applications (Vol. 207). Springer.

[Hoiem2012] Hoiem, D., Chodpathumwan, Y., & Dai, Q. (2012). Diagnosing error in object detectors. In Computer Vision–ECCV 2012 (pp. 340-353). Springer Berlin Heidelberg.

[Jia2014] Jia, Y., Shelhamer, E., Donahue, J., Karayev, S., Long, J., Girshick, R., ... & Darrell, T. (2014, November). Caffe: Convolutional architecture for fast feature embedding. In Proceedings of the ACM International Conference on Multimedia(pp. 675-678). ACM.

[Krizhevsky2012] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In Advances in neural information processing systems (pp. 1097-1105).

[Long2015] Long, J., Shelhamer, E., & Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 3431-3440).

[Martín2006] Martín, B., & Sanz, A. (2006). Redes neuronales y sistemas borrosos.

[Nagi2011] Nagi, J., Ducatelle, F., Di Caro, G. A., Cireşan, D., Meier, U., Giusti, A., ... & Gambardella, L. M. (2011, November). Max-pooling convolutional neural networks for vision-based hand gesture recognition. In Signal and Image Processing Applications (ICSIPA), 2011 IEEE International Conference on (pp. 342-347). IEEE.

[Oquab2014] Oquab, M., Bottou, L., Laptev, I., & Sivic, J. (2014). Learning and transferring mid-level image representations using convolutional neural networks. InProceedings of the IEEE Conference on Computer Vision and Pattern Recognition (pp. 1717-1724).

[Riedmiller1993] Riedmiller, M., and H. Braun (1993). "A direct adaptive method for faster backpropagation learning: The RPROP algorithm," Proceedings of the IEEE International Conference on Neural Networks, pp. 586–591.

[Sermanet2013] Sermanet, P., Eigen, D., Zhang, X., Mathieu, M., Fergus, R., & LeCun, Y. (2013). Overfeat: Integrated recognition, localization and detection using convolutional networks. arXiv preprint arXiv:1312.6229.

[Shlens2014] Shlens, J. (2014). A tutorial on principal component analysis. arXiv preprint arXiv:1404.1100.

[Sokolova2006] Sokolova, M., Japkowicz, N., & Szpakowicz, S. (2006, December). Beyond accuracy, F-score and ROC: a family of discriminant measures for performance evaluation. In Australasian Joint Conference on Artificial Intelligence (pp. 1015-1021). Springer Berlin Heidelberg.

[Stark1986] Stark, H., & Woods, J. W. (1986). Probability, random processes, and estimation theory for engineers. Englewood Cliffs: Prentice Hall, 1986, 1.

[Wang2013] Wang, X., Yang, M., Zhu, S., & Lin, Y. (2013). Regionlets for generic object detection. In Proceedings of the IEEE International Conference on Computer Vision (pp. 17-24).

[Welling2005] Welling, M. (2005). Fisher linear discriminant analysis. Department of Computer Science, University of Toronto, 3, 1-4.

[Zeiler2014] Zeiler, M. D., & Fergus, R. (2014). Visualizing and understanding convolutional networks. In Computer vision–ECCV 2014 (pp. 818-833). Springer International Publishing.

7 Apéndice: Abreviaturas

Tabla 4: Índice abreviaturas.

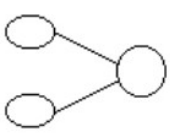
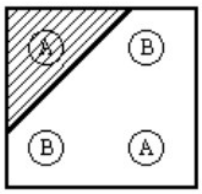
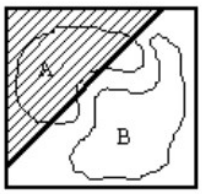
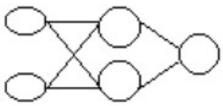
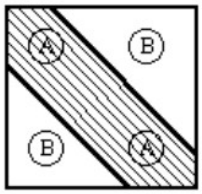
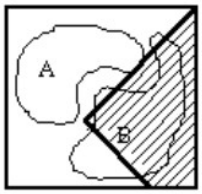
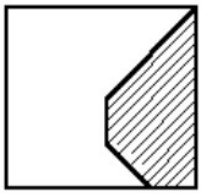
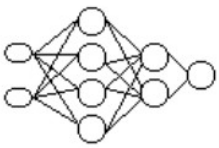
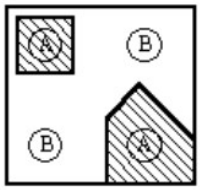
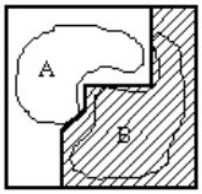
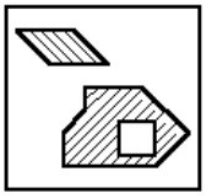
Abreviatura	Significado
BBDD	Bases de Datos
CNN	*Convolutional Neural Networks*
CPU	*Central Processing Unit*
DeCAF	*Deep Convolutional Activation Feature*
FN	Falso Negativo
FP	Falso Positivo
GPU	*Graphics Processing Unit*
LDA	*Linear Discriminant Analysis*
MLP	*Muylti Layer Perceptron*
PCA	*Principal Component Analysis*
RAM	*Random Access Memory*
R-CNN	*Regions with Convolutional Neural Networks features*
Rprop	*Resilient backProp*
SVM	*Support Vector Machine*
ReLUs	*Rectified Linear Units*
VN	Verdadero Negativo
VP	Verdadero Positivos
VOC	*Visual Object Classes*

