TESIS DOCTORAL

Reconocimiento y clasificación automatizada de especies de polen alergénicas

Ramón Gallardo Caballero

Programa de Doctorado en Modelización y Experimentación en Ciencia y Tecnología

2020

UNIVERSIDAD DE EXTREMADURA

TESIS DOCTORAL

Reconocimiento y clasificación automatizada de especies de polen alergénicas

Ramón Gallardo Caballero

Programa de Doctorado en Modelización y Experimentación en Ciencia y Tecnología

Conformidad de los directores:

La conformidad del director/es de la tesis consta en el original en papel de esta Tesis Doctoral.

Dr. D. Carlos Javier García Orellana

Dr. D. Antonio García Manso

2020

"Ciencia es creer en la ignorancia de los científicos"

Richard Phillips Feynman.

Resumen

El presente trabajo de tesis doctoral está centrado en la detección de granos de polen en imágenes palinológicas tomadas de muestras estándar utilizando técnicas de aprendizaje profundo. La localización y clasificación de granos de polen es una tarea manual, muy laboriosa, que llevan a cabo palinólogos experimentados para estimar las concentraciones de los tipos de polen atmosférico presentes en distintas áreas geográficas. Este proceso se realiza a partir de muestras obtenidas en captadores de partículas aerobiológicas que, tras un procesamiento, deben visualizarse con un microscopio óptico. La estimación de los distintos tipos de polen resulta de gran utilidad en varios campos de la ciencia como en alergología, agricultura, ciencias forenses o paleopalinología.

Desde el año 2012 el campo de la inteligencia artificial ha experimentado un desarrollo muy importante en detección de objetos en imágenes, gracias al exitoso desarrollo de técnicas basadas en redes neuronales convolucionales. Parte del éxito logrado se ha debido a la aparición en el mercado de unidades de procesamiento gráfico con grandes capacidades de cálculo paralelo, pero también resultó importante la recopilación de grandes conjuntos de imágenes clasificadas.

Esta tesis doctoral tiene por objetivo principal evaluar la idoneidad de un método basado en redes neuronales convolucionales, que permita realizar la localización y detección de granos de varios tipos de polen de forma robusta. Consideramos éste un primer paso para desarrollar un sistema que pudiese servir de ayuda en un laboratorio de palinología.

Palabras clave: detección de polen, aprendizaje profundo, redes neuronales, palinología.

Abstract

This doctoral thesis is focused on the detection of pollen grains in palynological images from standard samples, using deep learning techniques. The localization and classification of pollen grains is a very laborious manual task, which is carried out by experienced palynologists to estimate the concentrations of the different types of atmospheric pollen present in given geographic areas. This process is performed on the basis of samples obtained in aerobiological particle collectors, which after processing, must be visualized with an optical microscope. The estimation of the different pollen types is very useful in several areas of science such as allergology, agriculture, forensic science or paleopalinology.

Since 2012, the field of artificial intelligence has achieved a very important progress in image object detection, thanks to the successful development of techniques based on convolutional neural networks. Part of the success achieved has been due to the appearance on the market of graphics processing units with large parallel computing capabilities, but the collection of large sets of classified images was also important.

The main objective of this doctoral thesis is to evaluate the suitability of a method based on convolutional neural networks for the localization and detection of grains of pollen of various types in a robust way. This method is considered a first step in the development of a system that could be helpful in a palinology laboratory.

Keywords: pollen detection, deep learning, neural networks, palinology.

Agradecimientos

En primer lugar me gustaría mostrar mi agradecimiento a mis padres que me facilitaron estudiar una carrera sin dudar en ningún momento de mí. A mi hermana que siempre me ha apoyado a pesar de los momentos difíciles.

A todos los miembros del Grupo de Investigación de Clasificación de Patrones y Análisis de Imágenes (CAPI) de la Universidad de Extremadura que siempre han estado dispuestos para ayudarme en cuanto he necesitado. Gracias Horacio, Miguel, Carlos y Antonio.

No quiero olvidar a mis compañeros de la Escuela Politécnica, Geles, Montaña, María José, Jiménez, Paniagua, Gordillo, Jesús y Nacho. Gracias por vuestro ánimo constante e incansable.

También quiero agradecer a Rafael Tormo Molina y su equipo del laboratorio de Aerobiología de la Universidad de Extremadura todo su trabajo y apoyo. Sin ellos esta Tesis no habría sido posible.

A todas las personas que me han ayudado no sólo durante la realización de este trabajo, sino a lo largo de toda mi vida. En especial a mis amigos Alonso y Ramón.

Por último, me gustaría agradecer a mis directores de Tesis, Carlos Javier García Orellana y Antonio García Manso, que me enseñaron, ayudaron y animaron en tantas ocasiones hasta finalizar este trabajo.

Índice general

Índice de figuras

Índice de tablas

Abreviaturas

BBox	**B**ounding **B**ox
CHT	**C**ircular **H**ough **T**ransform
CNN	**C**onvolutional **N**eural **N**etwork
IoU	**I**ntersection **o**ver the **U**nion
FC	**F**ully **C**onnected
FCN	**F**ully **C**onnected **N**etwork
FLOPS	**F**loating Point **O**perations per **S**econd
FN	**F**alse **N**egative
FP	**F**alse **P**ostive
FPN	**F**eature **P**yramid **N**etwork
GPU	**G**raphics **P**rocessing **U**nit
MJPEG	**M**otion **J**oint **P**hotographic **E**xperts **G**roup
ML	**M**achine **L**earning
MLP	**M**ultilayer **P**erceptron
NMS	**N**on-**M**aximum **S**uppression
ReLu	**R**ectifier **L**inear **U**nit
ResNet	**R**esidual **N**etwork
ROI	**R**egion **o**f **I**nterest
RPN	**R**egion **P**roposal **N**etwork
SGD	**S**tochastic **G**radient **D**escent
SPP	**S**patial **P**yramid **P**ooling
SSD	**S**ingle **S**hot Multibox **D**etector
SVM	**S**upport **V**ector **M**achine

TN	**T**rue **N**egative
TP	**T**rue **P**ositive

Capítulo 1

Introducción

1.1. Introducción

El análisis de las muestras de polen es una tarea que requiere una gran cantidad de tiempo, es una actividad laboriosa y requiere un trabajo altamente especializado. Así por ejemplo una medición estándar requiere la clasificación de al menos entre 300 y 500 granos de polen en cada muestra. Se requiere un conteo más detallado si la muestra presenta un número muy elevado de granos de unos taxones determinados, o si debe realizarse una estimación estadística de la concentración de granos poco frecuentes.

El estudio de los tipos de polen resulta de gran utilidad en un buen número de campos, desde los estudios paleo-climáticos [1], pasando por las técnicas de estudio forense [2], agricultura [3–5] o alergología [6]. La monitorización diaria de las concentraciones de polen atmosférico es una tarea clave en la gestión de problemas alergénicos, estimación de cosechas o estudio del cambio climático. Hoy en día, múltiples unidades aerobiológicas en todo el mundo realizan tareas de conteo manual de granos de polen para proporcionar estimaciones de concentración polínica en sus zonas de trabajo.

El análisis de muestras microscópicas naturales es hoy en día la base de muchos desarrollos tecnológicos en campos como medicina o biología [7–9]. La detección automatizada de objetos en imágenes tomadas mediante microscopio óptico supone un reto en múltiples tipos de imágenes naturales debido a la gran variabilidad de las muestras. En el caso de las muestras palinológicas, deben tenerse en cuenta al menos el carácter volumétrico de los granos de polen, las grandes variaciones en su apariencia al microscopio, los detritos transportados por el aire y el sustrato no uniforme sobre el que se adhieren los granos. Estas características naturales de las muestras suponen importantes retos computacionales, pero el carácter volumétrico parece indicar, desde el primer análisis, la

imposibilidad de realizar un abordaje eficiente del problema basado en una única imagen por muestra.

En visión por computador, se utilizan los términos *clasificación* o *reconocimiento* para referirse a la tarea de identificar qué objeto contiene una imagen dada. El término *localización* se usa cuando el objetivo es encontrar objetos en la imagen, e indicar su ubicación y extensión, por ejemplo mediante un marco delimitador o *bounding box* (BBox). El término *detección* incluye las subtareas de localizar y clasificar todos los objetos presentes en una imagen. Y hoy en día, como tareas de mayor complejidad, encontramos los conceptos de *segmentación semántica* y *segmentación de instancias*, que proporcionan una clasificación de cada píxel de la imagen según el tipo de objeto al que pertenecen, e identificando por separado los distintos objetos en el caso de la segmentación de instancias.

Desde el punto de vista computacional, la localización e identificación de granos de polen es un problema de *detección de objetos* en una imagen, en nuestro caso granos de polen. Tradicionalmente, la detección de objetos en imagen se ha basado en el desarrollo de técnicas de extracción de características altamente significativas que se utilizaban como entrada a un clasificador, que generalmente estaba implementado mediante un perceptrón multicapa (MLP) o *Support Vector Machines* (SVM).

En los últimos años, la detección de objetos ha dado un salto muy importante gracias al desarrollo de las conocidas como técnicas de aprendizaje profundo (*deep learning*), que han superado ampliamente los parámetros de rendimiento logrados por las técnicas clásicas de localización y clasificación en imagen. El desarrollo acelerado de estas técnicas se ha basado en la recuperación de las redes neuronales convolucionales (CNN), propuestas desde 1989, pero sin un desarrollo exitoso hasta el año 2012.

Una de las desventajas que presentan las redes profundas es su elevado número de parámetros ajustables. Requieren un gran conjunto de prototipos de entrenamiento y una elevada potencia de cálculo para lograr un ajuste eficiente de la red. La obtención de un elevado número de prototipos, perfectamente segmentados, para ajustar eficientemente un modelo convolucional de clasificación de polen es una tarea compleja. Supondría el etiquetado manual de millones de imágenes lo que a priori supone un hándicap insalvable. Afortunadamente, uno de los resultados más interesantes obtenidos en el estudio de las CNN es la posibilidad de realizar lo que se conoce como transferencia de aprendizaje entre modelos (*transfer learning*) [10]. La transferencia de aprendizaje permite utilizar una red convolucional ya entrenada para una tarea específica, y reajustar sus parámetros para abordar una tarea distinta, utilizando un número de prototipos de entrenamiento reducido.

En términos históricos, el entrenamiento de grandes redes neuronales requería infraestructuras de cálculo muy potentes. Este requisito, que supuso un gran inconveniente en el pasado, hoy no es un problema significativo gracias a la aparición de las tarjetas gráficas modernas (GPU), que permiten paralelizar las operaciones e implementar algoritmos de entrenamiento convolucionales en tiempos razonables.

Durante los últimos años, las CNNs se han usado con mucho éxito tanto en tareas de clasificación como en detección de objetos. La primera propuesta que despertó interés como sistema de detección de objetos fue el algoritmo *Regions with CNN* (R-CNN) [11]. Más tarde, se desarrollaron otros algoritmos como *Fast R-CNN* [12], *Faster R-CNN* [13], *Single Shot Multibox Detector* (SSD) [14], *Mask R-CNN* [15], *RetinaNet* [16] o las distintas variantes de *YOLO* [17–19], que han permitido detectar objetos de forma cada vez más precisa y significativamente más rápida. El tiempo requerido para procesar imágenes con objetos macroscópicos con estas redes y GPUs de gama de consumo es tan reducido que permite incluso realizar detección en tiempo real. El modelo Faster R-CNN se considera en este momento como un modelo de referencia con el que comparar nuevas propuestas de detección integradas.

Esta tesis platea responder a las siguientes preguntas:

- ¿Puede un modelo neuronal localizar granos de polen de forma eficiente a partir de muestras estándar?.
- ¿Puede un modelo neuronal detectar granos de polen de forma eficiente a partir de muestras estándar?.
- ¿Podemos realizar una detección de granos de polen en tiempo real?.

1.2. Estructura de la tesis.

El presente trabajo de Tesis consta de 6 capítulos y un apéndice. El contenido de cada uno de los capítulos lo podemos resumir de la siguiente forma:

- En el **Capítulo 2** se muestra una introducción general a las técnicas actuales de captación, conteo y estimación de concentraciones de polen. Además, se expone una revisión del estado del arte de los métodos de localización, clasificación y detección en imágenes palinológicas, asistidos por ordenador.
- En el **Capítulo 3** se presentan las técnicas utilizadas en el desarrollo del presente trabajo.

- En el **Capítulo 4** se analizan los retos que presenta el espacio de imágenes a procesar. Además, se presentan los algoritmos desarrollados para detectar y clasificar las muestras palinológicas. Así mismo se aborda la integración de estos algoritmos con la base de datos SQL creada para la realización del trabajo.
- En el **Capítulo 5** se exponen los resultados obtenidos en los distintos experimentos que hemos realizado. Algunos resultados de interés a campo completo sobre las muestras utilizadas aparecen recogidos en un Apéndice.
- Finalmente, en el **Capítulo 6** se exponen las conclusiones finales del trabajo realizado y las líneas de trabajo que quedan abiertas para futuras investigaciones.

Capítulo 2

Identificación automatizada de granos de polen

2.1. Introducción

Los captadores utilizados con más frecuencia en estudios aerobiológicos para estudio e identificación de tipo de polen son de tipo Hirst [20], como el que se muestra en la Figura 2.1. En este tipo de captador, las partículas se recogen en una cinta adhesiva que se expone a un flujo de aire constante tomado del entorno en el que se ha colocado el equipo. Una sección diferente de la cinta adhesiva se expone cada hora, lo que permite estudiar la evolución horaria de la concentración polínica. Este método requiere un procesado posterior de la muestra que gracias a un tintado, facilita la observación de los elementos ornamentales de los granos de polen. A continuación, se procede a un conteo manual de los granos de polen encontrados en la muestra con un microscopio óptico siguiendo un protocolo establecido.

El proceso descrito es costoso en términos temporales, por lo que se han propuesto múltiples algoritmos que tratan de acelerar el proceso de identificación y conteo de granos de polen. Algunas de estas propuestas realizan un abordaje del problema utilizando conjuntos de datos que únicamente contienen un grano, previamente segmentado, por muestra [4, 21–23]. Sin embargo, la lista de trabajos que detallan una segmentación automática de los granos de polen en las muestras es corta, y en ellas los autores utilizan distintas técnicas clásicas como umbralizado, filtrado por forma y tamaño y análisis de textura [24], transformación por similitud de color [25], operadores morfológicos [4, 26] o uso de contornos activos [27].

Figura 2.1: Un captador tipo Hirst situado en una ubicación sin obstáculos.

2.2. El protocolo de conteo manual

En 1992 se puso en marcha la Red Española de Aerobiología (REA), que tiene entre sus objetivos obtener registros polínicos presentes en la atmósfera para la prevención de alergias polínicas, difundiendo la información entre la ciudadanía. Para ello estandarizaron una metodología [28] de uso común por todos los integrantes de la red.

En la REA se utilizan de forma normalizada captadores de partículas volumétricos por succión, basados en el principio de impacto, los ya mencionados captadores tipo Hirst [20]. Estos captadores permiten obtener datos homologables independientemente de las características biogeográficas y bioclimáticas de la zona que muestrean, con precisión horaria de 24 horas. Deben usar un caudal de succión de 10 litros de aire por minuto, similar al volumen de inhalación humano. Este mismo sistema se utiliza a nivel europeo en la European Aeroallergen Network (EAN).

Los muestreadores utilizados permiten capturar eficazmente partículas aerovagantes con diámetros comprendidos entre 1 y 100 micrómetros. Presenta una unidad de impacto con un orificio de entrada, de 14×2 mm, y un soporte circular (tambor) sobre el que se adhieren las partículas.

El tambor se encuentra conectado a un reloj con un mecanismo de giro que mueve el soporte 2 mm cada hora. Sobre el tambor se coloca una cinta especial, impregnada con un adhesivo, en la que se adhieren las partículas succionadas con cierta velocidad.

Como puede apreciarse en la Figura 2.1, el captador también cuenta con una veleta cuya función es mantener el orificio de entrada en la dirección de los vientos dominantes para aumentar la eficacia de captación.

Una vez transcurrido el tiempo de captación, se extrae el tambor y se procesa la muestra. La longitud de una muestra de 24 horas es de 48 mm. Cada segmento de 24 horas se dispone sobre un portaobjetos o platina para usar en el microscopio. La muestra suele tintarse con glicerogelatina teñida con fucsina para facilitar la mejor identificación y recuento de los granos de polen. Las muestras suelen sellarse con un cubreobjetos y laca-esmalte transparente.

El análisis de las muestras se realiza con microscopio óptico con un mínimo de 40×10 aumentos. Un valor inferior no permitiría la identificación de algunos tipos de polen. Dado que el recuento total de los granos de polen y esporas presentes en las muestras requeriría mucho tiempo, se realiza un sub-muestreo que represente como mínimo un 10 % del total de la muestra, según directrices de la EAN.

El protocolo REA, establece un recuento basado en 4 barridos horizontales continuos, equidistantes entre si y de los bordes de la preparación. La sub-muestra analizada sería en este caso del 12-13 % de la superficie total como muestra la Figura 2.2. A lo largo de cada barrido se cuenta el número de granos de polen de cada tipo polínico identificado con lo que se obtiene información sobre la concentración polínica del aire a lo largo del día.

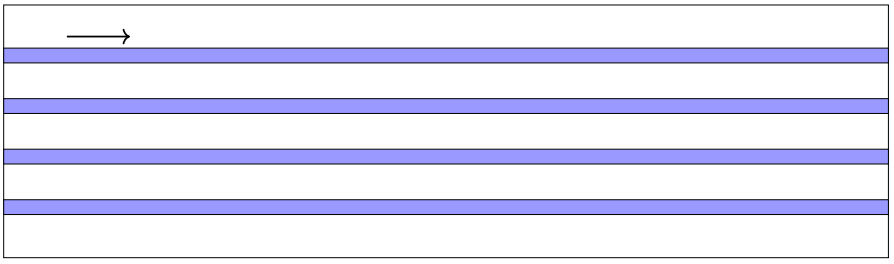


Figura 2.2: Recuento según metodología REA. Cuatro barridos horizontales equidistantes entre si y de los bordes superior e inferior.

Para conocer el número de granos que se registra en cada hora del día, se utiliza una reglilla de acetato con 24 separaciones de 2 milímetros que se coloca bajo la platina, como muestra la Figura 2.3. Y se establece el arranque de cada barrido mediante un rotulador permanente de punta fina. Así, se cuenta el número de granos de polen de cada hora y a continuación se suma el número total de granos de polen que se ha contabilizado para cada tipo polínico en un día determinado.

La concentración polínica se expresa como la media de granos de polen por metro cúbico de aire. Pero el valor de conteo debe corregirse en función del campo de visión del

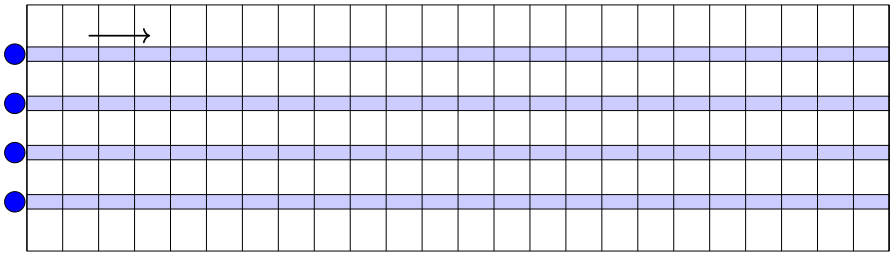


Figura 2.3: División horaria de la muestra para realizar el conteo horario de granos de polen según el protocolo REA.

microscopio utilizado en la escala de 40 × 10 aumentos. La Ecuación 2.1 se utiliza para especificar la concentración polínica diaria, siendo N el número de granos de polen contabilizado, S_T la superficie total muestreada, S_A la superficie analizada en los cuatro barridos y V_S el volumen diario de succión en metros cúbicos.

$$\nu = \frac{S_T \cdot N}{S_A \cdot V_S} \tag{2.1}$$

A modo de ejemplo se especifica en el protocolo que, con un volumen de succión de 10 l/min, siendo el diámetro del campo de visión del microscopio de 0.45 mm y el ancho de la cinta adherente de 14 mmm, el contenido de partículas por metro cúbico de aire sería $\nu = \mathrm{N} \times 0.54$.

2.3. Bases de datos previamente disponibles

Los experimentos reportados en este campo, generalmente se realizan a partir de conjuntos de datos auto recogidos. No obstante, al menos existen tres bases de datos de libre disposición que permitan llevar a cabo algún tipo de estudio en el campo de la identificación de granos de polen: *Duller's Pollen Dataset* [29], *POLEN23E* [30] o la recién publicada *POLLEN73S* [31].

La primera contiene un total de 630 imágenes en escala de gris con un tamaño de 25x25 píxeles. La resolución no parece excesivamente alta para apreciar detalles de ornamentación y al contener muestras en escala de gris se elimina una parte de la información de tintado de los granos.

El conjunto *POLEN23E*, contiene un total de 805 imágenes de 23 tipos de polen, con 35 muestras de color por tipo de polen, con al menos 250 píxeles de resolución en ambas dimensiones. En este caso, la resolución con la que puede estudiarse cada grano si es adecuada, pero al tratarse de una base de datos con granos ya segmentados no permite llevar a cabo estudios de detección de polen que se asemejen a una situación real. Además,

únicamente ofrece un plano focal para cada uno de los granos considerados escogido por el experto, por lo que tampoco permite estudiar el rendimiento a la hora de identificar el grano sin intervención humana.

POLLEN73S, de reciente aparición contiene 2523 imágenes de granos de polen segmentados habituales en la sabana brasileña. Recoge imágenes tomadas a distintos ángulos de 73 tipos de polen distintos. Y está creada específicamente para entrenar redes convolucionales profundas y mejorar la calidad de los modelos con un número mayor de prototipos.

La ausencia de bases de datos con muestras a campo completo bien naturales o generadas en laboratorio, lleva a que los investigadores desarrollen conjuntos de datos propietarios, lo que dificulta las posibilidades de realizar comparaciones significativas de rendimiento entre los distintos algoritmos previamente desarrollados, y los que pudiesen desarrollarse. Y en esta línea, debe entenderse que el análisis de los trabajos descritos en las Secciones 2.4 y 2.6, únicamente están referidos a conjuntos de datos capturados y etiquetados por los autores de cada uno de ellos.

2.4. Análisis del estado del arte en localización

Existen grandes diferencias en el tamaño y las características del conjunto de muestras utilizado en cada uno de los trabajos que referenciamos a continuación. En términos generales, la localización automática de granos de polen en muestras puras con objetos aislados alcanza buenos resultados de rendimiento con tasas de *precisión* y *sensibilidad* (*recall*) elevadas (ver Sección 4.10). Pero cuando las muestras son más realistas con granos solapados o agrupados y presencia de detritos, los indicadores de rendimiento de los sistemas propuestos decrecen de forma significativa.

La Tabla 2.1 muestra de forma resumida algunos datos de interés de varios trabajos recientes encontrados. Como puede observarse, los estudios de Landsmeer et al. [25] y Nguyen et al. [27] son los que presentan un menor número de granos a localizar y al mismo tiempo reportan las tasas más elevadas de rendimiento en localización.

Ranzato et al. realizaron un experimento para detectar partículas biológicas genéricas en imágenes microscópicas. Desarrollaron su clasificador utilizando un conjunto de partículas microscópicas encontradas en urianálisis y lo extendieron para localizar y clasificar un conjunto de datos de polen atmosférico. El conjunto de datos polínico constaba de 1429 imágenes que contenían 3686 granos pertenecientes a 27 tipos de polen. Su algoritmo de localización estaba basado en algoritmo de reconocimiento de objetos de Lowe [33], basado en características invariantes a la escala (SIFT). Los autores reportaron una tasa

Tabla 2.1: Resumen de las características y resultados de los trabajos de investigación más recientes que realizan localización automática de granos de polen.

	Año	Tipos	Platinas	Granos	Detector	Sens.	Prec.
Ranzato et al. [32]	2007	8	n/d	3686	SIFT	93.9	8.6
Landsmeer et al. [25]	2009	n/d	9	65	Similitud de color	86	61
Nguyen et al. [27]	2013	9	1	768	Contornos activos	93.8	89.5
Díaz-López et al. [24]	2015	12	12	4061	Forma y textura	81.92	18.5

de sensibilidad en localización del 93.9 % sobre el conjunto de muestras de polen, a costa de un una precisión del 8.6 %. El criterio utilizado para determinar una localización correcta es reproducible y está basado en la distancia entre de los centros de la predicción y una marca de certeza (*ground truth mark*), con una limitación de área como recoge la Ecuación 2.2.

$$\sqrt{(x_{c1} - x_{c2})^2 + (y_{c1} - y_{c2})^2} \leq \sqrt{\frac{A_1 + A_2}{2}} \tag{2.2}$$

Landsmeer et al. usaron un conjunto de 44 imágenes tomadas en planos focales sucesivos para entrenar su sistema. En primer lugar, el sistema usa una transformación basada en color que asocia un valor numérico a cada pixel de la imagen. A continuación, usan la Transformada Circular de Hough (CHT) para inferir objetos circulares en la imagen. Posteriormente, se realiza un filtrado por color y un clusterizado de objetos para determinar las propuestas finales de grano. Para evaluar el rendimiento de su sistema, usaron un conjunto de 17 muestras que contenían un total de 65 granos de polen, reportando una tasa de precisión del 61 % y una sensibildad del 86 %. A pesar de que el número de granos de polen utilizado para medir el rendimiento es bajo, lo que resta representatividad al estudio, este trabajo permite conocer con cierta claridad la separación realizada entre los conjuntos de entrenamiento y test. Sin embargo, no aporta un criterio claro para determinar qué requisitos requiere una localización correcta.

Nguyen et al. usaron una única muestra de polen recolectado por abejas escaneada a 40 aumentos, con fondo blanco y uniforme, en la que identificaron 768 granos de nueve tipos de polen distinto. Para localizar los bordes de cada grano de polen de forma precisa usaron contornos activos, estimando el contorno inicial del grano mediante CHT. A la hora de especificar el rendimiento en localización, decidieron considerar una localización correcta si la distancia de la propuesta al grano más cercano identificado manualmente, es menor que el diámetro del tipo de polen más pequeño considerado. Con este criterio, indican que su método proporciona una precisión del 89.5 % y con una sensibilidad del 93.8 %.

Finalmente, Díaz-López et al. desarrollaron un sistema de localización de granos que generaba propuestas mediante un algoritmo de cinco pasos. En primer lugar su algoritmo lleva a cabo una eliminación del fondo. A continuación, selecciona objetos de tono rosado mediante filtrado por color. Después, descarta objetos con baja ecentricidad y aquellos que no pueden considerarse granos de polen por su tamaño. Y finalmente, usa un clasificador que a partir de determinadas características de color, forma y textura, genera las propuestas de grano. Para medir el rendimiento de su sistema, usaron 12 muestras de polen atmosférico que contenían 3999 granos de polen identificados manualmente por un experto. No se hace mención expresa al uso de estas muestras para ajustar el clasificador. Como resultado final de su estudio, reportan una sensibilidad del 81.92 % con una precisión del 18.5 %. No se hace una mención expresa al criterio utilizado para calificar como acierto una detección generada por el clasificador. Nuevamente, en este caso podemos observar un valor de precisión bajo al utilizar un conjunto de muestras real y muy extenso. Estos valores llevan a los autores a concluir que el valor obtenido de precisión resulta insuficiente para desarrollar un sistema palinológico automatizado robusto.

2.5. Estado del arte en clasificación

Esta vertiente del trabajo con muestras polínicas es donde más trabajos podemos encontrar. Incluiremos en este apartado los estudios que, partiendo de una o más imágenes de granos segmentados, llevan a cabo la clasificación de estas muestras para asociarles una clase. Entran en esta categoría aquellos trabajos que utilizan las bases de datos *Duller's Pollen Dataset* y *POLEN23E*. La Tabla 2.2 recoge las principales características de los trabajos analizados en esta categoría. En esta tabla la columna CCR especifica la tasa de clasificación correcta definida por la Ecuación 2.3 [30].

$$CCR = \frac{TP + TN}{TP + TN + FP + FN} \tag{2.3}$$

El trabajo de Redondo et al. [4] del año 2015, construye una base de datos de 15 tipos de polen con 120 muestras por tipo a partir de muestras de polen de abeja. Para llevar a cabo la clasificación de las muestras analizan un amplio conjunto de características morfológicas y estadísticas, calculadas sobre la imagen y el contorno de cada grano ajustado manualmente. A partir de varias combinaciones de estas características y utilizando tres tipos de clasificadores (Fisher, SVM y árbol de decisión), estudian la configuración

Tabla 2.2: Resumen de las características y resultados de los trabajos de investigación más recientes que realizan clasificación automática de granos de polen.

	Año	DB	Tipos	Granos	Clasificación	Prec. (%)	Sens. (%)	CCR (%)
Redondo et al. [4]	2015	Propietaria	15	1800	Fisher, SVM, Árbol de decisión	n/d	n/d	99.4
Daood et al. [23]	2016	Propietaria	30	≈ 1000	CNN	92.0	90.3	89.9
Barbosa et al. [30]	2016	POLEN23E	23	805	SVM, Árbol de decisión, KNN	n/d	n/d	64.0
Sevillano y Aznarte [21]	2018	POLEN23E	23	805	CNN, LDA	97.8	99.6	97.2
Daood et al. [34]	2018	Propietaria	10	392	CNN+RNN	100	100	100
Sevillano et al.[35]	2020	Pública	46	19000	CNN	97.9	97.8	97.86
Astolfi et al. [31]	2020	POLLEN73S	73	2523	CNN	95.7	95.7	n/d

que proporciona mayor rendimiento. Así, a partir de una mezcla de descriptores morfológicos, estadísticos, Fourier, wavelets y logGabor con LDA reportan un rendimiento en clasificación de 99.4 %.

En el trabajo de Daood et al. [23] de 2016, se utiliza una base de datos propietaria generada por el laboratorio de Paleoecología del Instituto de Tecnología de Florida. La base contiene dos tipos de muestras, las primeras obtenidas con microscopía óptica y las segundas con microscopía de barrido electrónico (SEM). En total contenía unas 1000 imágenes de 30 tipos de polen. Lo primero que cabe resaltar de este trabajo es el gran nivel de detalle que aportan las muestras generadas con microscopio electrónico, por lo que centraremos el análisis en la vertiente óptica del trabajo. En el aspecto de clasificación utilizaron dos redes CNN entrenadas en CPU. La primera de ellas, tenía siete capas y se realizó un ajuste desde cero con técnicas de aumento de datos, rotando las muestras originales para pasar de 1000 a 14000 prototipos de tipo óptico. En segundo lugar, se entrenó una segunda CNN con distintas características usando transferencia de aprendizaje a partir de un modelo entrenado con ImageNet [36]. En el apartado de resultados, los autores reportan unas tasas de clasificación del 84.47 % para el modelo entrenado desde cero y del 89.95 % para el modelo que usa transferencia de aprendizaje. Adicionalmente, realizan una comparativa del rendimiento logrado por distintas técnicas previamente utilizadas en clasificación polínica sobre su conjunto de datos, concluyendo que sus resultados superan de forma significativa los obtenidos con técnicas anteriores.

El trabajo de Barbosa et al. [30] de 2016, presenta como punto más importante la definición de la base de datos *POLEN23E*, y sobre esta misma base de datos, proporciona referencias de clasificación tanto humana como automáticas. Como extractores

de características utilizan la técnica *Bag of Visual Words* (BOW) y otro conjunto de características basadas en color, forma y textura (CST). En cuanto a los clasificadores, encontramos dos variantes de SVM, árboles de decisión y *k-nearest neighbors* (KNN). Para el estudio del rendimiento de la capacidad de reconocimiento humana, contaron con la ayuda de 34 apicultores sin conocimientos en reconocimiento de polen, que escogían la clase de 46 imágenes usando un manual de referencia. Bajo estas condiciones, calcularon que el 64 % de las imágenes fueron clasificadas correctamente, con grandes variaciones en la tasa de acierto según el tipo de polen. Como resultado del estudio de clasificación automática, encontraron que la mejor CCR fue también del 64 %, utilizando un conjunto de características obtenido con los extractores CST+BOW y el clasificador C-SVM.

El trabajo de Sevillano y Aznarte de 2018 [21] utiliza la base de datos *POLEN23E*, y estudia el rendimiento de tres métodos de clasificación basados en redes neuronales convolucionales. En contraposición al uso habitual de las CNN como clasificador directo, usan este tipo de red como extractor de características que se emplean en otros clasificadores especializados. Además utilizan *Linear Discriminant Analysis* [37] como base de su clasificador (LD). De esta forma, comparan el rendimiento de tres estructuras de clasificación distintas:

1. Una red AlexNet con los pesos de referencia utilizada como extractor de características que se utilizan en un clasificador LD.
2. AlexNet reajustada al nuevo espacio de imagen como clasificador.
3. AlexNet reajustada como extractor de características para el clasificador LD.

Los autores obtienen los mejores resultados con el sistema basado en la red AlexNet reentrenada sobre el nuevo espacio de imágenes, como extractor de características para el clasificador LD. Con esta última configuración, alcanzan una sensibilidad del 99.64 % y una precisión del 97.77 %.

En 2018 Daood et al. [23] publicaron un nuevo trabajo, en el que utilizaron un nuevo conjunto de datos propietario basado en apilamientos de 10 imágenes de cada grano. En este caso, utilizaron una CNN entrenada para modelar el nuevo conjunto de datos y combinaron esta red con una *Recurrent Neural Network* (RNN) para determinar el tipo de polen a partir de las secuencias de imágenes del apilamiento. De los 392 granos disponibles, utilizaron 294 apilamientos para entrenar la CNN, basada en VGG16, y 98 como conjunto de test. Bajo estas condiciones, reportan una tasa de identificación del 100 %, aunque el reducido número de granos a identificar en el conjunto de test, puede que reduzca la representatividad de este resultado.

En junio de 2020, Sevillano et al.[35] aplicaron su técnica de clasificación desarrollada en [21] a un conjunto de datos considerablemente mayor, y por tanto con resultados más significativos. El conjunto de datos utilizado contiene más de 19000 imágenes de granos de 46 tipos de polen diferentes, obtenidos a partir de muestras con un tratamiento de acetólisis [38], y captadas con la técnica de microscopio de campo oscuro. Las imágenes se obtuvieron con el sistema *Classifynder* [39], que localiza automáticamente y escanea los granos de polen presentes en una muestra estándar. Para estimar el rendimiento de su sistema realizaron una validación cruzada de 10 iteraciones, reportando una precisión de 0.979 y una sensibilidad de 0.978, ambos parámetros con desviaciones estándar del orden de 0.03.

En octubre de 2020 se ha publicado una revisión de la base de datos *POLEN23E* denominada *POLLEN73S* [31], en la que de la misma forma que en [30] junto con la base de datos se publican unos valores de referencia obtenidos en este caso para varios modelos de CNN. El trabajo estudia el rendimiento de 8 modelos de CNN distintos entrenados con sus parámetros de configuración por defecto. El entrenamiento y validación de resultados se realizaron con un esquema de validación cruzada de 5 iteraciones (*5-fold cross validation*), usando el 80 % de los prototipos para entrenar y el 20 % restante para testar. Bajo esta estructura, obtuvieron los mejores resultados con un modelo *DenseNet-201* [40], alcanzando una precisión y sensibilidad del 95.7 %.

2.6. Estado del arte en detección

El concepto de detección de granos de polen implica la localización de propuestas de grano y la clasificación de estas propuestas de forma automatizada. La mayor parte de los trabajos que se analizan en esta sección ya han sido referenciados en la Seccion 2.4, por lo que en este caso únicamente se especificará los detalles asociados al proceso de clasificación. La Tabla 2.3 recoge las principales características de estos trabajos, incluyendo el clasificador utilizado y los indicadores de rendimiento que se especifican en cada trabajo.

Tabla 2.3: Resumen de las características y resultados de los trabajos de investigación más recientes que realizan detección automática de granos de polen.

	Año	Tipos	Platinas	Granos entren.	Granos test	Clasificador	CCR (%)
Ranzato et al. [32]	2007	8	n/d	3686	$\approx$ 368	Bayesiano	77
Holt et al. [39]	2011	6	4	n/d	809	MLP	n/d
Nguyen et al. [27]	2013	9	1	768	768	TaskTrAdaBoost	92

La etapa de clasificación del trabajo de Ranzato et al. [32], utiliza como extractores de características *local jets* estudiados en [41] e invariantes calculados sobre imágenes en escala de grises definidos en [42]. A partir de una mezcla de éstas características entrena un clasificador Bayesiano [43, cap. 10]. Como curiosidad, también usaron un clasificador convolucional [44] pero el modelo sufría de problemas de sobreajuste por el escaso número de ejemplos de entrenamiento y fue descartado. El rendimiento se testó usando una estrategia de validación cruzada de 10 iteraciones, repitiendo el experimento de entrenamiento y prueba 100 veces con los granos de polen segmentados por el experto y otras 100 con los granos de polen localizados automáticamente. El rendimiento en este caso se especifica en términos de la tasa de error, siendo el error medio del 21.8 % en el caso de los granos segmentados manualmente y creciendo hasta el 23.2 % al utilizar los granos segmentados por su algoritmo de localización. Por tanto, en valor medio, un 76.8 % de los granos del conjunto de test (≈ 368) se clasifican correctamente.

El trabajo de Holt et al. [39], no analizado en la Sección 2.4, presenta un estudio realizado sobre 4 preparaciones que contienen 6 tipos de polen. El objetivo principal del estudio es comparar la capacidad de detección de palinólogos humanos, con la del sistema semi-automatizado que han desarrollado (*AutoStage*, ahora conocido como Classifynder). El equipo utiliza un microscopio de campo oscuro robotizado en los tres ejes, con dos cámaras que proporcionan imágenes con resolución de 1280x1024 y aumentos asociados de x4 y x20. Tras un enfoque inicial sobre la platina el equipo realiza un primer barrido con la cámara de bajo aumento para localizar propuestas de grano. En un segundo barrido con la cámara de detalle, el equipo captura imágenes de los granos previamente localizados. Por último, los granos capturados son clasificados a partir de 43 características con un perceptrón con una capa oculta según el trabajo de Li et al. [45]. Este trabajo estudia un equipo ya entrenado, por lo que no se conoce el número de prototipos utilizado para su entrenamiento. El sistema permite una corrección manual de las clasificaciones generadas lo que permite reducir la carga de trabajo del palinólogo. No se realiza un estudio de rendimiento del sistema en localización, debido al tamaño del área a procesar manualmente. Sí se proporciona una comparación del rendimiento bruto del sistema, frente a la corrección realizada por los expertos en las cuatro platinas, y una segunda del conteo realizado por los expertos manualmente, frente al conteo automatizado corregido. Esta última comparativa permite conocer si la clasificación asistida con este equipo logra valores de conteo comparables a los obtenidos por los expertos de forma enteramente manual. Las mayores discrepancias entre el conteo automatizado y el corregido se deben a detritos identificados como granos de polen por el equipo. Ademas, en términos generales, los conteos automáticos corregidos son menores que los conteos humanos. El análisis de la varianza (ANOVA) que realizaron para esta comparación, les llevó a comprobar que con una certeza del 95 %, ambas distribuciones son diferentes, dado que su

equipo no localizaba correctamente los granos agrupados, que por tanto no pasaban a la fase de clasificación. Lo que les llevó a concluir que con la tecnología de aquel momento (2011), resultaba difícil construir un sistema de bajo coste que clasificase correctamente el 100 % de las muestras. Lo que vuelve a poner de manifiesto la dificultad de realizar esta tarea de forma eficiente y robusta.

En el apartado de clasificación, el trabajo de Nguyen et al. [27] presenta una versión mejorada del algoritmo de transferencia de aprendizaje TaskTrAdaBoost [46], que mejora la precisión del original utilizando un menor número de muestras etiquetadas. Como entrada al clasificador utilizan características de forma, textura y el número de espiculaciones detectadas en el grano. Las características de forma se obtienen a partir del contorno del grano localizado, las de textura a partir del contenido del BBox del grano y las espiculaciones se calculan a partir de las variaciones de luminancia en el contorno. El rendimiento en clasificación se calcula a partir de 50 repeticiones del algoritmo de entrenamiento utilizando una clase semilla distinta, especificando el rendimiento como el valor medio de detección correcta para todas las clases. Con estos criterios, observan un incremento de la precisión de su clasificador al incluir el número de espiculaciones como parámetro de entrada al clasificador reduciéndose la tasa de error en algo más del 3 %. Con esta última configuración, obtienen una tasa de clasificación correcta media del 92 % de los granos localizados.

Del análisis de los trabajos anteriores podemos inferir que la tarea de localización y clasificación de granos de polen no resulta sencilla. La inexistencia de amplias bases de datos con imágenes a campo completo obliga a los investigadores a construir su propio conjunto de imágenes. En este trabajo, hemos recopilado y etiquetado una base de datos, descrita en la Sección 4.3, que permitirá llevar a cabo estudios de detección reproducibles sobre los tipos de polen que hemos obtenido.

Capítulo 3

Descripción de las técnicas y herramientas utilizadas

3.1. Introducción

En este capítulo analizamos las distintas técnicas utilizadas en nuestro estudio. En primer lugar, analizamos un algoritmo de detección de círculos clásico, que ha sido la base de algunos trabajos analizados en la Sección 2.6. A continuación, realizamos una pequeña introducción a los principios de las redes neuronales y las primeras redes profundas. Después, repasamos los elementos constructivos más habituales en las redes convolucionales modernas y estudiamos algunos modelos de interés. Por último, analizamos los modelos de red que utilizamos en este estudio junto con las ideas previas en las que se basan.

3.2. La transformación circular de Houhg

La transformación circular de Houhg [47] (CHT) es una técnica de extracción de características clásica que permite localizar formas circulares imperfectas en imágenes. Utilizaremos un algoritmo de localización basado en esta transformada, como base de comparación con las técnicas clásicas de localización de granos de polen, dado que es uno de los más utilizados en la bibliografía referenciada en la Sección 2.4. De esta manera pretendemos estimar el rendimiento de los algoritmos clásicos de localización sobre nuestra base de datos. Nuestro algoritmo de localización está escrito con la implementación de la transformación circular de Hough de la librería OpenCV [48] mediante la función *HoughCircles*.

La CHT es una variante de la transformación de Hough [49], en la que los candidatos a círculo se producen por votación en un espacio de parámetros tridimensional. La principal ventaja de la transformada de Hough es su insensibilidad a la oclusión, lo que nos permitiría detectar incluso granos solapados, uno de los hándicaps detectados en nuestro espacio de imagen.

La implementación de la CHT que realiza OpenCV, permite localizar el centro (x, y) y el radio (r) de las formas circulares presentes en una imagen en escala de grises. Para ello utiliza el método del gradiente de Hough [50, p. 158]. En primer lugar se define un acumulador de dos dimensiones que sumará los votos para los centros de los círculos. Este método comienza aplicando un detector de bordes, en este caso de tipo Canny. A continuación, para cada punto distinto de cero en la imagen de bordes, se calcula su gradiente local mediante las derivadas de primer orden de Sobel. Todos los puntos en las direcciones marcadas por los gradientes, hasta un máximo dado, incrementan el valor del acumulador. Estas líneas deberían confluir por tanto, en el centro de círculos como muestra la Figura 3.1.

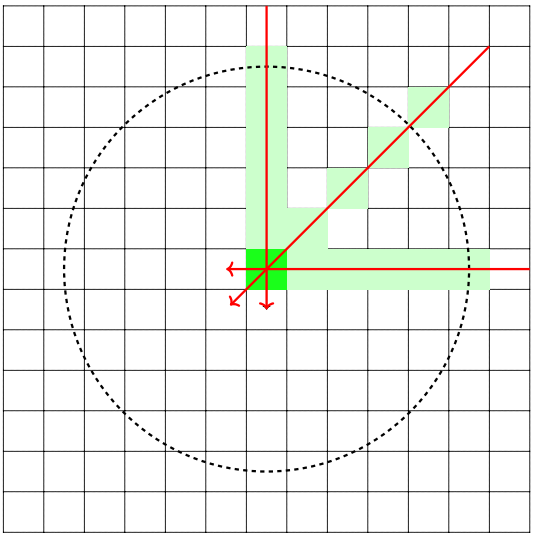


Figura 3.1: Determinación del centro de los círculos con el método del gradiente de Hough. Las líneas de gradiente deben confluir en el centro de un círculo.

De esta forma, los candidatos a centro de círculo serán aquellos píxeles del acumulador bidimensional que superen un valor umbral y tengan más votos que los píxeles de su entorno. Los candidatos a centro se ordenan descendentemente por su valor de acumulador de forma que los centros con mayor número de votaciones aparezcan primero. A continuación, para cada centro se tienen en cuenta los píxeles de la imagen de bordes distintos de cero y se ordenan según su distancia al centro. Teniendo en cuenta el rango de radios posibles especificado como parámetro en el algoritmo, se seleccionan los radios que mejor cubran los píxeles distintos de cero. Un radio así determinado se mantiene si

cubre un número suficiente de píxeles distintos de cero y presenta una distancia suficiente con otros centros previamente determinados.

