PROYECTO FIN DE GRADO

TÍTULO: Diseño de un analizador de muestras de suero sanguíneo mediante procesado de imagen y aprendizaje profundo en microscopía

AUTOR/A: Carlos Antonio Garijo Crespo

TITULACIÓN: Ingeniería de Sonido e Imagen

DIRECTOR/A: Gonzalo Rosa Olmeda

TUTOR/A: Miguel Chavarrías Lapastora

DEPARTAMENTO: Departamento de Ingeniería Telemática y Electrónica

VºBº TUTOR/A

Miembros del Tribunal Calificador:

PRESIDENTE/A: Rafael José Hernández Heredero

TUTOR/A: Miguel Chavarrías Lapastora

SECRETARIO/A: Eduardo Juárez Martínez

Fecha de lectura:

Calificación:

El Secretario/La Secretaria,

Resumen

Este PFG, con título "Diseño de un analizador de muestras de suero sanguíneo mediante procesado de imagen y aprendizaje profundo en microscopía", se centra en el estudio y desarrollo de técnicas alternativas y no invasivas de diagnóstico del grado de traumatismo craneoencefálico a través del análisis de muestras de suero sanguíneo secas. El fin de este proyecto es desarrollar un sistema que, a partir de una imagen de una muestra de este tipo, permita evaluar el nivel de lesión cerebral que sufre un paciente, ofreciendo una alternativa a las pruebas de diagnóstico actuales como la Tomografía Axial Computerizada (TAC) o Resonancia Magnética (RM) las cuales presentan algunas desventajas como la exposición a radiación ionizante, el elevado coste que suponen y su difícil acceso en algunas zonas debido a la falta de los recursos necesarios para su realización. Para ello se investigan, desarrollan e implementan distintas técnicas de tratamiento digital de la imagen, visión por ordenador y aprendizaje automático, con el objetivo de caracterizar las propiedades visuales de las muestras que permitan en última instancia el desarrollo de un sistema eficiente de diagnóstico del daño cerebral.

Para la elaboración de este proyecto se hace un estudio del estado del arte actual en áreas relacionadas con el procesado de imagen y aprendizaje automático. De esta forma se escoge *Python* como lenguaje de programación para el desarrollo del sistema y se investigan diversas bibliotecas y librerías como *OpenCV*, que implementa módulos y herramientas de procesado de imagen y visión por ordenador, así como *Scikit-learn* y *PyTorch*, los cuales implementan un gran número de funcionalidades que facilitan el desarrollo de sistemas de aprendizaje automático y *Deep Learning*. Los distintos experimentos realizados han sido aplicados sobre una base de datos conformada por imágenes de distintos recortes o *tiles* de muestras de suero sanguíneo secas de 117 pacientes facilitadas por el Hospital 12 de Octubre, y capturadas a través de un sistema de microscopía óptica y una cámara RGB. Estas muestras pertenecen a distintos pacientes los cuales han sido clasificados previamente en cuatro niveles de daño cerebral: control (sano), leve, moderado y grave.

El desarrollo de este PFG se conforma de cinco etapas. La primera etapa consiste en la captura de las imágenes de las muestras para la creación de la base de datos necesaria para realizar los experimentos. La segunda consiste en la creación de un algoritmo automatizado de extracción y representación de las características visuales pertinentes de las imágenes de la base de datos. Posteriormente, se realizan distintos experimentos con la información extraída con el objetivo de lograr una clasificación de las distintas muestras en los distintos niveles de grado de lesión, utilizando algoritmos de aprendizaje automático como *K-Neighbors* y máquinas de vectores soporte (SVM). En la cuarta etapa se realiza el entrenamiento de una red neuronal convolucional (CNN) con el mismo objetivo que los experimentos anteriores para, finalmente, en

la quinta etapa realizar una evaluación y comparación de los resultados obtenidos mediante los distintos métodos, con el fin de considerar cuál de ellos tiene una efectividad mayor para lograr el objetivo presentado en este proyecto.

Los resultados obtenidos son una tasa de aciertos del 52,8 % en el mejor entrenamiento realizado mediante SVM y un 75,0 % en el caso del entrenamiento de la CNN, ambos para los cuatro niveles de lesión de los pacientes. En el capítulo de descripción de la solución propuesta se detalla cómo se han conseguido estos resultados y, posteriormente en los apartados de resultados y conclusiones se presenta un análisis sobre las causas relacionadas con la obtención de estos valores, su relación con los datos extraídos y por qué puede no ser óptima la metodología que pretende entrenar modelos de aprendizaje automático con los vectores de características extraídos mediante el sistema de extracción de características visuales. A su vez, se presentan ciertas ventajas que ofrecen las redes neuronales convolucionales, las cuales están relacionadas con la obtención de una mayor tasa de aciertos. Por último, se analiza la importancia del tamaño de la base de datos de pacientes para obtener unos mejores rendimientos y generalización de los modelos de aprendizaje automático, así como la necesidad de repetir los experimentos planteados en este proyecto con un tamaño de la base de datos mayor.

Este PFG pretende influir un impacto positivo en términos sociales y sanitarios debido a la investigación realizada sobre el desarrollo de pruebas alternativas no invasivas. Por ello, se considera que está alineado con los Objetivos de Desarrollo Sostenible (ODS) definidos por la Asamblea General de las Naciones Unidas.

Abstract

This PFG, entitled "Design of a blood serum sample analyser using image processing and deep learning in microscopy", focuses on the study and development of alternative and non-invasive techniques for diagnosing the degree of traumatic brain injury through the analysis of dry blood serum samples. The aim of this project is to develop a system that, based on an image of a sample of this type, can evaluate the level of brain injury suffered by a patient, offering an alternative to current diagnostic tests such as Computerised Axial Tomography (CAT) or Magnetic Resonance Imaging (MRI), which have certain disadvantages such as exposure to ionising radiation, the high cost involved and their difficult access in some areas due to the lack of the necessary resources to carry them out. To this purpose, different digital image processing, computer vision and machine learning techniques are researched, developed and implemented, with the aim of characterising the visual properties of the samples that will ultimately enable the development of an efficient system for diagnosing brain damage.

For the development of this project, a study of the current state of the art in areas related to image processing and machine learning was carried out. Thus, *Python* was chosen as the programming language for the development of the system and different frameworks and libraries were used, such as *OpenCV*, which implements modules and tools for image processing and computer vision tasks, as well as *Scikit-learn* and *PyTorch*, which implement a large number of functionalities that facilitate the development of machine learning and deep learning systems. The various experiments conducted were applied to a database consisting of images of different tiles of dried blood serum samples from 117 patients provided by the Hospital 12 de Octubre, and captured through an optical microscopy system and an RGB camera. These samples belong to different patients who have been previously classified into four levels of brain damage: control (healthy), mild, moderate and severe.