8 Anexo: Complemento del estado del arte

8.1 Redes Neuronales

Siguiendo con el tema de las redes neuronales, nos centramos en las posibilidades de la arquitectura. Como se observar en la Tabla 5, una arquitectura simple puede resolver problemas simples, según sube la complejidad de la red, puede llegar a resolver problemas de mayor complejidad. Como se verá a continuación, la arquitectura del perceptrón simple, únicamente permite resolver problemas lineales.

Tabla 5: Arquitectura redes y su capacidad de clasificación.[Martín2006]

Arquitectura	Región de decisión	Ejemplo 1: XOR	Ejemplo 2: clasificación	Regiones más generales
Sin capa oculta 	Hiperplano (dos regiones)			
Una capa oculta 	Regiones polinomiales convexas			
Dos capas ocultas 	Regiones arbitrarias			

8.1.1 Perceptrón multicapa (MLP)

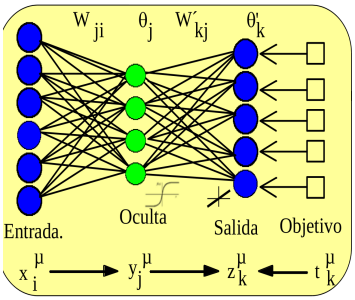


Figura 28: Modelo MLP [Martín2006].

Se define como un aproximador universal de funciones [Cybenko1989] porque permite modelar cualquier tipo de función, su uso más extendido es en clasificación. Sus salidas tienen menos limitaciones, pueden ser continuas o binarias, aunque se suelen emplear las binarias en los problemas de clasificación.

No hay un limite de capas ocultas, con una sola capa es completamente funcional, pero con más: el número de neuronas necesarias para modelar el mismo problema desciende. Las neuronas de las capas intermedias son de forma sigmoidea (ver tabla de la Figura 4).

Dado un patrón de entrada x_i^{μ} se puede expresar la red neuronal de la Figura 28 como: $z_i^{\mu} = \sum_j w'_{kj} f\left(\sum_i w_{ji} x_i^{\mu} - \theta_j\right) - \theta'_k$

El algoritmo de aprendizaje básico es *"back propagation"* (BP), en el cual se calcula w'_{kj} antes que w_{ji}. El objetivo de este algoritmo es reducir el error cuadrático entre la entrada y la salida. Se puede encontrar la descripción matemática de este algoritmo en el capitulo 2.5.2 de la referencia [Martín2006]. Este algoritmo es el más sencillo pero tiene algunas desventajas como son que se puede quedar atascado en mínimos locales y que es muy lento. Existen algunas técnicas de acelerar el entrenamiento, aunque existe la opción de emplear otros algoritmos de aprendizaje. Mencionaremos alguno de ellos en el apartado 8.1.3.

En cualquier caso resulta complejo comprender que ha aprendido la red, es como una caja negra.

8.1.2 Maquina de Vector Soporte (SVM)

Las SVM son más recientes, fueron propuestas en 1999 por Vapnik, V. en la publicación de "*An Overview of Statistical Learning Theory*".

Su aplicación más extendida es la clasificación. Su mayor ventaja con respecto a MLP, es que no sufre de mínimos locales, además de encontrar automáticamente el número óptimo de neuronas.

Los problemas que plantean estos sistemas tienen una resolución muy complejos puesto que hay que resolver funciones cuadráticas. Otro problema es la determinación de las funciones de kernel, que en el caso de MLP son sigmoideas.