Este algoritmo presenta algunas limitaciones, como un tiempo de ejecución variable y dependiente del umbral del acumulador, incapacidad de localización de círculos concéntricos y tendencia a sobreponderar círculos grandes frente a otros más pequeños. Pero ninguna de estas limitaciones nos afecta de forma significativa a la hora de localizar granos de polen.

3.3. Aprendizaje y Redes neuronales

La palabra aprendizaje en el campo del aprendizaje automático o *Machine Learning* (ML), describe el proceso de búsqueda automática que obtiene el mejor sistema de representación para un conjunto de datos en estudio. Se busca que la máquina aprenda sin ser expresamente programada para la tarea, y adquiera la capacidad de identificar patrones en los datos y realizar predicciones correctas.

El concepto de ML engloba múltiples técnicas de clasificación como árboles de decisión, algoritmos genéticos o redes neuronales artificiales [51]. En esta sección analizaremos de forma resumida los fundamentos de las redes neuronales artificiales, base sobre la que se sustentan redes neuronales convolucionales.

3.3.1. Regresión logística

Un problema de clasificación binaria responde a la pregunta de si una muestra n–dimensional $\vec{x}$, pertenece a una determinada categoría o no. Con este planteamiento, una muestra de entrenamiento estará formada por la dupla $(\vec{x}, y)$, donde y toma el valor 1 para indicar la pertenencia a la categoría y 0 para indicar la no pertenencia. En términos prácticos, se trataría de decidir por ejemplo, si una imagen de entrada contiene o no un gato.

En los problemas de clasificación binaria con aprendizaje supervisado, es habitual utilizar el algoritmo de *regresión logística* [52, p. 461], para determinar la probabilidad de pertenencia a la clase considerada. En términos numéricos, dada una muestra $\vec{x}$, queremos determinar su probabilidad de pertenencia a la clase $\hat{y}$, es decir $\hat{y} = P(y = 1|\vec{x})$. La función logística presenta la forma indicada en la Ecuación 3.1. Donde $\vec{x}$ es el vector que representa la muestra, $\vec{w}$ es un vector con el mismo número de elementos que $\vec{x}$ que suele denominarse *peso*, b es un número real que suele denominarse *sesgo* (*bias*) y σ es una función no lineal. La función σ suele ser la función sigmoide, dada por la Ecuación 3.2.

$$\hat{y} = \sigma\left(w^T x + b\right) \tag{3.1}$$

$$\sigma(z) = \frac{1}{1 + e^{-z}} \tag{3.2}$$

La función sigmoide garantiza que la salida $\hat{y}$ se encuentre entre 0 y 1, lo que la hace adecuada para expresar una probabilidad. En una regresión logística la tarea es encontrar los valores de los parámetros $\vec{w}$ y b, que hacen que $\hat{y}$ proporcione una buena estimación de la probabilidad de que y sea igual a 1.

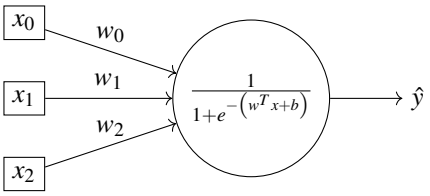


Figura 3.2: Modelo de regresión logística.

Para ajustar los parámetros del modelo de regresión logística, es necesario definir una función de coste. Así, dado un conjunto de muestras de entrenamiento y etiquetas asociadas de entrada $(x^{(i)}, y^{(i)})$, ajustaremos los parámetros $\vec{w}$ y b, para conseguir tras el entrenamiento, que las etiquetas estimadas $\hat{y}^{(i)}$ sean idealmente las etiquetas reales $y^{(i)}$, para todas las muestras del conjunto de entrenamiento. Para lograr este objetivo se define una *función de pérdida* o función de error $\mathcal{L}(\hat{y}, y)$ que mide la desviación respecto del funcionamiento ideal en la asignación de etiquetas.

Una de las posibles formas funcionales para la función de error viene dada por la función de error cuadrático (Ecuación 3.3), pero en regresión logística puede dar lugar a la elección de mínimos locales como respuesta óptima del modelo, por lo que no se considera muy robusta.

$$\mathcal{L}(\hat{y}, y) = (\hat{y} - y)^2 \tag{3.3}$$

Otra forma funcional para $\mathcal{L}$ es el error absoluto que viene dado por la Ecuación 3.4. Es una función que se considera más robusta ante la presencia de valores atípicos (*outliers*) en el conjunto de entrenamiento.

$$\mathcal{L}(\hat{y}, y) = |\hat{y} - y| \tag{3.4}$$

Y también puede utilizarse la función de pérdida de regresión logística, dada por la Ecuación 3.5. Esta función, presenta un comportamiento convexo y facilita la optimización, pero con un coste computacional superior.

$$\mathcal{L}(\hat{y}, y) = -[y \cdot log(\hat{y}) + (1 - y)log(1 - \hat{y})] \tag{3.5}$$

Las funciones de pérdida operan sobre muestras individuales. Para determinar el rendimiento de una configuración de los parámetros $\vec{w}$ y b sobre el conjunto de entrenamiento de m muestras, se define la *función de coste* $J(\vec{w}, b)$. La función de coste proporciona el promedio de la función de pérdida, para todas las muestras del conjunto de entrenamiento, como indica la Ecuación 3.6. Si se utiliza la función error cuadrático la función de coste se suele denominar función L2. En caso de utilizar la función error absoluto se denomina L1.

$$J(w, b) = \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}(\hat{y}^{(i)}, y^{(i)}) \tag{3.6}$$

Por tanto, para ajustar el modelo de regresión logística debemos encontrar el conjunto de parámetros $\vec{w}$ y b que minimizan la función de coste J. Para realizar esta minimización se usa el algoritmo de *gradiente descendiente*.

El algoritmo de gradiente descendiente trata de encontrar, mediante un procedimiento iterativo, los parámetros de la función de coste que minimizan la hipersuperficie de la función de coste J. En cada iteración de aprendizaje, se actualizan los valores de $\vec{w}$ y b en función del valor y dirección del gradiente en esa configuración. La Ecuación 3.7 expresa el ajuste a realizar, donde α es la *tasa o factor de aprendizaje* que controla la velocidad del ajuste.

$$\begin{aligned} w_i^+ &= w_i - \alpha \frac{\partial J(w, b)}{\partial w_i} \\ b^+ &= b - \alpha \frac{\partial J(w, b)}{\partial b} \end{aligned} \tag{3.7}$$

Los valores iniciales de $\vec{w}$ y b pueden establecerse a cero, rellenarse de forma aleatoria o usar funciones complejas de inicialización. Existen distintas estrategias de inicialización que tratan de minimizar la duración del entrenamiento, evitando que los gradientes tomen valores muy pequeños o muy grandes.

3.3.2. Redes neuronales

Una red neuronal [43] puede construirse apilando múltiples unidades como las utilizadas en el problema de regresión logística, donde la predicción de cada unidad ($\hat{y}$) se denomina activación. La Figura 3.3 muestra la estructura de una red neuronal sencilla. Cada una de las neuronas, representadas por un circulo, implementa el comportamiento de una unidad logística dado por la Ecuación 3.1.

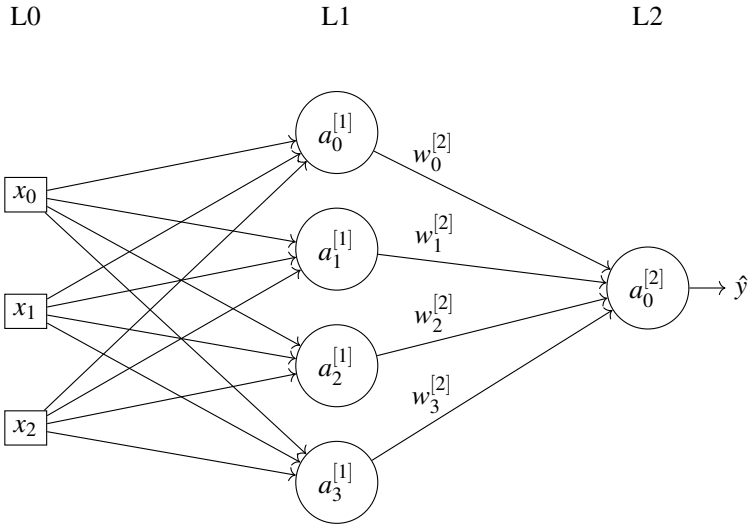


Figura 3.3: Estructura de una red neuronal básica con una única capa oculta

La salida de una neurona se denomina activación y se denota como $a_i^{[j]}$, donde el superíndice entre corchetes identifica la capa en la que se encuentra la neurona. En el caso presentado, la red consta de una única capa oculta que se identifica como L1, siendo L2 la capa de salida. La entrada no suele considerarse una capa ya que no presenta parámetros ajustables. Este tipo de estructura con una o más capas ocultas se denomina *perceptrón multicapa* (MLP).

Para ajustar los valores de los parámetros de una red neuronal (w_i y b_i), suele utilizarse el algoritmo de ajuste iterativo conocido como *backpropagation* [53]. En cada iteración, este algoritmo calcula la salida de la red ($\hat{y}$), aplicando los patrones del conjunto de entrenamiento (propagación directa). Comparando los valores de salida generados por la red con la etiqueta correcta de cada muestra, se calcula el valor de la función de coste J para la configuración de parámetros activa. Este algoritmo permite que la información de coste se propague hacia atrás en la red para calcular el gradiente. Así, se calculan las derivadas parciales de la función de coste con respecto a los pesos que unen la última capa oculta con la capa de salida, y entre las diferentes capas ocultas entre si, hasta la capa de entrada. Por último, se ajustan los parámetros de la red (w_i y b_i) usando la Ecuación 3.7 para minimizar el error en la dirección marcada por los gradientes. Este

procedimiento se repite hasta alcanzar, idealmente, el mínimo absoluto de la función de coste.

3.3.3. Redes neuronales profundas

Una red neuronal con pocas capas ocultas como la mostrada en la Figura 3.3 se suele considerar una red superficial. Del mismo modo si apilamos múltiples capas ocultas como muestra la Figura 3.4, obtenemos una red más 'profunda'. Las redes neuronales muy profundas pueden gestionar problemas que los modelos superficiales no pueden afrontar. Sin embargo, un modelo con mucha capacidad de aprendizaje puede sobreajustar su respuesta (*overfit*) al conjunto de aprendizaje. En este caso, dan lugar a un modelo que no tiene buena capacidad de *generalización*, es decir no proporciona respuestas correctas ante entradas no aprendidas durante el entrenamiento. No se puede decidir a priori qué profundidad es la necesaria para un problema dado.

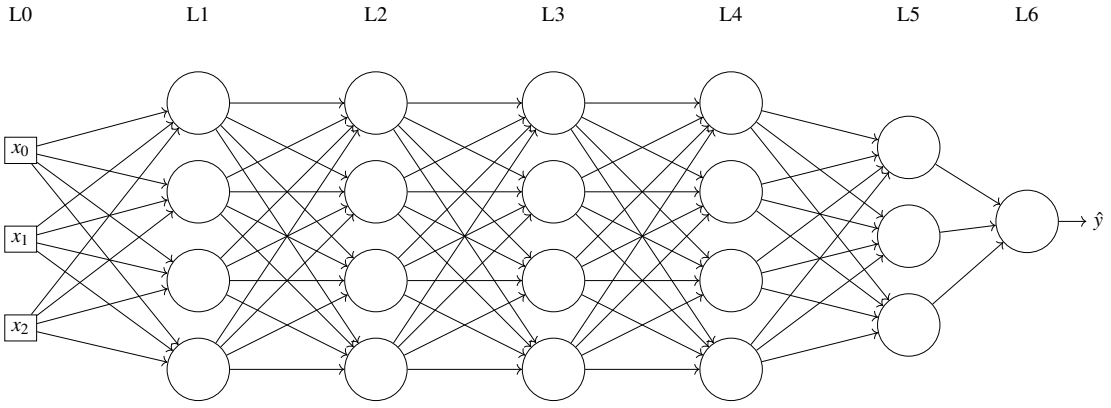


Figura 3.4: Estructura de una red neuronal más profunda que la mostrada en la Figura 3.3.

Cada neurona de una capa de la red de la Figura 3.4 utiliza todos los valores de activación de la capa anterior. Este tipo de conexionado da lugar a lo que se conoce como una capa completamente conectada (FC). Uno de los problemas fundamentales de las redes profundas completamente conectadas es la gran cantidad de parámetros a ajustar con el algoritmo de aprendizaje. Un número reducido de muestras de entrenamiento puede ser insuficiente para ajustar correctamente el modelo. El objetivo de una red neuronal es obtener un modelo que funcione bien, tanto sobre los datos que hemos usado para entrenarla, como sobre datos nuevos sobre los que debe realizar predicciones [54, p. 110].

Para reducir el sobreajuste se utilizan técnicas de *regularización* [54, p. 120], es decir cualquier modificación del algoritmo de aprendizaje que reduzca el error de generalización de la red. La forma más simple y habitual de regularización consiste en añadir

una penalización a la función de pérdida, proporcional a la magnitud de los pesos en el modelo. Esta técnica se conoce como regularización de pesos o *weight decay* [54, p. 427].

3.3.4. Gradiente descendente por minilotes y estocástico

Las redes profundas se ajustan mejor con conjuntos de datos muy grandes, pero entrenar con un conjunto de datos grande es un proceso muy lento. Por tanto, acelerar la fase de entrenamiento es importante. Una de las ideas que contribuyó a este objetivo es el algoritmo de gradiente descendente por minilotes (*mini-batches*).

La aplicación normal del algoritmo de gradiente descendente requiere procesar todos los ejemplos de entrenamiento (*epoch*) antes de aplicar una corrección de pesos. Esta estrategia no es preocupante con un número de ejemplos de entrenamiento bajo. Si el número de ejemplos es muy elevado, por ejemplo del orden de millones, la estrategia se vuelve poco práctica.

Un conjunto de entrenamiento muy grande puede dividirse en subconjuntos de entrenamiento más pequeños, y denominar a cada uno de estos conjuntos minilote. Podemos calcular la función de coste únicamente sobre los ejemplos del minilote, y aplicar *backpropagation* para actualizar los pesos de la red con este valor parcial de J. Se ha comprobado empíricamente, que este algoritmo permite reducir los tiempos de entrenamiento de las redes profundas con conjuntos de entrenamiento muy grandes.

La actualización de pesos asociada a un minilote de gradiente descendente se denomina *iteración*, obviamente tras procesar todos los minilotes del conjunto de entrenamiento habremos completado un ciclo de entrenamiento o *epoch*.

Un caso particular de gradiente descendente por minilotes se tiene al utilizar minilotes con un tamaño de lote unidad. En este caso cada ejemplo de entrenamiento constituye un minilote. Esta configuración se denomina *gradiente descendente estocástico* (SGD).

La evolución de la función de coste durante el proceso de entrenamiento en una configuración clásica (*batch*) suele ser monotónicamente descendente. Al utilizar minilotes, esta evolución se vuelve más ruidosa. Con SGD, la evolución del gradiente suele ir en dirección al mínimo global, pero en determinadas iteraciones puede diverger. Por lo que la evolución temporal de la función de pérdida puede ser rápida pero extremadamente ruidosa.

3.4. Plataformas de aprendizaje profundo

Aunque la implementación de redes neuronales puede realizarse desde cero por ejemplo con Python [55] y NumPy [56], lo habitual es trabajar con algún entorno de desarrollo que proporcione cierto grado de abstracción sobre los detalles de las redes con las que se trabaja.

En los últimos años, han aparecido distintos entornos de desarrollo con aprendizaje profundo como TensorFlow [57], Torch [58], Caffe [59] o Keras [60]. Estos entornos facilitan el desarrollo rápido de soluciones basadas en las tecnologías de aprendizaje profundo. Permiten utilizar las capacidades de cálculo paralelo ofrecidas por las modernas GPUs, para realizar el ajuste de los modelos de red en tiempos reducidos. Además, no requieren un conocimiento explícito de lenguajes para programación paralela tipo CUDA [61] u OpenCL [62]. Gracias a su diseño multiplataforma, permiten entrenar las redes en entornos con muchos recursos (GPU, memoria, disco). Y una vez ajustadas, y desplegar las redes entrenadas en entornos con menos recursos, como podrían ser teléfonos móviles u ordenadores monoplaca (SBC, *Single Board Computer*).

La Tabla 3.1 muestra algunas de las características de las plataformas más extendidas. Aunque, algunas de ellas son paquetes que requieren la instalación previa de otras plataformas base, como en el caso de Keras que requiere TensorFlow u otra plataforma que proporcione operaciones de tensores a bajo nivel.

Tabla 3.1: Caracteristicas de algunas de las plataformas de aprendizaje profundo más extendidas.

Nombre	Lenguajes	Fundado por	Licencia
Caffe	C++,Python,Cuda	Berkeley AI Research	BSD 2
PyTorch/Caffe2	C++,Python,Cuda	Idiap Research Institute	BSD modificada
TensorFlow	C++,Python	Google	Apache 2.0
Keras	Python	François Chollet	MIT
Theano	Python,Cuda	Universidad de Montreal	BSD 2
Apache MXNet	C++,Python,Cuda	Fundación Apache	Apache 2.0

La gran variedad de entornos disponibles genera dudas a la hora de escoger uno de ellos con cierta seguridad de uso a medio plazo. Suele haber una gran comunidad de desarrolladores detrás de cada uno de ellos y hasta con el soporte de grandes grupos empresariales. Esta efervescencia en el desarrollo provoca muchos cambios en el código e incompatibilidades en cada actualización y, en ocasiones, hasta en los archivos de configuración lo que en ocasiones genera más de un problema.

Además, a pesar de que existen herramientas de adaptación entre los formatos de configuración y datos utilizados por distintos entornos, en muchas ocasiones la compatibilidad no es perfecta. De esta forma, la elección de uno de ellos en un proyecto suele suponer una ligadura a largo plazo.

3.5. Redes neuronales convolucionales

Las arquitecturas clásicas de redes neuronales presentaban un comportamiento ineficiente en tareas de visión por computador. Una imagen en color supone una entrada muy grande para una red neuronal en términos de vector de entrada. En una red clásica completamente conectada el número de conexiones y parámetros asociados sería enorme para lograr un ajuste eficiente.

En *Machine Learning*, una red neuronal convolucional (CNN o ConvNet) es un tipo de red neuronal artificial en el que las interconexiones entre las neuronas se inspiran en la estructura del cortex visual animal [63]. Las neuronas corticales responden a los estímulos de una zona limitada denominada *campo receptivo*. Los distintos campos receptivos se solapan parcialmente hasta completar el campo visual.

Las neuronas del cortex visual actúan como filtros locales sobre el campo de entrada. Se han identificado células simples que responden a determinados patrones de borde y, otras complejas, con mayor campo receptivo que presentan invariancia en su respuesta a la posición exacta del patrón. La respuesta de una neurona cortical a un estímulo en su campo receptivo puede aproximarse matemáticamente por una operación de convolución. Las CNN son variantes de perceptrón multicapa bioinspiradas y con aprendizaje supervisado.

Puede considerarse que las actuales CNN están basadas en las ideas del *Neocognitron* de Kunihiko Fukushima [64] presentado en 1980. Tras casi dos décadas, en 1998 Yan LeCun et al. [44] introdujeron el ajuste de su modelo convolucional con backpropagation. Y más de una década después, en 2012, se produjo la explosión de los modelos convolucionales gracias al trabajo de Alex Krizhevsky et al. [65] y su modelo AlexNet, que fue el primero en utilizar GPUs para el entrenamiento de las redes convolucionales con resultados sobresalientes. Una CNN actual contiene varias capas ocultas especializadas y con una jerarquía definida, de forma que las primeras capas pueden detectar líneas o curvas y se van especializando hasta que las capas profundas son capaces de reconocer formas o conceptos complejos.

Los elementos habituales de una CNN actual incluyen capas convolucionales, capas u operaciones de agrupamiento (*pooling*), no linealidades (ReLU), capas completamente

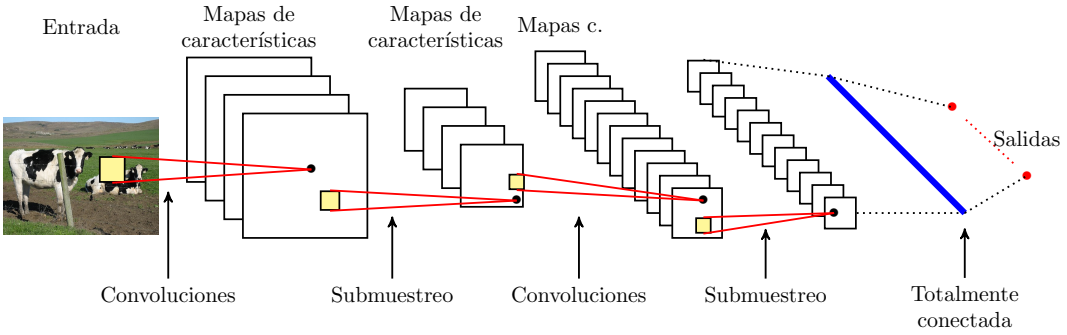


Figura 3.5: Estructura típica de una red neuronal convolucional.

conectadas (FC) y funciones de pérdida (*loos*). La Figura 3.5 muestra la arquitectura típica de una red neuronal convolucional.

3.5.1. Funciones de activación

La función de activación más utilizada en las ConvNets es la conocida como ReLu (*Rectifier Linear Unit*) que se define como la parte positiva de su argumento tal y como expresa la Ecuación 3.8, siendo x la entrada a una neurona.

$$ReLu(x) = max(0, x) \tag{3.8}$$

En 2011 se demostró que esta función permite realizar un entrenamiento más eficiente de las redes neuronales profundas que las funciones de activación más utilizadas hasta entonces [66] como la función logística o la tangente hiperbólica. Presentando entre sus ventajas un bajo coste de implementación computacional, mejor propagación del gradiente e invariancia a la escala ya que $max(0, ax) = a \cdot max(0, x)$ para cualquier $a \geq 0$. Y entre sus inconvenientes, esta función no es diferenciable en cero, no está centrada entorno a cero y no tiene una excursión limitada en Y.

3.5.2. Capas completamente conectadas y SoftMax

Las capas FC suelen aparecer en las etapas finales de una red convolucional utilizando las características generadas previamente por capas convolucionales. La Figura 3.7a presenta la estructura habitual de una capa tipo FC y pone de manifiesto uno de los hándicaps más importantes de este tipo de capas, el elevado número de parámetros que deben ajustarse w_i y b_i mediante el algoritmo de entrenamiento. En particular una capa de n neuronas con un vector de entrada de m variables requeriría el ajuste de $[(m + 1) \cdot n]$ parámetros.

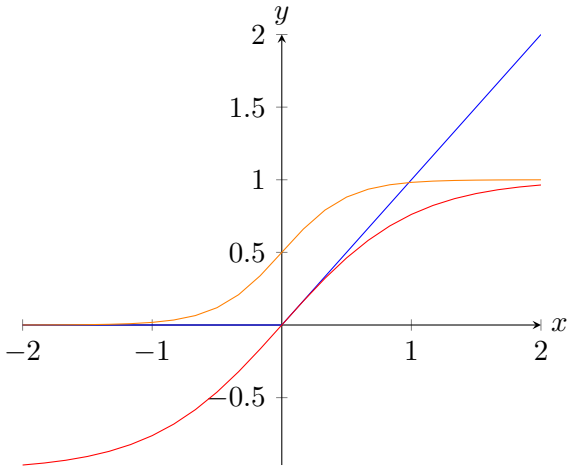


Figura 3.6: Comparativa del comportamiento de una función de activación tipo ReLu en azul, con la función sigmoidea en naranja y la función tangente hiperbólica en rojo.

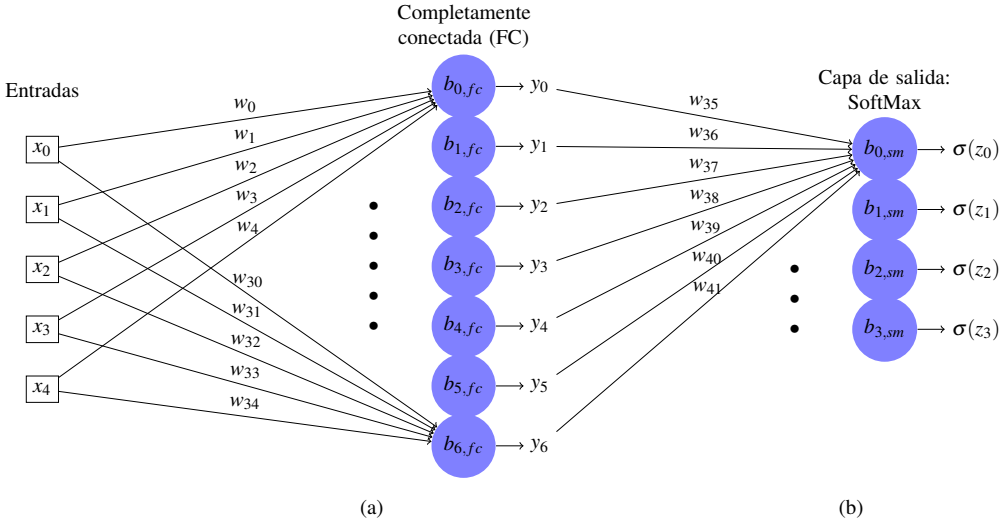


Figura 3.7: Estructura de una capa completamente conectada (a). Interconexión de una capa FC con una capa SoftMax para la determinación de las probabilidades de pertenencia a las categorías contempladas (b).

Una de las ubicaciones habituales de las capas FC es como entrada de la capa de salida. En clasificación, la capa de salida utiliza habitualmente como función de activación la función logística (Ecuación 3.1) para clasificación binaria, y *SoftMax* [67, p. 393] para clasificación multi-clase. Las capas ocultas, tradicionalmente usaban la función sigmoide, pero hoy en día, como ya se ha mencionado, es más habitual el uso de la función *ReLU* (Ecuación 3.8). Por tanto, una capa de salida tipo *SoftMax* utiliza una función de activación *SoftMax* o función exponencial normalizada, para proporcionar una probabilidad de pertenencia en el rango [0,1] a una de las categorías de clasificación contempladas.

La entrada z a la función de activación SoftMax se genera como en una unidad logística a partir de un conjunto de pesos w_i y sesgos b_i como expresa la Ecuación 3.9. Y a partir de

este valor se determina la activación de cada una de las salidas $\sigma(z_i)$ de la red neuronal según la expresión de la Ecuación 3.10.

$$z_i = w_0 y_0 + \cdots + w_{m-1} y_{m-1} + b_i \tag{3.9}$$

$$\sigma(z_i) = \frac{e^{z_i}}{\sum_{j=0}^{k-1} e^{z_j}} \tag{3.10}$$

De esta forma, una salida [0.1 0.8 0.1 0.0] para la capa de salida de la Figura 3.7b nos indicaría un 80 % probabilidad de pertenencia a la clase 1, un 10 % a las clases 0 y 2 y nula probabilidad de pertenencia a la clase 3.

3.5.3. Capas convolucionales

La capa convolucional es el bloque fundamental de una CNN, contiene un conjunto de filtros, o kernels, ajustables con pequeño campo receptivo. Una convolución[1] en ML realiza la operación producto escalar del filtro sobre el campo receptivo, es decir cada elemento de la matriz de salida es la suma de los productos, elemento a elemento, de las matrices de entrada y filtro. La Figura 3.8 muestra en las zonas resaltadas por un borde azul la obtención del primer elemento de la matriz de salida. A continuación, el filtro se desplaza sobre la imagen para obtener el resto de los valores de la matriz de salida. Habitualmente se utilizan filtros con tamaño de lado impar.

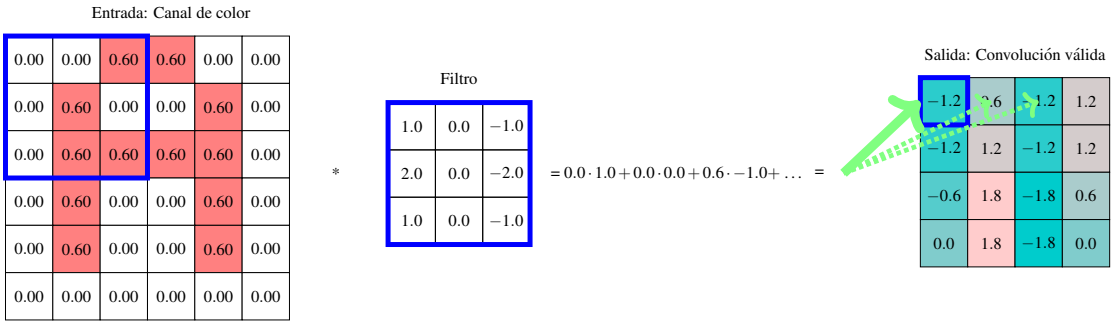


Figura 3.8: Convolución válida del canal de color rojo de una imagen de tamaño 6x6 con un filtro de tamaño 3x3.

Algebráicamente, la Ecuación 3.11 refleja la operación convolución G, según se entiende en ML, para cada elemento de salida $g(i, j)$. Siendo H un canal de la imagen y f la extensión del filtro F.

[1] Aunque se utiliza la denominación convolución en realidad se realiza una correlación cruzada o covarianza.

$$G = H * F \rightarrow g(i,j) = \sum_{u=-f}^{f} \sum_{v=-f}^{f} H[u,v]F[i+u, j+v] \tag{3.11}$$

Durante el proceso de aprendizaje de una capa convolucional, la red ajusta filtros que se activan al detectar características específicas en una posición de la entrada. En este caso, el filtro únicamente tiene 9 parámetros que ajustar.

3.5.3.1. Relleno

El proceso de desplazamiento del filtro sobre la imagen descrito, hace que los valores de los píxeles de los bordes se tengan en cuenta en la convolución un menor número de veces de lo que se tienen en cuenta los pixeles centrales. Esto haría que la información de los bordes contribuyese menos en una red convolucional. Para preservar la información de los bordes de la imagen, se definen algunas variantes [54] de *convolución con relleno* que se recogen en la Tabla 3.2. En las convoluciones con relleno, se añaden p pixeles en el borde de la imagen de entrada, que habitualmente tienen valor cero, para poder extender el recorrido del filtro. Por tanto, la convolución mostrada en la Figura 3.8 es de tipo válida donde el filtro se desplaza de pixel a pixel. Dado que el tamaño del filtro es $f = 3$ y la imagen es de tamaño $n = 6$, el tamaño de la matriz de salida en estas condiciones será $(n - f + 1) \times (n - f + 1) = 4 \times 4$.

Tabla 3.2: Variantes de la convolución con relleno para una imagen de lado n y un tamaño de filtro f.

Variante	Descripción	Tamaño de salida
Válida	No se añade relleno, el filtro se desplaza únicamente los píxeles de la imagen.	$(n - f + 1) \times (n - f + 1)$
Igual	Se añade un relleno a cero hasta conseguir que la salida tenga el mismo tamaño que la imagen.	$n \times n$
Completa	Se añade un relleno a cero en los bordes de la imagen para que cada pixel se visite k veces en cada dirección.	$(n + f - 1) \times (n + f - 1)$

3.5.3.2. Stridded Convolutions

Además existe un grado de libertad adicional a la hora de desplazar el filtro sobre la entrada. Si en vez de desplazar el filtro pixel a pixel en ambas direcciones se desplaza en más de un pixel realizando un submuestreo sobre la entrada con un tamaño de paso s mayor a uno, se habla de *Stridded Convolutions*. Obviamente, en este caso el tamaño de la matriz de salida será menor al de la convolución equivalente sin salto. De esta forma, utilizando la nomenclatura utilizada hasta ahora, el tamaño de la matriz de salida de

una operación de convolución genérica G vendrá dado por la Ecuación 3.12. Donde n es lado de la imagen, p es el número de píxeles de relleno, f es el tamaño del filtro, s es el tamaño de paso para desplazar el filtro y el operador $\lfloor x \rfloor$ denota la parte entera de x.

$$size(G) = \lfloor \frac{n + 2p - f}{s} + 1 \rfloor \times \lfloor \frac{n + 2p - f}{s} + 1 \rfloor \tag{3.12}$$

3.5.3.3. Convolución sobre volúmenes

Dado que la entrada a una red neuronal convolucional suele ser una imagen en color, debemos extender la definición de la convolución 2D a una implementación para las 3 matrices que almacenan los canales de color de una imagen.

En una convolución sobre imágenes RGB que contiene 3 canales de color, se definen tres filtros uno para cada canal de color. Cada filtro se aplica sobre un plano de color realizando la multiplicación elemento a elemento y sumando sus resultados, pero en este caso no se obtienen 3 matrices de salida sino una única matriz en la que cada elemento se obtiene al sumar las contribuciones de los tres filtros. La Figura 3.9 muestra gráficamente la aplicación de una convolución sobre una imagen RGB de tamaño 6×6 con un filtro $3 \times 3 \times 3$ con el que se obtiene un mapa de activación 4×4. En este caso, el filtro requeriría el ajuste de 27 parámetros.

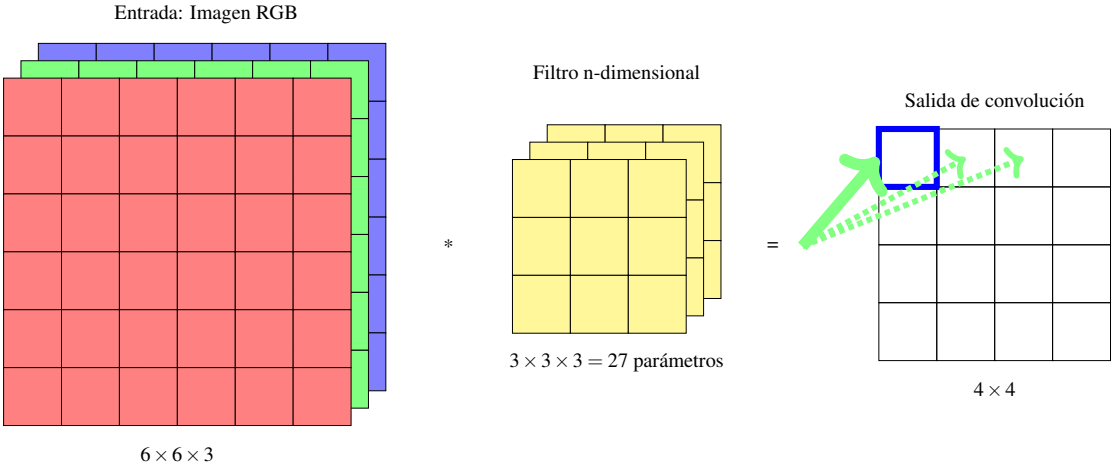


Figura 3.9: Convolución válida de una imagen RGB con un filtro de tamaño $3 \times 3 \times 3$ y tamaño de paso 1.

Una capa convolucional no usa un único filtro sino que define varios filtros que se ajustarán para detectar distintas características y que por tanto generarán varias salidas de convolución. A cada uno de estos mapas se les añade un sesgo (bias), que también se ajustará en el proceso de aprendizaje. Y por último, se aplicará una no linealidad de bajo coste computacional (ReLu habitualmente) para determinar la activación. Apilando las salidas anteriores de todos los filtros contemplados, generamos la salida de la capa

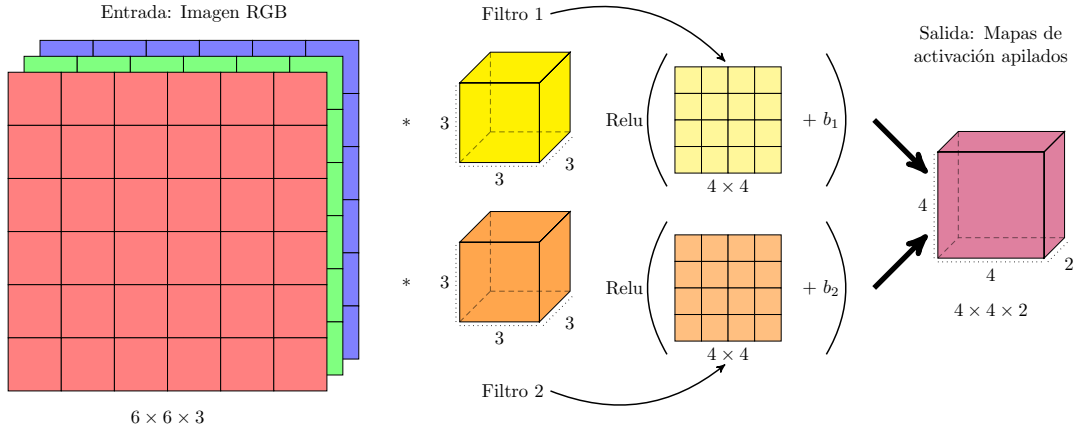


Figura 3.10: Ejemplo de capa convolucional con dos filtros, adición de sesgo y no linealidad. La salida muestra de forma agrupada los dos mapas de activación obtenidos.

convolucional como muestra la Figura 3.10. En este caso, al contener la capa únicamente dos filtros, la profundidad de la salida de la capa es únicamente dos. Es habitual indicar el número de canales que proporciona una capa convolucional como tercer parámetro por analogía con el número de canales de una imagen.

Una de las características de las capas convolucionales es el reducido número de parámetros ajustables en comparación con las capas completamente conectadas. Un detector de características que es útil en una parte de la imagen (como un detector de bordes horizontales) debe ser igual de válido en otra parte de la imagen. Teniendo en cuenta esta idea, no es necesario ajustar filtros distintos para distintas zonas de la imagen. Si comparamos el número de parámetros a ajustar en una primera capa FC frente a una capa convolucional observaremos que las capas convolucionales requieren menos parámetros ajustables. Cada filtro 3×3 requiere ajustar 27 parámetros más el *bias*, por lo que una capa más realista que la del ejemplo con diez filtros, únicamente requeriría ajustar 280 parámetros para generar 160 activaciones. Una capa FC desde la imagen RGB requeriría ajustar 17280 parámetros.

Otra de las características de las *ConvNets* es su capacidad de invariancia a la traslación, es decir que un perro desplazado unos cuántos pixeles debe seguir siendo detectado como un perro. La propia estructura de la red, filtros que cubren la imagen de entrada, hace que el perro desplazado proporcione un conjunto de características muy semejante, por lo que la etiqueta final debería ser la misma.

Además, desde el punto de vista del paralelizaje del cómputo, cada elemento de salida de una capa convolucional depende de un pequeño número de entradas. Este hecho facilita el fragmentado de operaciones en CPUs multicore o GPUs.

3.5.4. Capas de agrupamiento

Las *ConvNets* usan también capas de agrupamiento (*pooling*) que permiten realizar un submuestreo reduciendo el tamaño de la representación. Con esta operación se logra acelerar el procesamiento y además, se hace que las características generadas sean más robustas. Existen varios tipos de agrupamiento, pero uno de los más utilizados es el agrupamiento a máximo (*Max pooling*). En este algoritmo cada una de las regiones se representa a partir del valor máximo contenido en la región. El tamaño de filtro determina la región a muestrear y la ventana de selección se desplaza sobre la entrada en función del tamaño de paso definido. El caso representado en la Figura 3.11 representa una operación *Max pooling* con tamaño de filtro dos y tamaño de paso 2.

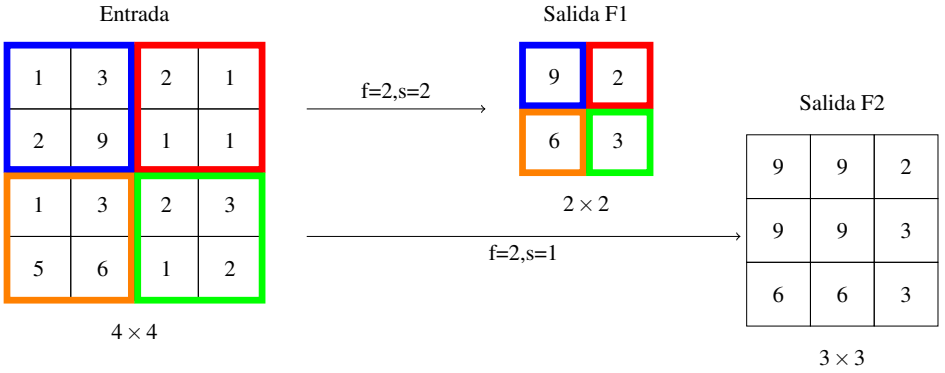


Figura 3.11: Aplicación de un agrupamiento a máximo con un filtro de tamaño 2×2 y paso 2 (Salida F1), y un filtro de tamaño 2×2 y paso 1 (Salida F2).

Existe un segundo tipo de agrupamiento que representa el área seleccionada por el filtro a partir de su valor medio. Este tipo de agrupamiento se denomina *Average pooling*.

Una capa de agrupamiento no tiene parámetros que deban ajustarse en la etapa de entrenamiento. Se elige en la fase de diseño el tipo de agrupamiento, el tamaño del filtro y el tamaño de paso. No es muy habitual añadir relleno en este tipo de capas.

3.5.5. De las primeras redes convolucionales a las redes modernas

Las CNN han tenido una evolución muy lenta hasta llegar a su nivel de utilización actual. Desde el *Neocognitron* de 1980, pasando por LeNet-5 en 1998, hasta el cambio que supuso el modelo AlexNet en 2012, ha habido grandes periodos en los que las CNN no han generado demasiado interés. En esta sección describimos las características de los modelos neuronales que han dado lugar a las *redes residuales profundas* que utilizamos como base de nuestro sistema de detección.

3.5.5.1. LeNet-5

En 1998 Yann LeCun et al. [44] propusieron una arquitectura de red neuronal para reconocimiento de caracteres tipográficos y manuscritos en escala de grises, que denominaron LeNet-5 cuya arquitectura original se muestra en la Figura 3.12.

Este modelo usa elementos que hoy en día no se usan en las ConvNet pero supone un primer modelo de las estructuras que se usan hoy en día. Por ejemplo, usa la función tangente hiperbólica como función de activación, además realiza agrupamientos promedio en vez de máximos y la capa de agrupamiento tiene también una función de activación. Por otro lado, tiene un número de parámetros ajustables en torno a 60000, muy inferior al que podemos encontrar hoy en día en el rango de 10 a 100 millones de parámetros.

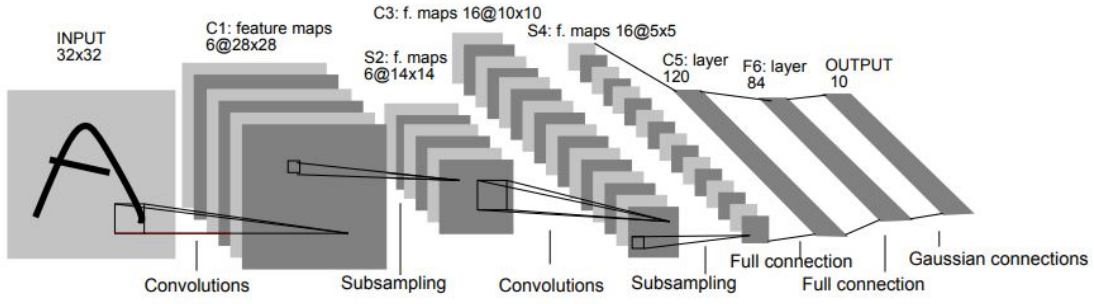


Figura 3.12: Imagen original de la arquitectura LeNet-5.

La arquitectura consta de dos capas convolucionales, cada una de ellas con su correspondiente capa de agrupamiento promedio (*subsampling*), seguidas de una *capa de aplanamiento convolucional* que aplana los mapas de características desenrollando sus componentes para utilizarlos como entrada a una capa FC. La salida aplanada se usa como entrada a un bloque de dos capas FC y un clasificador softmax.

3.5.5.2. El modelo AlexNet

El modelo AlexNet [65] logró el éxito al ganar la competición *ImageNet Large Scale Visual Recognition*. Alcanzó un error top-5[2] del 15.3 %, un 10.9 % inferior al del segundo mejor clasificado (26.2 %). AlexNet fue la primera red convolucional que usó una GPU para mejorar su rendimiento.

El resultado principal del trabajo fue que la profundidad del modelo era un punto clave para lograr su elevado rendimiento. Sin embargo, era computacionalmente muy costoso para el hardware de la época, pero fue posible gracias al reparto de la carga sobre dos

[2]Error top-5: Posibilidad de no encontrar la etiqueta correcta para una imagen dada entre las cinco mejores predicciones generadas por la red.