The development of this project is divided into five stages. The first stage involves capturing the images of the samples for the creation of the database needed to carry out the experiments. The second stage focuses on the creation of an automated algorithm for the extraction and representation of the relevant visual characteristics of the images in the database. Subsequently, different experiments are carried out with the extracted information with the aim of achieving a classification of the different samples in the different levels of injury, using machine learning algorithms such as *K-Neighbors* and support vector machines (SVM). In the fourth stage, a convolutional neural network (CNN) is trained with the same objective as the previous experiments. Finally, in the fifth stage, an evaluation and comparison of the results obtained using the different methods is carried out in order to consider which of them is most effective in achieving the objective presented in this project.

The results obtained are an accuracy of 52.8 % in the best training of the SVM model and 75.0 % in the case of the training of the CNN, both for the four levels of injury of the patients. The chapter describing the proposed solution details how these results have been achieved and, subsequently, in the sections results and conclusions, an analysis is presented on the causes related to these obtained values, their relationship with the extracted data and why the methodology that aims to train machine learning models with the feature vectors extracted using the visual feature extraction system may not be optimal. In addition, certain advantages offered by convolutional neural networks, related to obtaining a higher accuracy, are presented. Finally, the importance of the size of the patient database is analysez with regard to obtaining better performance and generalization of the machine learning models, as well as the need to repeat the experiments proposed in this project with a larger database size.

This PFG aims to influence a positive impact in social and health terms due to the research conducted on the development of non-invasive alternative tests. It is therefore considered to be aligned with the Sustainable Development Goals (SDGs) defined by the United Nations General Assembly.

Agradecimientos

Quiero agradecer a todas esas personas que me han acompañado durante estos cuatro años de carrera, junto con los que he crecido y me he desarrollado como persona.

A mi familia, por apoyarme siempre en mis decisiones y ofrecerme apoyo en aquellos momentos en los que lo he necesitado.

A Clara, por todo lo que me ha enseñado, y por su paciencia y cariño durante estos cuatro años conmigo.

A todos mis amigos que han hecho mi existencia más feliz, y junto con los que he realizado este camino.
En especial a Andrés, Patri, Sara e Isa, con los que he compartido exámenes, trabajos, momentos de estrés y momentos de liberación durante la carrera;
a Gogui, Nico, Tasio y Kike, con los que he podido compartir momentos de creatividad a través de la música, dándome en algunas ocasiones esos momentos de desconexión que tan necesarios han sido;
a Santi y Simón por haber estado siempre ahí desde mucho antes que empezase la carrera;
a mis amigos los Jinetes y a todo el resto de gente que han hecho que me convierta en la persona que soy a día de hoy.

También quiero agradecer a Miguel y a Gonzalo por la guía y la ayuda que me han ofrecido para el desarrollo de este proyecto con el que culmina mi carrera y comienza una nueva etapa.

Índice general

Índice de figuras

Índice de tablas

Listado de código

Lista de Acrónimos

ATR-FTIR *Attenuated Total Reflectance-Fourier Transform Infrared*. XV, 8

CITSEM Centro de Investigación en Tecnologías Software y Sistemas Multimedia para la Sostenibilidad. XV, 40

CNN *Convolutional Neural Network*. I–III, X, XV, 2, 32–34, 57, 58, 62, 63, 74–76, 81–83

GDEM Grupo de Diseño Electrónico y Microelectrónico. XV, 37, 38, 40

HSI *HyperSpectral Imaging*. XV, 7–9

IRM Imagen por Resonancia Magnética. XV

LDA *Linear Discriminant Analysis*. Algoritmo supervisado.. XV

ML *Machine Learning*. XIII, XV, 8, 21, 25, 29, 54–56, 61

MSI *MultiSpectral Imaging*. XV, 7–9

NIR *Near-Infrared*. XV, 9

ODS Objetivos de Desarrollo Sostenible. II, XI, XV, 4, 86

OpenCV Biblioteca de procesado de imagen de Python y C++. XV, 11, 17

PCA *Principle Component Analysis*. XV, 8, 9

PFG Proyecto Fin de Grado. I–IV, IX, X, XIII, XV, 1–5, 10, 11, 14, 17, 23, 29, 38–40, 63, 66, 73, 74, 79, 81, 86

PLS-DA *Partial Least Squares-Discriminant Analysis*. XV, 8, 9

RGB *Red Green Blue*. XV, 7, 8, 40, 43

RM Resonancia Magnética. I, XV, 1, 7, 8, 86

SVM *Support Vector Machine*. Algoritmo de Machine Learning básico de clasificación. I–III, IX, X, XIII, XV, 2, 9, 21, 24–27, 54, 55, 57, 69, 71–73

TAC Tomografía Axial Computerizada. I, XV, 1, 7, 8, 85, 86

TDI Tratamiento Digital de la Imagen. XV, 12, 13

Capítulo 1

Introducción

El diagnóstico rápido y preciso de traumatismos craneoencefálicos en pacientes que han sufrido accidentes o impactos en la cabeza es crítico para evitar posibles secuelas a largo plazo derivadas de lesiones cerebrales. La detección temprana de estas posibles lesiones es determinante para elaborar un tratamiento adecuado que pueda disminuir daños irreversibles en el futuro. Entre las tecnologías de diagnóstico más extendidas para evaluar traumatismos craneoencefálicos se encuentran la Tomografía Axial Computerizada (TAC) y la Resonancia Magnética (RM), siendo la primera la más utilizada. Estas pruebas presentan algunas problemáticas, tales como los recursos necesarios para poder realizarse o la radiación ionizante a la que se expone un paciente en el caso del TAC, lo cual puede suponer un riesgo potencial para la salud de la persona a largo plazo. En este sentido, hay una atención creciente hacia el desarrollo de sistemas de evaluación preliminar o triaje, que utilicen tecnologías más accesibles para discriminar aquellos pacientes que requieren una exploración más exhaustiva de los que no, reduciendo así a largo plazo costes económicos y consecuencias negativas en la salud asociadas al uso de métodos invasivos. Además, el desarrollo de pruebas alternativas accesibles resulta esencial para mejorar la atención médica, especialmente en áreas que por razones económicas o geográficas el acceso a determinados recursos y tecnologías es limitado.