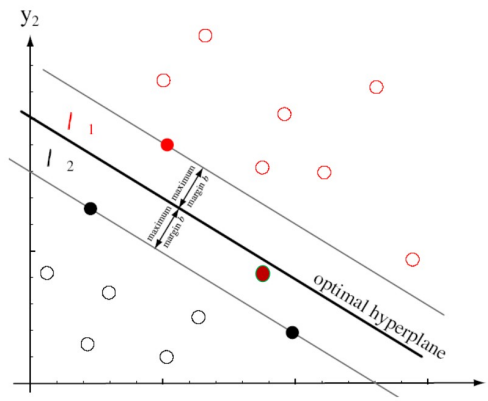


Figura 29: Ejemplo de separación de clases por un hiperplano de SVM.[Martín2006]

El algoritmo de entrenamiento de SVM busca el hiperplano equidistante entre las distintas clases. Los vectores de la base de datos que están más cerca del hiperplano se llaman "*support vectors*" y son los que definen la posición del hiperplano. Para determinarlos se buscan los elementos más cercanos a la frontera.

El aprendizaje se resuelve mediante el multiplicador de Lagrange α para encontrar el hiperplano que separa la clase. Los α_i que se quedan lejos de la frontera se van a cero ($\alpha_i = 0$) y los que están cerca de la frontera se quedan con $\alpha_i > 0$, como se muestra en el ejemplo de la Figura 29. Lo que significa que nos quedamos con 3 neuronas en el ejemplo porque se quedan

3 con $\alpha \neq 0$, se queda con el mínimo margen entre las 2 clases. El algoritmo es muy complejo de implementar, por suerte muchos sistemas, entre ellos MatLab ya lo conocen.

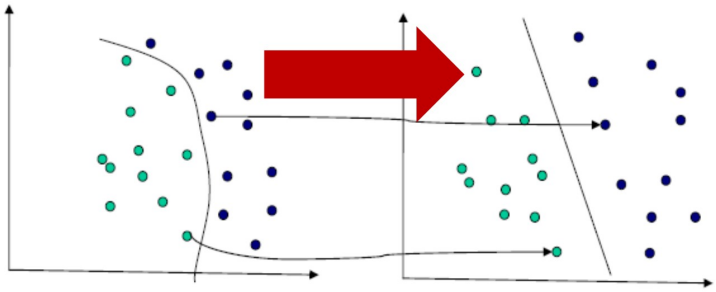


Figura 30: Transformación de Kernel.[Martín2006]

El algoritmo de SVM sabe trabajar con problemas lineales, para los casos en los que estos problemas no son lineales (la mayoría de los que existen), se emplean Transformaciones de Kernel (como la que se muestra en la Figura 30) que permiten convertir el problema en lineal. Esto provoca que se puedan ver otros sistemas de Redes Neuronales como un caso particular, entre ellos MLP.

Las transformaciones de Kernel, aplican el teorema de Cover [Cover1965] que recomienda elegir una transformación tal que el problema sea lineal en el nuevo espacio transformado. Esto provoca el aumento de la dimensionalidad del problema, pero al tratarse de un problema lineal es más fácil de abordar y resolver, puesto que los cálculos necesarios son más simples lo que hace que las clases sean más fáciles de separar.

8.1.3 Entrenamiento de la redes MLP

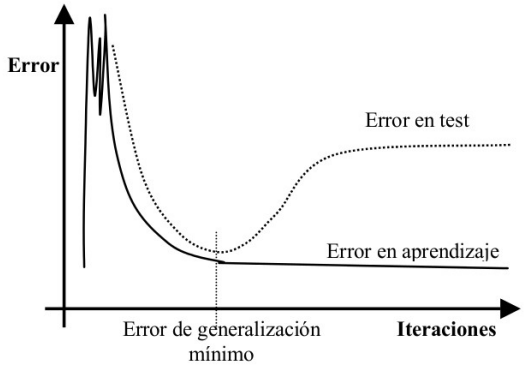


Figura 31: Efecto del error cuando se produce sobreajuste.[Martín2006]

De entre los algoritmos de entrenamiento vamos a destacar dos de los que se intentaron emplear en el sistema diseñado. El primero fue un sistema de segundo orden, el cual es considerado uno de los mejores para aproximación funcional, Levenberg-Marquardt, su mayor desventaja es el altísimo coste de memoria (fue el motivo por el que hubo que descartarlo).

El segundo algoritmo, es una de la variante más rápida de *backpropagation* (al menos entre las implementadas en MatLab), se conoce como *Resilient backProp* (Rpro) fue presentado en 1993 [Riedmiller1993].

En el entrenamiento de las redes neuronales se da un fenómeno conocido como *overfitting*, que viene a ser un sobreajuste o sobreentrenamiento. Éste consiste en que en un punto de las iteraciones que ajustan el peso de las neuronas, los pesos empiezan a ajustarse demasiado a los ejemplos concretos que se están empleando para entrenar. Se dice entonces que la red ha perdido capacidad de generalización. Esto se traduce en que el error de test comienza a crecer a pesar de que el error del aprendizaje siga disminuyendo. Se puede observar este efecto en la Figura 31.