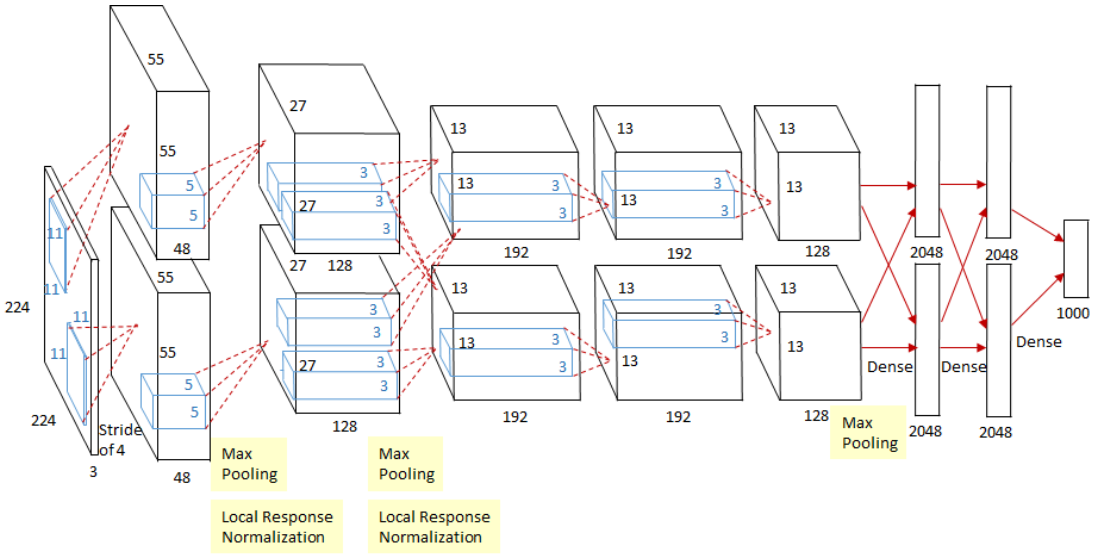


Figura 3.13: Estructura original de la red AlexNet que muestra la distribución de la carga entre las dos GPUs utilizadas.

GPUs durante el entrenamiento. La Figura 3.13 muestra la arquitectura de esta red según se publicó originalmente. En el gráfico se puede observar la distribución de recursos entre las dos GPUs y los requerimientos de interconexión entre las GPUs en determinados niveles de profundidad de la red. La arquitectura presenta 5 capas convolucionales, 3 capas de agrupamiento máximo, 2 capas de normalización, 2 capas completamente conectadas y una capa softmax.

Sin el requisito de reparto de la red entre dos GPUs, el modelo es similar a la estructura LeNet-5 ya descrita, pero mucho mayor. Además, AlexNet usa ReLu como función de activación y capas de agrupamiento máximo. Como elemento diferencial, utilizaba un conjunto de datos incomparablemente mayor: ImageNet [36].

Para minimizar el sobreajuste de la red usaron técnicas de aumento de datos para el conjunto de entrenamiento, y la técnica de regularización conocida como Dropout [68]. El aumento de datos se realizó reflejando la imagen original (*mirroring*) o extrayendo cortes aleatorios de menor tamaño. En la técnica de *Dropout*, una neurona se elimina de la red con una probabilidad del 50 %. Al eliminar la neurona, no contribuye en la propagación directa ni en la retropropagación. Así, cada muestra de entrenamiento atraviesa una red distinta, con lo que se ajustan los pesos de forma más robusta respecto de todas las muestras. Gracias al gran conjunto de datos disponible, sus aproximadamente 60 millones de parámetros ajustables podían entrenarse para obtener el sorprendente rendimiento reportado en el ILSVRC2012.

3.5.5.3. El modelo VGG

Otro modelo de red destacable es el VGG [69], desarrollado por Karen Simonyan y Andrew Zisserman. Trataron implementar un modelo sencillo, con el objetivo de reducir el número de hiperparámetros.

Una red VGG utiliza convoluciones de tipo igual, por lo que preserva el tamaño del mapa de activación. Todos los filtros que utiliza son de tamaño 3×3 y paso 1. Las capas de agrupamiento usan filtros de tamaño 2×2 y paso 2, lo que reduce a la mitad el tamaño de los mapas de características al aumentar la profundidad de la red.

En las primeras capas de la red se apilan dos capas convolucionales (x2 en la Figura 3.14), antes de aplicar una capa de agrupamiento máximo. En las capas más profundas, este apilamiento es triple (x3). Podemos observar que al aumentar la profundidad se reduce el tamaño del mapa de activaciones y se incrementa el número de canales, como muestra la Figura 3.14.

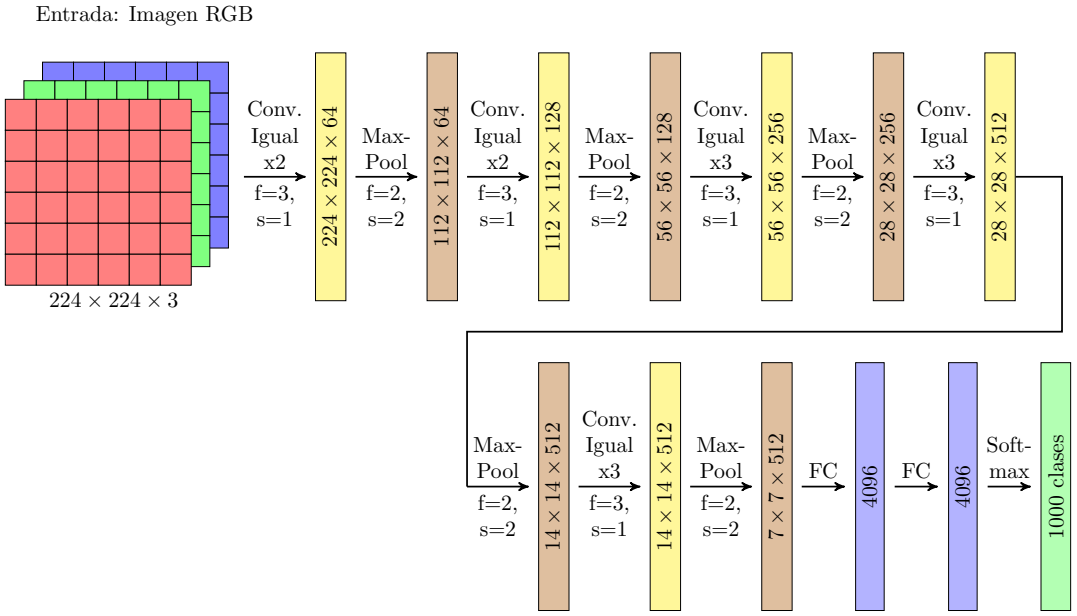


Figura 3.14: Estructura de capas de una red VGG-16.

Las redes estudiadas en el trabajo presentan de 11 a 19 capas. El rango de hiperparámetros entrenables va desde los 133 hasta los 144 millones. Este modelo estableció la idea de reducir el tamaño del mapa de activación a la mitad y duplicar el número de filtros tras cada submuestreo. De entre todas las variantes descritas en [69], el modelo VGG-16 en su variante D es quizá el más utilizado en trabajos posteriores, con un total de 138 millones de hiperparámetros.

3.5.6. Redes residuales profundas

Las redes neuronales muy profundas son difíciles de entrenar debido a dos tipos de fenómenos, el desvanecimiento del gradiente (*vanishing gradient problem*) y los gradientes explosivos (*exploding gradients*) [70, 71]. Un modelo que sufre desvanecimiento de gradientes, aprende muy despacio durante la fase de entrenamiento y el error se estanca en un valor elevado. En caso de aparecer gradientes explosivos, el modelo no aprende el conjunto de entrenamiento, presenta valores de error muy elevados, o grandes cambios en el error en cada actualización de parámetros.

Las redes residuales profundas (*ResNets*) definidas en [72], tratan de evitar estos dos efectos a partir de una nueva estructura de conexionado denominada *bloque residual*, que se muestra en la Figura 3.15. Esta estructura introduce una conexión que permite mezclar las activaciones de entrada a un bloque convolucional apilado, con el resultado del bloque convolucional ($\mathcal{F}(X)$), justo antes de aplicar la no linealidad (ReLu). Las conexiones de salto se denominan *shortcut connections* o *skipping connections* y no añaden parámetros entrenables adicionales a la red, por lo que el conjunto sigue pudiéndose entrenar de extremo a extremo utilizando SGD con *backpropagation*.

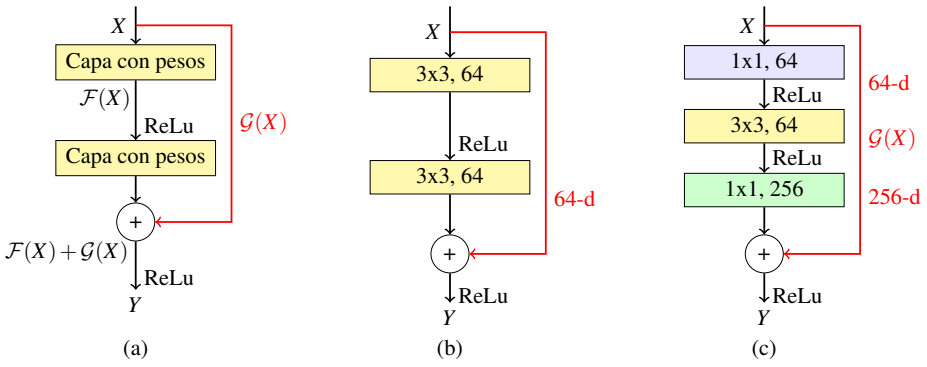


Figura 3.15: (a) Estructura de un bloque residual con el que se construyen las redes tipo ResNet. Aparece resaltada en rojo la conexión que conecta las activaciones de entrada con las de salida de las capas apiladas, justo antes de la no linealidad.

La conexión $\mathcal{G}(X)$ denota habitualmente la operación identidad. En determinadas ocasiones, cuando se ha producido un submuestreo, es necesario variar el tamaño del vector de entrada a sumar antes de aplicar la no linealidad. La salida de un bloque residual se puede expresar como indica la Ecuación 3.13. Donde, X e Y representan los vectores de entrada y salida de las capas consideradas. La función $\mathcal{F}(X, W_i)$ representa el mapeo residual que debe entrenarse. Y la matriz W_s permite obtener una proyección lineal del vector X, para igualar las dimensiones del vector antes de realizar la suma elemento a elemento. La función residual $\mathcal{F}$ suele incluir dos o tres capas convolucionales de igual o

distintas características como muestran las Figuras 3.15b y 3.15c. El número de capas más adecuado se determina de forma empírica.

$$Y = \mathcal{F}(X, W_i) + W_s X \tag{3.13}$$

Una red residual se construye apilando múltiples bloques residuales hasta conseguir la profundidad deseada como muestra la Figura 3.16b. Esta figura permite identificar como se construye una red residual a partir de una red convolucional estándar. Podemos apreciar la existencia de conexiones de salto, como las definidas en la Figura 3.15. Las conexiones que implementan la operación identidad están identificadas en rojo. Las conexiones dibujadas en azul, se encargan de hacer compatible el tamaño del mapa de activación de entrada, con el nuevo tamaño de mapa y número de filtros tras una operación de agrupamiento.

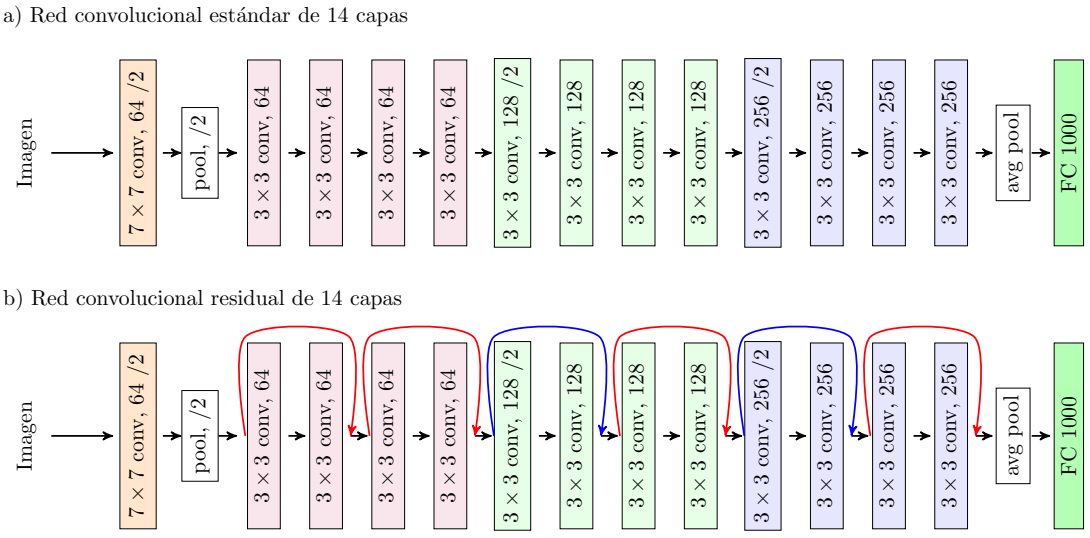


Figura 3.16: Comparación de la estructura de una red convolucional estándar de 14 capas apiladas (a), frente a una red convolucional residual de 14 capas. En (b) se puede observar las conexiones de salto.

Las capas convolucionales utilizadas se inspiran en la filosofía de las redes VGG descritas en la Sección 3.5.5.3. Utilizan filtros 3×3 y dos reglas de diseño básico: para un mismo tamaño del mapa de características las capas tienen el mismo número de filtros; y al reducir el mapa de activaciones a la mitad, el número de filtros se duplica. El submuestreo se realiza usando un tamaño de paso 2 en la primara capa del bloque residual. La red finaliza con una capa de agrupamiento promedio y una capa FC de 1000 unidades con algoritmo SoftMax.

3.5.6.1. ResNet50 y ResNet101

La Tabla 3.3 muestra las características de construcción de varias redes ResNet de interés. Las redes *ResNet50* y *ResNet101* utilizan bloques residuales compuestos por tres capas convolucionales con filtros de distinto tamaño, que se repiten un número variable de veces hasta alcanzar la capa FC. El coste computacional (FLOPs[3]) de una ResNet50 respecto de una ResNet34 es pequeño, gracias al menor tamaño de los filtros empleados en los bloques residuales.

Tabla 3.3: Caracteristicas de los bloques de construcción de algunas redes ResNet de interés para este trabajo.

Capa	Tamaño de salida	34 capas	50 capas	101 capas
conv1	112×112	7×7, 64, paso 2		
pool		3×3, agrupación máxima, paso 2		
conv2	56×56	$\begin{bmatrix} 3 \times 3, 64 \\ 3 \times 3, 64 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 64 \\ 3 \times 3, 64 \\ 1 \times 1, 256 \end{bmatrix} \times 3$
conv3	28×2	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 128 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$	$\begin{bmatrix} 1 \times 1, 128 \\ 3 \times 3, 128 \\ 1 \times 1, 512 \end{bmatrix} \times 4$
conv4	14×14	$\begin{bmatrix} 3 \times 3, 256 \\ 3 \times 3, 128 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 6$	$\begin{bmatrix} 1 \times 1, 256 \\ 3 \times 3, 256 \\ 1 \times 1, 1024 \end{bmatrix} \times 23$
conv5	7×7	$\begin{bmatrix} 3 \times 3, 128 \\ 3 \times 3, 256 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$	$\begin{bmatrix} 1 \times 1, 512 \\ 3 \times 3, 512 \\ 1 \times 1, 2048 \end{bmatrix} \times 3$
fc	1×1	agrupación promedio, 1000-uds., SoftMax		
FLOPs		3.6×10^9	3.8×10^9	7.6×10^9

La Tabla 3.4 compara el rendimiento de una VGG-19, con modelos no compuestos de ResNet, sobre el conjunto de validación de *ImageNet*. Como puede observarse, las arquitecturas ResNet preservan la reducción de la tasa de error *top-1* y *top-5* al aumentar la profundidad de las redes, incluso en un modelo tan profundo como el ResNet-152. La mejora obtenida por el modelo ResNet-50 frente al modelo VGG [69] supera los 3.5 puntos porcentuales en la tasa de error top-1.

[3]Operaciones de coma flotante (floating point operations)

Tabla 3.4: Comparativa de las tasas de error porcentuales de varios modelos residuales frente al modelo VGG sobre el conjunto de de validación del dataset ImageNet.

Modelo	Error top-1	Error top-5
VGG-19	24.4	7.1
ResNet-50	20.74	5.25
ResNet-101	19.87	4.60
ResNet-152	19.38	4.49

3.5.7. Transferencia de aprendizaje

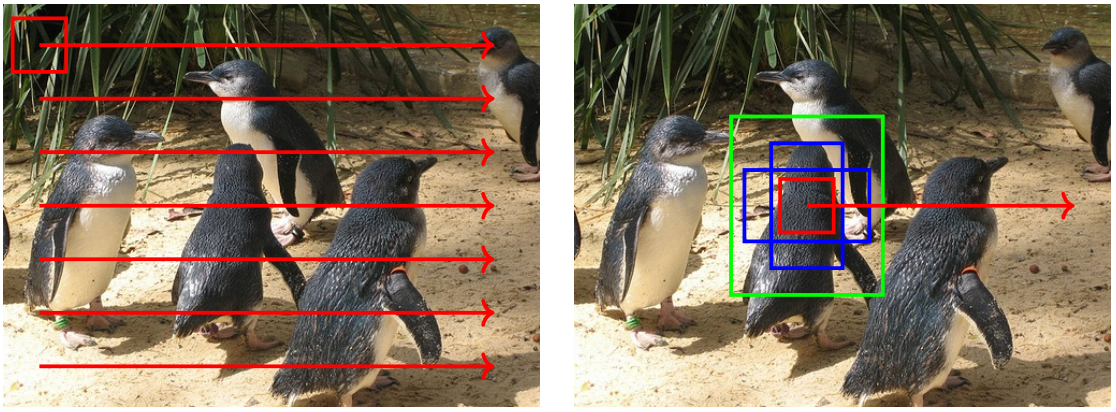
La transferencia de aprendizaje o *transfer learning* (TL), se basa en la reutilización del conocimiento adquirido al resolver un problema, aplicándolo a un problema distinto, pero relacionado [73].

En muchas aplicaciones, como en nuestro caso, es muy difícil construir grandes conjuntos de datos bien etiquetados. El coste de adquisición de los datos y el coste de anotación, limitan las posibilidades de desarrollo de grandes conjuntos. La transferencia de aprendizaje es una posible solución al problema de insuficiencia de datos en el conjunto de entrenamiento.

Las modernas redes neuronales profundas, tienden a aprender en las primeras capas características que recuerdan a filtros de Gabor o manchas de color [10]. Este fenómeno no solo ocurre con distintos conjuntos de entrenamiento, sino también al cambiar el objetivo de aprendizaje. También se sabe que las características que aprende la última capa, están fuertemente ligadas al conjunto de datos modelado y la tarea a ejecutar. Una red entrenada sobre un conjunto de imágenes masivo habrá aprendido características de bajo y alto nivel. Este aprendizaje se traduce en el ajuste de filtros, que pueden llegar a ser útiles en una tarea distinta. En nuestro caso, reutilizaríamos el conocimiento almacenado en una red entrenada para clasificar de imágenes macroscópicas, para localizar y clasificar granos de polen.

3.6. Detección de objetos con CNN

Para realizar la detección de un objeto, necesitamos conocer la clase del objeto y también la posición y tamaño del BBox que lo contiene. Como un objeto puede estar localizado en cualquier posisción y escala en la imagen, es natural buscarlo en todas las zonas de la imagen [74, 75]. En el enfoque clásico, para cada imagen, se utiliza una ventana deslizante para buscar en cada posición dentro de una imagen como muestra la Figura 3.17. Sin



(a) (b)

Figura 3.17: Ilustración del funcionamiento del método de la ventana deslizante (a) y su extensión al uso de ventanas con con distintas relaciones de aspecto y escalas (b).

embargo, los objetos pueden tener relaciones de aspecto distintas a un simple cuadrado. Además, el tamaño en la imagen dependerá del tamaño del objeto y de la distancia a la cámara. El proceso de análisis de una imagen será extremadamente lento, si usamos una red profunda para clasificar la imagen encerrada por la ventana, en cada posición y distintas escalas.

Las redes neuronales convolucionales profundas han mejorado de forma significativa la clasificación de imágenes y la precisión en la detección de objetos. La detección de objetos es una tarea mucho más compleja que la clasificación de imagen y requiere métodos más complejos para llevarla a cabo. Hasta la llegada del modelo *Fast R-CNN*, la mayor parte de los enfoques en detección se basaban en el entrenamiento de modelos multietapa con un funcionamiento bastante lento.

3.6.1. El algoritmo Selective Search

Selective Search (SS) [76] es un algoritmo de propuesta de regiones. Se diseñó para ser rápido y con muy alta sensibilidad (*recall*). Se basa en la agrupación jerárquica de regiones similares, basándose en parámetros como compatibilidad de color, textura, tamaño y forma, para realizar una segmentación independiente del tipo de objeto. La Figura 3.18[4] muestra la idea global de agrupamiento jerárquico del algoritmo, y un posible resultado simplificado en cuanto al número de BBoxes generado.

Como semilla inicial, SS utiliza los *oversegments* del algoritmo de Felzenszwalb y Huttenlocher [77]. A partir de los *oversegments* se realiza una operación iterativa según los pasos descritos en el Algoritmo 1, hasta obtener un listado de BBoxes (L) con las posiciones hipotéticas de los objetos presentes en la imagen.

[4]Imagen de: https://www.koen.me/research/selectivesearch/

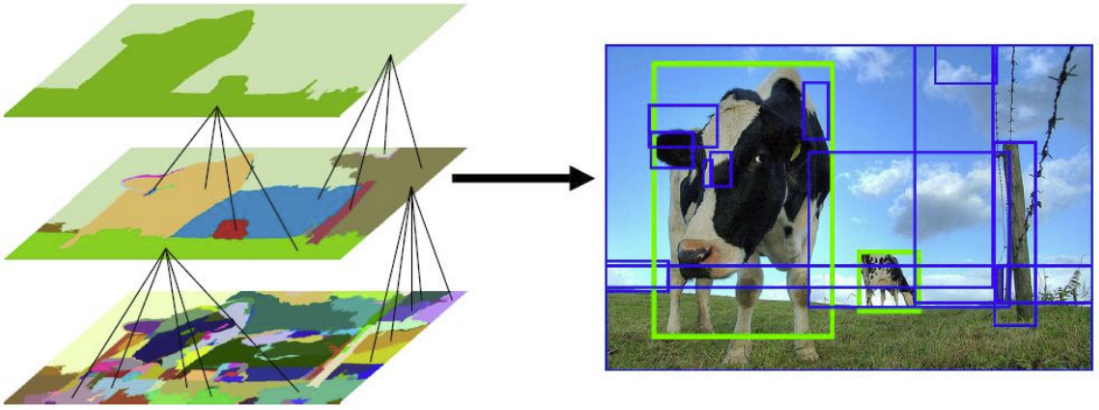


Figura 3.18: Ejemplo de propuesta de regiones mediante el algoritmo Selective Search.

Algoritmo 1: Algoritmo Selective Search

Data: Imagen en color
Result: Listado de BBoxes con posibles posiciones L
Obtiene regiones iniciales $R = [r_1, \ldots, r_n]$
Inicializa el conjunto de similitudes $S = \emptyset$
foreach *Par de regiones vecinas* **do**
    Calcula similitud $s(r_i, r_j)$
    $S = S \cup s(r_i, r_j)$
end
while $S \neq \emptyset$ **do**
    Obtén la mayor similitud $s(r_i, r_j) = max(S)$
    Fusiona las regiones i y j en $r_t = r_i \cup r_j$
    Elimina similitudes de la región i del conjunto S: $S = S \setminus s(r_i, r_\star)$
    Elimina similitudes de la región j del conjunto S: $S = S \setminus s(r_i, r_\star)$
    Calcula el conjunto de similitudes S_t de la región fusionada r_t y sus vecinas
    $S = S \cup S_t$
    $R = R \cup r_t$
end
Extrae BBoxes L para todas las regiones de R

En cada iteración, se generan segmentos de mayor tamaño y se añaden a la lista de propuesta de regiones. Por tanto, se generan nuevas propuestas de región a partir de regiones más pequeñas en un procedimiento *de abajo a arriba*.

Para el concepto de similitud entre dos regiones utiliza cuatro métricas basadas en color, textura, tamaño y compatibilidad de forma. Todas las métricas proporcionan valores en el rango [0,1] para facilitar su combinación. Además, todas deben poder propagarse en la jerarquía, para poder calcular las características de una región fusionada a partir de las regiones padre, sin acceder a los píxeles de la imagen.

Para obtener la similitud de color se calcula un histograma de color con 25 intervalos para cada canal de la imagen. A continuación, se concatenan los histogramas de todos

los canales para obtener un descriptor de color de 75 características. De esta forma, la similitud de color de dos regiones se calcula como la intersección de los dos histogramas como muestra la Ecuación 3.14, donde c_i^k es el valor del histograma para el intervalo k-ésimo del descriptor de color.

$$s_{color}(r_i, r_j) = \sum_{k=1}^{n} min(c_i^k, c_j^k) \tag{3.14}$$

Para la similitud de textura, las características se calculan a partir de las derivadas Gausianas en 8 orientaciones para cada canal de color. Para cada orientación y color, se calcula un histograma de 10 intervalos, lo que genera un vector de 240 características. De esta forma, la similitud de textura entre dos regiones se calcula como la intersección entre los dos histogramas como muestra la Ecuación 3.15, donde t_i^k es el valor del histograma para el intervalo k-ésimo del vector de características de textura.

$$s_{textura}(r_i, r_j) = \sum_{k=1}^{n} min(t_i^k, t_j^k) \tag{3.15}$$

La similitud de tamaño fomenta la fusión temprana de las regiones más pequeñas. Esto asegura que se formen propuestas de región en todas las escalas para todas las partes de la imagen. Si esta métrica no se usase, una región iría creciendo de escala a escala, y las propuestas en varias escalas sólo aparecerían en esa localización. La similitud de tamaño se define como indica la Ecuación 3.16.

$$s_{size}(r_i, r_j) = 1 - \frac{size(r_i) + size(r_j)}{size(imagen)} \tag{3.16}$$

Por último, la compatibilidad de forma mide como encajan entre si dos regiones. Si una región encaja en otra querremos fusionarlas para eliminar huecos, si ni siquiera se tocan no deberían fusionarse. La compatibilidad de forma se define como indica la Ecuación 3.17, donde $size(BB_{ij})$ es el BBox que engloba a r_i y r_j.

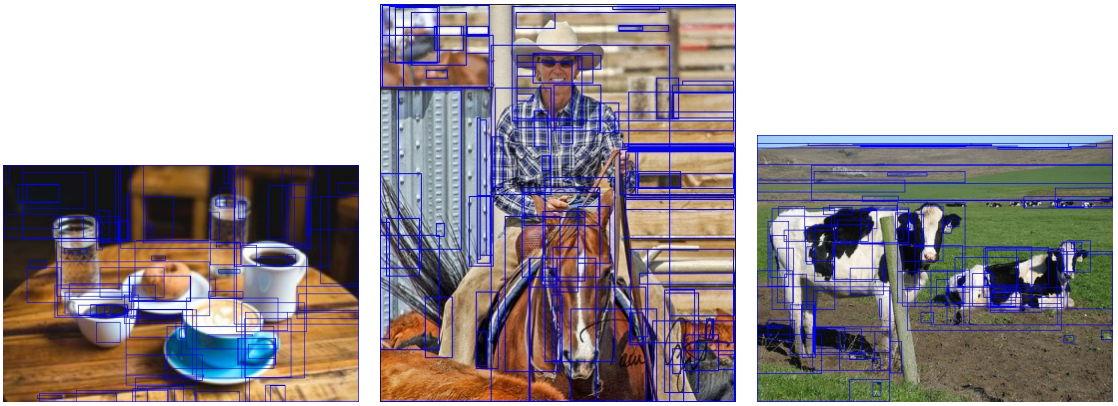
$$s_{forma}(r_i, r_j) = 1 - \frac{size(BB_{ij}) - size(r_i) - size(r_j)}{size(imagen)} \tag{3.17}$$

La similitud final entre dos regiones, se define como una combinación lineal de las cuatro métricas de similitud, como expresa la Ecuación 3.18. Donde r_i y r_j son dos regiones o segmentos de la imagen semilla, y $a_i \in [0, 1]$ indica si se usa o no cada métrica de similitud.

$$s(r_i, r_j) = a_1 s_{color}(r_i, r_j) + a_2 s_{textura}(r_i, r_j) + a_3 s_{size}(r_i, r_j) + a_4 s_{forma}(r_i, r_j) \quad (3.18)$$

Para obtener un listado ordenado de las hipótesis de objeto, se añaden en el orden en el que han sido generadas en cada estrategia de agrupación. Dado que se usan 80 estrategias de agrupamiento distintas, se sobrepuntuarían las regiones grandes. Para evitar esta sobrepuntuación se introduce una aleatorización asociada al nivel jerárquico en el que se genera cada región. Además, se realiza un filtrado de BBoxes duplicadas.

La Figura 3.19 muestra el resultado del algoritmo en su variante rápida sobre algunas imágenes limitando el número de propuestas a un máximo de 100.



(a) (b) (c)

Figura 3.19: Ejemplo de BBoxes de salida del algoritmo de propuesta de regiones *Selective Search* limitado a únicamente 100 propuestas para facilitar la visualización.

3.6.2. El modelo R-CNN

La detección de objetos con R-CNN [11] requiere varios módulos que pueden observarse en la Figura 3.20. El primero genera propuestas de regiones no ligadas a una categoría. Estas propuestas definen el conjunto de detecciones disponibles para el detector. El segundo módulo, es una red neuronal convolucional que extrae un vector de características de tamaño fijo de cada región. El tercer módulo es un conjunto de SVM lineales ajustados para cada clase que se utiliza para obtener la puntuación de cada propuesta. Y por último, cada propuesta de BBox se refina mediante regresión.

El modelo R-CNN no está limitado a un algoritmo fijo de propuesta de regiones, en la implementación original se utilizó *Selective Search* [76] para facilitar la comparación con otros trabajos previos. Respecto de la extracción de características, se utiliza un vector de 4096 características de cada propuesta de región usando una red tipo *AlexNet* [65]. Las

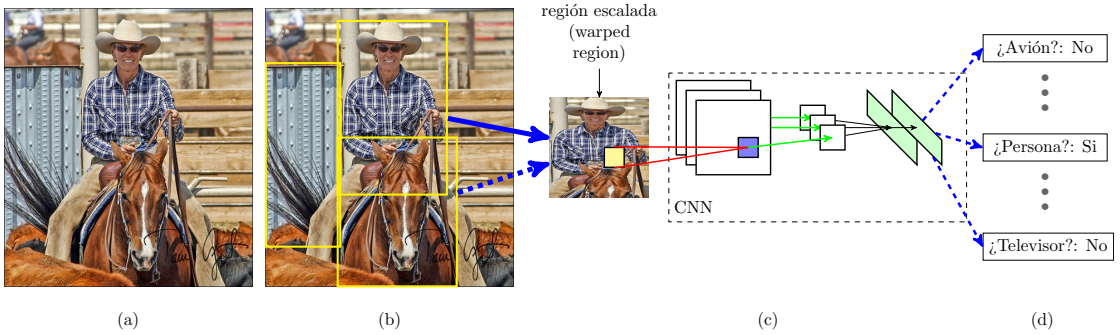


Figura 3.20: Esquema de funcionamiento del modelo R-CNN. (a) Imagen de entrada. (b) Extracción de propuestas de región con un algoritmo adecuado (~ 2000). (c) Cálculo de características profundas para todas las regiones escaladas a un mismo tamaño. (d) Clasificación de cada región.

características se obtienen propagando a través de la red de cinco capas convolucionales y dos capas completamente conectadas, una imagen RGB de 227x227 a la que se le ha restado la media.

Dado que la red requiere imágenes de tamaño fijo, se escogió escalar las regiones de entrada al tamaño requerido por la red añadiendo 16 píxeles de contexto respecto del BBox de certeza disponible. Obviamente, este escalado fijo e independiente del tamaño o relación de aspecto de la región original deformará la imagen a clasificar.

En modo detección, la red extrae 2000 regiones mediante el algoritmo Selective Search en modo *fast*. Se deforma cada propuesta y se propaga sobre la CNN para obtener el vector de características de cada propuesta. Después, se puntúa cada vector de características usando el conjunto de SVMs ajustado para cada clase. Con todas las las regiones puntuadas en una imagen se aplica un *algoritmo de supresión de no máximos* (NMS) básico y de forma separada para cada clase. Un algoritmo NMS descarta toda región que tenga un solapamiento superior a un umbral establecido con otra región de mayor puntuación.

Los aspectos que destacan en el modelo R-CNN respecto del estado del arte de la tecnología del año 2014, son la reutilización de la salida de la CNN para todas las clases y el reducido tamaño del vector de características que genera la CNN, significativamente menor que el requerido por otras propuestas de ese momento. De esta forma, los únicos cálculos que deben ser realizados para cada clase son el producto escalar entre el vector de características y los pesos SVM, y la supresión de no máximos.

3.6.2.1. Regresión de BBoxes

Por último, el modelo R-CNN usa una etapa de ajuste fino de la posición y tamaño de los BBoxes (regresión) para mejorar el rendimiento en localización. Por tanto para cada

propuesta se genera un nuevo BBox usando un *regresor* específico para la clase del BBox en cuestión. El ajuste se inspira en los modelos de piezas deformables utilizando en este caso las características proporcionadas por la CNN en vez de características geométricas.

La entrada de entrenamiento para el regresor se compone de un conjunto de parejas $(P^i, G^i)_{i=1,\ldots,N}$, donde $P^i = (P^i_x, P^i_y, P^i_w, P^i_h)$ especifica las coordenadas del píxel central de la propuesta i junto con el ancho y alto en píxeles. Se codifica de la misma forma las coordenadas y tamaño de los BBoxes de certeza G. El objetivo en este caso será entrenar una transformación que mapee una propuesta de *bounding box* de la red P a las coordenadas reales de un BBox de referencia G (*ground truth*).

La transformación se parametriza en base a cuatro funciones $d_x(P)$, $d_y(P)$, $d_w(P)$ y $d_h(P)$. Las dos primeras especifican una traslación invariante a la escala del centro del BBox, mientras que las restantes especifican traslaciones en espacio logarítmico del ancho y el alto del BBox. Tras entrenar estas funciones, podremos transformar una propuesta de entrada P en una predicción ajustada $\hat{G}$ aplicando la transformación de la Ecuación 3.19.

$$\begin{aligned} \hat{G_x} &= P_w d_x(P) + P_x \\ \hat{G_y} &= P_h d_y(P) + P_y \\ \hat{G_w} &= P_w exp(d_w(P)) \\ \hat{G_h} &= P_h exp(d_h(P)) \end{aligned} \tag{3.19}$$

Cada función $d_\star$ (donde $\star$ es x, y, h o w) se modela como una función lineal de las características de la capa $pool_5$ de la propuesta P en cuestión que se denota como $\phi_5(P)$. Por tanto tendremos que $d_\star(P) = w_\star^T \phi_5(P)$, donde $w_\star$ es un vector de parámetros ajustables. El vector $w_\star$ se ajusta por regresión de cresta (*ridge regression*) según indica la Ecuación 3.20.

$$w_\star = \underset{\hat{w}_\star}{argmin} \sum_i^N (t^i_\star - \hat{w}^T_\star \phi_5(P^i))^2 + \lambda ||\hat{w}_\star||^2 \tag{3.20}$$

Los objetivos de regresión $t_\star$ para la pareja de entrenamiento (P, G) se definen como indica la Ecuación 3.21.

$$
\begin{aligned}
t_x &= (G_x - P_x)/P_w \\
t_y &= (G_y - P_y)/P_h \\
t_w &= log(G_w/P_w) \\
t_h &= log(G_h/P_h)
\end{aligned}
\tag{3.21}
$$

Que puede resolverse de forma cerrada como un problema de mínimos cuadrados. El parámetro de regularización se ajustó a un valor de $\lambda = 1000$ en base a un conjunto de validación. No usaron propuestas P muy alejadas de un BBox de referencia G. Sólo se usaron propuestas con un IoU[5] mínimo de 0.6 con el BBox de certeza. El ajuste se realiza para cada una de las clases, con el objetivo de obtener un conjunto de regresores específico de cada clase de objeto.

En detección, tras puntuar cada propuesta se predice su nueva ventana únicamente una vez, ya que se ha encontrado empíricamente que un ajuste iterativo de los BBoxes generados no mejora los resultados.

3.6.3. Fast R-CNN

Fast R-CNN es un algoritmo que aprende de forma simultánea a clasificar propuestas de objetos y refinar sus ubicaciones. Por tanto, Fast R-CNN no autogenera propuestas de objeto, parte de un conjunto de Regiones de Interés (ROI, *Region of Interest*). El modelo surge como una mejora de R-CNN [11] y SPP-Net [78].

Uno de los inconvenientes del modelo R-CNN es su ajuste multietapa. En una R-CNN, en primer lugar se ajusta una CNN con propuestas de objetos usando una función de pérdida logarítmica. Después, se ajusta un SVM a las características CNN. Y por último, se entrenan los regresores de BBoxes. Por tanto, el entrenamiento de una red R-CNN es costoso en términos temporales y además, la detección es también muy lenta.

La lentitud del modelo R-CNN se debe a la aplicación de la CNN sobre cada una de las propuestas de objeto, se generan en torno a 2000 propuestas para cada imagen. El modelo SPP-Net [78], solventa este problema calculando el mapa de características de toda la imagen, para después incrustar las propuestas de ROI en los mapas de características mediante proyección. Esta estrategia permite mejorar el rendimiento de R-CNN entre diez y cien veces en modo test. El tiempo de entrenamiento también se reduce a un tercio, gracias a una extracción de características más rápida. Sin embargo, el modelo SPP-Net sigue necesitando ajuste multietapa de los distintos bloques que lo componen.

[5] Ver Sección 4.8.

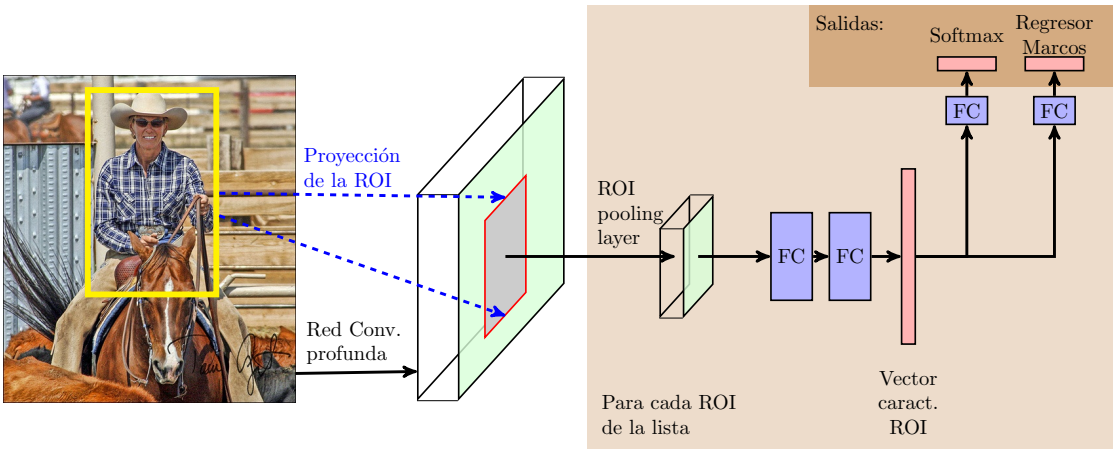


Figura 3.21: Diagrama de bloques del modelo Fast R-CNN.

Como podemos observar en la Figura 3.21, una Fast R-CNN usa como entrada un conjunto de propuestas de objeto y una imagen. La imagen atraviesa la red, compuesta de varias capas convolucionales y capas *max pooling* y genera un mapa de características. A continuación, mapea cada una de las propuestas de objeto sobre el mapa de características, proyectando la ROI según el procedimiento indicado en la Sección 3.6.3.1. La proyección de ROI permite localizar las coordenadas de las propuestas de región en el mapa de características asociado a la imagen original.

Para cada propuesta de objeto, una capa de agrupamiento de ROIs (ROI pooling layer) extraerá una un vector de características de tamaño fijo, que atravesará varias capas totalmente conectadas. Las capas FC se bifurcan en dos capas de salida, una que proporciona la probabilidad de pertenencia a $K+1$ clases y otra que se usa para proporcionar las coordenadas de los BBoxes para las K clases.

3.6.3.1. Proyección de ROIs

Como hemos mencionado previamente, en Fast R-CNN, las propuestas de región en la imagen original se proyectan en la salida de la última convolución del mapa de características. Obviamente, las dimensiones del mapa de características difieren de las de la imagen original, por lo que la traslación de coordenadas de cada ROI se realiza en función de la *tasa de submuestreo* del mapa de características.

De esta forma podemos ver, como muestra la Figura 3.22, que para una tasa de submuestreo de 1/16. Una imagen de dimensiones 688x920 dará lugar a un mapa de características de 43x58. Y una propuesta de región con centro en (340,450) y dimensiones 320x128 se proyectará sobre el mapa de características en el BBox con centro en (21,28) y dimensiones 20x8.

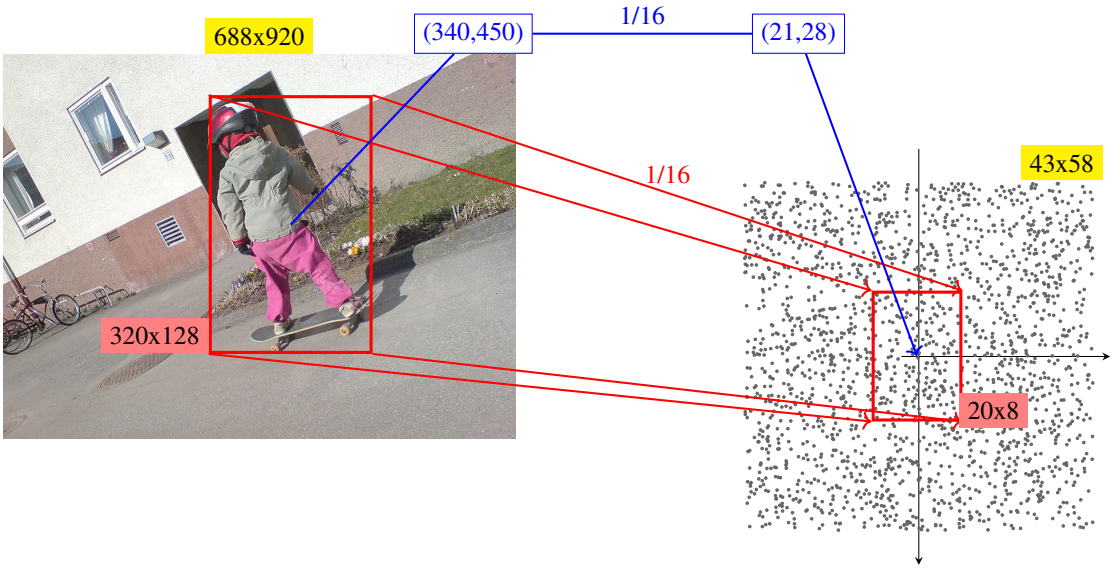


Figura 3.22: Proyección de ROIs sobre el mapa de características.

3.6.3.2. Agrupamiento de ROIs

Durante la fase de propuestas se generan muchas regiones para evitar la pérdida de objetos. La utilización de muchas regiones puede reducir la velocidad del sistema. El agrupamiento de ROIs (*ROI pooling*) surge como una solución a este problema representando cada ROI únicamente con una pequeña matriz.

La capa *ROI pooling* en Fast R-CNN es una variante de la capa *Spatial Pyramid Pooling* (SPP) [78] con un único nivel de pirámide. Tiene dos entradas, un mapa de características de tamaño fijo generado por la CNN y una matriz N x 5, que contiene la lista de regiones de interés. En la matriz anterior, N es el número de ROIs a evaluar y las cinco coordenadas especifican un índice, y las esquinas superior izquierda e inferior derecha la ROI.

La Figura 3.23 muestra gráficamente el procedimiento de agrupación para una ROI de tamaño 7x5, sobre un mapa de características 8x8. Cada ROI proyectada se escala a un tamaño fijo (2x2 en el ejemplo). El agrupamiento se realiza dividiendo la región proyectada en tantas secciones como deba tener la matriz de salida. Cada sección se representa a partir del valor máximo encerrado, como muestra la Figura 3.23. El tamaño de las regiones de interés (en este caso 7x5), no tiene porqué ser divisible de forma exacta por el número de regiones de agrupamiento (en este caso 2x2).