En este contexto, el aprendizaje automático es un área que puede contribuir a la creación de nuevas soluciones y funcionalidades para desarrollar sistemas de diagnóstico alternativos. Los modelos de visión artificial basados en técnicas de aprendizaje automático pueden analizar imágenes médicas de manera rápida y precisa e identificar en ellas patrones y características visuales que portan información relevante relacionada con un determinado diagnóstico. Esto las convierte en herramientas de gran valor que pueden mejorar el proceso de evaluación médica. Además, su capacidad para aprender y mejorar con el tiempo los convierte en herramientas versátiles y adaptativas, ya que su función puede variar según los datos con los que se entrenen, sin la necesidad de diseñar la lógica que subyace a su funcionamiento.

Por estos motivos, este PFG se centra en el estudio y análisis de distintas técnicas de apren-

dizaje automático con el fin de construir un sistema capaz de ofrecer un diagnóstico del grado de lesión cerebral que sufre un paciente a partir de una imagen de una muestra de suero sanguíneo seca. Dado que la literatura relacionada con el estudio de muestras de suero sanguíneo secas mediante técnicas de aprendizaje automático es reducida, el objetivo de este proyecto es investigar, implementar y evaluar a través de diferentes experimentos distintos métodos con la finalidad de hacer una evaluación de la eficacia que tienen en el desarrollo de dicho sistema de diagnóstico.

Para ello se hace uso de distintas técnicas de tratamiento digital de la imagen y visión por ordenador que permiten hacer las operaciones necesarias sobre las imágenes para extraer la información y características relevantes. Como modelos de aprendizaje automático se analiza el algoritmo *K-Neighbors*, las máquinas de vectores soporte (SVM) y una red neuronal convolucional (CNN).

1.1. Objetivos y metodología de trabajo

En este contexto, el objetivo principal de este PFG es desarrollar un sistema de diagnóstico del grado de lesión cerebral a partir de imágenes de muestras de suero sanguíneo secas comparando el rendimiento de distintos métodos de aprendizaje automático. Así, para el desarrollo del proyecto se ha seguido la siguiente metodología de trabajo:

- Realización de la investigación y documentación necesaria en lo referente a visión por ordenador, tratamiento digital de la imagen y aprendizaje automático en las distintas etapas del proyecto.
- Generación de una base de datos de imágenes que permita realizar y evaluar los experimentos realizados en este proyecto.
- Desarrollo en *Python* de los distintos sistemas y funcionalidades que permitan analizar las imágenes de la base de datos, extraer la información relevante, implementar los distintos modelos de aprendizaje automático y evaluar los resultados obtenidos.
- Selección de los distintos modelos de aprendizaje automático y ajuste de los hiperparámetros necesarios a partir de técnicas de validación cruzada a fin de optimizar su rendimiento.
- Entrenamiento de una red neuronal convolucional, adaptando su arquitectura a las características de la base de datos.
- Desarrollo de las distintas funciones necesarias que gestionen y adapten los datos, ya sean numéricos o visuales, de la forma requerida para entrenar y evaluar los diferentes modelos, y evitando cualquier sesgo que pueda producirse.
- Evaluación de los distintos modelos a partir de las métricas necesarias.

- Comparación de los resultados y conclusiones en función de la tasa de aciertos y la complejidad computacional asociada.

De esta forma, para el desarrollo de este PFG ha sido clave la consecución de los siguientes hitos principales:

- Diseño del sistema que extrae y caracteriza la información visual presente en las imágenes de las muestras de suero sanguíneo secas.
- Diseño de distintos clasificadores utilizando técnicas de aprendizaje automático que aprenden de la información extraída por el sistema de extracción y caracterización de las propiedades visuales de las muestras.
- Diseño de un clasificador utilizando una red neuronal convolucional que aprende directamente de las imágenes de muestras de suero sanguíneo secas de la base de datos.

1.2. Estructura de la memoria

El presente PFG se divide en los siguientes capítulos:

- **Capítulo 1. Introducción**. En este capítulo se presenta el contexto en el que se desarrolla este PFG así como los objetivos descritos para su elaboración.
- **Capítulo 2. Marco teórico y tecnológico**. En este capítulo se presentan las distintas áreas y tecnologías relacionadas con el desarrollo de este trabajo, principalmente sobre muestras de suero sanguíneo secas, Visión por Ordenador, Tratamiento Digital de la Imagen e Inteligencia Artificial aplicada a imagen.
- **Capítulo 3. Especificaciones y restricciones**. En este capítulo se muestran las especificaciones que el desarrollo de este proyecto debe cumplir así como las restricciones que lo limitan.
- **Capítulo 4. Descripción de la solución propuesta**. En este capítulo se describen las distintas implementaciones que conforman los sistemas desarrollados para la elaboración de este PFG.
- **Capítulo 5. Resultados**. En este capítulo se muestran los resultados obtenidos por los distintos sistemas utilizados en este proyecto.
- **Capítulo 6. Presupuesto**. En este capítulo se desglosan los gastos relacionados con el desarrollo de este proyecto.

- **Capítulo 7. Impacto social y ambiental**. En este capítulo se analiza el posible impacto que puede tener este proyecto a nivel económico, social y medioambiental, en el contexto de los ODS establecidos por la Asamblea General de las Naciones Unidas.
- **Capítulo 8. Conclusiones**. En este capítulo se presentan las evaluaciones finales sobre los distintos modelos y técnicas utilizadas así como una valoración general del trabajo. Además, se exponen posibles líneas futuras relacionadas con este PFG.

Capítulo 2

Marco teórico y tecnológico

En este apartado se introducen los diferentes conceptos que conforman el marco teórico y tecnológico sobre el que se ha desarrollado este PFG.

En el primer apartado se presenta una descripción general sobre el estado del arte en el uso de análisis de imagen para el diagnóstico y estudio de traumatismos creaneoencefálicos. En segundo lugar, se presenta una breve descripción del estado actual de la microscopía óptica, tecnología que ha sido clave en el desarrollo de este PFG.

En el tercer apartado se presenta una introducción a la visión por ordenador o *Computer Vision* y sus aplicaciones. Se detallan distintas funciones usadas en este proyecto así como las bibliotecas de programación que han sido utilizadas.

Por último, se detallan algunas características y aplicaciones de los algoritmos de Machine Learning y Deep Learning, así como sus aplicaciones en el ámbito del procesado de bioimágenes y salud.

2.1. Muestras de plasma sanguíneo secas

El secado de muestras de sangre y plasma sanguíneo ha sido objeto de estudio durante las últimas decadas, ya que ha demostrado ser de utilidad en distintas aplicaciones como arqueología, impresión de chips de ADN (*DNA microarray printing*), medicina e investigación forense. Al secar muestras de sangre y plasma sanguíneo sobre superficies sólidas no porosas se puede observar la formación de distintos patrones y estructuras sobre la muestra secada. En [1] Ruoyang Chen *et al.* ofrecen un análisis detallado y exhaustivo de la formación de los distintos patrones y estructuras en muestras secadas de plasma y sangre, su composición y aplicaciones que pueden tener en distintos áreas de investigación. En esta sección, se detallan algunas de las características más relevantes para la redacción de este informe.