Para visualizar con más claridad en que consiste el efecto de sobreentrenamiento en la Figura 32 se muestran tres gráficas de ejemplo con las distintas fases del entrenamiento. En ellas el punto rojo representa el ejemplo de test, mientras que las “X” representan los ejemplos de entrenamiento. La línea superpuesta no está indicando que es lo que saben los pesos de las neuronas.

En realidad el *overfitting* únicamente se da en el caso de no tener suficientes ejemplos para evitar el efecto, esto supone tener unos 10 ejemplos(p) por cada peso(w): $p \approx 10 \cdot w$. Como esto no siempre es viable, existen otros métodos para evitar este efecto. El que se va a emplear es uno conocido como “*early stopping*” que consiste en dividir los ejemplos disponibles en 3 grupos: unos para entrenamiento, otros para validar que no se está produciendo sobreentrenamiento, y parar el entrenamiento en caso de que ocurra, y el último para test, que permitirá evaluar si la red esta bien entrenada.

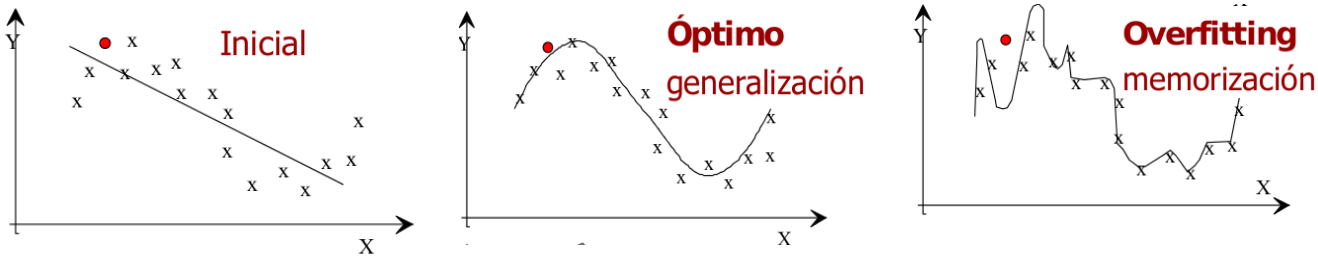


Figura 32: Fases del aprendizaje de una red neuronal [Martín2006]

Es muy importante que estos grupos no se mezclen, ya que no se puede emplear para entrenar y para test el mismo ejemplo, porque se estaría perdiendo lo que se quiere mirar que es la capacidad de generalización.

8.2 Métodos de reducción de características

Se trata de métodos que permiten reducir el espacio de características de forma que se facilite la resolución de los problemas matemáticos al tener unas dimensiones menores. También, pueden servir para extraer o agrupar la información más relevante de un espacio de características. Para poder discriminar correctamente las distintas clases en las que están agrupadas las características.

Poniendo un ejemplo, si se quiere modelar el ambiente de una habitación, se pueden tomar medidas de: temperatura, iluminación, viento, puerta abierta o cerrada, ventanas abiertas o cerradas, número de ocupantes, programación del aire acondicionado, etc. Todas esas medidas van a generar un espacio de características con muchas dimensiones, pero en realidad, parte de esas características son dependientes entre sí, la temperatura tendrá relación con que esté haciendo, el aire acondicionado y si la ventana esta abierta, incluso tendrá relación con el número de ocupantes de la habitación. Así que, no es descabellado pensar, que se pueden reducir esas características para obtener unas nuevas que modelen de forma más sencilla el ambiente de una sala.

Se van a ver dos técnicas distintas que se emplean para la reducción de características. Una es PCA (*Principal Components Analysis*) y la otra LDA *(Linear Discriminant Analysis*). La primera agrupa la información relacionada, ordenando así de más relevantes a menos, eso permite seleccionar, por ejemplo, aquellas característica que contengan el 95% de la información, ignorando las demás. La segunda técnica logra un espacio de características linealmente independiente, esto provoca que la reducción de características sea más drástica, no tiene perdida de información, pero la agrupación de la misma tiene criterios menos claros.

8.2.1 Principal Component Analysis (PCA)

El análisis PCA, que también se conoce como Expansión de Karhunen-Loêve [Stark1986], consiste en transformar un número de variables posiblemente correlacionadas en un número de variables igual o inferior, no correlacionadas (ortogonales) a las cuales se denomina componentes principales.

Este método es utilizado para reducir la dimensionalidad de los objetos, que se representan con componentes en cada una de las dimensiones de los espacios de características. Como sería el punto en un plano con sus componentes x e y. Con el método se logra la proyección que mejor representa los datos en el sentido de mínimos cuadrados, y elimina aquellas combinaciones lineales que presentan menor varianza, manteniendo aquellas que aportan mayor cantidad de información.