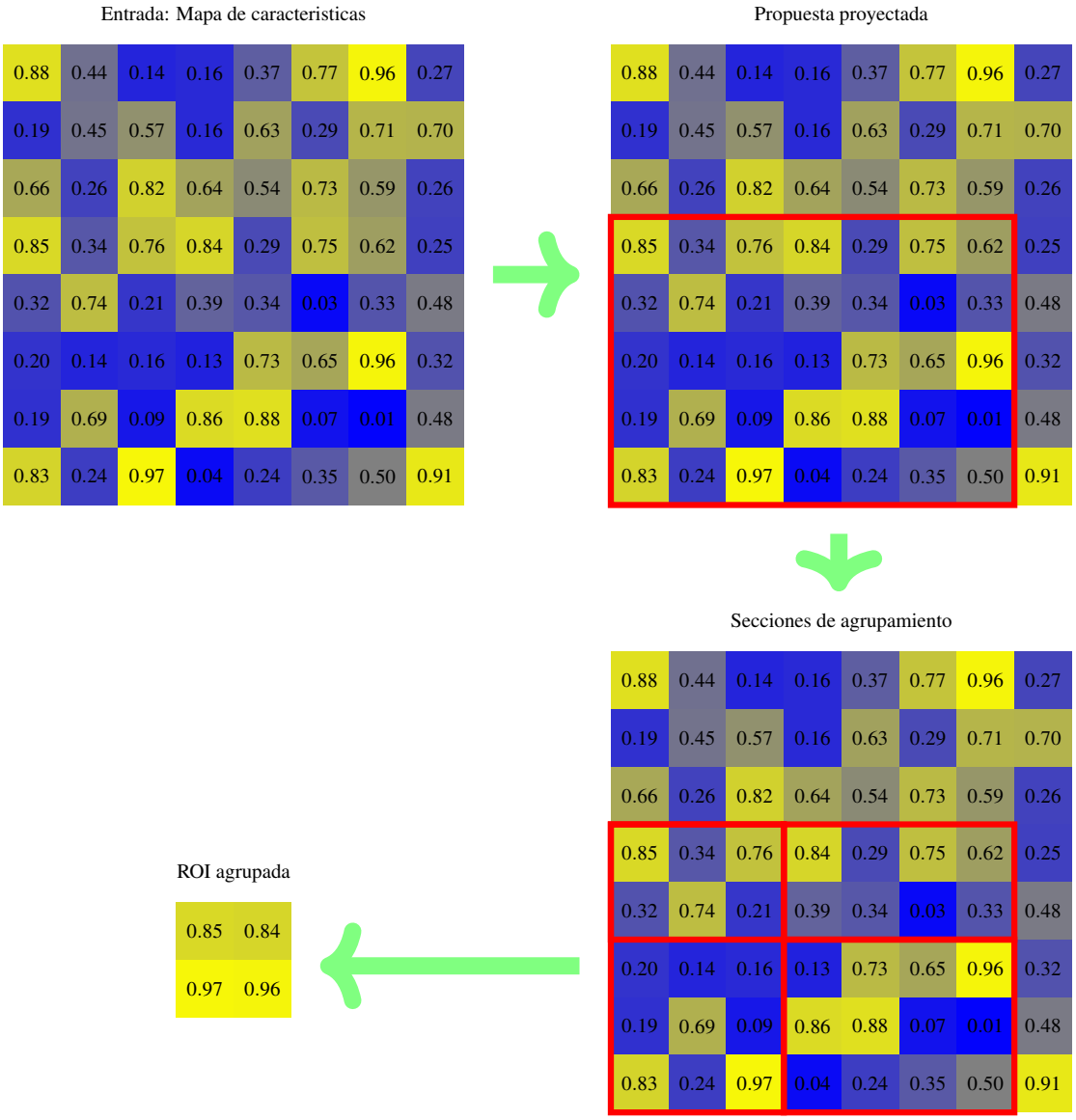


Figura 3.23: Proyección de una ROI sobre el mapa de características y agrupamiento de regiones (ROI pooling). Las regiones de agrupamiento pueden tener tamaño distinto.

3.6.3.3. Función de pérdida multitarea

Una arquitectura Fast R-CNN tiene dos capas de salida. Una capa predice la probabilidad discreta de pertenencia a $K+1$ clases $p = (p_0, ..., p_K)$ para cada ROI. Y la otra predice los desplazamientos de regresión de los BBoxes de cada objeto $t^k = (t_x^k, t_y^k, t_w^k, t_h^k)$, para cada uno de los K objetos, indexados en k. Donde t^k especifica una traslación invariante a la escala y desplazamiento logarítmico del ancho/alto respecto de una propuesta de objeto.

Cada ROI de entrenamiento se etiqueta con una clase u y un BBox de referencia v. Así, se define una función de pérdida multitarea (*multi-task loss function*) L, sobre cada ROI

etiquetada, que permite entrenar la red de forma simultánea en clasificación y regresión de BBoxes como especifica la Ecuación 3.22.

$$L(p, u, t^u, v) = L_{cls}(p, u) + \lambda[u \geq 1]L_{loc}(t^u, v) \tag{3.22}$$

Donde $L_{cls}(p, u) = -log(p_u)$ es la función de pérdida logarítmica para la clase correcta u. Y la segunda función de pérdida L_{loc}, se define sobre una tupla[6] de objetivos correctos de regresión de BBox para la clase u. El corchete de Iverson vale 1 cuando $u \geq 1$ luego se ignora la contribución de la clase fondo en localización.

La función de regresión de BBoxes utilizada en la función de pérdida tiene la expresión dada en la Ecuación 3.23, donde la función $smooth_{L_1}$ tiene la forma indicada en la Ecuación 3.24. Esta es otra diferencia con respecto a los modelos R-CNN y SPP-Net, en las que se usa L2 como función de coste (ver Ecuación 3.6).

$$L_{loc}(t^u, v) = \sum_{i \in \{x,y,w,h\}} smooth_{L_1}(t_i^u - v_i) \tag{3.23}$$

$$smooth\, L_1(t_i^u - v_i) = \begin{cases} 0.5x^2 & si \quad |x| < 1 \\ |x| - 0.5 & en\ el\ resto \end{cases} \tag{3.24}$$

Por último, el hiperparámetro λ controla el balance entre ambas contribuciones a la función de pérdida que regula el *backpropagation*. En este estudio, aplicaron la misma ponderación a las contribuciones de clase y localización ($\lambda = 1$).

3.6.3.4. El entrenamiento de una Fast R-CNN

En una Fast R-CNN todos los pesos de la red se ajustan mediante el algoritmo *backpropagation*, a diferencia de lo que ocurre en SPP-Net y R-CNN.

Para el entrenamiento de una Fast R-CNN, se muestrean jerárquicamente minilotes de gradiente descendente estocástico (SGD). Primero se muestrean N imágenes y luego R/N ROIs de cada imagen. Usando dos imágenes y tomando 128 ROIs por imagen, se obtiene un rendimiento mayor que muestreando una ROI de 128 imágenes diferentes, como implementan R-CNN y SPP-Net.

[6]Una tupla es una secuencia o lista ordenada finita de n objetos

3.6.3.5. Detección

Una vez que la red se ha entrenado, la detección consiste únicamente en un ciclo de propagación directa de la red. Obviamente, las propuestas de objeto están precalculadas, por ejemplo mediante Selective Search. La red utiliza una imagen como entrada y una lista de R propuestas de objeto a procesar. El número de propuestas suele estar entorno a 2000.

A continuación, se realiza el ciclo de propagación directa de la imagen y se obtienen las probabilidades de clase y predicciones de cada BBox. Por último, se realiza una supresión de no-máximos independiente para cada clase y se obtienen los mejores BBoxes no solapados para la imagen.

3.6.4. Faster R-CNN, el modelo de referencia

En el modelo Fast R-CNN el cuello de botella a nivel de cómputo se encuentra en la fase de generación de propuestas de región. Otros modelos de esta época (2016) ya reducían el tiempo de generación de propuestas respecto del esquema *Selective Search*. Así, por ejemplo en *EdgeBoxes* [79], la fase de propuesta de regiones supone casi el mismo tiempo de cómputo que la red de detección.

El modelo Faster R-CNN [13] introduce como gran novedad las Redes de Propuesta de Regiones o *Region Proposal Networks* (RPN). Una RPN utiliza las mismas capas convolucionales que las redes de clasificación. De esta forma, al compartir las convoluciones en tiempo de reconocimiento, el coste computacional de la fase de generación de propuestas se reduce.

A nivel de concepto, Faster R-CNN es un modelo de localización y clasificación compuesto por tres redes neuronales: una red profunda de características, una red de propuesta de regiones, y una red de clasificación. La red de características suele ser una red preentrenada, robusta y bien testada, como por ejemplo una ResNet50 [72] o VGG-16 [69], a la que nos referimos como *backbone*. La función de esta red es proporcionar un buen conjunto de características de la imagen de entrada. La red RPN es una red pequeña que genera regiones de interés (ROI) con alta probabilidad de contener un objeto. Finalmente, la red de clasificación usa las propuestas de la RPN y el mapa de características generado por el backbone, para afinar las BBoxes y asignarles una clase de entre las contempladas. El diagrama de bloques simplificado de esta arquitectura puede observarse en la Figura 3.24.

Ren et al. observaron que los mapas de características convolucionales usados por los detectores basados en regiones, como Fast R-CNN, también se podían usar para generar

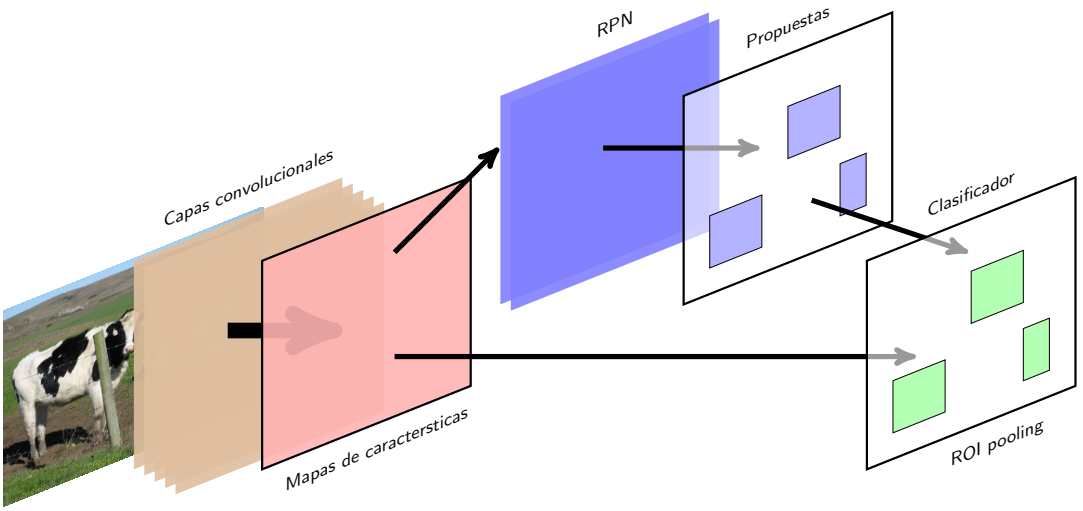


Figura 3.24: Diagrama de bloques del modelo Faster R-CNN en modo reconocimiento.

propuestas de región. De esta forma, sobre los mapas de características convolucionales construyeron una RPN, añadiendo algunas capas convolucionales adicionales. Como la RPN definida es un tipo de red totalmente conectada, puede entrenarse de extremo a extremo para la tarea de generar propuestas de localización.

Para generar las propuestas de región, se desliza una pequeña red sobre el mapa de características convolucionales de la última capa del *backbone*, como muestra la Figura 3.25a. La red usa una porción de tamaño $n \times n$ del mapa de características. Cada ventana se mapea a un vector de características de menor dimensión (256-d ó 512-d), que se usa como entrada a dos capas completamente conectadas: una capa de regresión de marcos (reg) y otra para clasificación (cls).

Las RPN introducen el concepto de *marcos de anclaje* (*anchor boxes*) que se utilizan como referencias para varias escalas y relaciones de aspecto. La Figura 3.25b muestra un posible conjunto de marcos de anclaje para una posición dada con dos escalas y tres relaciones de aspecto (1:1, 2:1 y 1:2), se representan con el mismo color los marcos asociados a una misma relación de aspecto.

En cada posición de la ventana, se predicen simultáneamente varias propuestas de región en función de los marcos configurados. El número máximo de propuestas en cada localización se denota como k. De esta forma la salida de la capa *reg* presenta $4k$ salidas que codifican las coordenadas de k marcos, y la capa *cls* proporciona $2k$ puntuaciones que estiman la probabilidad de objeto/no objeto para cada propuesta.

Los marcos de anclaje se referencian respecto del centro de la ventana deslizante, y se asocian con una escala y relación de aspecto. Si se usan como en la figura, dos escalas y tres relaciones de aspecto, se estudiarán $k = 6$ marcos de anclaje en cada posición de la

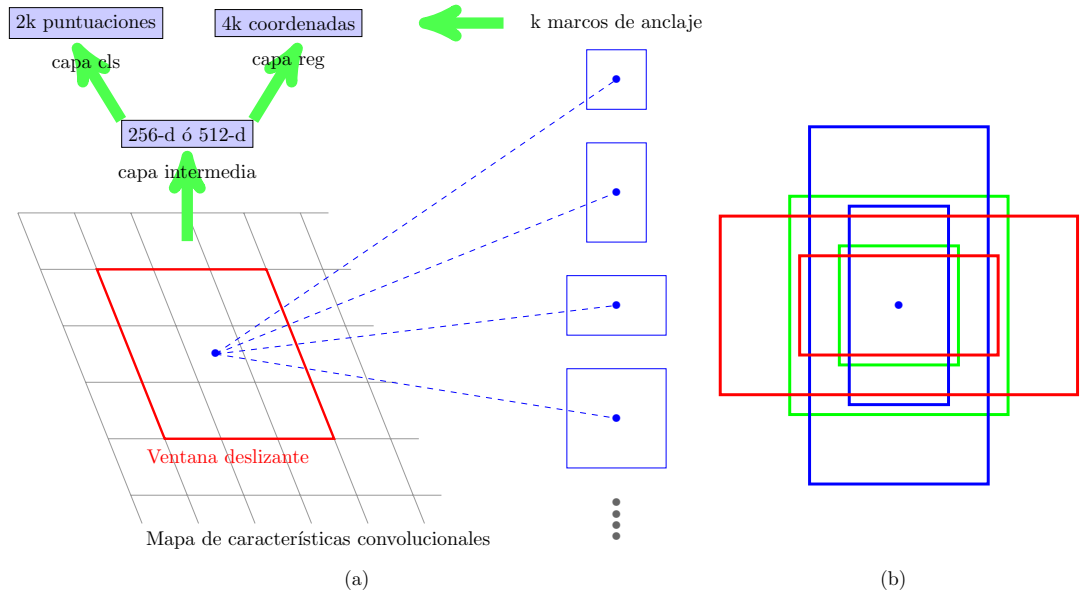


Figura 3.25: Funcionamiento de una RPN: (a) Uso de una ventana deslizante con $n = 3$ sobre el mapa de características para generar para cada marco de anclaje puntuaciones y coordenadas. (b) Posible superposición de marcos de anclaje para dos escalas y tres relaciones de aspecto.

ventana deslizante. Por tanto, para un mapa de características de tamaño $W \times H$ una RPN generará un máximo de $W \times H \times k$ marcos de anclaje.

3.6.4.1. Entrenamiento de la RPN y función de pérdida

En el entrenamiento de la RPN se asigna una etiqueta binaria a cada marco de anclaje (objeto vs. no objeto). Se asigna una etiqueta positiva a las anclas que tienen un solapamiento, en términos de IoU[7], mayor que 0.7 con cualquier BBox de referencia. Por tanto, un BBox de referencia puede etiquetar positivamente varios marcos de anclaje. Se asigna una etiqueta negativa a un ancla no positiva, si su solapamiento es inferior a 0.3 con todos los BBoxes de referencia del conjunto de entrenamiento. Aquellos BBoxes de referencia que no son positivos ni negativos no contribuyen al entrenamiento.

Con estas condiciones se minimiza una función objetivo multitarea cuya definición aparece en la Ecuación 3.25. Donde, i es el índice del ancla en un minilote de entrenamiento, y p_i es la probabilidad estimada de que el ancla i sea un objeto.

La probabilidad $p_i^\star$ vale 1 si el ancla es positiva y 0 si es negativa. El vector t_i contiene la predicción de las 4 coordenadas parametrizadas del BBox, y $t_i^\star$ es el BBox de referencia asociado con un ancla positiva.

[7]Ver Sección 4.8

$$L(p_i, t_i) = \frac{1}{N_{cls}} \sum_i L_{cls}(p_i, p_i^\star) + \lambda \frac{1}{N_{reg}} \sum_i p_i^\star L_{reg}(t_i, t_i^\star) \tag{3.25}$$

Para la función de pérdida de clasificación L_{cls}, se usa la función de pérdida logarítmica (objeto/no objeto). Y para la función de pérdida de regresión de BBoxes se usa la expresión dada por la Ecuación 3.26, donde R es la función *smooth* L_1. La función de pérdida sólo se activa para anclas positivas ($p_i^\star = 1$).

$$L_{reg}(t_i, t_i^\star) = R(t_i - t_i^\star) \tag{3.26}$$

Los dos términos de la Ecuación 3.25 se normalizan respecto del tamaño del minilote ($N_{cls} = 256$), y el número de posiciones de anclas ($N_{reg} \sim 2400$). Ambas contribuciones se balancean respecto de λ. El valor de λ se establece por defecto a 10 para que ambos términos presenten pesos semejantes. Según los autores, son valores seguros los comprendidos en el intervalo $[1, 100]$.

Para la regresión de BBoxes, se parametrizan las coordenadas siguiendo [11] como indica la Ecuación 3.27. Donde x,y,w y h denotan las coordenadas del centro del BBox, el ancho y el alto. Las variables sin subíndice están asociadas al BBox propuesto, con subíndice están referidas al ancla y el superíndice $\star$ denota el BBox de referencia.

$$\begin{aligned} t_x &= (x - x_a)/w_a \quad & t_x^\star &= (x^\star - x_a)/w_a \\ t_y &= (y - y_a)/h_a \quad & t_y^\star &= (y^\star - y_a)/h_a \\ t_w &= log(w/w_a) \quad & t_w^\star &= log(w^\star/w_a) \\ t_h &= log(h/h_a) \quad & t_h^\star &= log(h^\star/h_a) \end{aligned} \tag{3.27}$$

De esta forma, la operación de regresión ajusta una traslación de un marco de anclaje a un BBox de referencia cercano. En este modelo las características usadas para la regresión tienen siempre el mismo tamaño sobre el mapa de características (3×3), independientemente del tamaño del BBox de referencia. Para lograr el ajuste a tamaños variables, se ajusta un conjunto de k regresores de BBox. Cada regresor gestiona una escala y relación de aspecto, y no comparten pesos entre si. De esta forma se pueden predecir BBoxes de varios tamaños a partir de características de tamaño y escala fijas gracias al uso de los marcos de anclaje.

La RPN se entrena de extremo a extremo usando *backpropagation* y SGD. Se usa una estrategia semejante a la realizada en el modelo Fast R-CNN, donde cada minilote proviene de una misma imagen y contiene múltiples BBoxes de entrenamiento positivas

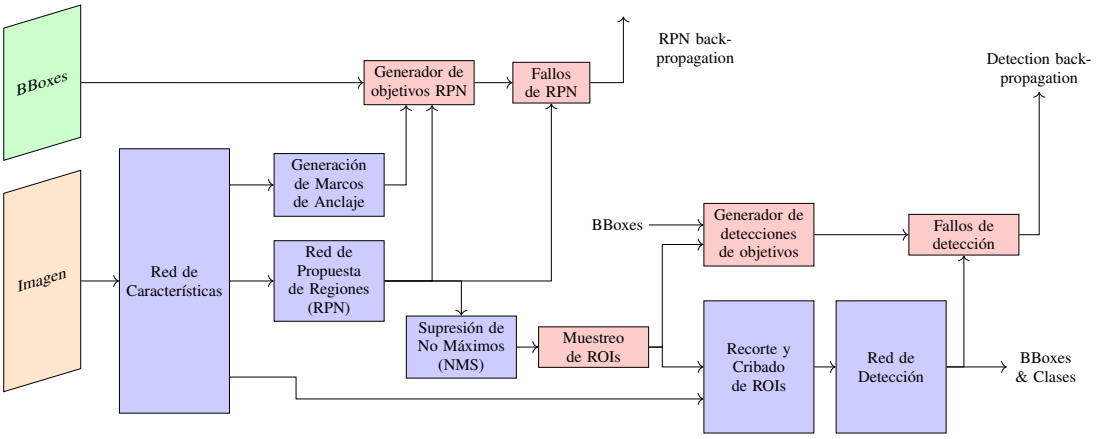


Figura 3.26: Diagrama de bloques del modelo Faster R-CNN. Los bloques coloreados en rojo o verde sólo están activos durante la fase de entrenamiento.

y negativas. Se muestrean 256 anclas en la imagen para calcular la función de coste del minilote. El algoritmo trata de obtener una relación de BBoxes positivas vs. negativas de 1 a 1. Si hay menos de 128 anclas positivas en una imagen, se completa el lote con BBoxes negativas. La Figura 3.26 muestra un esquema detallado de los distintos elementos que componen el modelo. Se indican en rojo los elementos que únicamente están activos durante la fase de entrenamiento de la red.

La Tabla 3.5 recoge un resumen de las principales características de los algoritmos analizados hasta este punto, comparando sus tiempos de reconocimiento y resaltando las principales limitaciones que presenta cada uno de ellos.

3.6.5. Redes piramidales de características

En una Red Piramidal de Características o *Feature Pyramid Network* (FPN), se usa el carácter multiescala de la forma piramidal que presentan las CNN para generar una pirámide de características. La estructura general de esta red puede observarse en la Figura 3.27.

En este tipo de red, el camino ascendente está constituido por una CNN robusta y testada. La CNN calcula una jerarquía de mapas de características para cada imagen de entrada. En el camino descendente, capas a la derecha en la Figura 3.27, la red piramidal combina en cada nivel características de baja resolución y fuerte significado, con características de mayor localización espacial pero menor significado que proceden de las conexiones laterales desde la CNN. De esta forma, se obtiene una pirámide de características (capas verdes de la Figura 3.27), en la que cada nivel de la pirámide puede usarse para detectar objetos a diferente escala.

Tabla 3.5: Resumen de las características de los algoritmos analizados para la detección de objetos en imágenes.

Algoritmo	Características	t_{pred}/ imagen	Limitaciones
CNN	Divide la imagen en múltiples regiones y después clasifica cada región en distintas clases	N/A	Necesita muchas regiones para predecir de forma eficiente y por tanto mucho tiempo de cómputo.
R-CNN	Usa *Selective Search* para generar regiones. Extrae entorno a 2000 regiones de cada imagen	40 – 50 s.	Requiere mucho tiempo de cómputo ya que cada región se inyecta en la CNN por separado, también usa tres modelos distintos para hacer predicciones.
Fast R-CNN	Cada imagen se inyecta en la CNN una sola vez y se obtienen mapas de características. Se usa Selective Search sobre los mapas para generar predicciones. Combina los tres modelos usados en R-CNN en uno solo.	2 s.	El uso de *Selective Search* limita la velocidad.
Faster R-CNN	Reemplaza *Selective Search* con una red de propuesta de regiones lo que acelera el algoritmo.	0.2 s.	Usa una arquitectura multietapa. El rendimiento total depende del funcionamiento de las etapas previas. La propuesta de regiones requiere un tiempo importante.

Una red Faster R-CNN estándar puede modificarse para usar una FPN como entrada a la RPN para mejorar su rendimiento [80]. En este caso se reemplaza el mapa de características monoescala con una red piramidal de características. La salida de las distintas capas de la FPN se usa ahora como entrada de la RPN para generar las propuestas de objeto, seleccionando la escala más adecuada de la FPN basándose en el tamaño de la ROI. Tras el ROI pooling, la salida se inyecta en el bloque final heredado del modelo Fast R-CNN para finalizar la predicción.

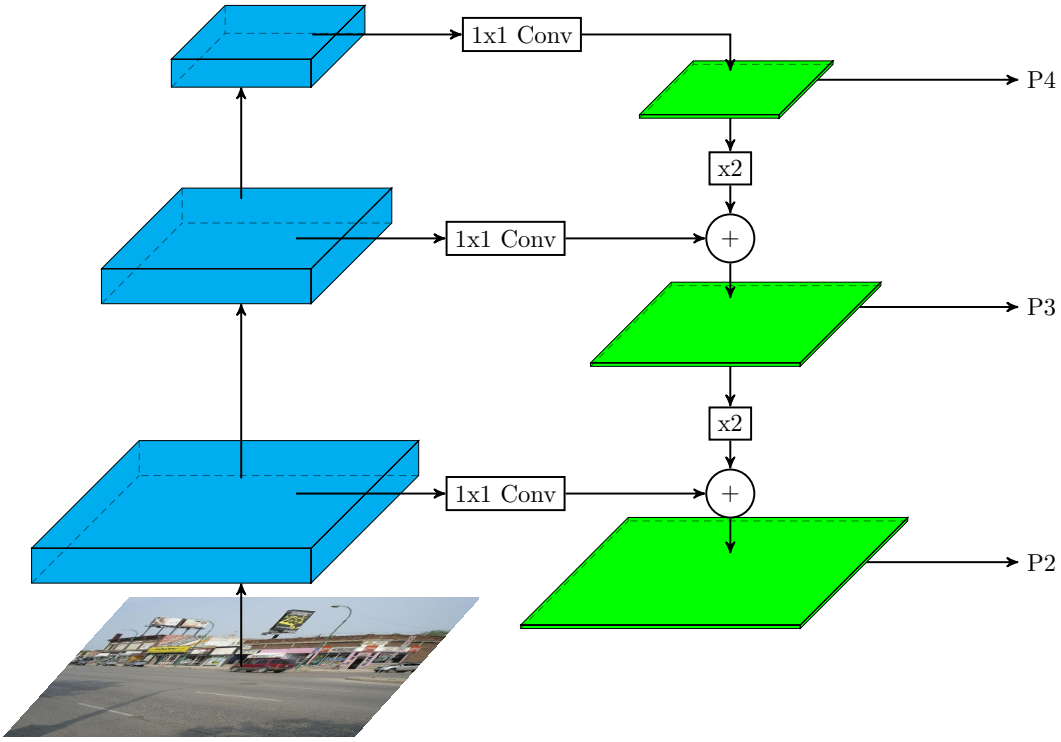


Figura 3.27: Estructura de una red piramidal de características. El camino ascendente es una red convolucional profunda. El camino descendente combina características de fuerte significado con características con mayor nivel de localización por medio de conexiones laterales.

3.6.6. RetinaNet

A diferencia de las redes de dos etapas, una red monoetapa opera sobre un conjunto uniforme y densamente muestreado de posibles localizaciones de objeto. Este tipo de estructura tiene el potencial de operar a mayor velocidad que una red de dos etapas gracias a su simplicidad. RetinaNet [16] es una red monoetapa que trata de superar la precisión de los detectores de dos etapas manteniendo una estructura simple. Está compuesta por un *backbone* FPN y dos subredes. Como elemento diferencial, que de hecho da título al artículo en el que se propone la arquitectura, usa una función de pérdida denominada *Focal Loss* (FL) para entrenar el clasificador de objetos.

La función de pérdida FL trata de concentrar el entrenamiento en un pequeño conjunto de ejemplos complejos. No realiza un muestreo aleatorio de los posibles prototipos en la imagen. De esta forma, trata de evitar que un alto número de ejemplos de fondo frente a un escaso número de objetos abrume al clasificador durante el entrenamiento.

La arquitectura de referencia RetinaNet se muestra en la Figura 3.28 de forma simplificada. Este modelo, siguiendo la idea desarrollada en [80], construye una FPN sobre una

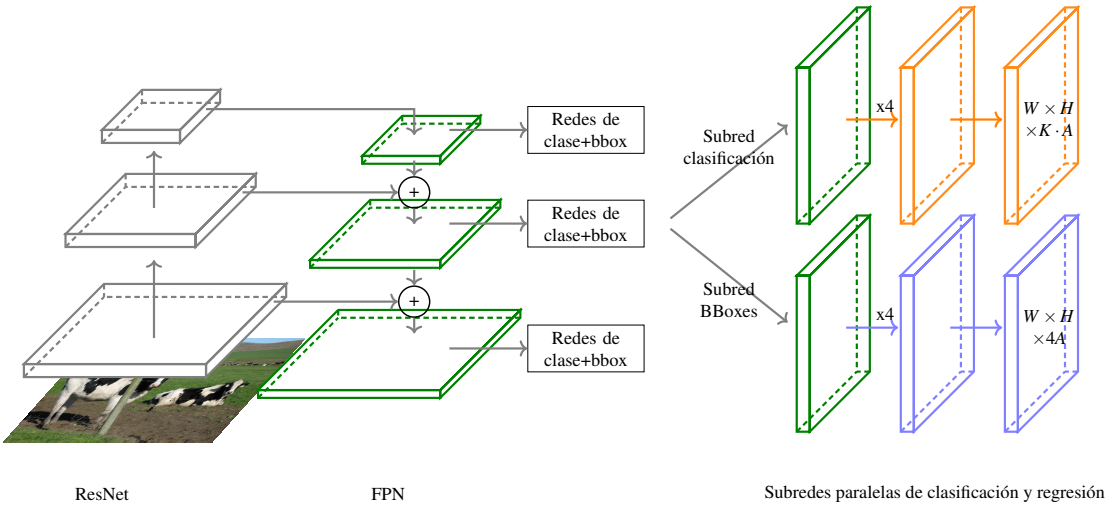


Figura 3.28: Estructura de un modelo RetinaNet. La generación de BBoxes y la clasificación se realiza a partir de la información de la FPN. El bloque de subredes paralelas para cada nivel de la pirámide realiza la clasificación y regresión de propuestas a distintas escalas de forma natural.

arquitectura ResNet, estudiando las variantes ResNet-50 y ResNet-101. La FPN proporciona una pirámide de características de la imagen de entrada. Cada nivel de la pirámide puede utilizarse para detectar objetos en una escala diferente. La pirámide del modelo de referencia usa los niveles de P_3 a P_7. Cada nivel l tiene una resolución 2^l menor que la imagen de entrada. Sólo los niveles de pirámide de P_3 a P_5 se calculan a partir de la correspondiente etapa residual de la ResNet como en una FPN estándar. La generación de P6 y P7 no sigue la estrategia de generación del modelo FPN. La capa P2 de ResNet no se usa para clasificar y o regresionar BBoxes por su alto coste computacional. Todos los niveles de la pirámide utilizan un mismo número de canales (256).

El modelo usa marcos de anclaje invariantes a la traslación como los usados en la RPN de [80]. Los marcos equivalen a un área equivalente de 32×32 en el nivel P_3 y de 512×512 en el nivel P_7. Al configurar la red se puede prescindir del nivel P_7 en implementaciones que únicamente vayan a detectar objetos pequeños.

En cada nivel de la pirámide se usan anclas con tres relaciones de aspecto a tres tamaños distintos, con lo que para cada posición muestreada se estudian 9 marcos de anclaje (A). A cada marco se le asigna un vector de K elementos para codificar la clase y otro para las coordenadas de la BBox objetivo. En entrenamiento, un ancla se asigna a un objeto si tiene un IoU > 0.5, y al fondo si el IoU ≤ 0.4[8]. Las anclas no asignadas no se usan para entrenar. Los objetivos de regresión se calculan como el desplazamiento entre el ancla y el BBox del objeto asignado.

[8]Ver definición de IoU en Sección 4.8

Cada subred de clasificación usa la salida de un nivel de la FPN para predecir la probabilidad de presencia de un objeto en las posiciones de las anclas muestreadas. Se usa FL como función de pérdida para el entrenamiento del clasificador. Cada una de estas subredes es una pequeña FCN conectada a un nivel de la FPN. Por diseño, los parámetros de las redes son idénticos en todos los niveles. El primer bloque de la red de clasificación está formado por 4 capas convolucionales 3×3 (en verde en la Figura 3.28). El segundo bloque es una capa convolucional 3×3 con $K \cdot A$ filtros. Y la final, contiene unidades con activación sigmoidea para generar las predicciones de cada posición espacial.

En paralelo con la red de clasificación corre una segunda subred. Esta subred realiza la regresión de cada marco de anclaje a un posible objeto, si es que finalmente es clasificado como tal. El diseño de esta subred es idéntico al de la red de clasificación, excepto en la capa de salida que proporciona $4 \times A$ salidas lineales por localización. Para su entrenamiento, usa como función de pérdida *Smooth L1*, ya definida en la Ecuación 3.24. Como novedad no realiza una regresión adaptada para cada clase, lo que le permite ahorrar parámetros. Según sus autores logra un rendimiento en regresión semejante a los modelos que entrenan regresores especializados.

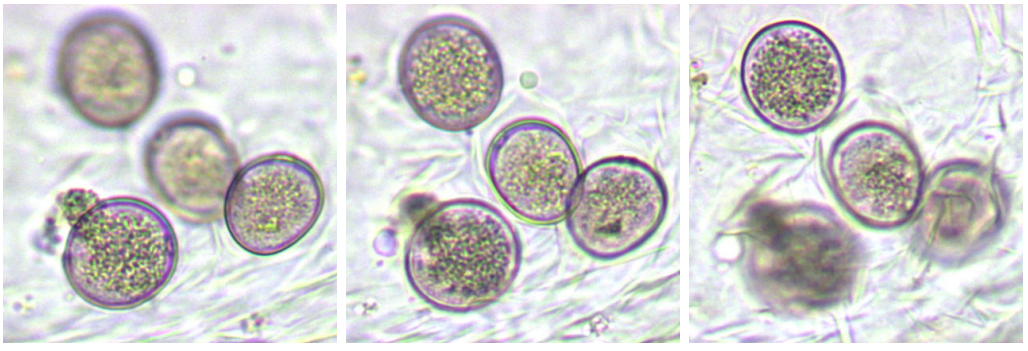
En este capítulo hemos analizado los distintos modelos de red que utilizamos en este trabajo. El número de modelos de red para detección de objetos a día de hoy es muy elevado. Cada nuevo modelo trata de mejorar la velocidad de inferencia con mejores índices de calidad de las propuestas. Los modelos utilizados en este trabajo han mostrado un excelente comportamiento en detección de objetos macroscópicos, en el capítulo siguiente evaluaremos su capacidad de detectar objetos microscópicos.

Capítulo 4

Descripción del sistema

4.1. Introducción

Existen dos problemas fundamentales al utilizar microscopía óptica para identificar automáticamente granos de polen [81]. En primer lugar, la existencia de imágenes con granos enfocados y otros parcialmente enfocados. Y en segundo lugar, las múltiples orientaciones que puede presentar un grano de polen en la imagen. Ambos problemas están asociados a que las imágenes son capturas bidimensionales de objetos volumétricos. De esta forma, una única vista de una muestra de polen puede ocultar parte de la ornamentación superficial de los granos, o en el peor caso, podemos ser incapaces de identificar el grano en un plano focal dado, como muestra la Figura 4.1. Por tanto, un análisis multifocal puede incrementar las posibilidades de localización e identificación correctas.



(a) (b) (c)

Figura 4.1: Distintos planos focales de un mismo área de una muestra. Podemos observar la alta influencia en la elección del plano focal para identificar los distinto objetos tipo grano presentes: los planos (a) y (c) muestran granos muy desenfocados, mientras que el plano (b) permite identificar los cuatro granos con más facilidad.

El tamaño de un grano de polen varía en el rango de los 8 a los 100 μm y visualmente se muestra como un objeto esferoidal. Una muestra de polen obtenida en un captador estándar además de granos de polen contiene polvo, detritos y esporas arrastrados por el aire. Estos elementos también pueden presentar aspecto esferoidal en la imagen, lo que complica la tarea de realizar una localización e identificación eficiente de los granos de polen presentes en una muestra natural. Además, la complejidad de la muestra puede variar en función de las condiciones atmosféricas de cada hora del día.

4.2. Esquema general

Dado que la visibilidad de la ornamentación de un grano de polen depende del enfoque de éste, como muestra la Figura 4.1. Un sistema que pretenda localizar y clasificar de forma eficiente todos los granos presentes en una muestra palinológica adquirida con un microscopio óptico, debe utilizar múltiples vistas de la muestra tomadas al recorrer el eje z (*z-stack*).

La Figura 4.2 ejemplifica de forma simplificada los efectos sobre la identificabilidad del grano en función del plano de enfoque analizado. Así, podemos observar como un grano que aparece nítido en uno de los planos (en verde en el plano F2), se vuelve borroso y difícil de identificar al desplazarnos en el eje Z. Como extensión de este mismo efecto, podemos encontrar granos únicamente identificables en planos muy alejados del plano central, como el mostrado en el plano F4 de la figura.

Además, desde el punto de vista de la clasificación, hay que tener en cuenta que la ornamentación puede ser observada únicamente en algunos planos de enfoque. Debemos tener en cuenta este hecho al procesar el *z-stack* y analizar el comportamiento de los modelos a la hora de determinar la clase más probable del objeto localizado.

Por tanto, en este trabajo proponemos un modelo de trabajo basado en el siguiente esquema de procesado:

1. Entrenamiento de varios modelos neuronales con muestras multienfoque etiquetadas a nivel de plano de enfoque.
    - Inclusión de varios planos por muestra con marcas de referencia corregidas.
2. Reconocimiento de *z-stacks* con los distintos modelos.
    - Supresión de no-máximos al fusionar los objetos identificados en el *z-stack*.
3. Evaluación de rendimiento con las marcas de referencia etiquetadas a nivel de muestra.

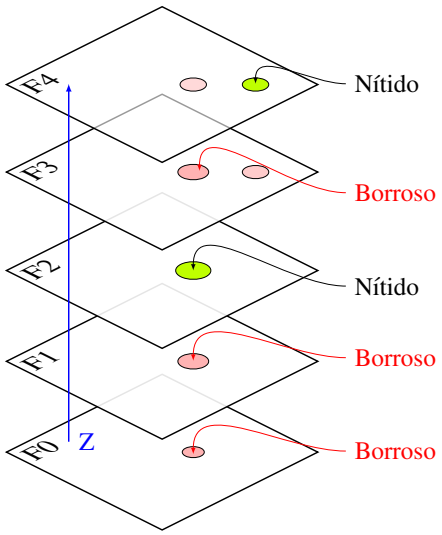


Figura 4.2: Variación de la identificabilidad de los granos de una muestra en función del plano de enfoque procesado. Un grano nítido (verde) puede aparecer borroso y con distinto tamaño (rojo) al variar el plano de enfoque, o puede ser únicamente visible en determinados planos de enfoque.

- Rendimiento en localización de granos de polen.
- Rendimiento en detección de granos de polen.

Adicionalmente, aplicaremos un algoritmo basado en la transformación de Hough sobre nuestro conjunto de datos test, que nos sirva como referencia a la hora de comparar el rendimiento de los distintos modelos neuronales en el apartado de localización de granos.

4.3. Construcción de la base de datos

Como hemos mencionado previamente, la utilización de varios planos focales parece ser imprescindible para llevar a cabo una localización y clasificación correcta de los distintos tipos de polen presentes en una muestra estándar. Por tanto, para poder utilizar la información tridimensional de los granos de polen a partir de distintos planos focales, decidimos grabar el proceso de enfoque de cada vista en formato de video MJPEG. De esta forma, almacenamos de forma compacta la información de los múltiples planos de enfoque que se visualizan al variar el eje Z (perpendicular al plano de la muestra), para facilitar su utilización posterior. Así podremos abordar la tarea de encontrar planos de enfoque adecuados para localizar todos los granos presentes en la muestra, o aquellos en los que se pueda visualizar la ornamentación superficial característica de algún tipo de grano que pueda ser clave en su identificación.

El conjunto de platinas que utilizamos está preparado en el laboratorio con el objetivo de que, salvo cierta contaminación inevitable, cada platina contenga exclusivamente

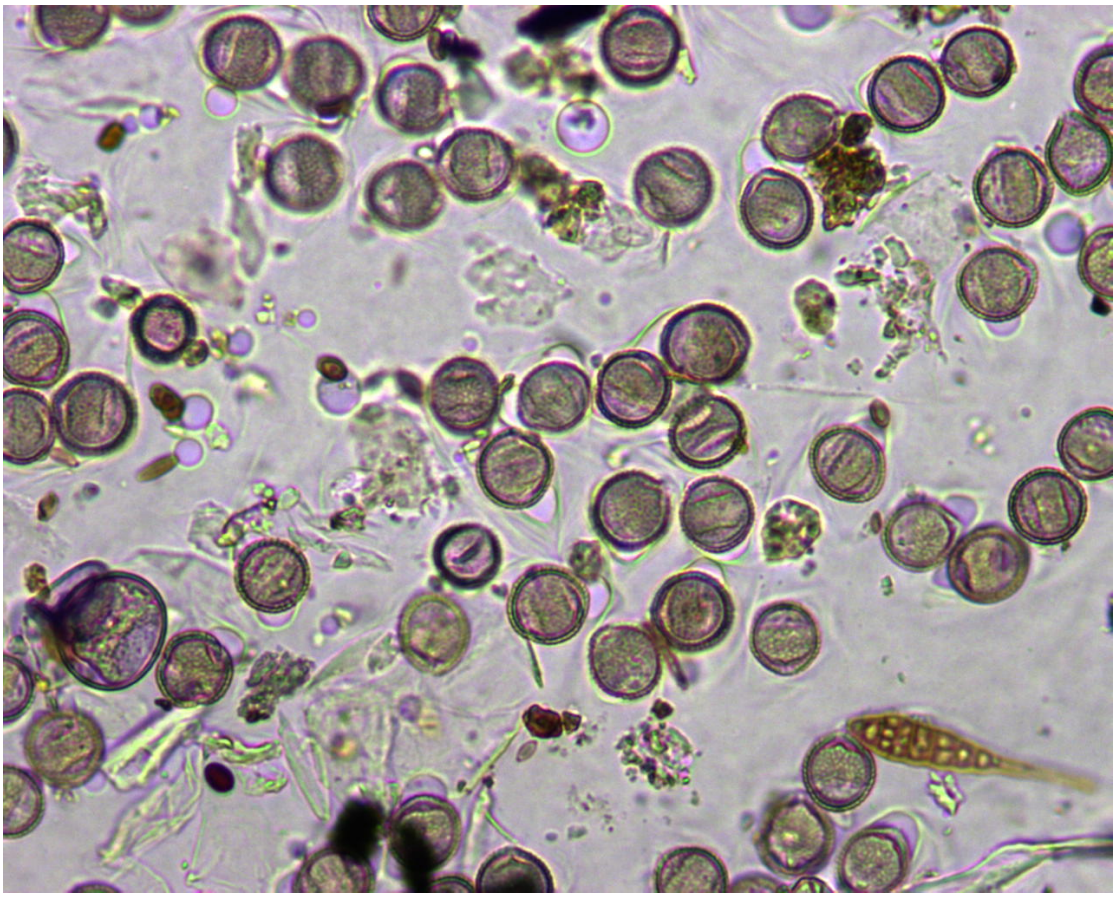


Figura 4.3: Un ejemplo del tipo de imagen a procesar. Se puede observar la presencia de granos solapados y granos de aspecto borroso en la parte inferior izquierda. Los residuos de adhesivo tintados y las burbujas que aparecen en el fondo incrementan la complejidad de la muestra.

un tipo de grano. No obstante, dado que el procedimiento de preparación es manual y requiere la manipulación de plantas, muchas muestras contienen detritos, esporas o restos visibles del adhesivo utilizado como sustrato para captar las muestras. Estos componentes naturales, acercan nuestro conjunto de datos a un entorno de trabajo más realista, pero sin la complejidad de una muestra real, lo que facilita las tareas de marcado de prototipos.

Las platinas se tintan con fucsina para facilitar la visualización de los elementos presentes. La Figura 4.3 muestra un ejemplo del resultado de una captura de las muestras que vamos a utilizar. Este ejemplo, permite apreciar, además de granos aislados, la existencia de granos agrupados o solapados, distintos niveles de tintado, burbujas, restos de adhesivo, polvo, detritos y esporas.

Utilizamos una cámara adaptable al objetivo del microscopio con una resolución de 1280x1024 píxeles. El microscopio se configuró con un aumento x40. Y de cada platina capturamos entre 15 y 40 muestras en función de la concentración de granos presente en

la platina. En total, capturamos 386 muestras (vídeos) de once tipos de polen distintos que aparecen recogidas en la Tabla 4.1. Esta tabla también recoge el número total de granos etiquetados de cada tipo de muestra, separando los granos que aparecen completos en la imagen de los parcialmente visibles.