La evaporación del líquido de las muestras da lugar a la formación de estructuras compuestas

por los componentes suspendidos no volátiles en la muestra una vez han cristalizado, generando así patrones definidos caracterizados por grietas que los conforman. Un fenómeno interesante es la formación del llamado *coffee ring* cuyo nombre se debe a la similitud que tiene con la formación del anillo de depósitos que se produce cuando se derrama café sobre alguna superficie sólida. Este fenómeno se produce debido a que cuando se evapora el líquido de los bordes de la muestra, el líquido que se encuentra en el centro fluye hasta los bordes arrastrando consigo los distintos elementos presentes en la muestra, produciendo así que haya una concentración mayor de los materiales en el borde que en el centro. Es por este fenómeno que los patrones que se producen se forman con mayor tamaño en la zona periférica, encontrándose estructuras más finas e irregulares en la parte central.

Figura 2.1. Muestra de ejemplo de plasma sanguíneo secado.

Diversos estudios han demostrado que variaciones en los componentes celulares y macromoleculares presentes en la sangre, variaciones en la concentración de elementos suspendidos y en la estructura química de la muestra producen diferentes patrones y estructuras, lo que las convierte en una fuente de información acerca del sujeto cuya sangre se está analizando, y un objeto de estudio para la identificación de diferentes estados o enfermedades presentes en el sujeto. En [2] se muestra un estudio y metodología para identificar si un paciente ha realizado un deporte exhaustivo recientemente en función de las diferencias entre los patrones presentes en las distintas muestras, obteniendo un resultado positivo.

2.2. Análisis de bioimágenes para el diagnóstico médico

Las herramientas de análisis de imagen albergan una gran utilidad dentro del sector médico. El análisis de imagen se basa en el estudio de alguna de las propiedades capturadas en una imagen para establecer relaciones y diagnósticos finales. Es por esto mismo que el objetivo a perseguir cuando se aplica análisis de imagen es capturar las características que encierren algún tipo de correlación con la o las condiciones que se desean estudiar. Y por esto mismo, es también importante asegurar que dichas características estén presentes en las muestras y sean detectables por el sistema de análisis. Por ejemplo, si se sabe que una condición concreta bajo estudio está relacionada con la absorción de un material a cierta frecuencia, es imprescindible asegurar que la fuente que excite dicho material contenga esa frecuencia, así como que el sistema de captura sea capaz de medirla.

Para lograr dicho objetivo se han desarrollado distintas tecnologías que abarcan desde el análisis clásico de imágenes monocromáticas y RGB (*Red Green Blue*) hasta análisis de imagen hiperespectral (HSI) y multiespectral (MSI)[1].

Un ejemplo del uso de imágenes monocromáticas es la Tomografía Axial Computerizada (TAC). En esta prueba se hace uso de rayos X para obtener la imagen correspondiente a un corte producido por un plano en el interior del cuerpo. La realización de estas imágenes tiene algunas contraindicaciones como pueden ser el elevado coste que supone y la exposición del paciente a radiación ionizante. Esta tecnología está muy extendida ya que permite detectar patologías como tumores, coágulos, fracturas y hemorragias internas, entre otras, con un alto grado de resolución y precisión. En el caso de los traumatismos craneoencefálicos, el TAC también sirve para obtener un diagnóstico fiable y descartar posibles lesiones, sin embargo, solamente el 7 % de los TACs que se realizan muestran realmente lesiones creaneoencefálicas [3], lo que supone un gasto innecesario, así como un riesgo para los pacientes que realizan la prueba.

Otra tecnología muy extendida es la Resonancia Magnética (RM). La RM, a diferencia del TAC, es una tecnología no invasiva que también se utiliza para la detección de lesiones y enfermedades. Esta tecnología consiste en la emisión de un campo magnético muy poderoso, de hasta 3T, que alinea los protones del cuerpo con dicho campo. Una vez alineados, se estimulan los protones mediante la emisión de una radiofrecuencia que altera su equilibrio y los hace girar a gran velocidad. Posteriormente, se cesa la emisión de dicha radiofrecuencia haciendo que los protones vuelvan a alinearse con el campo magnético, emitiendo energía que es capturada por los sensores de la máquina de RM. En función del tejido, entorno y naturaleza química de las moléculas, tardarán más o menos en realinearse con el campo magnético, emitiendo así una mayor o menor energía, cuyas diferencias son las que permiten la construcción de la imagen final.

[1] Aunque ambos términos hacen referencia a la captura de imágenes con un número mayor de canales de longitudes de onda que las clásicas imágenes RGB, éstas se diferencian en que en las MSI se seleccionan previamente un número y frecuencias concretas de longitudes de onda a capturar mientras que en las HSI se captura el espectro electromagnético de forma continua.

Las RM son especialmente indicadas para obtener imágenes de los tejidos blandos del cuerpo. Estas imágenes tienen mayor detalle y precisión que las obtenidas mediante TAC, por lo que, tanto por su naturaleza no invasiva como por su mayor calidad, es la tecnología preferida para la captura de imágenes del cerebro. Su desventaja frente al TAC es el elevado coste que supone la realización de este tipo de pruebas y la maquinaria necesaria.

Existen más tecnologías de captura de imágenes médicas como la medicina nuclear, con las SPECT (*Single Photon Emission Computed Tomography*) y las PET (*Positron Emission Tomography*). No obstante, las tecnologías más populares y utilizadas son el TAC y el RM.

Profundizando más en el análisis de imagen, posteriormente se desarrollaron las tecnologías HSI/MSI. A diferencia de las imágenes RGB, estas imágenes capturan más de tres frecuencias del espectro, permitiendo así obtener imágenes con un mayor volumen de información espectral. Las MSI y HSI pueden revelar información de suma importancia para ciertos sistemas de diagnóstico médico, que no podrían hacerse con imágenes RGB o monocromáticas. Esto es porque este tipo de imágenes permiten registrar las interacciones que tienen ciertas moléculas y tejidos con las distintas frecuencias del espectro electromagnético. Algunos ejemplos de aplicaciones en las que estas imágenes son útiles son la detección del cáncer de piel, la detección de márgenes tumorales, el diagnóstico de enfermedades cardiovasculares o la evaluación de tejidos cerebrales. S. Ortega *et al.* desarrollan en [4] el estado actual de estas tecnologías y sus diferentes aplicaciones en el campo de la medicina.