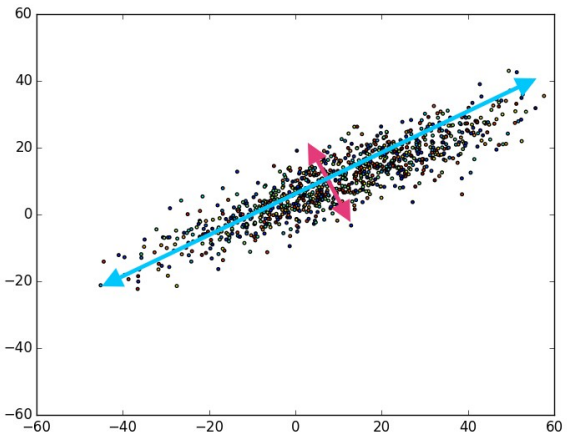


Figura 33: Ejemplo análisis PCA.

En la Figura 33 se observa la presentación de las dos primeras componentes tras una transformación PCA. En ella observamos la componente principal, la más larga y representada en azul, y la secundaria más corta y representada en rosa.

Al calcular PCA, se cumplen varios supuestos: que el problema tiene una base lineal, que las componentes principales que se van a obtener son ortogonales y que hay mucha varianza en los datos [Shlens2014].

Para el cálculo de PCA se puede emplear el algoritmo SVD (*Singular Value Decomposition*) implementado en MatLab.

8.2.2 *Linear Discriminant Analysis* (LDA)

En análisis LDA o Discirminante Linear de Fisher [Welling2005] emplea un proceso similar al análisis PCA. La diferencia consiste en que PCA buscaba las direcciones que mejor representaban las características de los objetos, sin tener en cuenta a que clase pertenecían estos objetos. En cambio, LDA buscaba las direcciones que mejor discriminan las diferentes clases a las que pertenecen los objetos de entrenamiento. Por eso mediante el empleo de LDA se obtienen mejores resultados a la hora de separar las clases.

LDA es capaz de encontrar el subespacio de proyección en el cual las muestras están más separadas. Por ello se emplea mucho en reconocimiento puesto que se esfuerza en discriminar mejor las clases. Al emplear LDA se está suponiendo que las muestras tienen una distribución gaussina. Consiste en transformaciones lineales, en las cuales se puede distinguir una matriz de variabilidad (*within-class* S_w) y otra de dispersión (*between-class* S_b).

El calculo de LDA no esta implementado en MatLab, por lo que a continuación se va a describir la forma de cálculo que emplea el algoritmo que lo calcula.

8.2.2.1 *Algoritmo LDA empleado*

Desde el punto de vista matemático, se define un conjunto de vectores (los ejemplos) $\vec{x}_i$ con $i=1,\ldots,N$, cada uno de ellos posee una dimensión $Mx1$. A su vez se normalizan estos vectores a media cero y varianza unidad. Esta es la información que se utilizará para calcular la transformación, empleando los siguientes pasos:

1. Cálculo del objeto medio total: $\vec{\mu}_x=\frac{1}{N}\sum_{i=1}^{N}\vec{x}_i$; y el objeto medio de cada clase $\vec{\mu}_c=\frac{1}{n_c}\sum_{i=1}^{n_c}\vec{x_{c,i}}$. Siendo n_c el número de elementos de la clase c, y $x_{c,i}$ el i-ésimo elemento de la clase c.
2. Obtención de la matriz de varianza inter-clases, que define la varianza entre las medias de las clases: $S_b=\sum_{i=1}^{C} n_c(\vec{\mu}_c-\vec{\mu}_x)(\vec{\mu}_c-\vec{\mu}_x)^T$.

3. Cálculo de la matriz de varianza intra-clases, que define la varianza dentro de cada clase:

$$S_w = \sum_{c=1}^{C} \sum_{i=1}^{n_c} (\vec{y_{c,i}} - \vec{\mu_c})(\vec{y_{c,i}} - \vec{\mu_c})^T$$

4. El método finaliza realizando la búsqueda de la transformación lineal U que maximiza el ratio:

$$max \frac{\left| U^T S_b U \right|}{\left| U^T S_w U \right|} = max \frac{\left| \widetilde{S_b} \right|}{\left| \widetilde{S_w} \right|}$$

Una aproximación apropiada viene dada por la matriz $U = (\vec{u_1}, \vec{u_2}, \ldots, \vec{u_p})$ cuyas columnas son los vectores propios de la $S_w^{-1} S_b$. Se puede expresar la relación entre las matrices como: $S_b \vec{u_k} = \lambda_k S_w \vec{u_k}$ donde k es el número de vectores propios, que como mucho puede coincidir con $C-1$ porque es el rango máximo de S_b; λ_k corresponden a los valores propios del sistema.