Cada grano se define en la base de datos por un cuadro delimitador (BBox). El cuadro delimitador se marca en el plano de enfoque en el que el borde del grano aparece más nítido. Por tanto, las dimensiones y ubicación de cada grano marcado únicamente son correctas en el plano de enfoque seleccionado en la base de datos. De esta forma, cada muestra puede tener granos definidos en distintos planos de enfoque para cubrir todos los presentes en la muestra. Obviamente, como se puede apreciar en la Figura 4.3 las muestras pueden contener granos parcialmente visibles en los bordes de la imagen. Estos granos deben identificarse para poder evitar su uso, en su caso, en las tareas de ajuste de los modelos de detección de objetos. Así, la última columna de la Tabla 4.1 identifica los granos parcialmente visibles en las imágenes que han sido marcados en las muestras de este trabajo.

Tabla 4.1: Composición de nuestra base de datos de polen, se indica a nivel de tipo de polen el número de muestras procesado, el número total de granos etiquetados y el desglose de granos etiquetados completo y parcialmente visibles.

Tipos de Pollen	Muestras	Granos Etiquetados	Granos Completos	Granos Parciales
Avena sativa	15	82	62	20
Avena sterilis	17	102	83	19
Cedrus	40	599	390	209
Cupressus	22	90	62	28
Dactylis	60	153	143	10
Lolium	20	260	200	60
Olea	59	496	441	55
Phalaris	20	104	78	26
Plantago	40	344	308	36
Platanus	20	911	753	158
Quercus	73	969	750	219
Total	386	4112	3272	840

4.3.1. Marcado de prototipos

Para etiquetar los granos de polen hemos desarrollado un programa con el entorno de desarrollo Qt [82] que nos permite realizar las distintas tareas de gestión gráfica y sobre la base de datos. El interfaz de etiquetado de prototipos puede verse en la Figura 4.4.

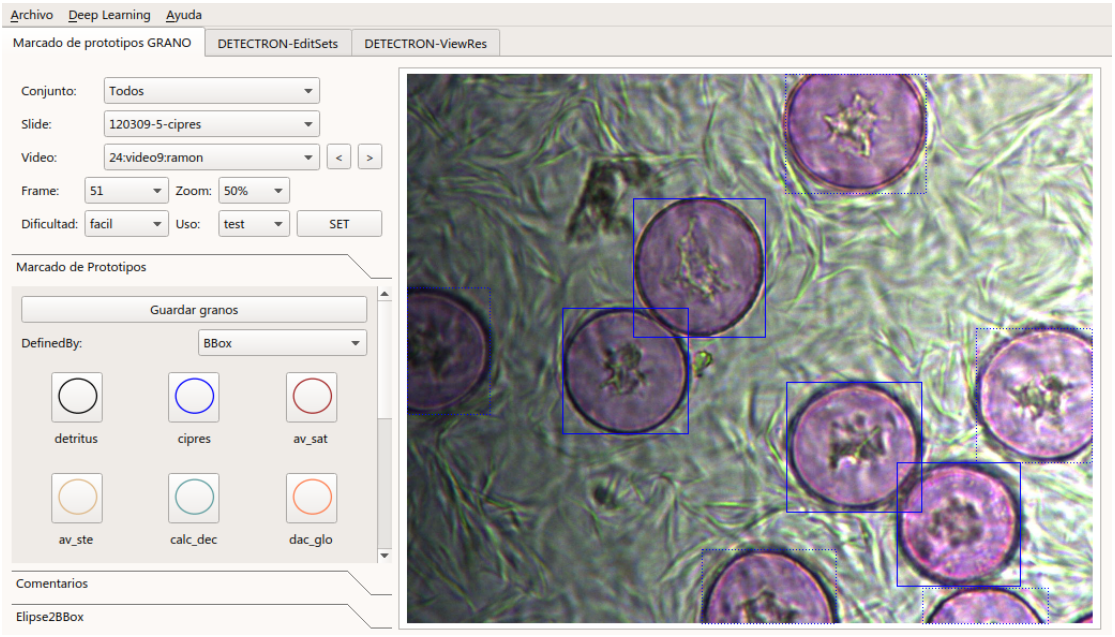


Figura 4.4: Interfaz del programa desarrollado para el etiquetado de granos de polen. Los granos parcialmente visibles en la muestra se muestran con un bounding-box de borde punteado.

La imagen de cada plano de enfoque se establece como fondo de la escena sobre la que se pueden definir y arrastrar objetos gráficos de tipo rectangular o elipsoidal. Estos objetos permiten almacenar tanto la posición y dimensión de cada grano como el resto de características de interés de los objetos a definir.

La clase a la que asociar cada uno de los granos se escoge a priori desde la caja de herramientas de marcado de prototipo, no obstante, puede cambiarse posteriormente mediante el menú contextual del objeto. Para permitir el ajuste fino de la posición y tamaño de cada uno de los objetos, hemos habilitado el uso de los cursores y combinaciones de teclas rápidas que permiten alterar un objeto seleccionado en la escena.

Como ya hemos mencionado, la apariencia de los granos varía en función del plano de enfoque utilizado, por eso, el programa permite etiquetar cada uno de los granos en un plano de enfoque activo, o actualizar el plano una vez insertado en caso necesario. La elección del plano de enfoque se facilita mediante el uso de la rueda del ratón que permite cambiar rápidamente el fondo sobre el que se dibujan los BBoxes.

4.3.2. Distribución de muestras en conjuntos disjuntos

Otros trabajos en este campo, suelen utilizar únicamente granos segmentados, por lo que no suele ser necesario tomar ningún tipo de precaución respecto de la distribución

de los granos, más allá de obtener una distribución razonable de los granos de cada clase o realizar estudios de validación cruzada.

Sin embargo, la complejidad de las muestras que utilizamos en este trabajo varía desde aquellas que únicamente presentan un único grano sobre un fondo prácticamente blanco, a otras que contienen varias decenas de granos, visibles en planos de enfoque distantes, sobre fondos complejos y con presencia de múltiples contaminantes. Por tanto, cualquier distribución de muestras entre conjuntos de entrenamiento y test debe reproducir esta variabilidad para poder obtener resultados con el menor sesgo posible.

Dado que uno de los objetivos de este trabajo es la localización de granos, debemos basar el criterio de distribución respecto del número de granos que contiene cada conjunto sin ignorar el resto de elementos que afectan al problema que abordamos. De esta forma, hemos decidido realizar una distribución de las muestras disponibles cuyo primer objetivo es intentar conseguir que el 60 % de los granos completos se asignen al conjunto de entrenamiento y el 40 % restante se asignen a cualquier actividad de testeo o validación, si fuera necesaria. Obviamente, el segundo objetivo de esta distribución debe ser tratar de igualar la complejidad de ambos conjuntos.

El algoritmo de asignación de muestras a cada conjunto comienza generando listas ordenadas de las muestras de cada tipo de polen. Cada lista se ordena en orden descendente respecto del número de granos completos que contiene. De esta forma, asociamos el número de granos de polen de una muestra a la idea de complejidad de la muestra para cada tipo de polen. A continuación, recorriendo las listas, asignamos cada muestra a uno de los dos conjuntos tratando de no exceder la relación sobre el número de granos entrenamiento/test preestablecida. Al usar las listas ordenadas, las últimas muestras que se asignan suelen contener únicamente un grano, por lo que resulta posible llegar a alcanzar la proporción inicialmente planteada. En la Tabla 4.2, resumimos las características de los dos conjuntos generados respecto del número de muestras asignadas y el número de granos completos que contiene.

Tabla 4.2: Caracteristicas de los conjuntos de datos establecidos para ajustar y testar los clasificadores.

Conjunto	Muestras	Granos Completos
Entrenamiento	251	2037
Test	135	1234

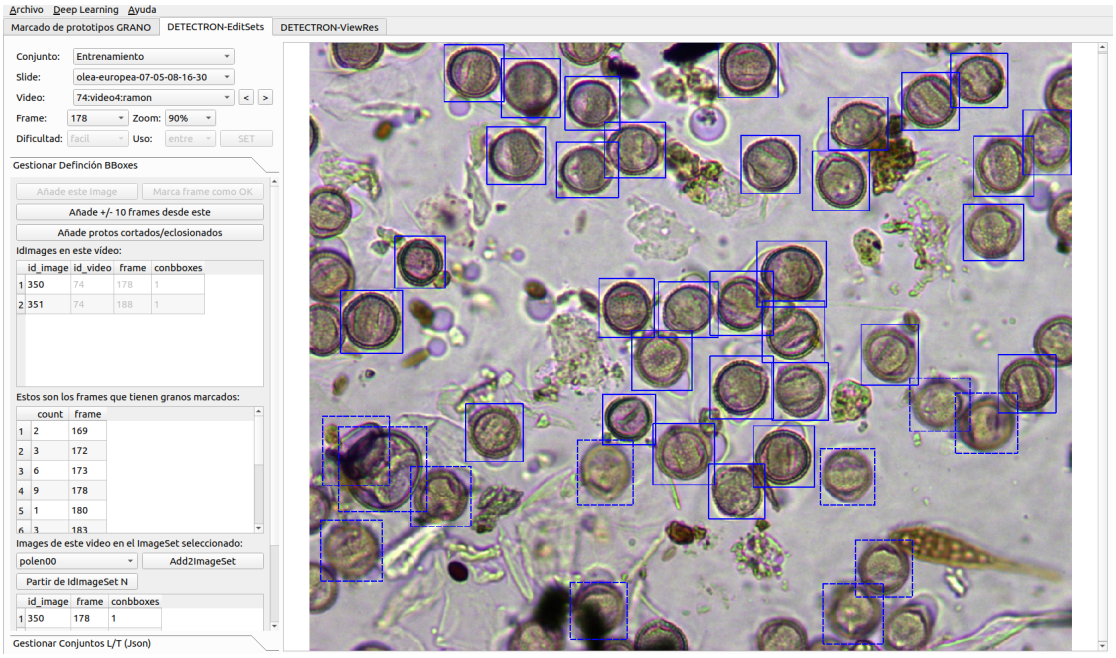


Figura 4.5: Interfaz del programa desarrollado para el etiquetado de granos de polen. Los granos parcialmente visibles en la muestra se muestran con un bounding-box de borde punteado.

4.4. Etiquetado de prototipos para análisis multienfoque

Como hemos indicado en la sección anterior, utilizaremos varios planos de enfoque de cada región a procesar para tratar de maximizar las posibilidades de localización e identificación. Sin embargo, la posición de cada grano de polen y su tamaño varían ligeramente al observar los distintos planos de enfoque. Más aún, en determinados planos el grano puede no ser visible o identificable, mientras otros granos aparecen bien definidos. Dado que los modelos de red considerados, requieren que para cada imagen del conjunto de entrenamiento se definan BBoxes sobre cada uno de los objetos a utilizar como ejemplo, los BBoxes definidos en la base de datos inicial no reflejan la información necesaria para el entrenamiento, pero servirán de base para realizar el etiquetado de prototipos de entrenamiento.

Para realizar este segundo etiquetado de muestras para el entrenamiento de los modelos neuronales, desarrollamos un nuevo módulo en nuestra aplicación de marcado que se muestra en la Figura 4.5. Desde esta pestaña, podemos en primer lugar añadir los planos de enfoque que vamos a utilizar de cada muestra. Por defecto, siempre se añade el plano de la muestra que contiene más granos etiquetados manualmente, y para decidir qué planos adicionales se añaden, se dispone de una lista ordenada con los planos que contienen mayor número de marcas[1].

[1]En la base de datos, cada grano está asociado al plano de enfoque en el que aparece más nítido.

Al añadir un plano, se copian en éste todos los granos definidos de la muestra. A continuación, debemos ajustar las posiciones y tamaño de cada uno de los granos. También, podemos marcar un grano como *borroso*, si a criterio del marcador, resulta difícil su identificación en esa vista. O puede eliminarse completamente si no es visible en el plano o es extremadamente borroso.

Dado que es muy costoso, en términos temporales, realizar este segundo etiquetado sobre todos los planos de enfoque grabados, decidimos utilizar aquellos planos de enfoque que proporcionasen una vista relevante para la identificación de todos los granos presentes en la muestra. De esta forma, sólo se añade una imagen al conjunto de entrenamiento si proporciona una vista en la que la ornamentación de un grano es visible en ésta, y no en una imagen previamente considerada. O cuando presenta granos enfocados que no aparecían en una muestra previa.

Los algoritmos considerados recomiendan el uso de objetos completamente visibles en las imágenes de entrenamiento, por tanto, hemos ignorado en este segundo etiquetado la corrección de las posiciones y tamaños de los granos cortados, que pueden aparecer en los bordes de cada imagen. Esta decisión no debería suponer un problema en el funcionamiento del sistema final, dado que un grano cortado presente en una muestra, debería aparecer completo en una vista adyacente al desplazar la posición de la platina en el microscopio, usando un solapamiento entre muestras suficiente.

A la hora de elegir las imágenes que utilizaremos en el conjunto de test, debemos tener en cuenta que no es necesario realizar este etiquetado detallado para cada imagen a procesar. La evaluación de rendimiento se realizará por ubicación del grano en la posición correcta en los experimentos de localización. Y en los experimentos de detección, además de una localización correcta, se tendrá en cuenta que la clase asignada sea correcta. En este caso por tanto, únicamente debemos asegurarnos de que el conjunto de imágenes a utilizar contiene todos los planos de enfoque necesarios para llevar a cabo la localización de todos los granos visibles en la muestra. Tras analizar la base de datos, concluimos que con la velocidad de desplazamiento en el eje Z utilizada para capturar las imágenes, debemos estudiar al menos 10 imágenes en ambas direcciones del eje Z respecto del plano de referencia[2]. De esta forma, para cada muestra del conjunto de test deberemos procesar 21 imágenes antes de proporcionar un resultado. La Tabla 4.3 resume la composición final del conjunto de entrenamiento y test en términos del número de muestras asignado a cada uno de ellos, el número de granos completos que incluye, y el número de imágenes utilizadas.

[2]Plano con mayor número de granos nítidos etiquetados.

Tabla 4.3: Caracteristicas globales de los conjuntos de datos establecidos para ajustar y testar los modelos de detección y clasificación.

Conjunto	Muestras	Granos completos	Imágenes
Train	251	2038	582
Test	135	1235	2863

Dado que el trabajo de etiquetado fue iterativo, el módulo de edición de conjuntos de nuestra aplicación permite definir nuevos conjuntos de entrenamiento y test a partir de un conjunto previo, añadiendo sobre este nuevos planos de enfoque y sus correspondientes correcciones sobre los granos visibles. Por ello, algunos experimentos detallados en el capítulo de resultados pueden contener un número distinto de imágenes.

4.5. Descripción de la base de datos SQL

La base de datos SQL para la realización del trabajo ha sido creada utilizando PostgreSQL [83]. PostgreSQL es un sistema de gestión de bases de datos con licencia *Open Source*.

La estructura de nuestra base de datos SQL es la representada en la Figura 4.6. Como se puede observar, en esta figura están representadas las principales tablas que la forman así como las relaciones que existen entre ellas.

A continuación se comentan brevemente cada una de las tablas que integran nuestra base de datos. Las tablas en fondo naranja son tablas de configuración, aquellas que tienen fondo amarillo contienen resultados que han requerido de marcado manual y el resto de tablas contienen resultados brutos del clasificador (azul) o resultados finales (verde).

- **Slides:** Esta tabla almacena información relativa a la platina utilizada. Recoge el tipo de granos utilizado para su preparación, el proceso de tintado utilizado y la fecha de preparación.
- **Vídeos:** Almacena información relativa a cada una de las muestras que se extraen de una platina. Incluye el nombre base de las imágenes, el número de imágenes de la muestra, el índice de la muestra más nítida y una asignación de uso para el conjunto de entrenamiento o test.
- **Clases:** Recoge las distintas clases consideradas en este trabajo, asociadas a los distintos tipos de grano considerados, el fondo y otras clases auxiliares útiles para

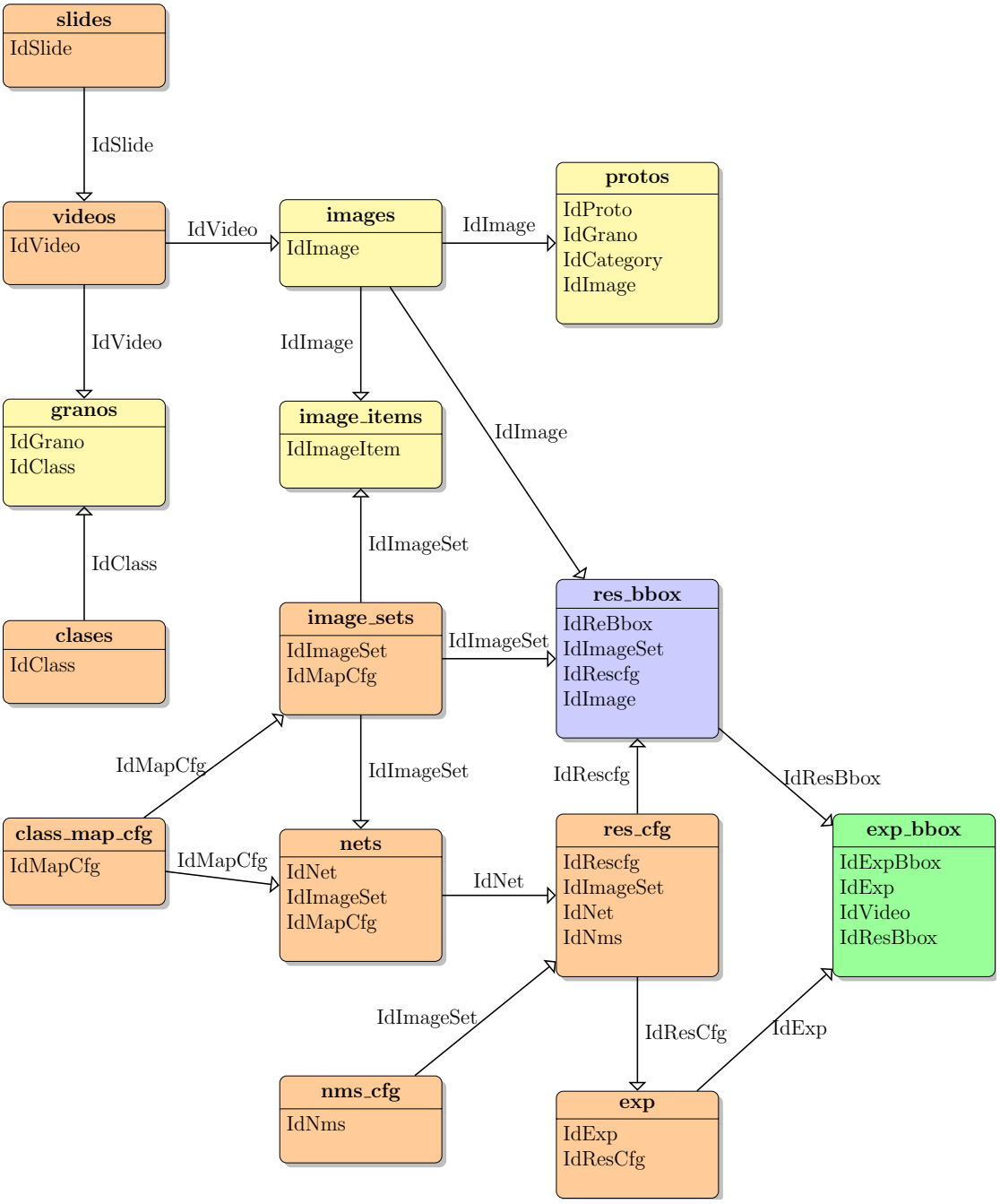


Figura 4.6: Estructura y relaciones principales de la base de datos

estudiar agrupaciones de tipos de grano, marcar la extensión de algunos detritos o componentes extraños en la imagen.

- **Class Map Cfg:** Define posibles agrupamientos entre las clases consideradas. Por ejemplo para entrenar redes especializadas en localización de granos independientemente de su clase o realizar agrupamientos a nivel de especie.
- **Granos:** La tabla de granos contiene la definición de las marcas de referencia para cada uno de los granos definidos sobre una muestra. Recoge las coordenadas de

la esquina superior izquierda, el ancho y el alto del BBox que encierra el grano sin desbordar las dimensiones de la imagen. Además, contiene la clase a la que pertenece el grano, el índice de la imagen en la que el grano se considera enfocado y se ha marcado. Y por último, se consideró útil añadir otros detalles sobre el grano como si está solapado con otro grano o aparece cortado en el borde de la imagen.

- **Images:** Esta tabla recoge información asociada a cada una de las imágenes sobre las que definir prototipos a utilizar en los procesos de ajuste de los clasificadores. Por compatibilidad, también incluye aquellas imágenes a utilizar en las tareas de evaluación del rendimiento de los clasificadores, aunque sobre estas imágenes no se definan prototipos.

- **Protos:** La tabla de prototipos recoge las marcas de referencia a utilizar para cada grano en una de las imágenes de entrenamiento. Adicionalmente, se especifica si el grano en ese plano de enfoque es identificable o aparece borroso.

- **Image Sets:** Almacena el nombre y las características de los distintos conjuntos de imágenes definidos. Además, recoge los parámetros de generación de las imágenes para su utilización con la red neuronal, como por ejemplo el escalado respecto de la imagen original. También controla si junto con las imágenes, debe generar las anotaciones para los prototipos de la imagen con coordenadas escaladas para el entrenamiento permitiendo la exclusión de granos cortados o borrosos.

- **Image Items:** Es una tabla auxiliar que contiene todas las entradas de imagen a utilizar en un conjunto de imágenes determinado y la distribución respecto de su uso.

- **Nets:** La tabla de redes recoge las características de configuración para entrenamiento y rutas de los archivos para las distintas redes entrenadas.

- **Res Cfg:** La tabla de configuración de resultados agrupa los parámetros de configuración a utilizar para generar los BBoxes de salida sobre un conjunto de imágenes dado. Entre estos se incluye la configuración del algoritmo NMS a usar a nivel de imagen.

- **Exp:** La tabla de experimentos permite establecer los parámetros de fusión de BBoxes de la pila de imágenes para determinar las detecciones finales para una configuración de resultados dada. Además, contiene los parámetros de configuración del algoritmo de rendimiento. Y por último, almacena las estadísticas de rendimiento del experimento configurado una vez finalizado.

- **Res BBox:** Contiene las coordenadas y puntuación de los candidatos a grano generados por la red neuronal para cada una de las imágenes del conjunto de entrenamiento.
- **Exp BBox:** Contiene las coordenadas de los BBoxes resultado de la fusión de la pila de imágenes de cada muestra. Adicionalmente, almacena el resultado de aplicar el criterio de rendimiento del experimento al comparar el BBox con la marca de certeza almacenada en la tabla granos.

La adición de redes, configuración de resultados y experimentos se realiza mediante tres asistentes añadidos a nuestro programa PolenProt. El primero de ellos se muestra en la Figura 4.7 y permite que nuestro entorno gestione las rutas a los archivos de configuración y pesos. Además, recoge la asignación de clases que realiza la red, el conjunto de imágenes utilizado para su entrenamiento y las características de la red como el modelo o la red utilizada como backbone.

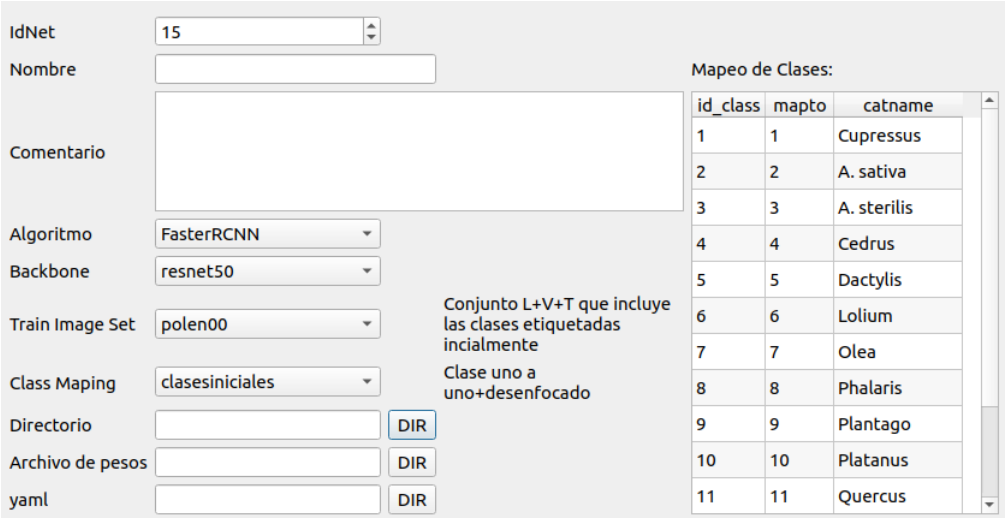


Figura 4.7: Asistente de configuración de nuevo modelo de red.

El asistente para configurar una nueva salida de red mostrado en la Figura 4.8, nos permite asociar una red ya definida con un conjunto de test dado y seleccionar el algoritmo NMS que se aplicará a nivel de imagen.

Por último, el asistente para añadir un experimento mostrado en la Figura 4.9, nos permite definir un experimento para una configuración de resultados previamente definida. En este caso, una vez seleccionada una configuración de resultados asociamos un algoritmo de fusión de los BBoxes del apilamiento con sus parámetros correspondientes. Y finalmente, seleccionamos el algoritmo de medida de la exactitud de la localización, generalmente IoU, y establecemos sus parámetros.

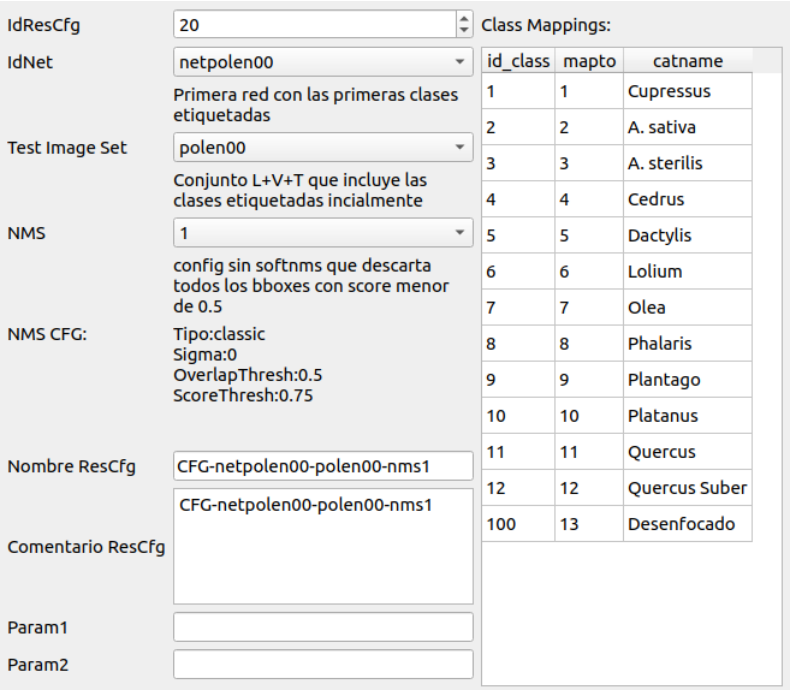


Figura 4.8: Asistente de configuración de nueva configuración de resultados.

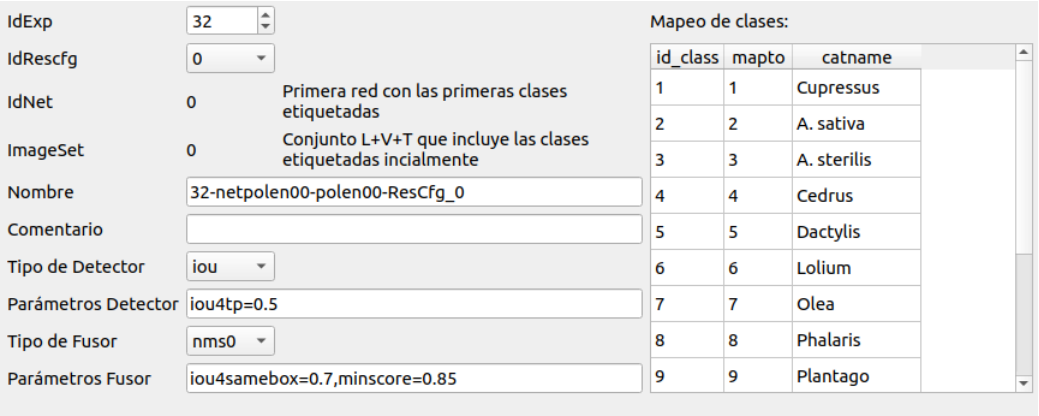


Figura 4.9: Asistente de configuración de nuevo experimento.

Todos los identificadores definidos con los asistentes nos permiten enlazar las distintas relaciones posibles entre los componentes del sistema y facilitar la gestión de los resultados. De esta forma, mantenemos la integridad de la base de datos al no requerir la inserción manual con el posible error asociado a la consulta de identificadores previa a la inserción.

4.6. Plataforma de aprendizaje profundo

Como mencionamos en la Sección 3.4 existen múltiples plataformas de aprendizaje profundo sobre las que se han generado distintos entornos que facilitan el desarrollo de

modelos neuronales. En este trabajo hemos usado el entorno *Detectron* [84], que se desarrolló con el API de Caffe2 que a día de hoy está integrado dentro de PyTorch [85]. El objetivo del entorno Detectron es ser lo suficientemente flexible para permitir la implementación y evaluación rápida de proyectos de investigación.

Este entorno incluye, bajo un entorno de configuración común, implementaciones de los algoritmos más recientes junto con los modelos considerados de referencia. Entre ellos disponemos de Faster R-CNN, RetinaNet o Mask R-CNN. Algunos de los operadores de detectron no poseen implementaciones en CPU; por tanto, se requiere una GPU para llevar a cabo la detección. Afortunadamente, el coste de las GPU con grandes cantidades de memoria se ha reducido mucho, y una simple tarjeta gráfica con prestaciones de usuario permite ejecutar el algoritmo.

En este trabajo usamos una tarjeta gráfica GTX 1070 Ti con 8 GB de RAM. En reconocimiento, si se requiere un sistema compacto, el algoritmo puede correr sobre una plataforma Jetson [86], que adicionalmente proporciona un conjunto de GPIOs para gestionar los servomotores del microscopio.

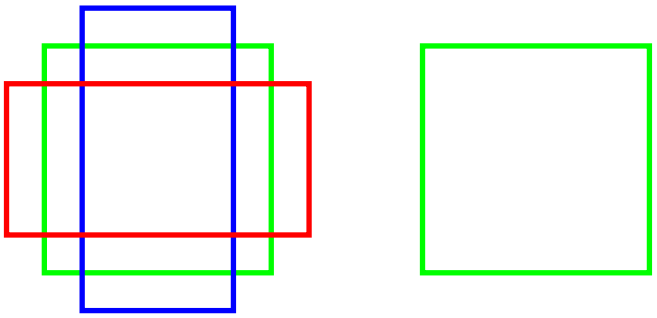
4.7. Modelos de red utilizados

Los experimentos realizados con Faster R-CNN sobre los conjuntos PASCAL VOC [87] y MS COCO [88] alcanzaron elevada precisión en detección, junto con una elevada velocidad de procesamiento, con una velocidad media de 5 imágenes por segundo. La velocidad de procesamiento alcanzada y los grandes resultados obtenidos en detección nos llevó a considerar la idoneidad de este algoritmo como base de resolución de nuestro problema. Sin embargo, debíamos superar dos dificultades previas: el tipo de imágenes a procesar y el relativamente bajo número de imágenes etiquetadas.

Las imágenes utilizadas para ajustar los *backbone* convolucionales de referencia pertenecen a la vida real macroscópica, con categorías como persona, bicicleta, coche o moto. Este espacio de imágenes da lugar a un determinado ajuste de los filtros del *backbone*, que puede no ser el más adecuado para abordar el estudio de un conjunto de imágenes microscópicas, con una estructura totalmente distinta a las imagenes de la vida diaria. Por otro lado, el ajuste desde cero de la red completa resultaba poco esperanzador debido a que el número de imágenes etiquetadas que teníamos disponibles nos parecía insuficiente. Por lo que debíamos verificar que la transferencia de aprendizaje descrita en la Sección 3.5.7 nos permitiría reutilizar las redes de referencia preentrenadas y reentrenarlas para nuestro nuevo espacio de imágenes.

En este trabajo hemos utilizado tanto un detector de dos etapas (Faster R-CNN con FPN) como un detector de una etapa (RetinaNet), para evaluar el rendimiento en localización y detección de granos de polen sobre nuestro conjunto de datos. En ambos casos, las redes operan a nivel de imagen y las regiones candidatas generadas para una misma muestra se procesan posteriormente para determinar las propuestas finales según el algoritmo descrito en la Sección 4.9.

La implementación estándar del modelo Faster R-CNN usa tres marcos de anclaje de referencia como se muestran en la Figura 4.10a. Los marcos rectangulares son útiles para modelar objetos alargados en los ejes X o Y. Como pudimos comprobar empíricamente que los granos de polen de nuestro conjunto de datos son generalmente circulares, decidimos prescindir de los marcos de anclaje rectangulares y limitar la generación de anclas de entrenamiento a marcos cuadrados, como los de la Figura 4.10b. El modelo RetinaNet también utiliza marcos con las mismas relaciones de aspecto por defecto, por lo que en este caso también decidimos limitar las relaciones de aspecto a 1:1.



(a) (b)

Figura 4.10: (a) Marcos de anclaje que usa el modelo Faster R-CNN por defecto con relaciones de aspecto 1:1 (verde), 1:2 (azul) y 2:1 (rojo). (b) Único marco de anclaje base usado en nuestra implementación con relación de aspecto 1:1.

4.8. Precisión en la localización

Uno de los problemas que surgen a la hora de poder realizar una valoración de la representatividad que aporta al estado del arte cada uno de los trabajos, es la heterogeneidad de criterios y métricas utilizadas en cada uno de ellos para mostrar sus resultados.

En este contexto, una de las métricas más aceptadas actualmente para medir la exactitud con la que una propuesta ha localizado un objeto en una imagen es la *Intersección sobre la Unión* (IoU) [89]. Esta métrica es simplemente un cociente de dos áreas como muestra la Ecuación 4.1, donde el numerador es el área en el que se solapan la predicción (P) y la marca de referencia (G) y el denominador es el área total cubierta tanto por P como por G, como muestra gráficamente la Figura 4.11. Por tanto, aquellas propuestas que

mejor se ajusten a la posición y tamaño de la referencia marcada G tendrán un índice de IoU superior a otras con menor solapamiento.

$$IoU(P, G) = \frac{|P \cap G|}{|P \cup G|} \tag{4.1}$$

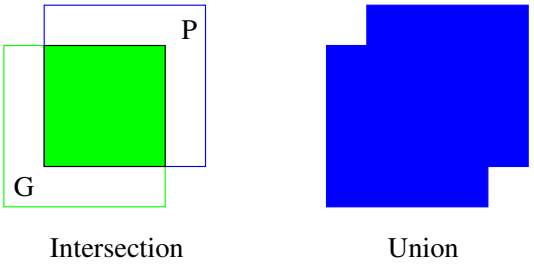


Figura 4.11: Definición gráfica del concepto de Intersección sobre la Unión

Como referencia, en retos internacionales de detección de obejetos macroscópicos, un índice de IoU > 0.5 se considera generalmente un un objeto bien localizado [89]. La Figura 4.12 muestra un conjunto de posibles configuraciones de solapamiento entre marcas de referencia (en verde) y diferentes propuestas de localización proporcionadas por un sistema dado (en azul). Como puede observarse, el valor de referencia de citado (IoU > 0.5) puede no ser indicativo de una localización muy acertada en cuanto al tamaño del objeto localizado por la propuesta. Sin embargo, cuando este índice se aproxima a 0.9, puede considerarse que la exactitud en la localización y tamaño indicados por una propuesta es elevada.

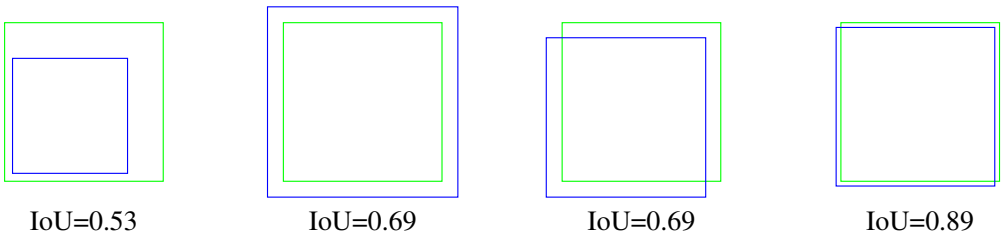


Figura 4.12: Representación gráfica de distintos valores de Intersección sobre la Unión

En este trabajo, especificamos la precisión en localización a partir de los valores medios de de IoU de cada propuesta correcta con el marco de referencia almacenado en la base de datos. Adicionalmente, indicamos la desviación estándar de los valores medios calculados para proporcionar información respecto de la dispersión de las medidas obtenidas.

4.9. Fusión de la información multifocal

Como ya hemos expuesto en la Sección 4.2, para cada muestra en estudio, consideramos varias imágenes obtenidas al variar el eje z para buscar de forma exhaustiva los granos presentes en la muestra, emulando la actuación del experto al recorrer la platina. Si el clasificador es robusto, obtendremos todos los granos identificables en cada imagen. Teniendo en cuenta los granos encontrados en todos las imágenes procesadas, podemos encontrar todos los granos presentes en la muestra, independientemente de su grado de visibilidad o identificabilidad en un plano de enfoque específico, como ya mostramos esquemáticamente en la Figura 4.2.

En la fase de reconocimiento, se determina un plano de referencia mediante autoenfoque[3] para cada muestra y procesamos el plano de referencia junto con 10 planos por encima y por debajo de éste. Por tanto, cada muestra se procesa en al menos 21 planos de enfoque y, como resultado obtenemos una lista de BBoxes asociados con las propuestas realizadas por la red para cada imagen junto con una categoría y puntuación (*score*) asociadas.

Se realiza un primer filtrado de esta lista de propuestas, descartando aquellas que tengan una puntuación inferior a un umbral de certeza dado. De esta forma, se rebaja la carga computacional del resto de los algoritmos que es necesario aplicar hasta obtener las propuestas finales.

La salida de los modelos neuronales utilizados, proporciona una lista de BBoxes que requiere la aplicación de un algoritmo NMS para eliminar las propuestas asociadas a un mismo objeto. Detectron usa por defecto el algortimo conocido como *Greedy NMS*, que aparece recogido en Algoritmo 2. Este algoritmo comienza ordenando la lista de propuestas generadas por la red descendentemente respecto de su *score*. A continuación, comenzando con el BBox de mayor puntuación, determina el solapamiento con el BBox con *score* inmediatamente inferior y lo elimina del listado si el solapamiento N_t supera un determinado umbral. Tras repetir el procedimiento con el resto de los BBoxes que van quedando en el listado, únicamente permanecerán los BBoxes de mayor puntuación no solapados D asociados a la imagen. La métrica de solapamiento utilizada habitualmente en este algoritmo es la intersección sobre la unión definida en la Ecuación 4.1.

[3]Esta posición de referencia podría realizarse manualmente al posicionar la platina en el microscopio.

Algoritmo 2: Algoritmo Greedy NMS

Entrada: Umbral de solapamiento N_t
Datos: Lista de BBoxes con puntuación R
Resultado: Lista de BBoxes no solapados D
Ordena R por puntuación descendente en D
for $i = 1$ **to** *D.conteo* **do**
  for $j = i + 1$ **to** *D.conteo* **do**
    if $IoU(d_i, d_j) > N_t$ **then**
      Elimina d_j
    end
  end
end
Devuelve D

4.9.1. Fusión básica de la pila

El algoritmo más básico de fusión de la pila de imágenes consiste en considerar todas las propuestas generadas por la red para todas las imágenes de una muestra, independientemente de la imagen de procedencia. En este caso tendríamos un listado de BBoxes en el que muchas de las propuestas estarían solapadas. Tras descartar aquellas que presenten un *score* inferior a un valor dado aplicamos el algoritmo *Greedy NMS* para proponer como granos finales aquellas BBoxes con mayor tasa de certeza y solapamiento entre propuestas inferior al indicado como argumento del NMS.

Este método de fusión elimina la información asociada al plano de enfoque en el que se localizan las propuestas y presenta mucha dependencia respecto de la variación en valor del índice de certeza. Un modelo que tienda a sobrepuntuar las propuestas de objeto, provoca que el algoritmo NMS retenga siempre las primeras propuestas no solapadas que se añaden a la lista.

4.9.2. Fusión de la pila con filtrado en imagen

Una segunda opción de fusión del apilamiento de imágenes aplicaría el algoritmo de supresión de no máximos a nivel de imagen, para retener únicamente las propuestas de mayor interés de cada imagen. Una vez determinados los BBoxes con mayor puntuación y no solapados de una imagen, aplicamos un segundo algoritmo de supresión de no máximos para agrupar los granos detectados en las imágenes del apilamiento, que pertenecen a distintas vistas de un mismo objeto. De esta forma obtendremos idealmente los granos finales presentes en la muestra. Sin embargo, hemos comprobado empíricamente que la posición y tamaño de los granos en el campo visual de una muestra, varía ligeramente

al cambiar el plano de enfoque del microscopio, por lo que debemos tener en cuenta este efecto a la hora de analizar el rendimiento del algoritmo NMS utilizado.

Un segundo problema que encontramos al fusionar los BBoxes procedentes de distintos planos fue la no eliminación de BBoxes muy pequeños incluidos en otras propuestas con mayor área. El problema se debía a la definición estándar de la IoU, que puede proporcionar un valor bajo si la diferencia entre las áreas de las regiones solapadas es muy grande aún cuando una región esté totalmente solapada con otra como muestra la Figura 4.13. En este caso, el área de la intersección se corresponde con el área de la propuesta más pequeña, y al dividir por el área de la unión podemos llegar a obtener un valor de solapamiento inferior al necesario para eliminar una de las dos propuestas con lo que ambas sobrevivirían al filtrado.

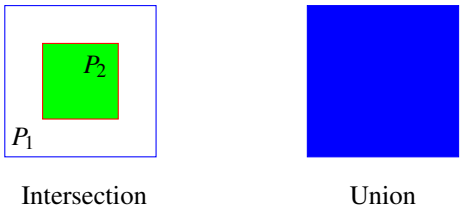


Figura 4.13: Intersección sobre la unión cuando un BBox muy pequeño (P_2) está totalmente incluido en uno mucho mayor (P_1). El área de la intersección en este caso se corresponde con el área de P_2.

Como resultado de este efecto, ambas propuestas de grano deberán ser evaluadas para calcular el rendimiento del sistema, degradando las métricas de rendimiento global. Para eliminar estas situaciones, utilizamos una variación de la definición estándar de intersección sobre la unión que proporciona un valor de solapamiento total cuando el área de la intersección se corresponde con el área de cualquiera de las dos propuestas de entrada como muestra la Ecuación 4.2.

$$IoU(P_1, P_2) = \begin{cases} 1.0 & \text{Si } |P_1 \cap P_2| = P_1 \\ 1.0 & \text{Si } |P_1 \cap P_2| = P_2 \\ \frac{|P_1 \cap P_2|}{|P_1 \cup P_2|} & \text{Resto} \end{cases} \tag{4.2}$$

4.9.3. Supresión de no máximos en detección

A la hora de llevar a cabo el estudio de detección, podemos encontrarnos ante problemas adicionales al tener en cuenta que la red puede asignar distintas categorías a un mismo grano al analizar las distintas imágenes del apilamiento, como muestran los granos rojo y verde de la Figura 4.14. La variación en la clase puede ser debida por ejemplo a

la aparición de ornamentación característica en una vista determinada, lo que sería determinante aunque únicamente apareciese en una vista. La eliminación en estos casos de la propuesta con menor puntuación puede no ser fácil cuando ambas presentan el mismo valor de puntuación, por lo que puede que a la vista de los resultados, debamos tener en cuenta algoritmos de supresión más complejos que un simple Greedy NMS.

Además, podemos encontrar un problema adicional al aplicar el algoritmo NMS cuando nos encontramos con dos o más granos distintos muy solapados que en el mejor de los casos se detectan en planos muy distantes y con puntuaciones distintas como muestran los granos naranja y azul de la Figura 4.14. La aplicación estándar del algoritmo NMS eliminaría el segundo grano que entrase en el listado al presentar un IoU muy elevado con la primera propuesta.