En [3] A. G. Theakstone *et al.* propone un método basado en el análisis espectral de muestras de suero sanguíneo líquidas mediante *Machine Learning*, de aquí en adelante ML, para identificar y evaluar el grado de lesión de traumatismo cerebral. Este método se propone con miras a desarrollar una posible prueba de triaje de forma que se pueda evitar que personas con accidentes leves de cabeza se expongan a la radiación que produce el TAC de forma innecesaria (este es el caso del 93 % de los pacientes según se describe en [3]), además de crear una prueba que pueda ser accesible en aquellos sitios en los que no se disponga de las máquinas necesarias para realizar un TAC. En él se utiliza la espectroscopía ATR-FTIR (*Attenuated Total Reflectance-Fourier Transform Infrared*) para evaluar la composición de las muestras de suero sanguíneo midiendo la absorción molecular que se produce en frecuencias medias del rango del espectro de infrarrojos, concretamente $4000 - 450\,cm^{-1}$. Se realizó un examen con muestras de 222 pacientes con lesiones craneales y 87 muestras de control de pacientes sanos. Analizando la información espectral, y utilizando el método de *Principle Component Analysis* (PCA) y distintos modelos de clasificación lograron una sensibilidad[2] del 96,0 %, una especificidad[3] del 98,1 % y una exactitud balanceada[4] del 97,1 % con un modelo de clasificación de tipo PLS-DA[5] (*Partial Least Squares-Discriminant*

[2]Sensibilidad: porcentaje de todas las muestras positivas que han sido evaluadas como positivas.

[3]Especificidad: porcentaje de todas las muestras negativas que han sido evaluadas como negativas.

[4]Exactitud balanceada: media aritmética de la sensibilidad y la especificidad.

[5]PLS-DA es un método supervisado de reducción de la dimensionalidad que, a diferencia del PCA, realiza la reducción de componentes utilizando las etiquetas de los datos de salida para encontrar aquellas variables que presentan mayor varianza y correlación con el dato de salida.

Analysis).

En [5] E. Gómez González *et al.* presentan un método basado en el uso de HSI en el rango de luz visible y NIR (*Near-Infrared*) para la detección de partículas virales en muestras líquidas utilizando redes neuronales y PLS-DA. En este artículo se detalla una comparativa entre ambos métodos de clasificación, logrando unos mejores resultados con el método que utiliza redes neuronales.

En resumen, a día de hoy existen diferentes herramientas con las que se pueden realizar diversos estudios acerca de las imágenes biomédicas. Por una parte existen las tecnologías de captura y representación de la información y su formato de imagen final, entre las que se han visto las imágenes monocromáticas, RGB, MSI o HSI. Por otra parte está la tecnología de análisis de la información o elaboración del diagnóstico. Esta tecnología aúna multitud de técnicas distintas, entre ellas técnicas de aprendizaje automático que se pueden utilizar para resumir, comprimir y analizar la información capturada y, en última instancia, diseñar sistemas de predicción que puedan servir como complemento al diagnóstico. Entre estas tecnologías se ha mencionado el PCA o el PLS-DA, pero existen muchos sistemas más como la máquina de vectores soporte (SVM), los regresores lineales y las redes neuronales. Estos sistemas se detallan en la sección 2.5 de este mismo capítulo.

2.3. Microscopía óptica

La microscopía en el ámbito médico es una técnica que permite la observación y análisis visual de diferentes estructuras imposibles de observar a simple vista. Las estructuras observadas mediante técnicas de microscopía pueden ser a nivel celular y subcelular y el análisis de éstas ofrece información clave para el diagnóstico de distintas patologías así como el estudio de distintas propiedades biológicas. En la Figura 2.2 se muestra un diagrama básico de un microscopio óptico.

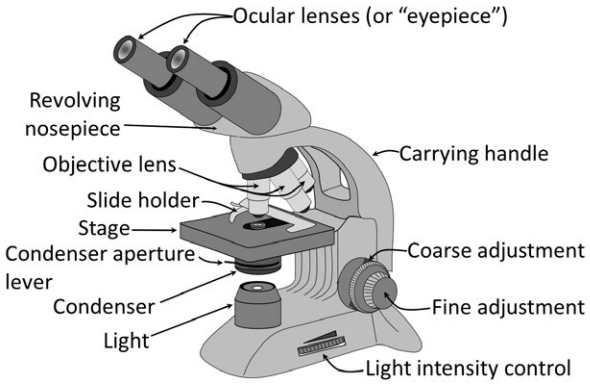


Figura 2.2. Diagrama básico de un microscopio óptico de iluminación transmitida. Imagen extraída de [6].

Existen distintos tipos de microscopía como la microscopía óptica, microscopía electrónica y la microscopía de sonda local. En esta sección se presenta de forma resumida información acerca de la primera de las tres. La microscopía óptica ilumina las muestras con luz visible proveniente de un foco. Esta luz llega hasta los oculares desde los cuales el usuario puede visualizar la muestra. Este tipo de microscopía es la más popular. No obstante, su resolución está limitada debido a la difracción de la luz, lo que causa que el aumento máximo alcanzable mediante esta tecnología sea de 1500x.

Según la manera de iluminar la muestra, la técnica de iluminación puede ser de dos tipos:

- Iluminación transmitida. En este caso, la luz incide por la parte trasera de la muestra y la atraviesa. Esta luz luego llega hasta los ojos o la cámara. Este tipo de iluminación se utiliza normalmente con muestras transparentes o semitransparentes.
- Iluminación reflejada. En este caso la luz incide por la parte superior de la muestra y la imagen captada por el sistema de visión se conforma por la luz reflejada por el objeto.

En la Figura 2.3 se muestra un ejemplo en el que aparece la misma imagen capturada mediante las dos técnicas distintas.

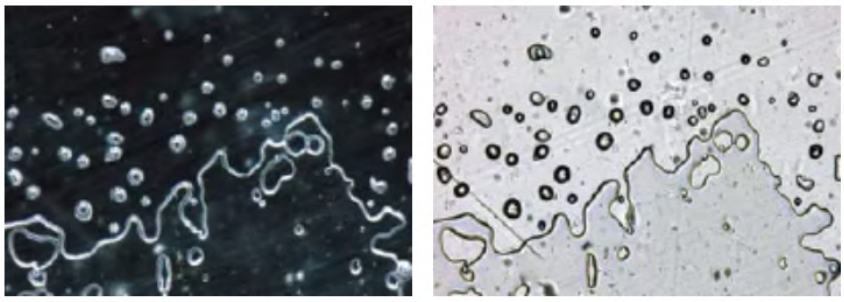


Figura 2.3. Captura de una misma imagen mediante dos técnicas de iluminación distinta. A la izquierda la muestra es capturada mediante iluminación reflejada y a la derecha mediante iluminación transmitida. Imagen extraída de [7].