8.2.3 Perceptrón simple

Ideado por Rosenblatt en 1959, el perceptrón simple solamente tiene dos capas, la de entrada y la de salida. La neuronas que emplea son de tipo escalón, por lo que sus salidas únicamente pueden tomar los valores 0 o 1. Su objetivo es la clasificación, pero tiene como limitación que únicamente puede tratar problemas linealmente separables. La solución que fue introducir más capas para poder tratar problemas no lineales, se verá en el siguiente apartado.

El modelo de aprendizaje de los pesos del perceptrón tiene la forma $\Delta w_{ij}^{\mu}(t) = \varepsilon \cdot (t_i^{\mu} - y_i^{\mu}) \cdot x_j^{\mu}$ si se consideran entradas y salidas digitales de -1 y +1 queda como:

$$\Delta w_{ij}^{\mu}(t) = \begin{cases} 0 \ , \ si \ \ t_i^{\mu} = y_i^{\mu} \\ 2 \cdot \varepsilon \cdot t_i^{\mu} \cdot x_j^{\mu} \ , \ si \ \ t_i^{\mu} \neq y_i^{\mu} \end{cases}$$

El aprendizaje se detiene cuando todos los patrones están bien clasificados. Rosemblatt demostró que convergía para problemas linealmente separables.

9 Anexo: Más resultados

Tabla 6: Todos los objetos encontrados según los distintos umbrales.

Categorías	Reales	Umbrales empleados									
		0.5	0.55	0.6	0.65	0.7	0.75	0.8	0.85	0.9	0.95
Avión	34	48	1404	1116	701	438	271	182	128	87	56
Bicicleta	32	47	2443	2003	1457	996	660	479	358	286	189
Pájaro	41	50	5623	4750	3474	2213	1121	701	473	313	171
Barco	39	60	775	619	482	356	275	216	160	126	84
Botella	48	79	743	576	398	273	165	115	62	38	12
Autobús	25	48	700	577	384	259	168	120	86	51	40
Coche	42	128	640	519	419	336	266	215	184	146	77
Gato	28	50	1463	1155	809	466	219	138	105	74	53
Silla	54	150	1279	917	579	360	219	107	46	17	8
Vaca	40	69	487	356	224	144	89	58	34	24	13
Mesa comedor	7	53	981	773	583	407	280	199	134	82	35
Perro	36	57	1275	957	659	406	261	165	90	61	47
Caballo	33	53	3253	2819	2184	1577	1091	833	666	520	387
Moto	30	50	1641	1370	1053	760	513	371	302	241	172
Persona	6580	7272	5848	5526	5153	4717	4170	3480	2716	1915	931
Maceta	58	103	2584	2039	1509	973	595	379	236	154	90
Oveja	42	90	692	482	311	200	144	116	85	68	51
Sofá	10	55	2852	2306	1679	1116	770	431	235	101	35
Tren	18	48	1767	1499	1151	806	537	402	291	203	132
Monitor TV	42	67	2973	2536	1985	1459	935	680	441	332	237
Total	7239	8577	39423	32895	25194	18262	12749	9387	6832	4839	2820

Tabla 7: Objetos encontrados y no asignados (FP) según los distintos umbrale.

Categorías	Reales	Umbrales empleados									
		0.5	0.55	0.6	0.65	0.7	0.75	0.8	0.85	0.9	0.95
Avión	48	1357	1069	654	392	228	140	88	49	24	7
Bicicleta	47	2402	1962	1416	956	624	443	323	254	165	109
Pájaro	50	5577	4704	3429	2169	1078	659	435	276	137	47
Barco	60	719	563	428	306	225	168	116	88	58	24
Botella	79	698	533	361	242	141	95	49	33	11	1
Autobús	48	665	543	352	229	141	95	63	30	20	9
Coche	128	552	433	338	260	204	164	146	120	63	35
Gato	50	1416	1108	764	421	174	94	63	32	13	4
Silla	150	1198	843	525	316	181	80	30	7	1	0
Vaca	69	434	305	182	104	57	30	15	8	4	2
Mesa comedor	53	948	744	555	380	257	177	113	67	24	11
Perro	57	1218	906	612	364	221	127	57	33	21	8
Caballo	53	3207	2773	2140	1533	1049	794	631	489	361	279
Moto	50	1591	1322	1004	712	468	329	261	203	135	86
Persona	7272	519	557	582	584	592	588	550	479	281	50
Maceta	103	2496	1953	1428	904	533	321	188	108	53	5
Oveja	90	612	404	233	125	72	48	21	10	3	3
Sofá	55	2815	2272	1648	1089	748	411	221	92	30	10
Tren	48	1719	1452	1106	762	493	358	248	162	96	53
Monitor TV	67	2913	2477	1932	1412	889	636	401	296	205	122
Total	8577	33056	26923	19689	13260	8375	5757	4019	2836	1705	865

Tabla 8: Objetos etiquetados pero no encontrados (FN) según distintos umbrales.