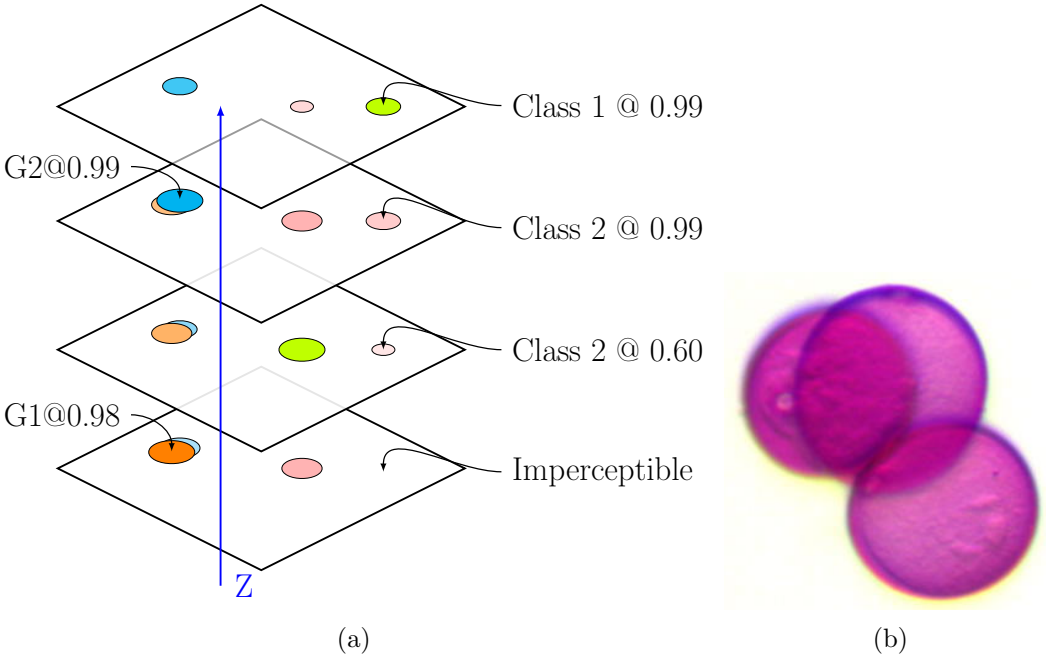


Figura 4.14: Supresión de no máximos en detección. (a) Ejemplificación de una condición de clasificación indefinida al usar varios planos de enfoque. Los granos verdes representan objetos nítidos y los granos rojos objetos borrosos. Los granos naranja y azules representan un patrón de alto solapamiento entre granos. (b) Granos altamente solapados en una muestra de Dactilys.

Para tratar de gestionar este segundo problema, utilizamos umbrales distintos para los algoritmos NMS que aplicamos a nivel de imagen y a nivel del de apilamiento. Así, usaremos un umbral de solapamiento menor a nivel de imagen, por ejemplo $N_t = 0.5$, lo que genera un radio de acción importante del NMS, sobre las propuestas cercanas con menor índice de certeza. Y reducimos el radio de acción del NMS al gestionar las imágenes del apilamiento, usando por ejemplo un segundo umbral más restrictivo $N_t = 0.7$. Este criterio se basa en la idea de que dos propuestas asociadas a un mismo grano en planos consecutivos del apilamiento deben estar muy cercanas. Pero si nos encontramos ante dos

granos solapados, generalmente las posiciones de las propuestas con mayor puntuación en el apilamiento estarán más alejadas, por lo que un mayor valor de N_t puede que preserve esta propuesta al procesar el apilamiento. De esta forma tratamos de preservar las propuestas de granos localizables en planos alejados que se visualizan solapados al fusionar el apilamiento en z. El rendimiento de este criterio obviamente, está ligado a que los modelos de red proporcionen BBoxes muy ajustados a los bordes de los objetos localizados.

4.10. Métricas de rendimiento

El rendimiento de cada sistema se expresa midiendo la precisión (P), sensibilidad o exhaustividad (*recall*, R), *F1-score* ($F1$), *weighted–averaged F1-score* ($F1_W$) y el valor medio de IoU de las propuestas generadas con las marcas de certeza almacenadas. Todos estos parámetros se calculan sobre todas las muestras del conjunto de test de la Tabla 4.3. Las imágenes de entrada únicamente se escalan a un tamaño de 640x512 sin realizar ningún procesamiento adicional.

En el cálculo de las métricas de rendimiento para localización, consideramos un verdadero positivo (TP) cuando una predicción generada por el sistema presenta una IoU con una de las marcas de referencia almacenadas en la base de datos de al menos 0.5. Contamos un falso negativo (FN) si ninguna de las propuestas generadas por la red para una muestra dada alcanza el valor mínimo de solapamiento para un grano dado. Finalmente, consideramos un falso positivo (FP) siempre que una propuesta generada por el sistema no alcance el valor mínimo de solapamiento con ninguna de las marcas de certeza recogidas en la base de datos para la muestra o si se asocia más de un verdadero positivo a un grano dado. En base a estas definiciones, las métricas de precisión (P) y sensibilidad (R) vienen dadas por las expresiones habituales recogidas en las Ecuaciones 4.3 y 4.4 respectivamente. Donde la precisión nos indica la fracción de propuestas generadas que son correctas, y la sensibilidad es la fracción de objetos a localizar o detectar que lo han sido correctamente.

$$P = \frac{TP}{TP + FP} \tag{4.3}$$

$$R = \frac{TP}{TP + FN} \tag{4.4}$$

Aunque no hemos usado los granos parcialmente visibles situados en el borde las imágenes para entrenar los modelos, en algunos casos, cuando el grano es bastante visible el

entorno puede generar propuestas en el borde. En estos casos, la propuesta no se considera un FP si realmente cubre un grano parcialmente visible que está almacenado en la base de datos. En estos casos, identificamos las propuestas como verdaderos positivos en el borde (TP_E) pero no contribuyen a las métricas de rendimiento global. Obviamente, al desplazar la posición de la platina en el microscopio en dirección X o Y, el grano de polen detectado parcialmente en la muestra inicial aparecerá completo en otra vista en donde sí se contabilizará como un TP.

La precisión en localización se especifica mediante los valores medios de solapamiento, en términos de IoU, entre las propuestas TP y las marcas de referencia almacenadas en la base de datos. Los valores se proporcionan tanto de manera global como para cada uno de los tipos de polen considerados en este trabajo. De esta forma podremos identificar los posibles fallos sistemáticos de los modelos asociados a tipo de polen particular. En caso de obtener un buen rendimiento en la determinación de los BBoxes asociados a cada grano, podríamos usar estas dimensiones para llevar a cabo una estimación media de los granos de polen en cada periodo.

Obviamente, en los experimentos de detección, se requiere que además del criterio de solapamiento mínimo una propuesta tenga también la clase correcta para ser contabilizada como un verdadero positivo. Sin embargo, un error en la clase asignada no evita que esa propuesta pueda usarse para realizar una estimación de la concentración polínica global, por lo que etiquetaremos estas propuestas como clase errónea (WC). Obviamente y como es habitual, las detecciones tipo WC se contabilizan como FP a la hora de analizar el rendimiento en detección, por lo que la precisión en este caso viene dada por la Ecuación 4.5.

$$P = \frac{TP}{TP + FP + WC} \tag{4.5}$$

Por último, en cualquier clasificador existe un balance entre precisión y sensibilidad, y un modelo puede ser optimizado (por ejemplo ajustando el umbral de confianza o del NMS) para priorizar la precisión o la sensibilidad dependiendo del objetivo. Bajo esta premisa, el índice F1 pesado y promediado (habitualmente *weighted-F1*) permite reunir en un único marcador la información en la que precisión y sensibilidad alcanzan sus valores máximos e idénticos sobre la curva precisión vs. sensibilidad sobre todas las clases en estudio.

El índice F1 estándar se define según la Ecuación 4.6 y varía entre 0 (el peor) y 1 (óptimo) y si el modelo se ajusta para favorecer la precisión o la sensibilidad el índice F1 descenderá.

$$F1 = 2 \cdot \frac{P \cdot R}{P + R} \tag{4.6}$$

En nuestro caso, dado que no disponemos del mismo número de granos de cada clase en el conjunto de test, la utilización directa del índice F1 falsearía el rendimiento. Por tanto, usamos la versión pesada de este índice, que pondera los índices F1 de cada clase respecto del número de granos que aporta cada clase. La Ecuación 4.7 presenta la definición de este parámetro donde N es el número total de granos a detectar, N_i es el número de granos a detectar de la clase i-ésima y por último $F1_i$ es el índice $F1$ de la clase i-ésima.

$$F1_W = \frac{1}{N} \sum_{i=1}^{N_{cls}} N_i \cdot F1_i \tag{4.7}$$

4.11. Gestión de resultados

Para gestionar los resultados hemos añadido una tercera pestaña a nuestro programa de gestión de prototipos (ver Figura 4.15). Desde esta pestaña, podemos visualizar los resultados que proporciona una red sobre cada una de las imágenes procesadas y filtrar las propuestas en base a su *score*.

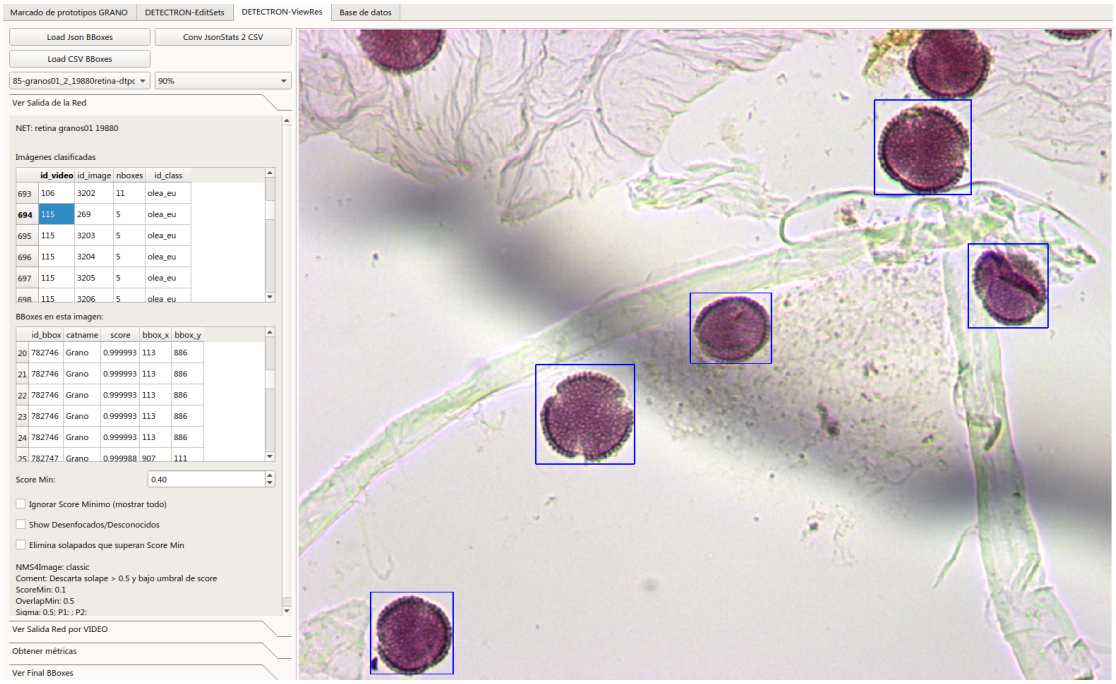


Figura 4.15: Interfaz del programa desarrollado para la visualización de resultados. Vista de resultados para cada imagen.

También, podemos mostrar los resultados solapados para todas las imágenes de una misma muestra. Esta vista, nos permite observar las distintas BBoxes propuestas por la red para todas las imágenes de la muestra y tras seleccionar un nivel de filtrado, apreciar la precisión en la localización o los posibles errores de la red para intentar corregirlos en los nuevos modelos a entrenar. La Figura 4.16 muestra un ejemplo de localización en una platina con granos de ciprés mostrando todas las propuestas que presentan un *score* superior a 0.4. Como puede apreciarse, en este caso todos los granos completos se localizan con BBoxes que cubren con mayor o menor éxito. Tras la aplicación del algoritmo de fusión del *z-stack*, únicamente se retendrán los BBoxes con mejor *score*.

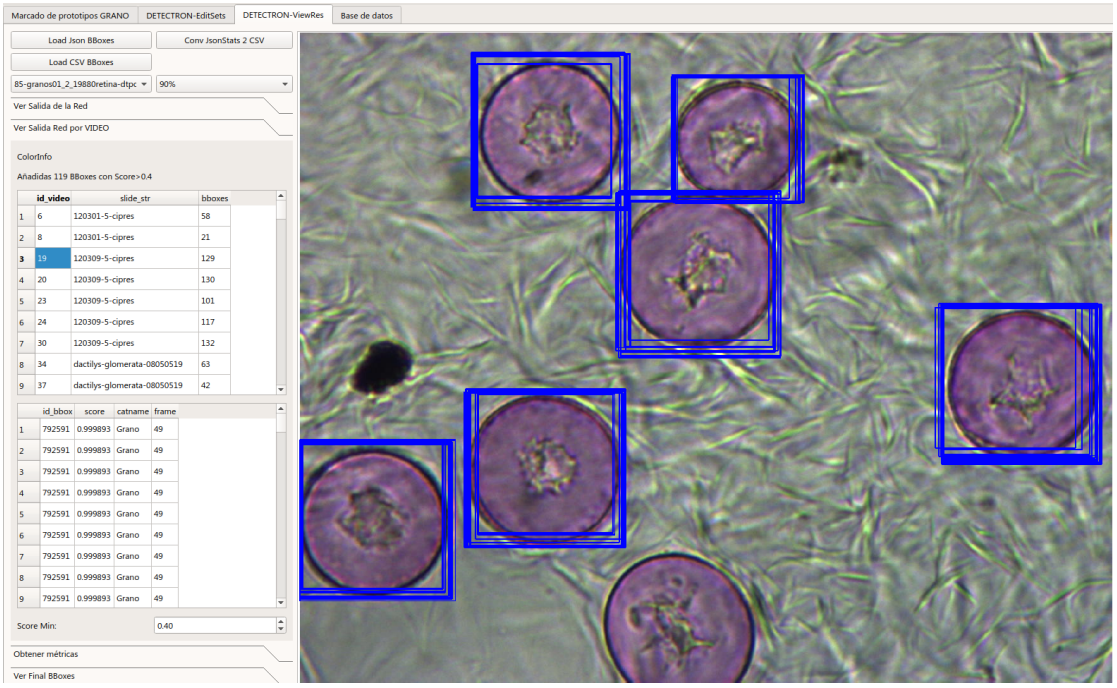


Figura 4.16: Interfaz del programa desarrollado para la visualización de resultados. Vista de resultados del *z-stack*.

Una tercera vista, nos permite obtener las distintas métricas de rendimiento del experimento configurado. Y para visualizar los resultados obtenidos en la herramienta anterior, la vista final mostrada en la Figura 4.17, nos permite visualizar el resultado del algoritmo de fusión con la red y muestra seleccionadas.

4.12. Configuración y entrenamiento de las redes

El entrenamiento de ambos modelos se realizó con la GPU previamente mencionada. En ambos casos se usó como *backbone* la red de referencia ResNet50 del proyecto Detectron [90].

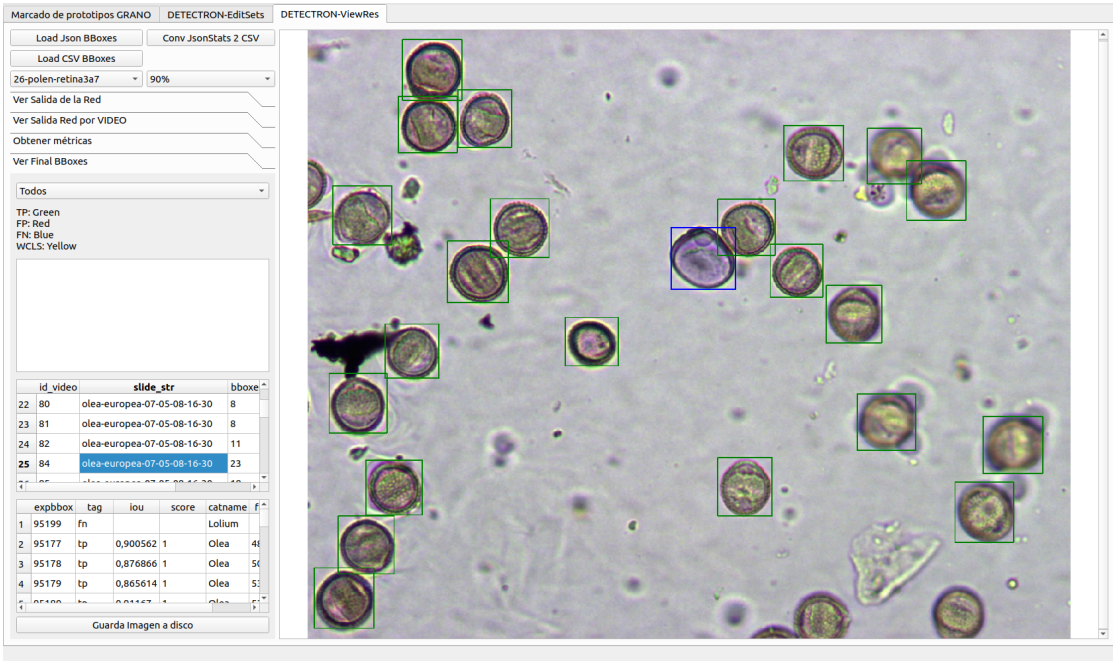


Figura 4.17: Interfaz del programa desarrollado para la visualización de resultados. Vista de resultados finales.

El entrenamiento de los modelos Faster R-CNN se realizó en la modalidad de extremo a extremo. El *solver* se se configuró para usar gradiente descendiente estocástico (SGD) con minilotes de dos imágenes por GPU y 256 ROIs por imagen, por lo que el número total de ROIs por minilote de entrenamiento es de 512. Usamos un decaimiento de pesos de 0.0001 y un momento de 0.9. Aplicamos un calentamiento lineal de 1/3 a la tasa de aprendizaje durante las primeras 500 iteraciones de entrenamiento para evitar cambios bruscos al comienzo del entrenamiento.

Los modelos se entrenaron durante un número de iteraciones ajustado a un múltiplo del número de imágenes disponibles en el conjunto de entrenamiento hasta completar un numero entero de periodos (*epochs*). La tasa de aprendizaje inicial se establece en todos los caso en 0.0025, y se reduce en un factor 10 cada 1/3 y 2/3 del número total de iteraciones. Además, usamos inversión horizontal de imágenes como algoritmo de aumento de datos.

Respecto de la configuración de la RPN del modelo Faster R-CNN, la FPN usa los niveles 2 a 5 del backbone, con marcos de anclaje cuadrados de tamaño mínimo de 32 píxeles.

Los modelos RetinaNet también se entrenan con la modalidad de extremo a extremo con SGD y los mismos parámetros de tamaño de imagen y configuración de *solver*. Como *backbone* de la red usamos la FPN de [80] que está basada en una Resnet50. Configuramos el modelo para considerar un marco de anclaje cuadrado por localización

con 3 escalas en los niveles de la pirámide 3 a 7. Los parámetros de la función de pérdida focal se ajustaron en los valores recomendados en [16] ($\alpha = 0.25$ and $\gamma = 2.0$), y la beta de la función *Smooth L1* para la regresión de BBoxes se estableció en 0.11.

4.13. Experimentos

4.13.1. Utilidad de los granos borrosos en localización

Al variar el plano de enfoque, algunos granos presentan un aspecto borroso por lo que cabe preguntarse sobre la utilidad de incluir o no estos granos en el conjunto de entrenamiento.

Al variar el eje z, cada grano debería aparecer nítidamente identificable en alguna de las imágenes de entrada a la red, por lo que debería poder localizarse sin la necesidad de incluir granos borrosos en el conjunto de entrenamiento. Pero la ornamentación algunas veces aparece en vistas en las que el grano aparece borroso. Además, al excluir los granos borrosos reducimos de manera significativa el número de ejemplos de entrenamiento. Por tanto, debemos evaluar el impacto sobre el rendimiento en localización que provoca la inclusión de ejemplos borrosos de grano en el conjunto de entrenamiento.

Hemos planificado dos tipos de experimento de localización con los dos modelos de red que vamos a utilizar. En primer lugar, los experimentos tipo A usan todos los granos definidos en cada imagen para entrenar la red, independientemente del aspecto del grano en la vista en particular. Este conjunto de entrenamiento etiquetará las redes tipo A.

En segundo lugar, los experimentos tipo B usan únicamente aquellos granos de polen marcados como nítidos en las imágenes de entrenamiento. Por tanto, usaremos un número de ejemplos de entrenamiento menor y los granos borrosos serán enseñados a la red con la categoría fondo, lo que podría provocar a priori una reducción del rendimiento en localización. Este segundo conjunto de entrenamiento dará lugar a redes que denominaremos como tipo B.

En el momento de planificar estos experimentos el número de imágenes etiquetadas en los conjuntos de entrenamiento era de 497, por lo que tras estudiar la evolución de las curvas de error durante 60000 iteraciones y comprobar la rápida convergencia de ambos modelos de red, se decidió entrenar durante 19880 iteraciones lo que se traduce en 80 *epochs*.

4.13.2. Experimentos de detección

En los experimentos de localización, no resulta especialmente relevante conocer el detalle del número de imágenes utilizado de cada tipo de polen en el conjunto de entrenamiento para ajustar los modelos. Sin embargo, a la hora de estudiar el comportamiento en detección, es importante detallar este desglose para poder ponderar la representatividad de los resultados para cada clase, en función del número de ejemplos de entrenamiento disponibles.

Tabla 4.4: Composición del conjunto de entrenamiento. $\overline{I}$ indica el número medio de imágenes por muestra que se ha utilizado para entrenar el sistema.

Tipo de grano	Muestras	Granos	Imágenes	$\overline{I}$
Avena sativa	11	42	22	2.0
Avena sterilis	12	39	24	2.0
Cedrus	29	228	32	1.1
Cupressus	15	33	30	2.0
Dactylis	42	83	90	2.1
Lolium	15	117	35	2.3
Olea	38	246	108	2.8
Phalaris	13	42	45	3.4
Plantago	28	190	97	3.4
Platanus	13	573	21	1.6
Quercus	35	445	78	2.2
Total	251	2038	582	2.3

La Tabla 4.4 muestra los detalles del conjunto de imágenes utilizado para entrenar los modelos neuronales en detección. Como puede observarse, el número medio de imágenes por grano $\overline{I}$ no es constante para cada tipo de polen, dado que hemos usado una estrategia de etiquetado dependiente de la complejidad del tipo de grano a describir. Así, hemos etiquetado más imágenes en las que hay muchos granos etiquetados en planos distantes o cuyo aspecto difiere mucho al variar el plano de enfoque. Por ejemplo, en nuestro conjunto de imágenes, las aperturas superficiales del *Plantago Lagopus* sólo son visibles en planos muy alejados de aquel en el que el borde del grano aparece más nítido, por lo que $\overline{I}$ alcanza el valor de 3.4. De esta forma, el número total de imágenes del conjunto de entrenamiento utilizado para detección es superior, alcanzando 582 imágenes.

Evidentemente, el número de imágenes utilizado en el conjunto de test en detección sigue siendo el mismo (2863) ya que modela el barrido del microscopio en el eje Z usando 21 imágenes por muestra.

4.13.3. Configuración monoimagen frente a apilamientos

En una aplicación típica de clasificación, las métricas de rendimiento se calculan utilizando una única imagen por muestra, en la que todos los objetos de interés son visibles y están etiquetados. En nuestra tarea, el uso de una única imagen por muestra, puede dar lugar a la pérdida de granos como ya hemos indicado.

En un estudio basado en el uso de una única imagen por muestra, resulta de gran importancia la selección del plano de enfoque más adecuado de forma automática, pero incluso esta tarea resulta no trivial, como comprobamos empíricamente, debido a la variabilidad en las muestras y la presencia de fondos con patrones de alta frecuencia.

Para evaluar el impacto en el rendimiento que genera nuestra implementación multifocal, estudiaremos el comportamiento de los modelos neuronales entrenados sobre un conjunto de test que únicamente contiene una imagen por muestra (SIPS). Para generar este conjunto de test, de entre todas las imágenes de cada muestra, hemos seleccionado aquella que presenta un mayor número de granos etiquetados manualmente en la base de datos. Por tanto, dado que la determinación de este plano no resulta trivial, los resultados obtenidos sobre este conjunto deben entenderse como aquellos que que maximizarían las posibilidades de acierto en localización y detección.

Capítulo 5

Resultados

5.1. Resultados en localización

En esta sección recogemos los resultados obtenidos en los distintos experimentos de localización realizados. Además, realizamos el análisis de estos resultados tanto de forma numérica como gráficamente, mostrando algunos detalles que no reflejan los parámetros numéricos.

5.1.1. Tiempos de entrenamiento y localización

Los tiempos de entrenamiento de los cuatro modelos para un mismo número de iteraciones superaron ligeramente la hora, siendo algo más rápido el ajuste de los modelos RetinaNet. El detalle de los tiempos de entrenamiento para los cuatro modelos se refleja el la Tabla 5.1. En cuanto a los requerimientos de memoria durante el entrenamiento, los modelos tipo Faster alcanzaron valores de pico de 2491 MB y los RetinaNet 2027 MB.

Con respecto a la evolución de las funciones de error durante el entrenamiento de los modelos tipo Faster R-CNN, la Figura 5.1 muestra la evolución de las funciones de pérdida de clase y ubicación (BBox) durante el entrenamiento. Como puede apreciarse, ambos marcadores alcanzan sus valores asintóticos con bastante rapidez siendo más evidente en el caso de las propuestas de BBox. Respecto de las oscilaciones debidas al uso de SGD podemos apreciar que son más evidentes para la subred de clasificación que para subred de regresión de BBoxes.

En cuanto a la evolución del entrenamiento de los modelo RetinaNet, la Figura 5.2 nos permite comparar ambos modelos. En primer lugar cabe destacar que estos modelos no proporcionan una función de error diferenciada para clasificación y regresión de BBoxes.

Si que observamos que en el modelo que usa únicamente granos nítidos se produce una oscilación en el entorno del primer *epoch*, que después converge sin nuevos picos significativos hasta alcanzar la estabilidad pasadas las 5000 iteraciones.

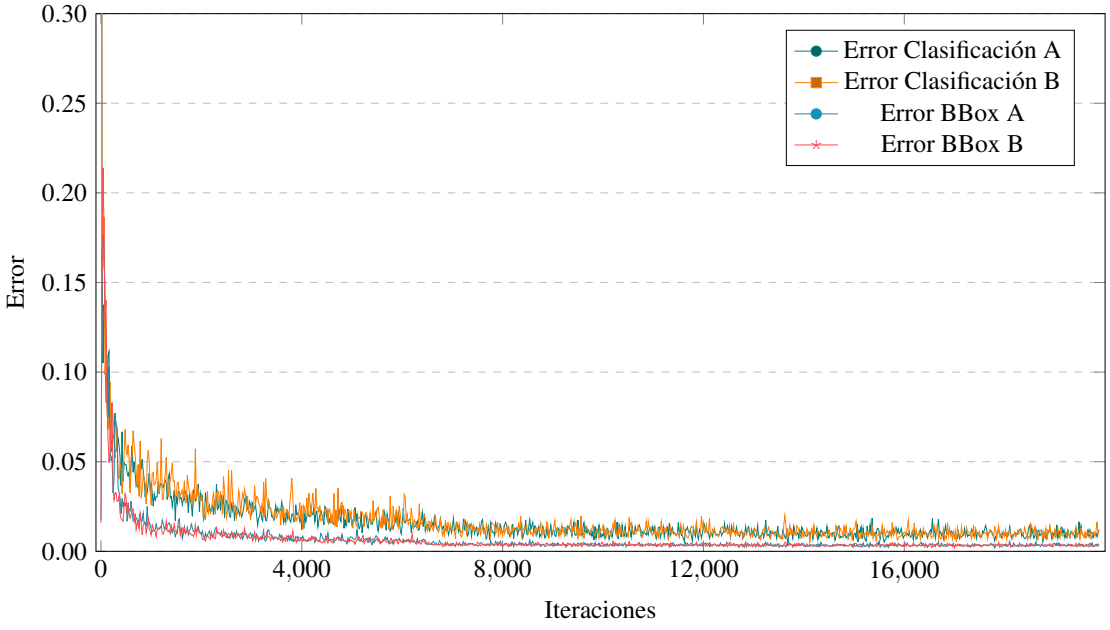


Figura 5.1: Evolución del valor de las funciones de coste para los modelos Faster R-CNN durante su entrenamiento.

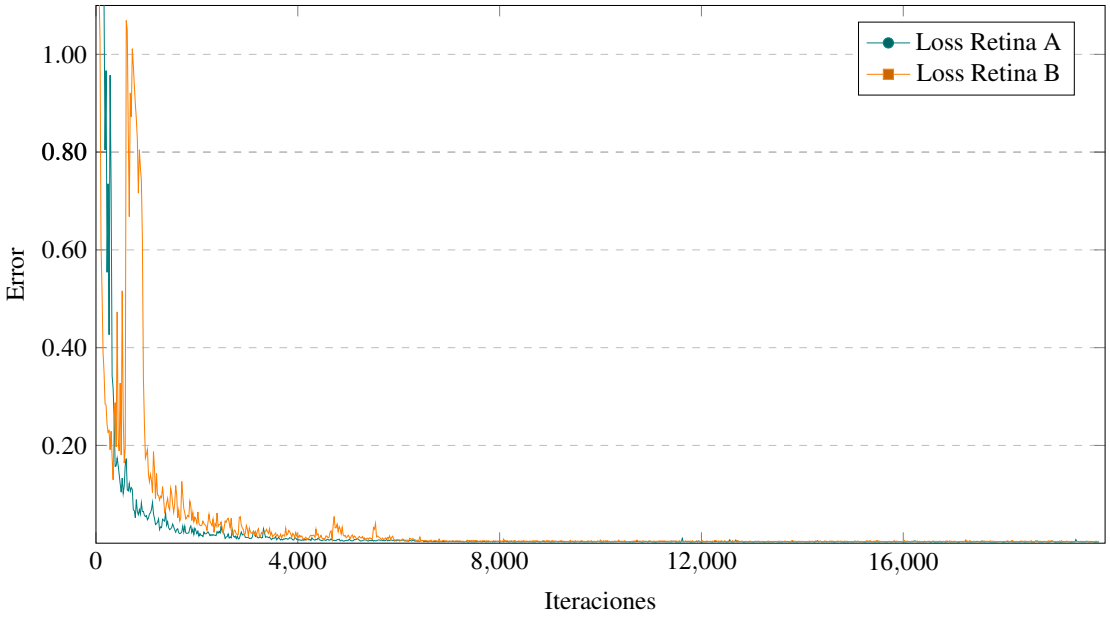


Figura 5.2: Evolución del valor de las funciones de coste para los modelos RetinaNet durante su entrenamiento.

Una vez disponemos de las redes entrenadas, realizamos el reconocimiento de las las imágenes incluidas en el conjunto de test de localización con un umbral de *score* muy bajo (0.1) y NMS configurado para descartar propuestas solapadas con un IoU superior

al 0.5. De esta forma, podemos estudiar la evolución de los parámetros de rendimiento del algoritmo de fusión en función del *score*.

El tiempo total de inferencia para las 2863 imágenes incluidas en el conjunto de test fue de 154.7 s en el modelo Faster R-CNN Tipo A y 155.2 s para el modelo B de la misma arquitectura. Por tanto, la velocidad de procesamiento por imagen fue inferior a los 55 ms en nuestra GPU. De esta forma, el tiempo procesamiento de red de las 21 imágenes que consideramos para cada muestra requeriría menos de 1.14 s con este hardware.

Realizando el mismo análisis con las dos variantes del modelo RetinaNet, observamos una reducción de los tiempos de inferencia sobre el conjunto de test que se sitúa en una media de 46 ms por imagen. Por tanto, necesitaríamos un mínimo de 966 ms por muestra en el apartado de inferencia. Por tanto, como esperábamos, la velocidad del modelo monoetapa es superior al modelo de dos etapas. La Tabla 5.1 muestra de forma resumida los tiempos de entrenamiento e inferencia para las redes consideradas.

Tabla 5.1: Resumen de los tiempos de entrenamiento e inferencia para las redes estudiadas.

Red	Tiempo de entrenamiento (horas)	Tiempo de inferencia (s)	Tiempo medio de inferencia (ms)
Faster R-CNN A	1:10:35	154.7	54
Faster R-CNN B	1:10:36	155.2	54
RetinaNet A	1:08:23	131.1	46
RetinaNet B	1:08:22	131.5	46

Como podemos observar, no hay una variación significativa entre los tiempos de entrenamiento necesarios para las redes tipo A o B. Ambos tipos de red han usado un número diferente de prototipos de grano, siendo menor en las tipo B. Este resultado era esperable dado que ambos modelos generan automáticamente un número de BBoxes de entrenamiento fijo a partir de las BBox de certeza indicadas como entrada.

5.1.2. Rendimiento por fusión básica de propuestas

En este trabajo de localización, el número total de granos de polen a localizar es de 1235, contenidos en 135 muestras preparadas en el laboratorio (vídeos). Los resultados iniciales de esta técnica los publicamos en [91]. Una de las tareas a realizar es conocer la evolución de los parámetros de rendimiento de nuestro algoritmo, respecto de la puntuación de las propuestas de grano (score) que genera cada modelo de red. Para estudiar este comportamiento, hemos realizado un barrido respecto de este parámetro en el rango

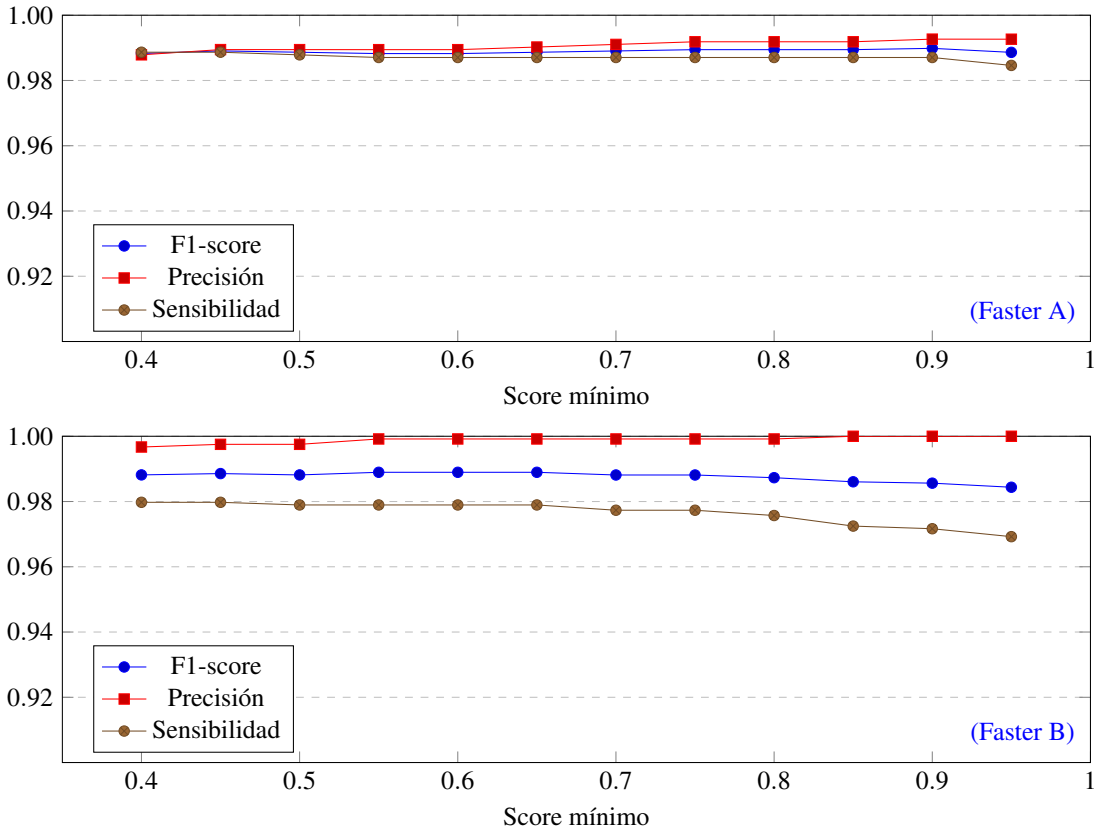


Figura 5.3: Variación de las métricas de rendimiento con el *score* mínimo de las propuestas para las cuatro redes entrenadas en localización.

0.4 a 0.95 para. La Figura 5.3 muestra la variación de los parámetros de precisión, sensibilidad y F1-score para las dos redes tipo Faster R-CNN.

La primera conclusión que podemos extraer de ambas gráficas es la escasa variación de los parámetros de rendimiento respecto del requisito de puntuación mínima requerida a las BBoxes. Podemos apreciar que el índice F1 se mantiene prácticamente plano en todo el rango estudiado en el caso de la red entrenada con prototipos borrosos, alcanzando su valor más elevado para un *score* de 0.90. Los valores de precisión y sensibilidad asociados a este punto son 0.9927 y 0.9870. El comportamiento de la red entrenada con prototipos nítidos es semejante en cuanto a la dependencia en el *score*, manteniendo siempre elevados valores del índice F1, en este caso el valor máximo se localiza para un valor de 0.65, con una precisión asociada de 0.9992 y sensibilidad de 0.9789. Resulta interesante observar que no crece de forma importante el número de falsos postivos al reducir el umbral de *score* hasta un valor tan bajo como 0.4, hecho que es atribuíble a la capacidad de la red de modelar el concepto grano a pesar de la gran variación en el aspecto de éstos.

La Figura 5.4 muestra los resultados obtenidos al repetir el análisis para las dos redes RetinaNet. En este caso, podemos observar como en el caso de la red Retina tipo A

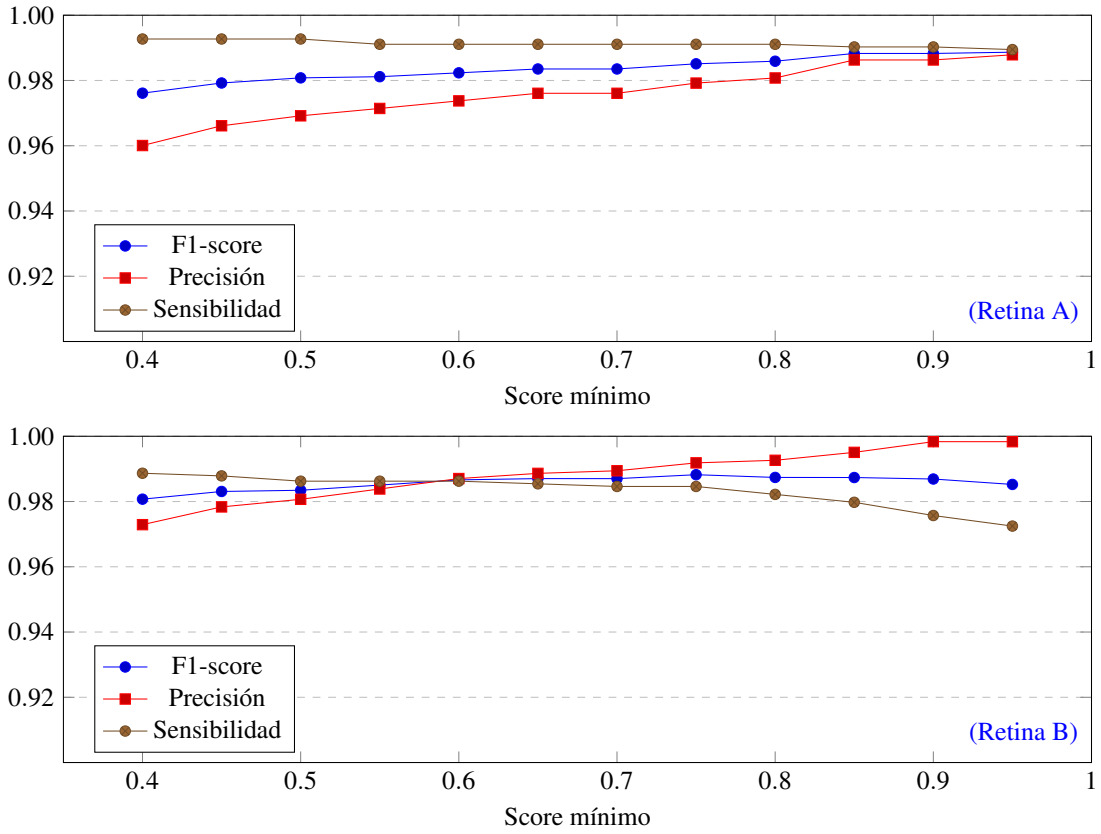


Figura 5.4: Variación de las métricas de rendimiento con el *score* mínimo de las propuestas en las redes tipo Faster R-CNN entrenadas en localización.

el índice F1 alcanza su valor máximo (0.988673) para el valor de filtrado más elevado (score=0.95), siendo la precisión en este punto 0.9879 y la sensibilidad 0.9895. Además, observando las curvas de precisión y sensibilidad, podemos observar un comportamiento distinto a los modelos Faster, donde el comportamiento era más independiente del valor de *score* seleccionado. En este caso, se observa que el modelo RetinaNet genera un mayor número de falsos positivos a bajos niveles de filtrado que daña el parámetro de precisión, que crece de forma monotónica al aumentar el índice de certeza exigido.

Al observar el comportamiento del modelo Retina entrenado únicamente con granos nítidos (Retina B), observamos nuevamente que a bajos valores de filtrado, el modelo genera más falsos positivos que la configuración equivalente tipo Faster. Sin embargo tiene valores de sensibilidad más elevados utilizando ambos el mismo conjunto de entrenamiento. En este caso, el índice F1 más elevado se logra para un *score* de 0.75, para el que se obtiene una precisión de 0.9918 y una sensibilidad de 0.9846.

La Tabla 5.2 muestra los parámetros de rendimiento de las cuatro redes sobre el conjunto de test, para la configuración de nivel de filtrado (score) que proporciona el mayor valor de índice F1 en cada una de las redes. Se indican en cada caso, el número de granos

Tabla 5.2: Resultados de localización para las cuatro redes analizadas con la configuración de *score* que logra un índice F1 más elevado.

Parámetro	Faster A		Faster B		RetinaNet A		RetinaNet B	
	Valor	%	Valor	%	Valor	%	Valor	%
Score	0.90		0.65		0.95		0.75	
TP	1219	98.70	1209	97.89	1222	98.95	1216	98.46
FN	16	1.30	26	2.11	13	1.05	19	1.54
FP	9	0.73	1	0.08	15	1.21	10	0.81
TP_E	31	2.51	38	3.07	33	2.67	40	3.24
Precisión	-	99.27	-	99.92	-	98.79	-	99.18
Sensibilidad	-	98.70	-	97.89	-	98.94	-	98.46
F1	-	98.98	-	98.89	-	98.87	-	98.82

de cada métrica y los porcentajes normalizados respecto del número de granos totales a localizar en el conjunto de test.

El primer resultado que se puede destacar de la Tabla 5.2 es que el algoritmo de fusión logra que con todas las redes se localice la casi totalidad de los granos presentes en las muestras. El modelo Faster tipo B es el que proporciona un valor más bajo en la tasa de localización (97.89 %). Este hecho, nos sugiere que el problema se puede abordar de forma robusta con cualquiera de las las arquitecturas de red que hemos considerado.

El número de falsos positivos en ambos modelos tipo Faster es muy bajo, siendo especialmente bajo en el modelo tipo B, donde únicamente se han estimado un grano en zonas en las que no estaba recogido en la base de datos. El modelo retina que incluye prototipos borrosos es el único que genera una tasa de falsos positivos superior al 1 %, por lo que podríamos pensar que este modelo de red tiende a generar respuestas menos fiables partiendo del mismo conjunto de entrenamiento.

Finalmente, como se mencionó en la Sección 4.10, cuando la porción de grano visible en las imágenes es suficiente, la red genera propuestas de grano asociadas a granos cortados en los bordes (TP_E), que realmente existen en la muestra y están etiquetados como tal en la base de datos. Como ya se especificó, estas detecciones se indican por separado en los resultados, pero no contribuyen en las tasas de acierto.

Respecto de la utilidad de los granos difusos en la tarea de localización, observamos que en ambos modelos se obtiene una mayor tasa de sensibilidad al utilizar estos granos frente a las redes que no los usan. Es decir se incrementa las posibilidades de localización de granos, pero por contra, cae la tasa de precisión de ambas redes por lo que deberíamos

aceptar un mayor número de falsos positivos en funcionamiento. En una aplicación como ésta en la que queremos maximizar tanto la precisión como la sensibilidad, el parámetro que debe guiar la elección del modelo es el índice F1, y éste alcanza su valor más elevado para la red Faster A que incluye los prototipos borrosos. El segundo índice F1 más elevado lo logra también el segundo modelo Faster R-CNN, por lo que éste modelo parece funcionar mejor que RetinaNet en localización.

No obtante, en una implementación física, el tiempo de inferencia también debe ser tenido en cuenta en el balance, y localización las redes RetinaNet son más rápidas.