Los microscopios ópticos de iluminación transmitida más utilizados y extendidos son los microscopios compuestos. Estos están formados por varios sistemas ópticos. En el caso de los microscopios binoculares, el número de sistemas ópticos es de dos, los cuales aplican ampliaciones a la imagen en pasos distintos. La primera ampliación la realiza el objetivo (ver Figura 2.2) del cual, generalmente, se dispone de múltiples opciones que permiten seleccionar entre distintos aumentos. El segundo aumento es realizado por el ocular, el cual está formado por el sistema de lentes por el que el usuario visualiza la muestra.

Existe un tipo de microscopio denominado microscopio trinocular. Este se diferencia del anterior por tener un sistema óptico adicional al que, generalmente, se une una cámara para hacer una captura digital de los datos. Este tipo de microscopio es el utilizado en el desarrollo de este PFG (ver Figura 4.2).

A través de microscopía óptica se pueden aplicar distintas técnicas de visualización de la muestra como puede ser:

- Contraste de fases: técnica que convierte las diferencias de fase de la luz recibida, debido a la textura de la muestra, en diferencias de contraste.
- Microscopía óptica de fluorescencia: técnica que consiste en excitar mediante ciertas longitudes de onda sustancias autofluorescentes o muestras a las que se le han aplicado fluorocromos[6] y visualizar la luz emitida por estas moléculas debido a su excitación.
- Microscopía óptica confocal: técnica que consiste en la eliminación de la luz desenfocada a través del uso de un orificio denominado *pinhole* el cual elimina la luz proveniente de planos distintos al plano focal. Las imágenes tomadas mediante esta técnica se hacen escaneando punto a punto la muestra, logrando de esta manera imágenes con precisión mucho mayor que las obtenidas utilizando fuentes de luz difusa.

Estas técnicas no se analizan en detalle dado que no han sido utilizadas para la elaboración de este PFG. Utilizando técnicas simples de iluminación transmitida ha sido posible la captura correcta de las muestras de suero, debido a sus propiedades transparentes y a su tamaño, el cual es suficientemente grande como para realizar la captura con un objetivo de 4x aumentos.

En resumen, la microscopía es una tecnología que permite hacer visibles estructuras de un tamaño muy reducido. Esto hace posible el análisis y estudio de las propiedades que poseen estas estructuras, lo cual puede servir para comprender mejor distintos funcionamientos biológicos.

2.4. Visión por ordenador

La visión por ordenador es un área de la informática que permite a las máquinas interpretar y comprender imágenes del mundo real. Consiste en un conjunto de técnicas para extraer, analizar y procesar imágenes mediante la utilización de distintos algoritmos y herramientas con el fin de identificar patrones y características relevantes de las imágenes, imitando la capacidad de la visión humana. Algunos ejemplos de aplicaciones son la detección y clasificación de objetos o el seguimiento del movimiento en tiempo real.

Este proyecto se ha desarrollado utilizando Python como lenguaje de programación y haciendo uso de *OpenCV* [8] como biblioteca de referencia para la elaboración de ciertas etapas de la solución diseñada. La decisión de utilizar esta biblioteca se debe a los numerosos métodos y librerías que comprende en el campo de la visión por ordenador, entre las que se incluyen algunas funcionalidades usadas en este proyecto. En las siguientes subsecciones, se citan algunas de estas funcionalidades que han sido utilizadas durante el desarrollo de este PFG.

[6]Los fluorocromos son componentes de una molécula a la que dotan de propoiedades fluorescentes. Estos componentes absorben energía en una longitud de onda específica y vuelven a emitirla en una longitud de onda mayor, es decir, con menos energía.

2.4.1. Binarización

La binarización es una operación usada comúnmente en Tratamiento Digital de la Imagen, en adelante TDI, que convierte los valores de los píxeles de una imagen en negro (0) o blanco (255 en imágenes de 8 bits) mediante la aplicación de un umbral. Esta operación se aplica sobre imágenes monocromáticas y consiste en escribir un valor de negro en aquellos píxeles cuyo valor de brillo se encuentra por debajo de dicho umbral y blanco en aquellos que se encuentren por encima. En este caso, además, se realiza una inversión del valor de los píxeles resultantes. Esta operación se ejemplifica en la Figura 2.4.

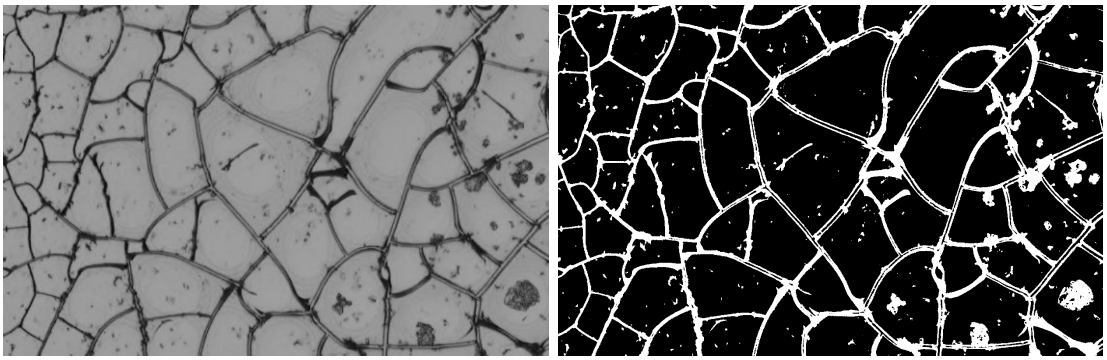


Figura 2.4. Ejemplo de binarización sobre una imagen de una muestra de suero sanguíneo.

La elección del umbral puede ser un proceso heurístico, aunque existen algoritmos que determinan un posible umbral óptimo según la imagen correspondiente. Un ejemplo de este tipo de técnicas es el algoritmo de *Otsu*. Este algoritmo se basa en encontrar el umbral óptimo mediante la minimización de la varianza de cada grupo de píxeles y la maximización de la varianza intra-grupal, en función de la siguiente fórmula (ver Figura 2.5):

$$\sigma_W^2 = c_o\sigma_o^2 + c_f\sigma_f^2 \tag{2.1}$$

donde σ_W^2 es la varianza intra-grupal, σ_o^2 es la varianza de una partición de píxeles, σ_f^2 es la varianza de la otra partición, c_o y c_f representan el número de píxeles que pertenecen a cada grupo.