Categorías	Reales	Umbrales empleados									
		0.5	0.55	0.6	0.65	0.7	0.75	0.8	0.85	0.9	0.95
Avión	48	1	1	1	2	5	6	8	10	16	26
Bicicleta	47	6	6	6	7	11	11	12	15	23	32
Pájaro	50	4	4	5	6	7	8	12	13	16	23
Barco	60	4	4	6	10	10	12	16	22	34	44
Botella	79	34	36	42	48	55	59	66	74	78	79
Autobús	48	13	14	16	18	21	23	25	27	28	40
Coche	128	40	42	47	52	66	77	90	102	114	119
Gato	50	3	3	5	5	5	6	8	8	10	18
Silla	150	69	76	96	106	112	123	134	140	143	148
Vaca	69	16	18	27	29	37	41	50	53	60	68
Mesa comedor	53	20	24	25	26	30	31	32	38	42	44
Perro	57	0	6	10	15	17	19	24	29	31	37
Caballo	53	7	7	9	9	11	14	18	22	27	29
Moto	50	0	2	1	2	5	8	9	12	13	22
Persona	7272	1943	2303	2701	3139	3694	4380	5106	5836	6622	7171
Maceta	103	15	17	22	34	41	45	55	57	66	76
Oveja	90	10	12	12	15	18	22	26	32	42	52
Sofá	55	18	21	24	28	33	35	41	46	50	54
Tren	48	0	1	3	4	4	4	5	7	12	14
Monitor TV	67	7	8	14	20	21	23	27	31	35	41
Total	8577	2210	2605	3072	3575	4203	4947	5764	6574	7462	8137

Tabla 9: Sensibilidad según los distintos umbrales empleados.

Categorías	Umbrales empleados									
	0.5	0.55	0.6	0.65	0.7	0.75	0.8	0.85	0.9	0.95
Avión	91,67	87,50	85,42	85,42	83,33	79,17	75,00	66,67	56,25	39,58
Bicicleta	87,23	85,11	82,98	76,60	70,21	55,32	53,19	40,43	19,15	12,77
Pájaro	92,00	90,00	88,00	88,00	86,00	82,00	80,00	68,00	68,00	58,00
Barco	86,67	81,67	75,00	73,33	61,67	50,00	48,33	41,67	33,33	21,67
Botella	60,76	56,96	50,63	46,84	39,24	30,38	25,32	20,25	11,39	6,33
Autobús	64,58	64,58	56,25	47,92	45,83	43,75	41,67	37,50	27,08	18,75
Coche	69,53	67,19	60,16	49,22	42,97	36,72	25,00	18,75	16,41	10,16
Gato	94,00	92,00	90,00	88,00	90,00	90,00	86,00	78,00	66,00	58,00
Silla	56,67	52,67	42,00	32,00	24,00	20,67	16,00	9,33	6,67	3,33
Vaca	65,22	55,07	53,62	46,38	34,78	30,43	23,19	15,94	5,80	2,90
Mesa comedor	49,06	49,06	45,28	39,62	35,85	28,30	20,75	15,09	11,32	9,43
Perro	96,49	92,98	82,46	75,44	71,93	64,91	57,89	50,88	49,12	31,58
Caballo	81,13	79,25	71,70	71,70	60,38	52,83	47,17	41,51	28,30	18,87
Moto	94,00	94,00	92,00	90,00	82,00	74,00	70,00	68,00	52,00	28,00
Persona	95,71	94,21	92,42	89,51	85,74	80,86	74,33	64,00	48,34	20,26
Maceta	82,52	78,64	63,11	57,28	53,40	48,54	42,72	37,86	33,01	22,33
Oveja	91,11	88,89	85,56	83,33	76,67	70,00	65,56	58,89	55,56	30,00
Sofá	41,82	36,36	32,73	21,82	18,18	12,73	7,27	5,45	1,82	0,00
Tren	100,00	100,00	100,00	97,92	97,92	95,83	85,42	77,08	70,83	58,33
Monitor TV	76,12	71,64	68,66	65,67	61,19	53,73	53,73	53,73	46,27	35,82

Tabla 10: Precisión según los distintos umbrales empleados.