5.1.3. Rendimiento de las técnicas clásicas sobre nuestra base de datos

La Tabla 5.3 muestra una comparativa de las diferentes propuestas de localización de granos de polen analizadas en la Sección 2.4, con los resultados de localización obtenidos en este trabajo. Como puede observarse, el resultado más destacable es el elevado índice de precisión obtenido en nuestro estudio (99.27 %), respecto de los reportados por los trabajos mencionados. La tasa de sensibilidad (98.70 %) también supera la reportada en el resto de los estudios referenciados. Sin embargo, como ya se mencionó en la Sección 2.4, resulta muy difícil realizar una comparación significativa y justa de los distintos estudios, debido a las grandes diferencias existentes entre los conjuntos de datos utilizados por los autores referenciados, en particular respecto del número de granos en el conjunto de test.

Tabla 5.3: Comparación de los resultados obtenidos en los últimos trabajos publicados que realizan localización de granos de polen con los resultados obtenidos en este trabajo. Un *n/d* en el campo indica la inexistencia del dato en la publicación.

	Tipos	Platinas	Granos Entren.	Granos Test	Sensibilidad	Precisión
Ranzato et al. [32]	8	n/d	3318	368	93.90 %	8.60 %
Landsmeer et al. [25]	n/d	9	n/d	65	86.00 %	61.00 %
Nguyen et al. [27]	9	1	n/d	768	93.80 %	89.5 %
Díaz-López et al. [24]	12	12	n/d	3999	81.92 %	18.50 %
CHT	11	20	2037	1235	79.82 %	83.40 %
Nuestro algoritmo	11	20	2037	1235	98.70 %	99.27 %

Para tratar de obtener una idea del rendimiento sobre nuestra base de datos con las técnicas clásicas reportadas, hemos procesado nuestro conjunto de datos con una implementación de uno de los algoritmos utilizados en los trabajos analizados. Tanto el trabajo de Díaz-López et al. [24] como el de Nguyen et al. [27] utilizaron la transformación circular de Hough (CHT) para localizar granos de polen en alguna etapa su algoritmo de

localización. Así, hemos implementado un algoritmo desarrollado con OpenCV [48], que trata de reproducir la etapa de localización del algoritmo de Nguyen et al.

Nuestra implementación del algoritmo de localización usa una única imagen por muestra, que en primer lugar se convierte a escala de grises. Utilizaremos por tanto el conjunto de test que previamente denominamos SIPS, que de entre todas las imágenes de cada muestra, selecciona aquella que presenta un mayor número de granos etiquetados manualmente. Cada imagen se difumina usando un filtro de la mediana y se buscan círculos usando CHT. Por último, aplicamos un algoritmo NMS sobre el listado de propuestas de círculo. Con estas premisas, estudiamos el tamaño del filtro más adecuado y los parámetros de la función *HoughCircles* para maximizar el número de detecciones correctas, usando un IoU > 0.5 con las marcas de certeza almacenadas como criterio de acierto.

Tras varias pruebas, escogimos un tamaño de filtro de 17 píxeles y los siguientes parámetros para la CHT: resolución del acumulador idéntica a la de la imagen de entrada, distancia mínima entre centros de círculo de 25 píxeles, gradiente de 50, umbral del acumulador de centros de 30 y un rango de radio de los objetos entre 20 y 200 pixeles.

Bajo la configuración anterior, obtuvimos 985 TP, 196 FP y 249 FN. Por tanto, este algoritmo de localización alcanzó una tasa de sensibilidad del 78.8 % y una precisión del 83.4 %. Estos resultados aparecen reflejados en la Tabla 5.3 etiquetados como CHT.

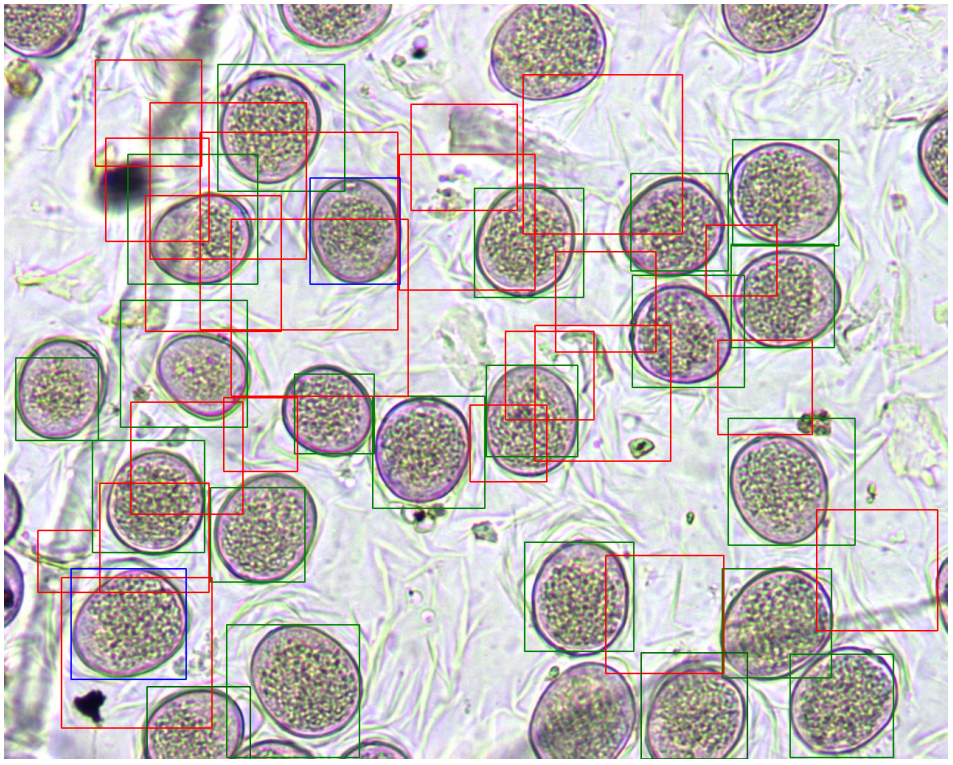


Figura 5.5: Localización de granos de tipo *Lolium* con el algoritmo basado en la transformación circular de Hough. La existencia de un fondo no uniforme provoca un alto número de falsos positivos.

La Figura 5.5 muestra un resultado obtenido al aplicar este algoritmo a una imagen con múltiples objetos sobre un fondo complejo. Las principales fuentes de error observadas usando nuestra implementación de localizador basado en CHT se deben a:

- Fondos con ruido muy elevado debido a la superficie del adhesivo.
- Granos con bajo contraste respecto de la tintura del fondo.
- Granos que no tienen una apariencia circular.

Afortunadamente, todos estos retos inherentes al espacio de imagen con el que trabajamos, parecen ser bien gestionados por nuestro sistema basado en aprendizaje profundo a la vista de los resultados obtenidos.

5.1.4. Precisión de las propuestas localización

La Tabla 5.4 muestra la exactitud en la localización de las predicciones que genera cada modelo de red en términos de IoU respecto de las marcas de referencia almacenadas para cada grano. Los datos se desglosan a nivel de tipo de polen y el valor medio global, especificando junto con la desviación estándar de cada medida.

Tabla 5.4: Exactitud en la localización de los granos de polen para las clases en estudio. La exactitud se expresa en términos del IoU entre las predicciones y las marcas de certeza almacenadas para los granos de la muestra. La desviación estándar para cada tipo de grano se indica entre paréntesis y el número de predicciones con #.

Polen	Faster A		Faster B		RetinaNet A		RetinaNet B	
	IoU ($\boldsymbol{\sigma}$)	#	IoU ($\boldsymbol{\sigma}$)	#	IoU ($\boldsymbol{\sigma}$)	#	IoU ($\boldsymbol{\sigma}$)	#
Cupressus	0.90 (0.03)	29	0.90 (0.04)	29	0.93 (0.02)	29	0.90 (0.03)	29
Avena sativa	0.86 (0.09)	17	0.88 (0.06)	16	0.88 (0.06)	18	0.89 (0.06)	17
Avena sterilis	0.86 (0.07)	43	0.88 (0.05)	43	0.89 (0.06)	43	0.90 (0.06)	43
Cedrus	0.91 (0.03)	162	0.91 (0.02)	162	0.93 (0.02)	162	0.93 (0.02)	162
Dactylis	0.89 (0.06)	59	0.91 (0.05)	59	0.91 (0.06)	60	0.92 (0.05)	60
Lolium	0.89 (0.04)	82	0.88 (0.03)	80	0.91 (0.04)	82	0.91 (0.03)	75
Olea	0.89 (0.04)	194	0.89 (0.04)	192	0.90 (0.04)	194	0.90 (0.04)	191
Phalaris	0.94 (0.03)	36	0.93 (0.02)	36	0.93 (0.04)	36	0.94 (0.01)	36
Plantago	0.89 (0.04)	116	0.89 (0.04)	116	0.89 (0.04)	116	0.89 (0.03)	116
Platanus	0.85 (0.04)	180	0.86 (0.03)	180	0.84 (0.04)	180	0.86 (0.03)	180
Quercus	0.90 (0.03)	303	0.91 (0.02)	303	0.90 (0.03)	306	0.91 (0.03)	301
Valor medio	0.89 (0.04)	1221	0.89 (0.04)	1216	0.90 (0.05)	1226	0.90 (0.04)	1210

Como referencia, un IoU de 1.0 representa una precisión total a la hora de estimar la ubicación y el tamaño de cada grano en la muestra, como se indicó en la Section 4.8.

Un valor medio de 0.89 como el obtenido en en el caso de los clasificadores tipo Faster es indicativo de una alta precisión como mostramos gráficamente en la Figura 4.12. Cabe resaltar, como muestra la Tabla 5.4, que las implementaciones RetinaNet parecen proporcionar un mejor ajuste de la posición y el tamaño de los granos de polen, por lo que podrían ser una implementación más adecuada para estimar el marco que mejor se ajusta a los bordes de un grano.

Además, los valores de desviación estándar que presentan los valores de solapamiento son muy bajos, lo que nos indica que la dispersión de los valores de IoU es muy baja y por tanto la mayor parte de las propuestas finales del sistema cubren casi perfectamente la extensión de los granos marcados en la base de datos.

Aunque alguna de las publicaciones recogidas en la Sección 2.4 abordan la localización y la estimación del tamaño de grano, resulta imposible realizar una comparación con nuestros resultados, debido a la ausencia de cualquier métrica reproducible en esos trabajos.

5.1.4.1. Resultados gráficos

Aunque los resultados numéricos nos permiten obtener una idea general del funcionamiento del sistema, algunos detalles del rendimiento en localización quedan ocultos en los datos numéricos globales. Por tanto, mostraremos algunos resultados gráficos de especial interés en esta sección. En el Apéndice A.1 pueden encontrarse las imágenes a campo completo de las que se han obtenido los cortes que se detallan en esta sección.

En las imágenes siguientes, una localización correcta se indica mediante un recuadro verde, los falsos positivos se muestran en color rojo, y finalmente, los falsos negativos aparecen en color azul. La posición de las propuestas finales se almacena en la base de datos asociándola con el plano de enfoque en el que ha sido localizada, por lo que la marca correcta puede parecer ligeramente desplazada respecto del grano visualizado si el plano que se muestra es distinto de aquel en el que la propuesta fue localizada.

Como indican los resultados numéricos mostrados previamente en la Sección 5.1.2, nuestro sistema es capa de localizar eficientemente los granos presentes en el conjunto de test definido. Más aún, los datos de la Sección 5.1.4 muestran una alta independencia de las propuestas finales respecto del tipo de polen a localizar, la complejidad del fondo o la retroiluminación utilizada en las distintas platinas. Sin embargo, hay determinadas configuraciones no controlables en las muestras reales que provocan que el sistema ignore ciertos granos, o genere propuestas de grano que alcanzan el *score* mínimo en áreas donde no se aprecia ningún grano al variar el plano de enfoque.

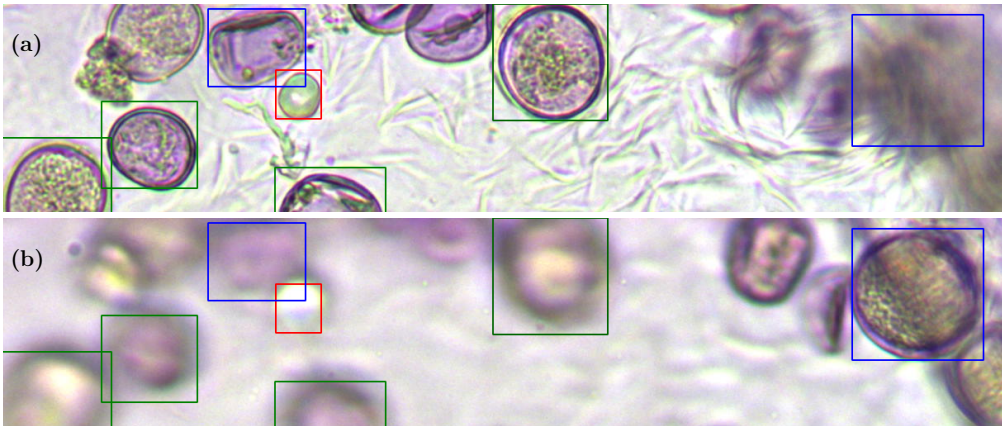


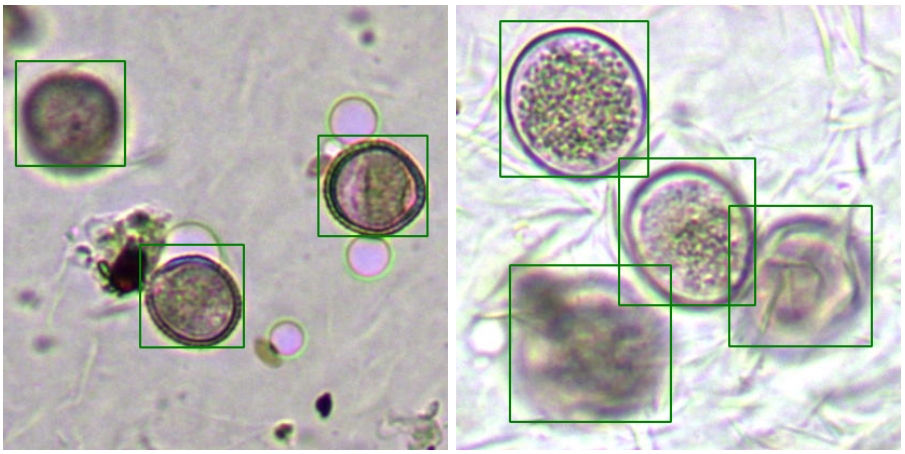
Figura 5.6: Dos cortes del mismo área de una muestra en planos de enfoque distintos que muestran dos tipos de error: un falso positivo (rojo) en el que una burbuja se etiqueta como grano de polen (a) y un falso negativo (marca azul en la esquina superior derecha) que no llega a presentar una vista nítida en la excursión de enfoque (b).

Así, la Figura 5.6, obtenida usando la red Faster tipo B, muestra dos planos diferentes de la misma muestra, en la que el sistema ha identificado erróneamente una burbuja como un grano de polen (en rojo) y dos falsos negativos (en azul). Uno de los falsos negativos está asociado con un grano localizado en el borde superior del campo visible, junto a la burbuja, y el otro, en la esquina superior derecha, se muestra muy borroso en la mayor parte del recorrido del eje z, y únicamente se aprecia con cierta nitidez en una de las imágenes consideradas en el conjunto de test (Figura 5.6b), alejada del plano de enfoque central.

Afortunadamente, la localización de burbujas del sustrato como granos no es un comportamiento sistemático de nuestro sistema. En general, el sistema diferencia correctamente el concepto de grano y burbuja. A modo de ejemplo gráfico, la Figura 5.7a muestra una sección de una de las muestras de tipo *Olea* con varias burbujas, detritos y polvo que han sido ignorados, mientras que los tres granos visibles de *Olea* han sido localizados correctamente. Además, podemos apreciar en este mismo corte el aparente error de localización del grano que aparece difuso en la esquina superior izquierda, que obviamente, se debe a que el plano que hemos decidido mostrar en la imagen no coincide con el plano en el que la red localizó el grano.

Adicionalmente, la Figura 5.7b muestra un corte de una muestra de *Lolium* en la que los cuatro granos visibles han sido localizados correctamente, a pesar de que cada uno de ellos sólo se visualiza nítidamente en planos de enfoque muy distantes entre si y dos de ellos aparecen agrupados presentando deformación del borde.

La localización de granos adyacentes o parcialmente solapados es otro de los retos que habíamos identificado al analizar el conjunto de muestras del que disponíamos. La Figura 5.8 presenta los resultados de localización de tres de estas situaciones con el sistema



(a) (b)

Figura 5.7: Dos ejemplos de localización de granos en muestras de *Olea* (a) y *Lolium* (b), en las que los problemas identificados en la Figura 5.6 han sido gestionados correctamente.

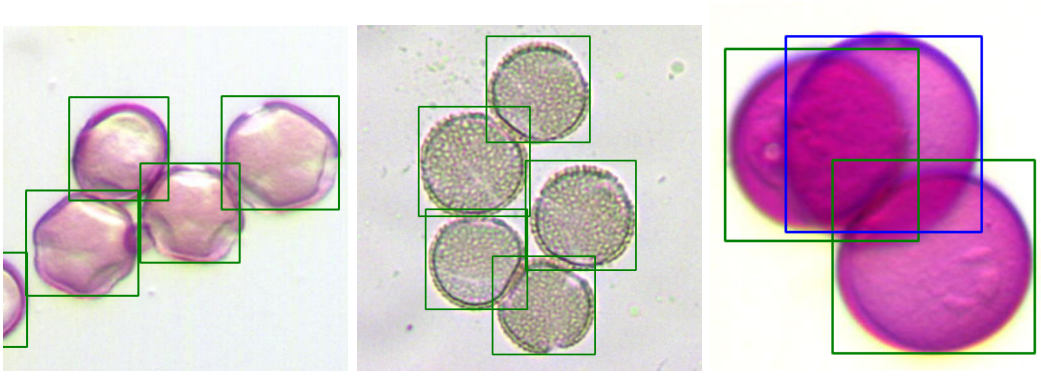
basado en la red Faster tipo B. Así, la Figura 5.8a muestra un conjunto de granos de tipo *Quercus* donde, además de las evidentes variaciones de apariencia visual entre ellos, los cuatro granos adyacentes han sido localizados correctamente.

La Figura 5.8b también muestra una operación correcta con cinco granos adyacentes muy agrupados y con bordes deformados. Esta muestra procede de una platina de *Olea* distinta a la mostrada en el corte de la Figura 5.7a y como se puede comparar presentan fondos y niveles de tintura muy distintos.

Por último, la Figura 5.8c muestra el resultado de procesar una muestra de *Dactylis* en la que el elevado nivel de solapamiento entre los dos granos superiores hacen que nuestro sistema sea incapaz de separar e identificar ambos granos dando lugar a la pérdida del grano central.

5.2. Resultados en detección

En esta sección recogemos los resultados obtenidos en los distintos experimentos de detección realizados, discutiendo al mismo tiempo sus implicaciones. Hemos entrenado una red tipo Faster R-CNN y otra tipo RetinaNet sobre el conjunto de entrenamiento descrito en la Sección 4.13.2. Los tiempos medios de procesamiento por imagen de ambas redes son en este caso 59 ms para el modelo Faster R-CNN, y 58 ms para el modelo RetinaNet. La configuración de tamaños máximos de imagen de entrada en ambos casos es la misma, por lo que parece que al aumentar el número de clases la diferencia de velocidad de inferencia se reduce. En ambos casos, se ha utilizado la misma GPU y número de



(a) (b) (c)

Figura 5.8: (a) Un ejemplo de localización correcta de todos los granos de tipo *Quercus* a pesar de las variaciones morfológicas visibles, (b) un segundo ejemplo de una muestra de *Olea* con diferentes niveles de tintura y deformación de bordes, y (c) un alto nivel de solapamiento en la muestra de *Dactylis* no permite la localización del grano central.

imágenes (2863) para calcular estos valores medios. En todos los casos consideraremos 1235 granos a localizar.

En el proceso de inferencia, hemos configurado que sólo se acepten las propuestas de la red con alta certeza. Por tanto, hemos establecido un umbral de *score* de 0.75 y mantenido el umbral de solapamiento para el algoritmo NMS en 0.5. Esta decisión se ha tomado tras comprobar con los resultados de localización que la mayor parte de los granos se localizan con *scores* elevados. El algoritmo de fusión del apilamiento usará un umbral de 0.7 según el argumento desarrollado en la Sección 4.9.3.

5.2.1. Utilizando una imagen por muestra

Al utilizar una imagen por muestra cabe esperar una tasa de FN superior. La Tabla 5.5 muestra los parámetros de rendimiento global obtenidos al reconocer el conjunto de test SIPM con los dos modelos de red entrenados. La entrada *Propuestas* indica el número de candidatos finales resultantes de la aplicación del algoritmo NMS. Y los parámetros de precisión, sensibilidad y *weighted-averaged F1–score* vienen dados por las Ecuaciones 4.5, 4.4 y 4.7 respectivamente.

La primera observación que cabe resaltar es que ninguno de los modelos genera propuestas para cubrir la totalidad de los granos presentes en el conjunto de test. Este hecho podría venir motivado por la elección de los parámetros del algoritmo NMS, pero en este caso, al analizar los resultados sobre las imágenes, observamos que los granos perdidos se corresponden con situaciones en las que el grano no es visible en la imagen utilizada.

Cabe destacar que en el modelo Faster R-CNN la precisión en localización es del 100 % al no generar falsos positivos, pero al considerar los fallos en la asignación de la clase como

Tabla 5.5: Resumen de los resultados obtenidos con ambos modelos de red utilizando una imagen por muestra. Los porcentajes se calculan respecto del número de granos en el conjunto de test.

	Faster R-CNN		RetinaNet	
Parámetro	Granos	(%)	Granos	(%)
Propuestas	1198	-	1210	-
TP	1155	93.52	1159	93.85
WC	13	1.05	19	1.54
FN	67	5.43	57	4.61
FP	0	0.00	2	0.16
TP_E	30	-	30	-
Precisión	-	98.88	-	98.22
Sensibilidad	-	94.57	-	95.38
$F1_W$	-	95.99	-	95.90

FP, este parámetro es inferior. Además, llama la atención el escaso número de falsos positivos generados en ambos modelos. A la hora de buscar una justificación para estas métricas de rendimiento tan positivas, debemos tener en cuenta la generosa elección del plano de enfoque más benevolente a la hora de generar el conjunto de test SIPM.

5.2.2. Utilizando nuestra propuesta multienfoque

La Tabla 5.6 muestra los parámetros de rendimiento global obtenidos al reconocer el conjunto de test con un apilamiento de 21 imágenes por muestra, con los dos modelos de red utilizados en la sección anterior. De esta forma, podremos cuantificar la variación en los parámetros de rendimiento que se obtiene al utilizar esta propuesta a igualdad de red.

En este caso, ambos modelos generan un número de propuestas de grano superior a las realmente presentes en el conjunto de test. Como primer resultado, cabe destacar que ambos modelos ubican y clasifican correctamente más del 96 % de los granos presentes en el conjunto de test.

En segundo lugar, observamos que el número de falsos positivos sigue siendo muy bajo, manteniéndose el mismo comportamiento positivo del modelo Faster R-CNN frente al modelo RetinaNet. Este resultado resulta más interesante al recordar que en este caso, al usar 21 imágenes por muestra, la red podría haber generado más candidatos asociados a grano por las imperfecciones del espacio de imágenes ya mencionadas.

Tabla 5.6: Resumen de los resultados obtenidos con ambos modelos de red utilizando apilamiento de imágenes. Los porcentajes se calculan respecto del número de granos en el conjunto de test.

	Faster R-CNN		RetinaNet	
Parámeter	Grains	(%)	Grains	(%)
Propuestas	1241	-	1255	-
TP	1188	96.27	1189	96.28
WC	17	1.30	23	1.86
FN	30	2.43	23	1.86
FP	0	0.00	7	0.57
TP_E	36	-	36	-
Precisión	-	98.59	-	97.54
Sensibilidad	-	97.57	-	98.14
$F1_W$	-	97.31	-	97.09

Si comparamos los indicadores de rendimiento monoimangen frente a esta segunda propuesta multienfoque, podemos observar que el porcentaje de granos correctamente detectado en ambos modelos crece un 2.75 % en el modelo Faster R-CNN y un 2.43 % en el modelo RetinaNet. Como cabría esperar, también se reduce la tasa de falsos negativos en ambos modelos de red, sin generar un incremento significativo del número de granos clasificados erróneamente, 4 granos más en ambos casos. Por tanto, la mayor parte de los nuevos granos encontrados contribuyen a mejorar tanto la tasa de precisión como la de sensibilidad. El bajo número de falsos positivos que genera la red en ambos casos es un indicativo de la capacidad de estas dos técnicas de aprendizaje profundo de modelar correctamente este nuevo espacio de imagen y del elevado nivel de *score* escogido para aceptar las propuestas.

A la hora de escoger uno de los dos modelos de red bajo las mismas condiciones de NMS, en base a las métricas obtenidas, encontramos que el *índice F1 pesado* más elevado lo logra el modelo Faster R-CNN, aunque por un escaso margen. Ambos modelos presentan altos indicadores de rendimiento y prácticamente los mismos tiempos de inferencia en nuestro entorno de prueba, por lo que en estas condiciones la elección de uno de ellos no resulta clara.

5.2.2.1. Rendimiento a nivel de tipo de polen

Entrando a analizar la distribución de errores de detección a nivel de tipo de polen, podemos observar la distribución de éstos con ayuda de las matrices de confusión de

ambos modelos sobre el conjunto de test. La Tabla 5.7 muestra la matriz de confusión para el modelo Faster R-CNN con los distintos tipos de polen etiquetados como T_i para simplificar la tabla. También se muestran las tasas de precisión, sensibilidad e índice F1 para cada tipo de polen.

Tabla 5.7: Matriz de confusión para detección de tipo de polen usando el modelo Faster R-CNN sobre el conjunto de test de 21 imágenes apiladas.

		Clase predicha												
Clase real		T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9	T_{10}	T_{11}	FN	Sens.
Cupressus	T_1	29											0	1.00
A. sativa	T_2		15	1									4	0.75
A. sterilis	T_3		8	33									3	0.75
Cedrus	T_4				161					1			0	0.99
Dactylis	T_5					57	1						2	0.95
Lolium	T_6						73					1	9	0.88
Olea	T_7							191					4	0.98
Phalaris	T_8						1		36				0	0.97
Plantago	T_9									115			3	0.97
Platanus	T_{10}										180	1	0	0.99
Quercus	T_{11}									2	1	298	5	0.97
FP		0	0	0	0	0	0	0	0	0	0	0		
Precisión		1.00	0.65	0.97	1.00	0.98	0.97	1.00	1.00	0.97	0.99	0.99		
F1-score		1.00	0.70	0.85	1.00	0.97	0.92	0.99	0.99	0.97	0.99	0.98		

La matriz de confusión nos muestra que la mayor parte de las propuestas generadas por la red se sitúan en la diagonal principal de la matriz, lo que indica que, en términos generales, el modelo presenta un rendimiento muy bueno en la localización e identificación de los distintos granos de polen estudiados. Sin embargo, se aprecia un error de identificación evidente cuando el modelo identifica algunos granos de *Avena sterilis* como *Avena sativa*. La apariencia visual de ambos tipos de grano es muy similar en las muestras que tenemos, y el modelo parece favorecer el tipo *sativa* frente a la variante *sterilis*. En cualquier caso, el escaso número de granos presente en las muestras de *Avena sativa* reduce la significación de este resultado.

El funcionamiento del modelo RetinaNet sobre el conjunto de test puede observarse en la Tabla 5.8. Como cabría esperar a la vista de las métricas macro, en este caso la mayor parte de las propuestas de grano también se sitúan en la diagonal principal. También observamos un comportamiento idéntico respecto de las dos clases de avena que contiene nuestro estudio. Y como aspecto diferencial, únicamente apreciamos la

presencia de celdas activas aisladas en distintas posiciones entre ambos modelos, aunque de mayor magnitud en el modelo RetinaNet.

Tabla 5.8: Matriz de confusión para detección de tipo de polen usando el modelo RetinaNet sobre el conjunto de test de 21 imágenes apiladas.

		Clase predicha												
Clase real		T_1	T_2	T_3	T_4	T_5	T_6	T_7	T_8	T_9	T_{10}	T_{11}	FN	Sens.
Cupressus	T_1	29											0	1.00
A. sativa	T_2		14	1		1							4	0.70
A. sterilis	T_3		10	32									2	0.73
Cedrus	T_4				161							1	0	0.99
Dactylis	T_5			2		56							2	0.93
Lolium	T_6						74						9	0.89
Olea	T_7							193					2	0.99
Phalaris	T_8								37				0	1.00
Plantago	T_9					1				115			2	0.97
Platanus	T_{10}										180	1	0	0.99
Quercus	T_{11}									2	4	298	2	0.97
FP		0	1	2	0	0	0	3	0	0	0	1		
Precisión		1.00	0.58	0.91	1.00	0.95	1.00	0.99	1.00	0.98	0.98	0.99		
F1-score		1.00	0.64	0.81	1.00	0.94	0.94	0.99	1.00	0.98	0.99	0.98		

5.2.2.2. Exactitud de las propuestas de localización

En las secciones previas hemos presentado los resultados asumiendo que el solapamiento de las propuestas (P) con las marcas de referencia (G) de los granos almacenados en la base de datos, supera el criterio establecido para el concepto de acierto, IoU(P,G) > 0.5. Cabe preguntarse en este punto por el ajuste de las propuestas generadas al contorno de cada tipo de grano de polen. De esta forma, podríamos identificar tipos de polen mal modelados por la red y añadir, en su caso, un mayor número de imágenes de estos tipos.

Como ya hicimos para mostrar la exactitud en la tarea de localización, recogemos en la Tabla 5.9 los valores medios de solapamiento en términos de IoU entre las propuestas generadas correctamente por la red (TP) y sus correspondientes marcas de referencia. La desviación estándar de cada medida se muestra entre paréntesis.

El valor medio global de ambas redes, nos indica que la precisión de los BBoxes generados en ambos casos es muy elevada, con una desviación típica media global de 0.04 en ambos casos. Por tanto, este experimento nos permite comprobar nuevamente que la calidad

Tabla 5.9: Exactitud de la localización de los granos de polen para las clases estudiadas con los modelos multiclase entrenados.

	Faster R-CNN		RetinaNet	
Tipo de Polen	IoU (σ)	Predicciones	IoU (σ)	Predicciones
Cupressus	0.91 (0.03)	29	0.92 (0.03)	29
Avena sativa	0.91 (0.03)	15	0.91 (0.04)	14
Avena sterilis	0.91 (0.04)	33	0.93 (0.03)	32
Cedrus	0.93 (0.02)	161	0.93 (0.02)	161
Dactylis	0.91 (0.06)	57	0.92 (0.06)	56
Lolium	0.89 (0.04)	73	0.90 (0.03)	74
Olea	0.90 (0.04)	191	0.90 (0.03)	193
Phalaris	0.94 (0.02)	36	0.94 (0.02)	37
Plantago	0.92 (0.04)	115	0.90 (0.04)	115
Platanus	0.87 (0.03)	180	0.87 (0.03)	180
Quercus	0.91 (0.03)	298	0.91 (0.03)	298
Valor medio	0.91 (0.04)	1188	0.91 (0.04)	1189

de las propuestas RetinaNet es semejante a la lograda por el modelo Faster R-CNN, a pesar de ser este último un modelo de dos etapas. Además, los valores medios de IoU coinciden en muchos de los tipos de polen, poniendo de manifiesto que el tipo de polen peor modelado en términos de área es el *Platanus hispanica* con un IoU de 0.87. Y por contra, el mejor modelado en ambos casos sería el tipo *Phalaris minor* con un IoU de 0.94.

El resultado más interesante aparece al comparar la exactitud de las BBoxes generadas por los modelos entrenados en localización (Tabla 5.4) y los entrenados para realizar detección (Tabla 5.9). Al comparar ambas tablas, observamos que la exactitud de los BBoxes en detección supera en términos generales los valores equivalentes en localización, 0.91 frente a 0.90 en el mejor de los casos. La justificación a este resultado puede estar ligada al bloque de regresión de marcos en el caso del modelo Faster R-CNN, que ajusta una subred para cada una de las clases aprendidas. Cabe esperar, que sea más eficiente regresionar BBoxes con las características particulares de cada tipo de grano, que tratar por igual todos los tipos de grano, sean alargados o circulares. Sin embargo, este efecto no tiene la misma explicación en el caso del modelo RetinaNet, donde la subred de regresión se ajusta independientemente de la clase del marco de entrenamiento. Por tanto, es posible que esté ligado al mejor modelado que realizan ambas redes con el nuevo conjunto de entrenamiento.

5.2.2.3. Comparación con otros trabajos de detección

Una comparación significativa con otros trabajos en esta vertiente resulta compleja, ya que no todos los estudios abordan tanto las fases de localización como de clasificación de los objetos encontrados.

Al considerar ambas tareas en un estudio integrado, el número de trabajos se reduce y también los parámetros de rendimiento. Además, la comparación de métricas resulta más complicada, no solo por la diferente composición de los conjuntos de test, sino también por la ausencia de detalle de las métricas de detección como se puso de manifiesto en la Sección 2.6. El trabajo de Ranzato et al. de 2007 reporta una tasa de clasificación correcta del 77 %. Mientras que Nguyen et al. [27] lleva a cabo un estudio de detección sobre 768 granos aislados, reportando únicamente un valor de sensibilidad del 96.4 % en identificación. A pesar de que la comparación no es muy significativa, nuestro trabajo de detección utilizando nuestra técnica multienfoque mejora ambos estudios con tasas de precisión (98.59 %) y sensibilidad (97.57 %) sobre un conjunto de 1235 granos de 11 tipos distintos.

En este capítulo, hemos estudiado el comportamiento de dos modelos de detección basados en técnicas de aprendizaje profundo para realizar localización y detección de distintos tipos de polen. Los resultados parecen indicarnos que el uso de apilamiento de imágenes de campo completo permite mejorar la el rendimiento en localización. El uso de granos borrosos parece mejorar ligeramente la localización, pero el coste temporal de su etiquetado para realizar detección no parece compensar el posible incremento en detección. En el siguiente capítulo expondremos las principales conclusiones extraídas en este trabajo.

Capítulo 6

Conclusiones

La localización y clasificación de granos de polen en muestras con sustrato adhesivo tintado es una tarea compleja. En este trabajo, hemos abordado la aplicación de dos de las técnicas de detección de objetos más recientes en inteligencia artificial (Faster R-CNN y Retina Net), obteniendo resultados prometedores y utilizando equipamiento de bajo coste, tanto en el bloque de adquisición como en el apartado computacional.

Hemos confirmado que es posible llevar a cabo un ajuste de extremo a extremo de los dos modelos neuronales analizados, utilizando transferencia de aprendizaje, con un conjunto de muestras palinológicas relativamente pequeño.

Por otro lado, hemos desarrollado un sistema que mejora los resultados de localización expuestos en la bibliografía, tanto en términos de sensibilidad como de precisión, utilizando granos parcialmente borrosos en el conjunto de entrenamiento.

En detección, hemos conseguido que nuestra implementación logre tasas de precisión y sensibilidad muy elevados, aunque la comparación con métricas equivalentes de otros trabajos no resulte posible. Además, la precisión de los marcos generados es incluso más elevada que la lograda en localización, por lo que debemos concluir que el modelado de los tipos de grano estudiados parece efectivo.

Una contribución importante de este trabajo es la construcción de una base de datos de granos de polen que permite la realización de estudios de detección. Utiliza apilamientos de imágenes, emulando el comportamiento de un palinólogo. Que tengamos conocimiento, es la única que permite llevar a cabo esta tarea actualmente. Hemos puesto a disposición de la comunidad científica esta base de datos en nuestra página web [92], habiendo atendido hasta este momento 9 peticiones desde varios países.

Los reducidos tiempos de detección, entorno a 59 ms por imagen, nos hacen esperar una operación cercana al tiempo real, si se realiza una integración del sistema con un microscopio robotizado. En este caso, el tiempo requerido para ajustar cada nueva posición, puede ser utilizado para procesar con la GPU la imagen capturada previamente.

La alta precisión lograda por nuestro sistema sobre el conjunto de test, y la velocidad de procesamiento alcanzada, nos lleva a concluir que esta técnica puede ser adecuada para desarrollar un sistema automático de estimación de la concentración de polen a partir de muestras estándar.

Finalmente, como continuación natural del trabajo desarrollado en esta tesis, hay varias líneas de investigación futura que podemos abordar. La primera debería ser la extensión del número de muestras y tipos de polen a localizar, para obtener una representación más significativa de las especies que se pueden encontrar en nuestra zona. Como requisito previo para abordar esta tarea, deberíamos construir un microscopio robotizado o estudiar la automatización de un microscopio estándar por medio de motores paso a paso. Una vez automatizada la captura de muestras, deberíamos estudiar la dependencia del rendimiento en localización en función del tamaño de paso configurado en el eje vertical. Tras integrar la captura automática de muestras, podríamos estudiar su uso en un laboratorio de palinología, con la adecuada supervisión de expertos para controlar y mejorar su funcionamiento. La inclusión de muestras obtenidas en condiciones meteorológicas adversas, serviría para estudiar la robustez de estas técnicas ante muestras con gran cantidad de polvo y detritos.

El coste temporal de adición de nuevos tipos de polen puede verse reducido con el uso de nuestros modelos ya entrenados. Hemos comprobado que podemos utilizar una red ya entrenada para ubicar nuevos prototipos de clases desconocidas para, a posteriori, corregir las propuestas estimadas por la red, y establecer manualmente la clase de cada prototipo. Este cambio en la forma de trabajo aceleraría significativamente la adición de soporte para nuevos tipos de polen.

En el largo plazo, se podría pensar extender el número de tipos de polen para gestionar los más habituales en otras zonas. Para este trabajo, deberíamos contar con el apoyo de otros grupos de investigación interesados en esta misma línea trabajo.

Apéndice A

Resultados gráficos

A.1. Resultados en localización

En este apéndice se muestran algunos resultados de interés en localización, con las imágenes a campo completo.

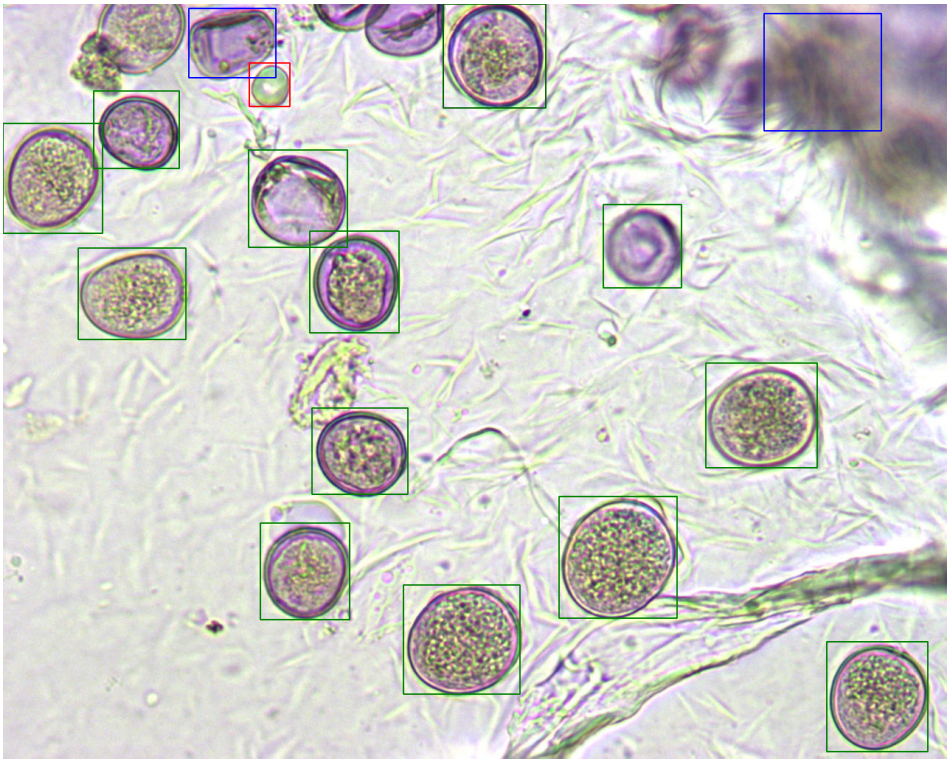


Figura A.1: Muestra de *Lolium rigidum* con una burbuja detectada como grano y falsos negativos a consecuencia de su baja visibilidad en las imágenes del apilamiento.

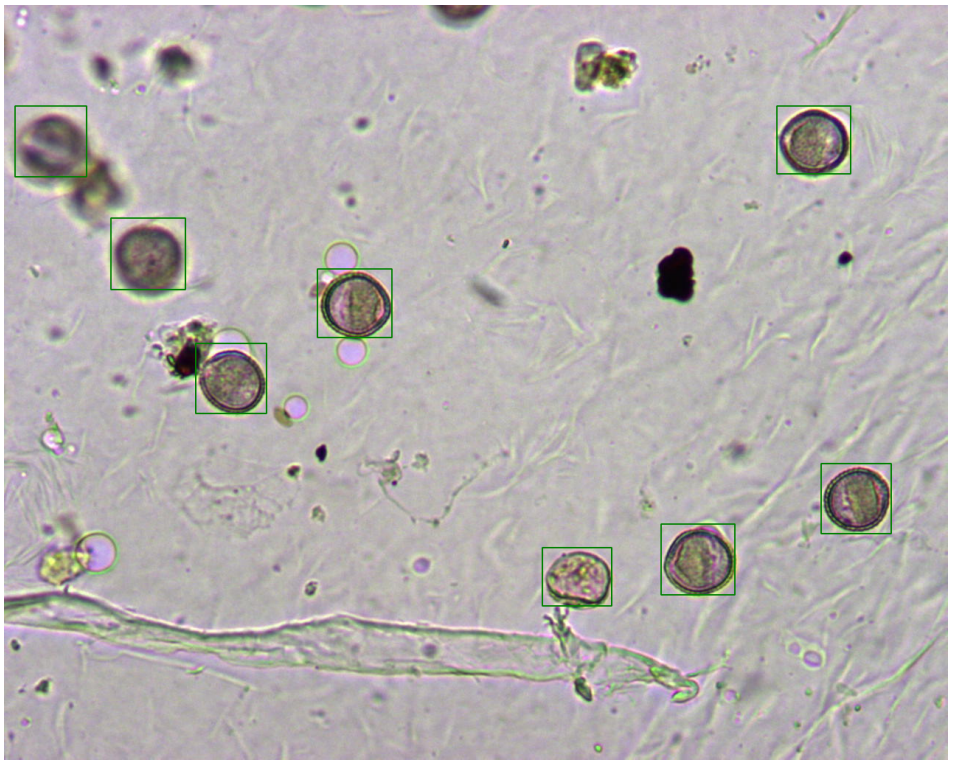


Figura A.2: Muestra de *Olea europea* en la que la presencia de burbujas no da lugar a falsos positivos. También puede apreciarse la existencia de granos correctamente localizados visibles en planos de enfoque distantes.

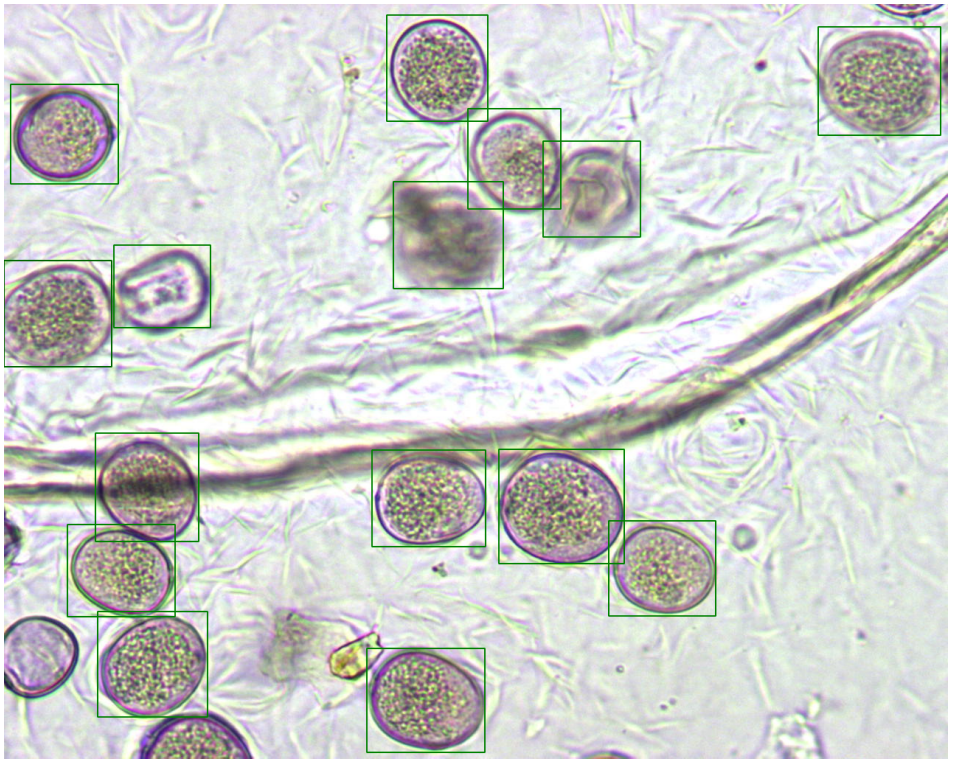


Figura A.3: Muestra de *Lolium rigidum* en la que puede apreciarse la presencia de granos correctamente localizados visibles en planos de enfoque distantes.