Existen métodos más avanzados para aplicar binarización como la umbralización adaptativa. Este método calcula el valor resultante de cada píxel en función del valor de intensidad de su vecindad, lo cual puede ser mucho más efectivo en imágenes cuyo fondo no mantiene un valor de intensidad constante a lo largo y ancho de la imagen.

Además, existen métodos más modernos, precisos y costosos que utilizan algoritmos más complejos para establecer el valor óptimo del umbral de binarización. Un ejemplo de estos métodos es el propuesto por Li *et al.* en [9], en el cual el valor de umbralización que se aplica en cada zona de la imagen se calcula a partir de un algoritmo genético que correlaciona la forma de los contornos extraídos tras la umbralización con los vectores de características de las imágenes

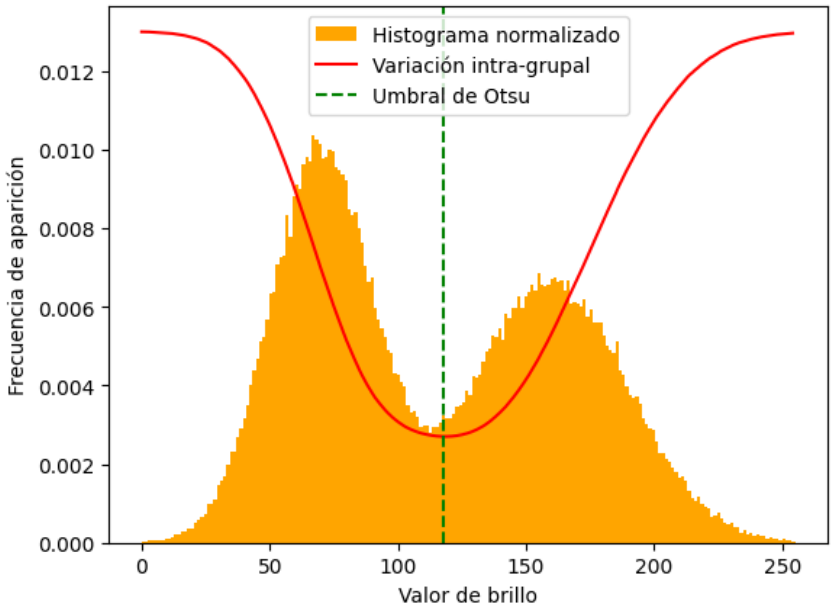


Figura 2.5. Ejemplo del funcionamiento del algoritmo de Otsu. La escala de la varianza intra-grupal ha sido reducida para poder visualizarse en contraste con el histograma normalizado de la imagen.

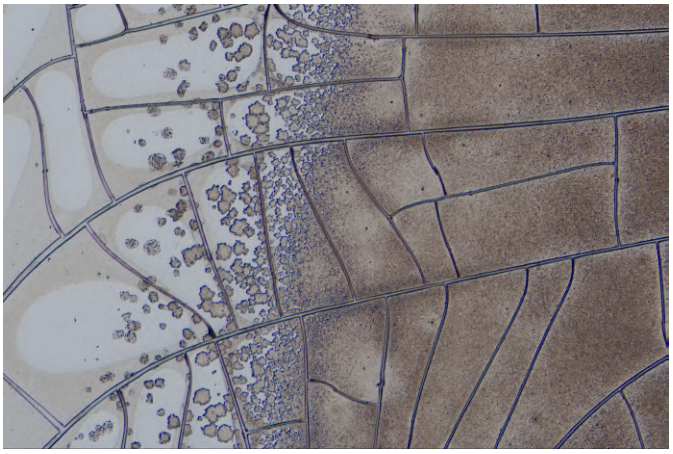
médicas, obteniendo así el umbral óptimo en cada zona de la imagen que asegura una extracción de contornos de la mayor calidad posible.

2.4.2. Operaciones morfológicas

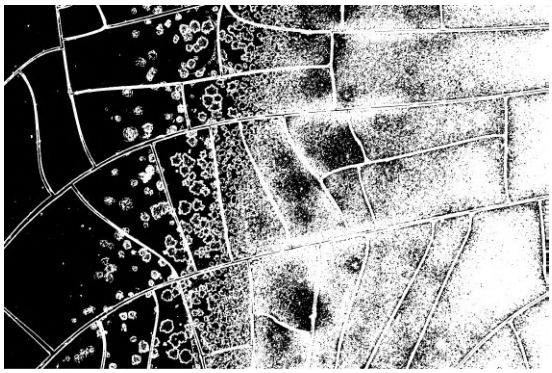
Las operaciones morfológicas se utilizan en TDI con la finalidad de eliminar ruidos o artefactos que pueden contaminar la imagen. Las operaciones morfológicas más básicas son la erosión y la dilatación. A partir de estas dos operaciones se pueden construir los procesados de apertura u *opening* y cerramiento o *closing*, tal y como se describe a continuación. Todas estas operaciones morfológicas se aplican sobre imágenes que han sido previamente binarizadas, por lo que todos sus píxeles tienen valores de 0 o 255 (en el caso de imágenes con profundidad de bit de 8) [10].

2.4.2.1. Erosión

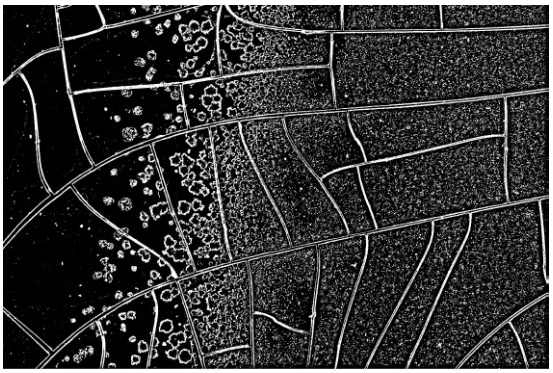
La erosión se utiliza para eliminar detalles o líneas finos de una imagen. Es por eso que una de sus principales utilidades es eliminar el ruido de tipo granulado de las imágenes. Su funcionamiento es sencillo: Para aplicar la erosión se define un núcleo o *kernel* de forma y tamaño arbitrarios, aunque lo más común suele ser utilizar núcleos cuadrados o elípticos. Este *kernel* recorre todos los píxeles de la imagen de forma que para un píxel arbitrario (x, y) de la imagen de entrada, su valor de salida será 255 exclusivamente si todos los píxeles que abarca el *kernel* tienen valor 255, incluido el propio píxel (x, y). De lo contrario el valor de salida será 0.



(a) Imagen original.



(b) Umbralización mediante Otsu.



(c) Umbralización adaptativa.

Figura 2.6. Ejemplo de distintos métodos de umbralización sobre una misma imagen de suero sanguíneo seco del dataset de capturas utilizado en este PFG.