Categorías	Umbrales empleados									
	0.5	0.55	0.6	0.65	0.7	0.75	0.8	0.85	0.9	0.95
Avión	3,35	4,21	6,70	10,50	15,87	23,08	31,25	43,68	57,14	75,86
Bicicleta	1,68	2,05	2,81	4,02	5,45	7,52	9,78	11,19	12,70	12,10
Pájaro	0,82	0,97	1,30	1,99	3,84	5,99	8,03	11,82	19,88	36,49
Barco	7,23	9,05	11,20	14,04	18,18	22,22	27,50	30,16	30,95	40,00
Botella	6,06	7,47	9,30	11,36	14,55	17,39	20,97	13,16	8,33	0,00
Autobús	5,00	5,89	8,33	11,58	16,07	20,83	26,74	41,18	50,00	47,06
Coche	13,75	16,57	19,33	22,62	23,31	23,72	20,65	17,81	18,18	20,45
Gato	3,21	4,07	5,56	9,66	20,55	31,88	40,00	56,76	75,47	88,89
Silla	6,33	8,07	9,33	12,22	17,35	25,23	34,78	58,82	87,50	100,00
Vaca	10,88	14,33	18,75	27,78	35,96	48,28	55,88	66,67	69,23	33,33
Mesa comedor	3,36	3,75	4,80	6,63	8,21	11,06	15,67	18,29	31,43	45,00
Perro	4,47	5,33	7,13	10,34	15,33	23,03	36,67	45,90	55,32	71,43
Caballo	1,41	1,63	2,01	2,79	3,85	4,68	5,26	5,96	6,72	7,92
Moto	3,05	3,50	4,65	6,32	8,77	11,32	13,58	15,77	21,51	24,56
Persona	91,13	89,92	88,71	87,62	85,80	83,10	79,75	74,99	69,82	66,89
Maceta	3,41	4,22	5,37	7,09	10,42	15,30	20,34	29,87	41,11	84,38
Oveja	11,56	16,18	25,08	37,50	50,00	58,62	75,29	85,29	94,12	92,68
Sofá	1,30	1,47	1,85	2,42	2,86	4,64	5,96	8,91	14,29	9,09
Tren	2,72	3,14	3,91	5,46	8,19	10,95	14,78	20,20	27,27	39,08
Monitor TV	2,02	2,33	2,67	3,22	4,92	6,47	9,07	10,84	13,50	17,57

Tabla 11: Medida-F según los distintos umbrales empleados.

Categorías	Umbrales empleados									
	0.5	**0.55**	**0.6**	**0.65**	**0.7**	**0.75**	**0.8**	**0.85**	**0.9**	**0.95**
Avión	6,47	8,08	12,55	18,93	26,96	36,52	45,45	56,30	61,54	57,14
Bicicleta	3,29	4,00	5,45	7,67	10,18	13,69	17,28	19,22	20,34	17,54
Pájaro	1,62	1,92	2,55	3,89	7,34	11,19	14,53	20,39	30,77	43,55
Barco	13,41	16,49	19,93	24,04	29,85	34,78	40,00	40,86	36,11	32,00
Botella	10,95	13,13	15,51	17,61	19,67	20,62	18,44	8,55	2,20	0,00
Autobús	9,36	10,88	14,81	19,54	25,00	29,76	34,33	42,42	45,45	24,62
Coche	22,92	26,58	29,62	32,76	31,47	29,74	24,36	18,98	13,66	10,47
Gato	6,21	7,80	10,48	17,44	33,46	46,81	54,19	67,74	77,67	74,42
Silla	11,34	13,87	14,81	17,25	20,60	21,01	16,33	11,98	8,86	2,63
Vaca	19,06	24,00	28,67	37,56	40,51	44,09	36,89	34,41	21,95	2,78
Mesa comedor	6,38	7,02	8,81	11,74	13,81	17,46	22,46	22,22	25,00	24,66
Perro	8,56	10,06	13,13	18,14	25,16	34,23	44,90	47,46	50,00	47,06
Caballo	2,78	3,20	3,93	5,40	7,34	8,80	9,74	10,82	11,82	13,48
Moto	5,91	6,76	8,88	11,85	15,99	19,95	23,30	26,12	33,33	34,15
Persona	81,23	77,65	73,58	68,95	62,54	53,79	43,37	31,26	15,85	2,72
Maceta	6,55	8,03	10,05	12,83	17,77	24,07	28,32	35,80	38,34	40,00
Oveja	20,46	27,27	38,90	51,72	61,54	66,02	73,14	73,42	68,09	58,02
Sofá	2,55	2,88	3,58	4,61	5,33	8,23	9,66	11,54	11,11	3,03
Tren	5,29	6,08	7,51	10,30	15,04	19,56	25,37	32,67	40,00	50,37
Monitor TV	3,95	4,53	5,17	6,16	9,18	11,78	15,75	18,05	21,05	24,19