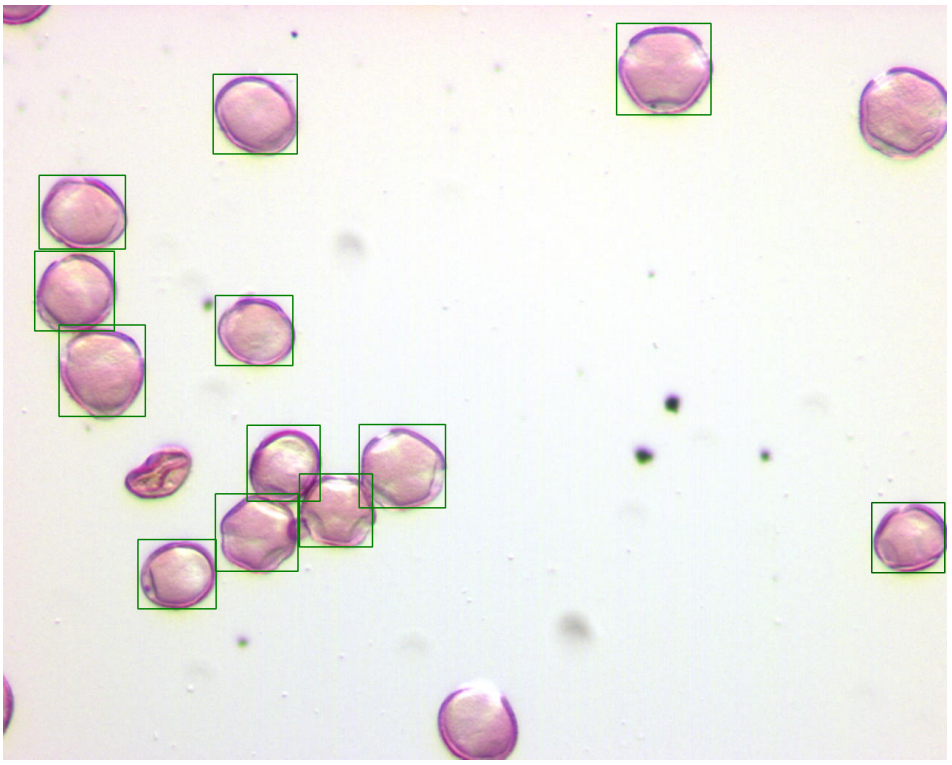


Figura A.4: Muestra de *Quercus rotundifolia* en la que puede apreciarse la presencia de granos multiformes correctamente localizados.

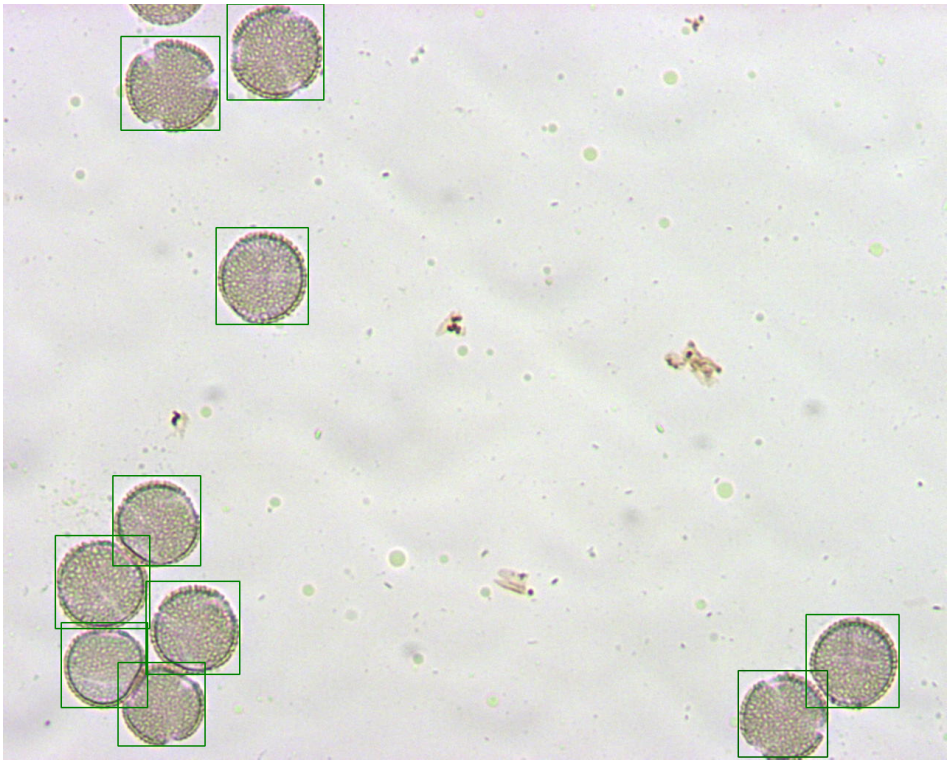


Figura A.5: Muestra de *Olea europea* en la que puede apreciarse la presencia de granos adyacentes con el borde deformado y correctamente localizados.

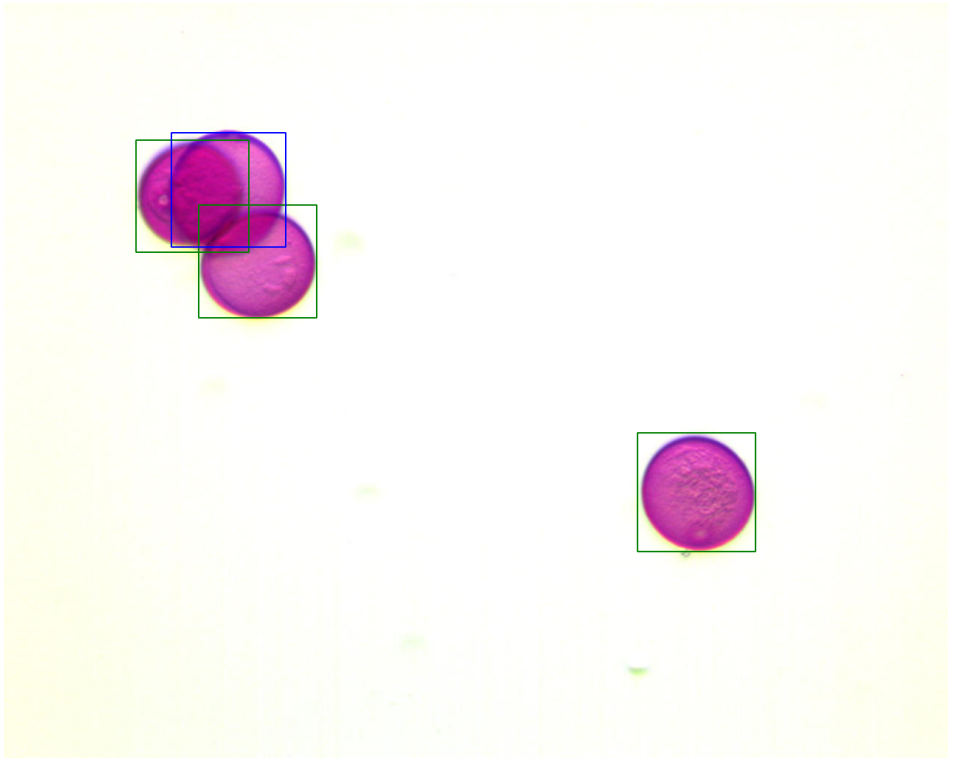


Figura A.6: Muestra de *Dactilys glomerata* en la que el alto solapamiento existente entre los granos impide la localización correcta del grano central.

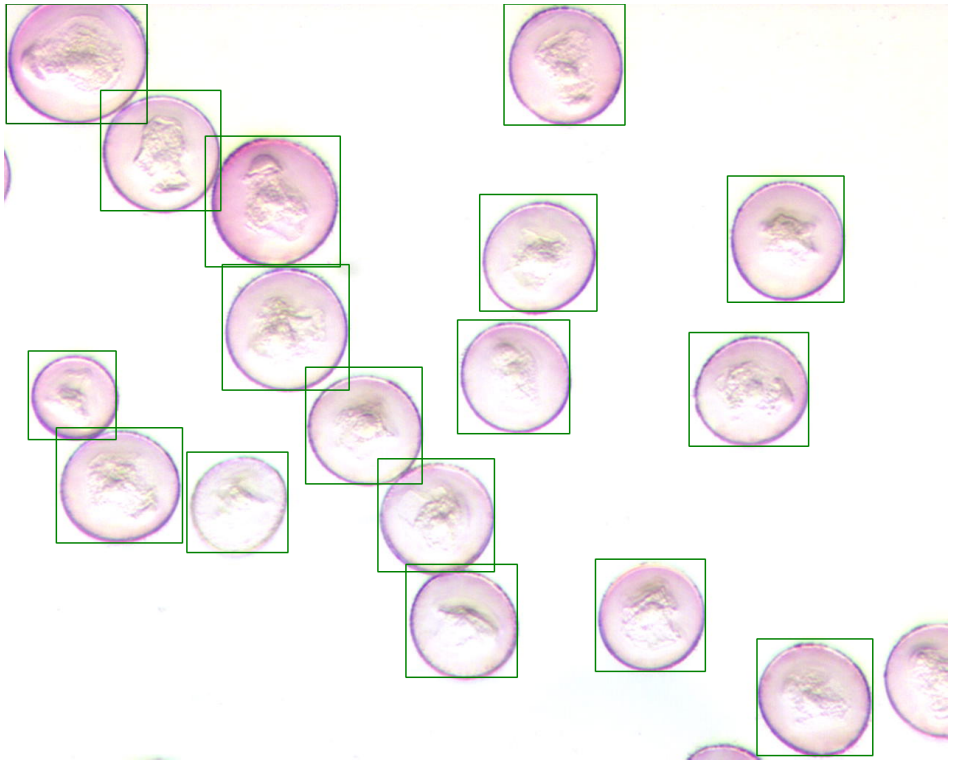


Figura A.7: Muestra de *Calocedrus decurrens* con un gran número de granos visibles y distintos tamaños.

A.2. Resultados en detección

En este apéndice se muestran algunos resultados de interés en detección.

Figura A.8: Faster R-CNN: La muestra de *Lolium Rigidum* previamente analizada en localización, sigue siendo compleja en detección. Varios falsos negativos (azul).

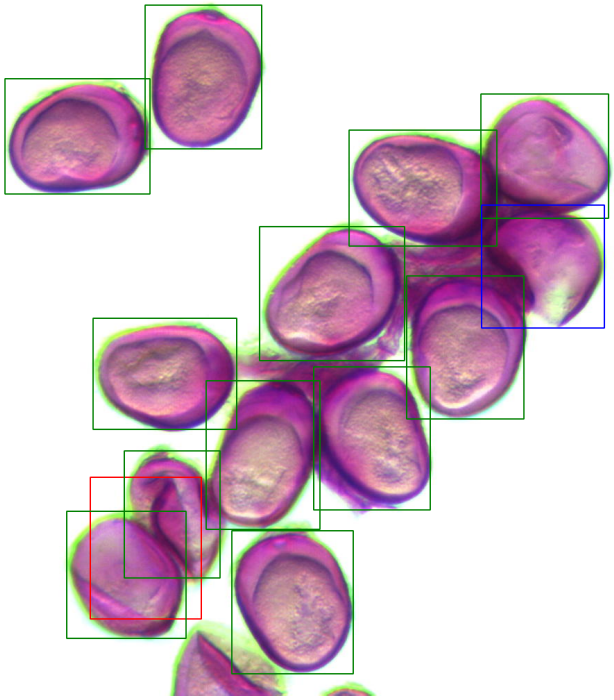


Figura A.9: RetinaNet: Falso positivo que el algoritmo NMS no elimina (rojo) y falso negativo (azul) en una muestra de *Avena Sterilis*.

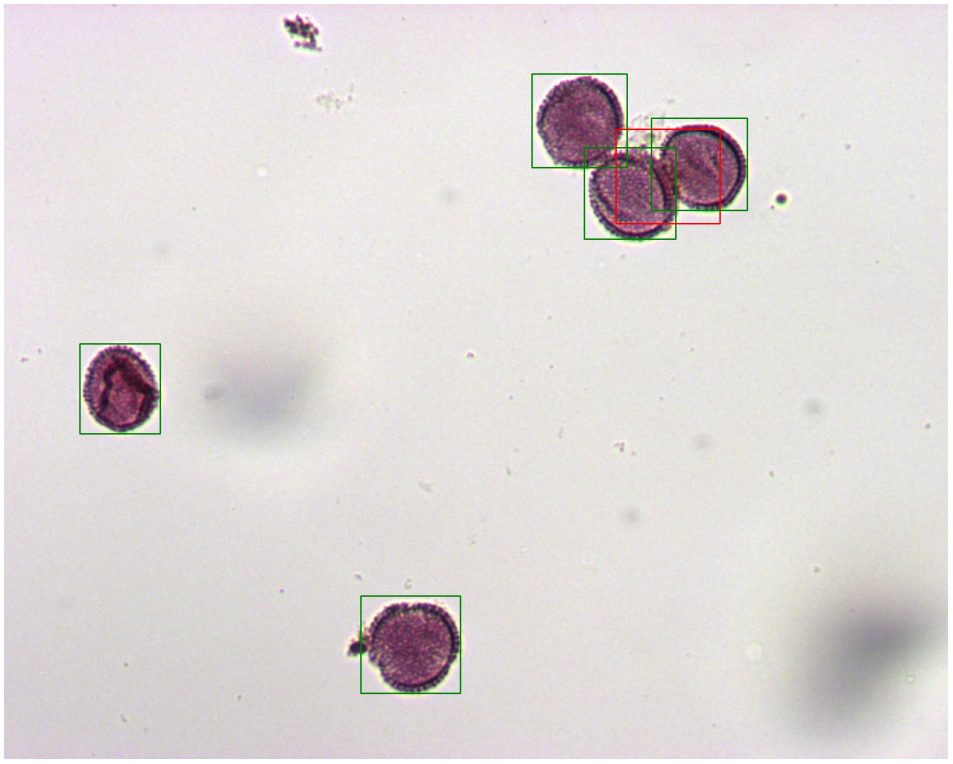


Figura A.10: RetinaNet: Un falso positivo con dos granos adyacentes.

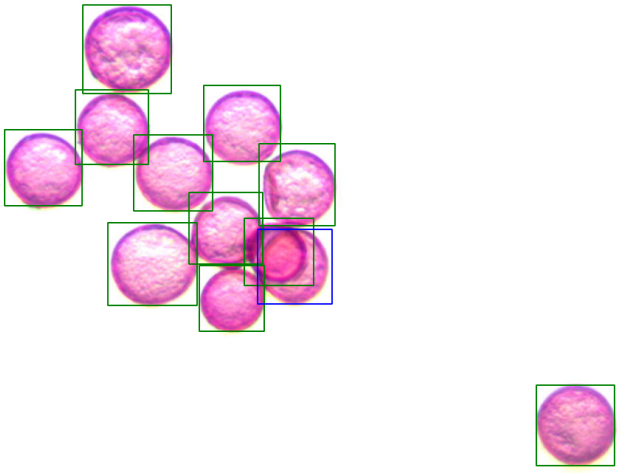


Figura A.11: Faster R-CNN: Grano solapado de *Plantago Lagopus* que no es capaz de separar en ningún plano del *z-stack*.

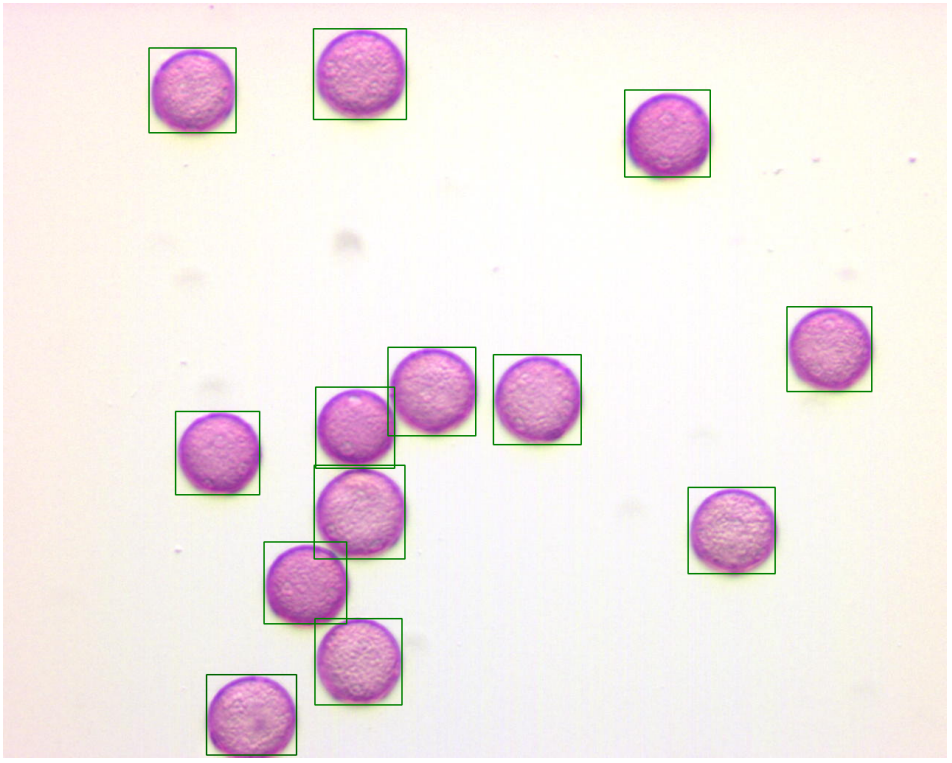


Figura A.12: Faster R-CNN: Localiza correctamente todos los granos de *Plantago Lagopus* presentes en la vista.

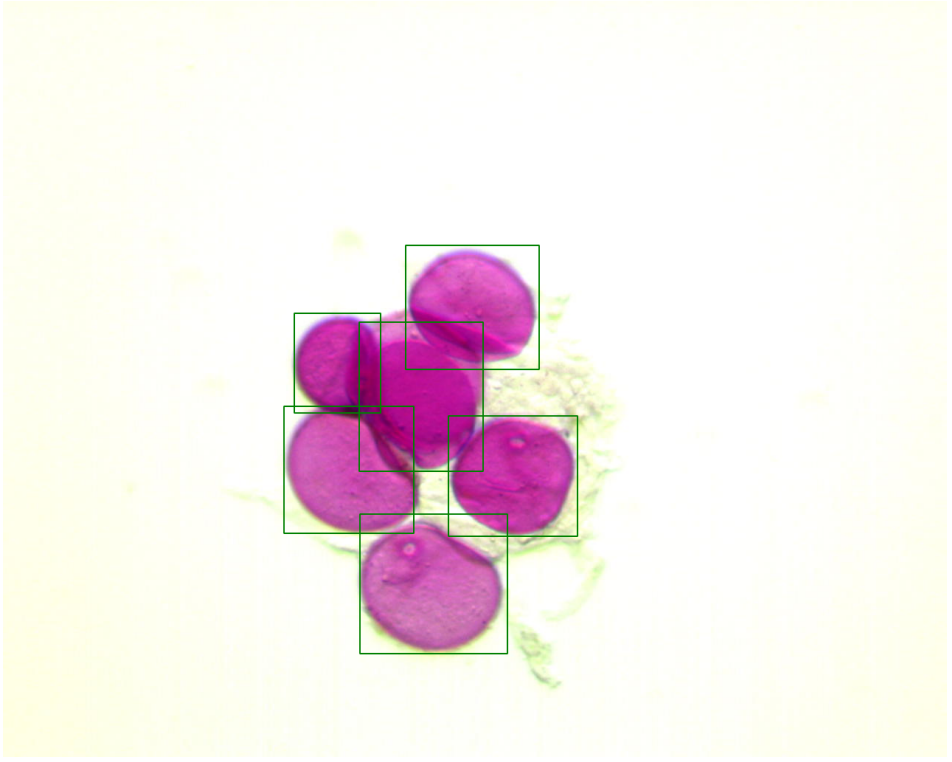


Figura A.13: Faster R-CNN: Localiza correctamente todos los granos de *Dactylis Glomerata* presentes en la muestra a pesar del alto solapamiento.

Bibliografía

[1] P. C. Tzedakis, V. Andrieu, J.-L. De Beaulieu, S. Crowhurst, M. Follieri, H. Hooghiemstra, D. Magri, M Reille, L. Sadori, N.J. Shackleton, y T.A. Wijmstra. Comparison of terrestrial and marine records of changing climate of the last 500,000 years. *Earth and Planetary Science Letters*, 150(1):171–176, 1997. ISSN 0012-821X. doi: 10.1016/S0012-821X(97)00078-2.

[2] D.C. Mildenhall, P.E.J. Wiltshire, y V.M. Bryant. Forensic palynology: Why do it and how it works. *Forensic Science International*, 163(3):163 – 172, 2006. ISSN 0379-0738. doi: 10.1016/j.forsciint.2006.07.012.

[3] Carmen Galán, Luis Vázquez, Herminia García-Mozo, y Eugenio Domínguez. Forecasting olive (olea europaea) crop yield based on pollen emission. *Field Crops Research*, 86(1):43 – 51, 2004. ISSN 0378-4290. doi: 10.1016/S0378-4290(03)00170-9.

[4] Rafael Redondo, Gloria Bueno, François Chung, Rodrigo Nava, J. Víctor Marcos, Gabriel Cristóbal, Tomás Rodríguez, Amelia Gonzalez-Porto, Cristina Pardo, Oscar Déniz, y B. Escalante-Ramírez. Pollen segmentation and feature evaluation for automatic classification in bright-field microscopy. *Computers and Electronics in Agriculture*, 110:56 – 69, 2015. ISSN 0168-1699. doi: 10.1016/j.compag.2014.09.020.

[5] Werner Von Der Ohe, Livia Persano Oddo, Maria Lucia Piana, Monique Morlot, y Peter Martin. Harmonized methods of melissopalynology. *Apidologie*, 35, 2004. doi: 10.1051/apido:2004050.

[6] G. D' Amato, L. Cecchi, S. Bonini, C. Nunes, I. Annesi-Maesano, H. Behrendt, G. Liccardi, T. Popov, y P. Van Cauwenberge. Allergenic pollen and pollen allergy in Europe. *Allergy*, 62(9):976–990, 2007. doi: 10.1111/j.1398-9995.2007.01393.x.

[7] Yoshimasa Kawazoe, Kiminori Shimamoto, Ryohei Yamaguchi, Yukako Shintani-Domoto, Hiroshi Uozaki, Masashi Fukayama, y Kazuhiko Ohe. Faster R-CNN-Based glomerular detection in multistained human whole slide images. *Journal of Imaging*, 4(7), 2018. ISSN 2313-433X. doi: 10.3390/jimaging4070091.

[8] Robert T. Krivacic, Andras Ladanyi, Douglas N. Curry, H. B. Hsieh, Peter Kuhn, Danielle E. Bergsrud, Jane F. Kepros, Todd Barbera, Michael Y. Ho, Lan Bo Chen, Richard A. Lerner, y Richard H. Bruce. A rare-cell detector for cancer. *Proceedings of the National Academy of Sciences*, 101(29):10501–10504, 2004. ISSN 0027-8424. doi: 10.1073/pnas.0404036101.

[9] Rufeng Li, Yibei Wang, Hong Xu, Baowei Fei, y Binjie Qin. Micro-droplet detection method for measuring the concentration of alkaline phosphatase-labeled nanoparticles in fluorescence microscopy. *Sensors*, 17(11), 2017. ISSN 1424-8220. doi: 10.3390/s17112685.

[10] Jason Yosinski, Jeff Clune, Yoshua Bengio, y Hod Lipson. How transferable are features in deep neural networks? En *Proceedings of the 27th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'14, páginas 3320–3328, Cambridge, MA, USA, 2014. MIT Press.

[11] Ross Girshick, Jeff Donahue, Trevor Darrell, y Jitendra Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. En *Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition*, páginas 580–587, Washington, DC, USA, 2014. IEEE Computer Society. ISBN 978-1-4799-5118-5. doi: 10.1109/CVPR.2014.81.

[12] Ross Girshick. Fast R-CNN. En *2015 International Conference on Computer Vision (ICCV)*, páginas 1440–1448, Santiago, Chile, 2015. doi: 10.1109/ICCV.2015.169.

[13] Shaoqing Ren, Kaiming He, Ross Girshick, y Jian Sun. Faster R-CNN: Towards real-time object detection with region proposal networks. En C. Cortes, N.D. Lawrence, D.D. Lee, M. Sugiyama, y R. Garnett, editores, *Advances in Neural Information Processing Systems*, volumen 28, 2015.

[14] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott Reed, Cheng-Yang Fu, y Alexander C. Berg. Ssd: Single shot multibox detector. En Bastian Leibe, Jiri Matas, Nicu Sebe, y Max Welling, editores, *Computer Vision – ECCV 2016*, páginas 21–37. Springer International Publishing, 2016. doi: 10.1007/978-3-319-46448-0_2.

[15] Kaiming He, Georgia Gkioxari, Piotr Dollar, y Ross Girshick. Mask R-CNN. En *2017 IEEE International Conference on Computer Vision (ICCV)*, páginas 2961–2969, Venecia, Italia, 2017. doi: 10.1109/ICCV.2017.322.

[16] T. Lin, P. Goyal, R. Girshick, K. He, y P. Dollár. Focal loss for dense object detection. En *2017 IEEE International Conference on Computer Vision (ICCV)*, páginas 2999–3007, Octubre 2017. doi: 10.1109/ICCV.2017.324.

[17] J. Redmon, S. Divvala, R. Girshick, y A. Farhadi. You only look once: Unified, real-time object detection. En *2016 IEEE Conference on Computer Vision and Pattern Recognition*, páginas 779–788, 2016. doi: 10.1109/CVPR.2016.91.

[18] J. Redmon y A. Farhadi. Yolo9000: Better, faster, stronger. En *2017 IEEE Conference on Computer Vision and Pattern Recognition*, páginas 6517–6525, Julio 2017. doi: 10.1109/CVPR.2017.690.

[19] Joseph Redmon y Ali Farhadi. Yolov3: An incremental improvement. *arXiv preprint arXiv:1804.02767*, 2018.

[20] J. M. Hirst. An automatic volumetric spore trap. *Annals of Applied Biology*, 39(2): 257 – 265, 1952. doi: 10.1111/j.1744-7348.1952.tb00904.x.

[21] Víctor Sevillano y José L. Aznarte. Improving classification of pollen grain images of the polen23e dataset through three different applications of deep learning convolutional neural networks. *PLOS ONE*, 13(9):1–18, Septiembre 2018. doi: 10.1371/journal.pone.0201807.

[22] D. G. Arias, M. V. M. Cirne, J. E. Chire, y H. Pedrini. Classification of pollen grain images based on an ensemble of classifiers. En *2017 16th IEEE International Conference on Machine Learning and Applications (ICMLA)*, páginas 234–240, Diciembre 2017. doi: 10.1109/ICMLA.2017.0-153.

[23] Amar Daood, Eraldo Ribeiro, y Mark Bush. Pollen grain recognition using deep learning. En George Bebis, Richard Boyle, Bahram Parvin, Darko Koracin, Fatih Porikli, Sandra Skaff, Alireza Entezari, Jianyuan Min, Daisuke Iwai, Amela Sadagic, Carlos Scheidegger, y Tobias Isenberg, editores, *Advances in Visual Computing*, páginas 321–330. Springer International Publishing, 2016. ISBN 978-3-319-50835-1. doi: 10.1007/978-3-319-50835-1_30.

[24] Estela Díaz-López, M. Rincón, J. Rojo, C. Vaquero, A. Rapp, S. Salmeron-Majadas, y R. Pérez-Badia. Localisation of pollen grains in digitised real daily airborne samples. En José Manuel Ferrández Vicente, José Ramón Álvarez-Sánchez, Félix de la Paz López, Fco. Javier Toledo-Moreo, y Hojjat Adeli, editores, *Artificial Computation in Biology and Medicine*, páginas 348–357. Springer International Publishing, 2015. ISBN 978-3-319-18914-7. doi: 10.1007/978-3-319-18914-7_37.

[25] Sander Landsmeer, Emile Hendriks, Letty De Weger, Johan Reiber, y Berend Stoel. Detection of pollen grains in multifocal optical microscopy images of air samples. *Microscopy research and technique*, 72:424–30, Junio 2009. doi: 10.1002/jemt.20688.

[26] R.S. Sabeenian, M.E. Paramasivam, y P.M. Dinesh. Identification and counting of fertile pollen grains using morphological operators, FSF and CGF. *International Journal of Computer Applications*, 42:36–40, Marzo 2012. doi: 10.5120/5787-8041.

[27] N. R. Nguyen, M. Donalson-Matasci, y M. C. Shin. Improving pollen classification with less training effort. En *2013 IEEE Workshop on Applications of Computer Vision (WACV)*, páginas 421–426, Enero 2013. doi: 10.1109/WACV.2013.6475049.

[28] Carmen Galán Soldevilla, Paloma Cariñanos González, Purificación Alcázar Teno, y Eugenio Domínguez Vilches. *Manual de Calidad y Gestión de la Red Española de Aerobiología*. 2007.

[29] A Duller, G Guller, I France, y H Lamb. A pollen image database for evaluation of automated identification systems. *Quaternary Newsletter*, páginas 4–9, 1999.

[30] Ariadne Barbosa Gonçalves, Junior Silva Souza, Gercina Gonçalves da Silva, Marney Pascoli Cereda, Arnildo Pott, Marco Hiroshi Naka, y Hemerson Pistori. Feature extraction and machine learning for the classification of brazilian savannah pollen grains. *PLOS ONE*, 11(6):1–20, Junio 2016. doi: 10.1371/journal.pone.0157044.

[31] Gilberto Astolfi, Ariadne Barbosa Gonçalves, Geazy Vilharva Menezes, Felipe Silveira Brito Borges, Angelica Christina Melo Nunes Astolfi, Edson Takashi Matsubara, Marco Alvarez, y Hemerson Pistori. POLLEN73S: An image dataset for pollen grains classification. *Ecological Informatics*, página 101165, 2020. ISSN 1574-9541. doi: 10.1016/j.ecoinf.2020.101165.

[32] M. Ranzato, P.E. Taylor, J.M. House, R.C. Flagan, Y. LeCun, y P. Perona. Automatic recognition of biological particles in microscopic images. *Pattern Recognition Letters*, 28(1):31 – 39, 2007. ISSN 0167-8655. doi: 10.1016/j.patrec.2006.06.010.

[33] D. G. Lowe. Object recognition from local scale-invariant features. En *Proceedings of the Seventh IEEE International Conference on Computer Vision*, volumen 2, páginas 1150–1157 vol.2, 1999. doi: 10.1109/ICCV.1999.790410.

[34] Amar Daood, Eraldo Ribeiro, y Mark Bush. Sequential recognition of pollen grain z-stacks by combining CNN and RNN. En *Proceedings of the Thirty-First International Florida Artificial Intelligence Research Society Conference, FLAIRS 2018, Melbourne, Florida, USA. May 21-23 2018.*, páginas 8–13, 2018.

[35] Víctor Sevillano, Katherine Holt, y José L. Aznarte. Precise automatic classification of 46 different pollen types with convolutional neural networks. *PLOS ONE*, 15(6): 1–15, Junio 2020. doi: 10.1371/journal.pone.0229751.

[36] Olga Russakovsky, Jia Deng, Hao Su, Jonathan Krause, Sanjeev Satheesh, Sean Ma, Zhiheng Huang, Andrej Karpathy, Aditya Khosla, Michael Bernstein, Alexander C. Berg, y Li Fei-Fei. Imagenet large scale visual recognition challenge. *International Journal of Computer Vision*, 115(3):211–252, Diciembre 2015. ISSN 1573-1405. doi: 10.1007/s11263-015-0816-y.

[37] R. A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7(2):179–188, 1936. doi: 10.1111/j.1469-1809.1936.tb02137.x.

[38] Gunnar Erdtman. The acetolysis method, a revised description. *Sven Bot Tidskr*, 54:561–564, 1960.

[39] K. Holt, G. Allen, R. Hodgson, S. Marsland, y J. Flenley. Progress towards an automated trainable pollen location and classifier system for use in the palynology laboratory. *Review of Palaeobotany and Palynology*, 167(34):175–183, 2011. ISSN 0034-6667. doi: 10.1016/j.revpalbo.2011.08.006.

[40] G. Huang, Z. Liu, L. Van Der Maaten, y K. Q. Weinberger. Densely connected convolutional networks. En *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, páginas 2261–2269, 2017.

[41] J. J. Koenderink y A. J. van Doorn. Representation of local geometry in the visual system. *Biological Cybernetics*, 55(6):367, 1987. doi: 10.1007/BF00318371.

[42] C. Schmid y R. Mohr. Local grayvalue invariants for image retrieval. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(5):530–535, 1997.

[43] C. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, 1995.

[44] Y. LeCun, L. Bottou, Y. Bengio, y P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, Noviembre 1998. doi: 10.1109/5.726791.

[45] P. LI, W. J. Treloar, J. R. Flenley, y L. Empson. Towards automation of palynology 2: the use of texture measures and neural network analysis for automated identification of optical images of pollen grains. *Journal of Quaternary Science*, 19 (8):755–762, 2004. doi: 10.1002/jqs.874.

[46] Y. Yao y G. Doretto. Boosting for transfer learning with multiple sources. En *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, páginas 1855–1862, 2010. doi: 10.1109/CVPR.2010.5539857.

[47] Carolyn Kimme, Dana Ballard, y Jack Sklansky. Finding circles by an array of accumulators. *Commun. ACM*, 18(2):120–122, Febrero 1975. ISSN 0001-0782. doi: 10.1145/360666.360677.

[48] G. Bradski. The OpenCV Library. *Dr. Dobb's Journal of Software Tools*, 2000.

[49] Richard O. Duda y Peter E. Hart. Use of the hough transformation to detect lines and curves in pictures. *Commun. ACM*, 15(1):11–15, Enero 1972. ISSN 0001-0782. doi: 10.1145/361237.361242.

[50] Gary Bradski y Adrian Kaehler. *Learning OpenCV: Computer Vision with the OpenCV Library*. O'Reilly Media, first edition, 2008. ISBN 978-0-596-51613-0.

[51] Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006. ISBN 0-38-731073-8.

[52] David A. Forsyth y Jean Ponce. *Computer Vision: A Modern Approach, 2nd edition*. Pearson, 2012. ISBN 9780136085928.

[53] David E. Rumelhart, Geoffrey E. Hinton, y Ronald J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986.

[54] Ian Goodfellow, Yoshua Bengio, y Aaron Courville. *Deep Learning*. MIT Press, 2016. URL deeplearningbook.org. Consultado 2020-10-20.

[55] G. van Rossum. Python tutorial. Technical Report CS-R9526, Centrum voor Wiskunde en Informatica (CWI), Amsterdam, 1995.

[56] The SciPy community. Numpy, 2020. URL numpy.org. Consultado 2020-10-20.

[57] Martín Abadi, Ashish Agarwal, Paul Barham, Eugene Brevdo, Zhifeng Chen, Craig Citro, Greg S. Corrado, Andy Davis, Jeffrey Dean, Matthieu Devin, Sanjay Ghemawat, Ian Goodfellow, Andrew Harp, Geoffrey Irving, Michael Isard, Yangqing Jia, Rafal Jozefowicz, Lukasz Kaiser, Manjunath Kudlur, Josh Levenberg, Dandelion Mané, Rajat Monga, Sherry Moore, Derek Murray, Chris Olah, Mike Schuster, Jonathon Shlens, Benoit Steiner, Ilya Sutskever, Kunal Talwar, Paul Tucker, Vincent Vanhoucke, Vijay Vasudevan, Fernanda Viégas, Oriol Vinyals, Pete Warden, Martin Wattenberg, Martin Wicke, Yuan Yu, y Xiaoqiang Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL www.tensorflow.org. Consultado 2020-10-20.

[58] R. Collobert, K. Kavukcuoglu, y C. Farabet. Torch7: A matlab-like environment for machine learning. En *BigLearn, NIPS Workshop*, 2011.

[59] Yangqing Jia, Evan Shelhamer, Jeff Donahue, Sergey Karayev, Jonathan Long, Ross Girshick, Sergio Guadarrama, y Trevor Darrell. Caffe: Convolutional architecture for fast feature embedding. *arXiv preprint arXiv:1408.5093*, 2014.

[60] François Chollet et al. Keras, 2015. URL keras.io. Consultado 2020-10-20.

[61] NVIDIA, Péter Vingelmann, y Frank H.P. Fitzek. Cuda, release: 10.2.89, 2020. URL developer.nvidia.com/cuda-toolkit. Consultado 2020-10-20.

[62] Khronos Group. Opencl (open computing language), 2020. URL www.khronos.org/opencl. Consultado 2020-10-20.

[63] D. H. Hubel y T. N. Wiesel. Receptive fields and functional architecture of monkey striate cortex. *The Journal of Physiology*, 195(1):215–243, 1968. doi: 10.1113/jphysiol.1968.sp008455.

[64] Kunihiko Fukushima. Neocognitron: A self-organizing neural network model for a mechanism of pattern recognition unaffected by shift in position. *Biological Cybernetics, volume = "36", number = "4", pages = "193-202", year = 1980, doi= 10.1007/BF00344251.*

[65] Alex Krizhevsky, Ilya Sutskever, y Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. En F. Pereira, C. J. C. Burges, L. Bottou, y K. Q. Weinberger, editores, *Advances in Neural Information Processing Systems 25*, páginas 1097–1105. Curran Associates, Inc., 2012. doi: 10.1145/3065386.

[66] Xavier Glorot, Antoine Bordes, y Yoshua Bengio. Deep sparse rectifier neural networks. volumen 15 of *Proceedings of Machine Learning Research*, páginas 315–323, Fort Lauderdale, FL, USA, Abril 2011. JMLR Workshop and Conference Proceedings.

[67] Trevor Hastie, Robert Tibshirani, y Jerome Friedman. *The Elements of Statistical Learning. Data Mining, Inference, and Prediction, 2nd edition.* Springer, 2009. ISBN 978-0-387-84858-7.

[68] Geoffrey E. Hinton, Nitish Srivastava, Alex Krizhevsky, Ilya Sutskever, y Ruslan Salakhutdinov. Improving neural networks by preventing co-adaptation of feature detectors. *arXiv preprint arXiv:1207.0580*, 2012.

[69] Karen Simonyan y Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. En *ICLR 2015 Conference Proceedings*, San Diego, USA, 2015.

[70] Y. Bengio, P. Simard, y P. Frasconi. Learning long-term dependencies with gradient descent is difficult. *IEEE Transactions on Neural Networks*, 5(2):157–166, 1994. doi: 10.1109/72.279181.

[71] Xavier Glorot y Yoshua Bengio. Understanding the difficulty of training deep feedforward neural networks. volumen 9 of *Proceedings of Machine Learning Research*,

páginas 249–256, Chia Laguna Resort, Sardinia, Italy, Mayo 2010. JMLR Workshop and Conference Proceedings.

[72] Kaiming He, Xiangyu Zhang, Shaoqing Ren, y Jian Sun. Deep residual learning for image recognition. *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, páginas 770–778, 2016. doi: 10.1109/CVPR.2016.90.

[73] Chuanqi Tan, Fuchun Sun, Tao Kong, Wenchang Zhang, Chao Yang, y Chunfang Liu. A survey on deep transfer learning. En Věra Kůrková, Yannis Manolopoulos, Barbara Hammer, Lazaros Iliadis, y Ilias Maglogiannis, editores, *Artificial Neural Networks and Machine Learning – ICANN 2018*, páginas 270–279. Springer International Publishing, 2018. ISBN 978-3-030-01424-7.

[74] N. Dalal y B. Triggs. Histograms of oriented gradients for human detection. En *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, volumen 1, páginas 886–893 vol. 1, 2005. doi: 10.1109/CVPR.2005.177.

[75] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, y D. Ramanan. Object detection with discriminatively trained part-based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 32(9):1627–1645, 2010. doi: 10.1109/TPAMI.2009.167.

[76] J. R. R. Uijlings, K. E. A. van de Sande, T. Gevers, y A. W. M. Smeulders. Selective search for object recognition. *International Journal of Computer Vision*, 104(2): 154–171, Septiembre 2013. ISSN 1573-1405. doi: 10.1007/s11263-013-0620-5.

[77] Pedro F. Felzenszwalb y Daniel P. Huttenlocher. Efficient graph-based image segmentation. *International Journal of Computer Vision*, 59(2):167–181, Septiembre 2004. ISSN 1573-1405. doi: 10.1023/B:VISI.0000022288.19776.77.

[78] K. He, X. Zhang, S. Ren, y J. Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 37(9):1904–1916, 2015. doi: 10.1109/TPAMI.2015.2389824.

[79] C. Lawrence Zitnick y Piotr Dollár. Edge boxes: Locating object proposals from edges. En David Fleet, Tomas Pajdla, Bernt Schiele, y Tinne Tuytelaars, editores, *Computer Vision – ECCV 2014*, páginas 391–405. Springer International Publishing, 2014. ISBN 978-3-319-10602-1.

[80] T. Lin, P. Dollar, R. Girshick, K. He, B. Hariharan, y S. Belongie. Feature pyramid networks for object detection. En *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, páginas 936–944, Los Alamitos, CA, USA, Julio 2017. IEEE Computer Society. doi: 10.1109/CVPR.2017.106.

[81] JR Flenley. The problem of pollen recognition. *Problems in Picture Interpretation, Clowes MB, Penny JP (eds). CSIRO: Canberra*, páginas 141–145, 1968.

[82] The Qt Company Ltd. Qt, a cross-platform software development framework, 2020. URL www.qt.io. Consultado 2020-10-20.

[83] The PostgreSQL Global Development Group. PostgreSQL. URL www.postgresql.org. Consultado 2020-10-20.

[84] Ross Girshick, Ilija Radosavovic, Georgia Gkioxari, Piotr Dollár, y Kaiming He. Detectron, 2018. URL github.com/facebookresearch/detectron. Consultado 2020-10-20.

[85] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, y Adam Lerer. Automatic differentiation in pytorch. En *NIPS-W*, 2017.

[86] NVIDIA Corporation. Nvidia jetson systems, 2019. URL www.nvidia.com/en-us/autonomous-machines/embedded-systems. Consultado 2020-10-20.

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

[88] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, y C. Lawrence Zitnick. Microsoft coco: Common objects in context. En David Fleet, Tomas Pajdla, Bernt Schiele, y Tinne Tuytelaars, editores, *Computer Vision – ECCV 2014*, páginas 740–755. Springer International Publishing, 2014. ISBN 978-3-319-10602-1.

[89] Mark Everingham, S. M. Ali Eslami, Luc Van Gool, Christopher K. I. Williams, John Winn, y Andrew Zisserman. The pascal visual object classes challenge: A retrospective. *International Journal of Computer Vision*, 111(1):98–136, Enero 2015. ISSN 1573-1405. doi: 10.1007/s11263-014-0733-5.

[90] Ross Girshick, Ilija Radosavovic, Georgia Gkioxari, Piotr Dollár, y Kaiming He. Detectron model zoo and baselines, 2018. URL github.com/facebookresearch/Detectron/blob/master/MODEL_ZOO.md. Consultado 2020-10-20.

[91] R. Gallardo-Caballero, C.J. García-Orellana, A. García-Manso, H.M. González-Velasco, R. Tormo-Molina, y M. Macías-Macías. Precise pollen grain detection in bright field microscopy using deep learning techniques. *Sensors*, 19:3583.1 – 3583.19, 2019. ISSN 1424-8220. doi: 10.3390/s19163583.

[92] Ramón Gallardo-Caballero, Antonio García-Manso, Carlos J. García-Orellana, Horacio M. González-Velasco, Rafael Tormo-Molina, y Miguel and Macías-Macías. CAPIPollen DB1, 2019. URL capi.unex.es/pollendb1. Consultado 2020-10-20.