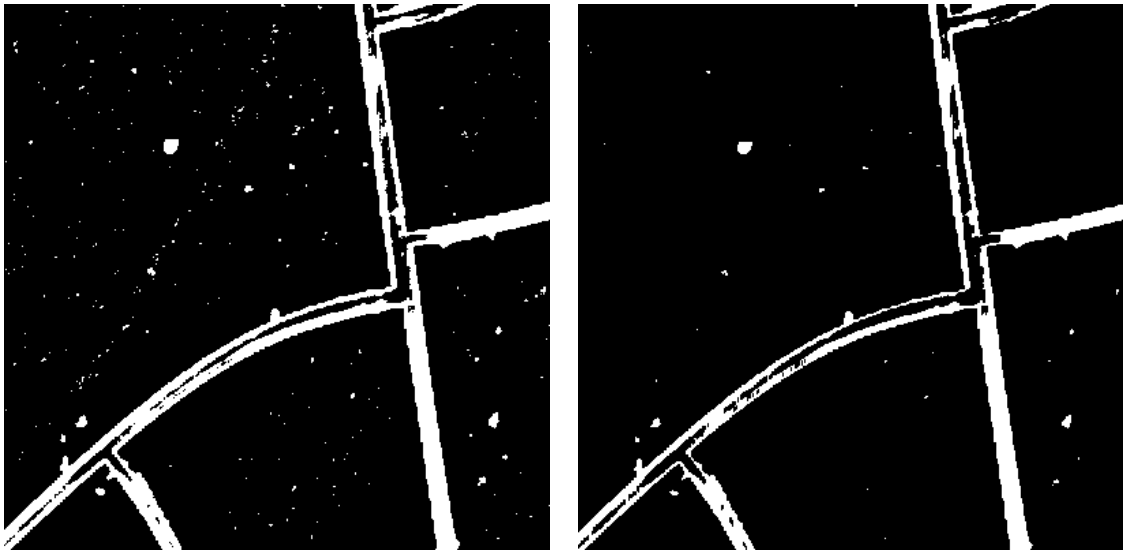
2.4.2.2. Dilatación

La dilatación es la operación complementaria a la erosión, es decir, se usa para unir segmentos que, inicialmente, deberían estar juntos y han acabado separados debido, por ejemplo, a la umbralización. Esta técnica también se puede utilizar para rellenar pequeños huecos que queden sobre el objeto ya umbralizado.

Su funcionamiento es muy similar al de la erosión. En este caso, el píxel de la imagen de entrada (x, y) tendrá un valor de salida de 255 si al menos uno de los píxeles bajo la influencia del *kernel* tiene un valor de 255. De lo contrario, el valor de este píxel será 0.

2.4.2.3. *Opening*

La operación morfológica de apertura es simplemente aplicar la operación de erosión y luego la operación de dilatación, en ese orden. Esta operación es útil para eliminar el ruido granulado que aparece sobre el fondo debido a la binarización, pero sin afectar al grosor del objeto resaltado.



(a) Imagen binarizada. (b) Imagen tras aplicar la operación de erosión.

Figura 2.7. Ejemplo de la operación morfológica de erosión. Como se puede apreciar sobre la imagen, el ruido de fondo existente en el fondo debido a la binarización desaparece, pero los contornos de las líneas también se hacen más finos.

Esta operación parte del hecho de que, al aplicar la erosión el ruido de fondo desaparecerá al tener un tamaño pequeño, pero el objeto simplemente se estrechará, por lo que al aplicar posteriormente la dilatación, el grosor del objeto volverá a su tamaño original.

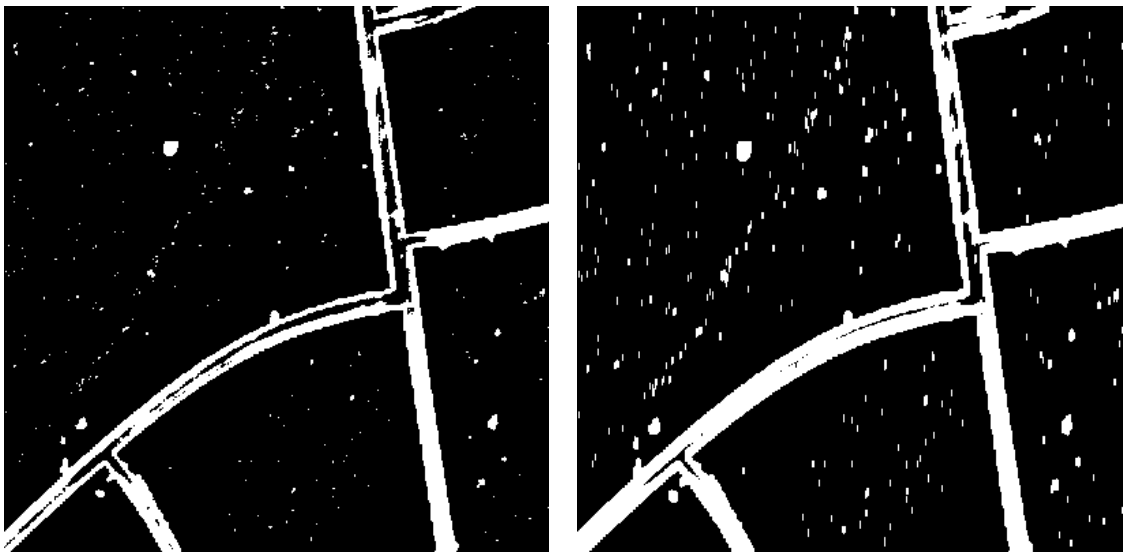
2.4.2.4. *Closing*

La operación de cerramiento consiste en aplicar la operación de apertura a la inversa, es decir, realizar la dilatación y la de erosión, en ese orden. Esta operación, al contrario que la apertura, sirve para cerrar o unir posibles huecos en el objeto resaltado. Mientras que el *opening* se aplica para eliminar píxeles que resultan blancos tras la binarización cuando pertenecen al fondo, el *closing* se utiliza para eliminar píxeles que resultan negros, es decir, tomados como fondo, que aparecen en el interior del objeto.

2.4.3. Detección de contornos

La detección de contornos es un área de la visión por ordenador que consiste en detectar bordes adyacentes entre distintos objetos presentes en la imagen. Generalmente, los algoritmos que realizan estas operaciones se basan en encontrar zonas en la imagen con grandes variaciones de intensidad, y se suelen aplicar sobre imágenes previamente binarizadas.

La estructura más básica de los detectores de contornos suelen los operadores de gradiente. Estos operadores son filtros derivativos que al convolucionarse con la imagen dan como resultado



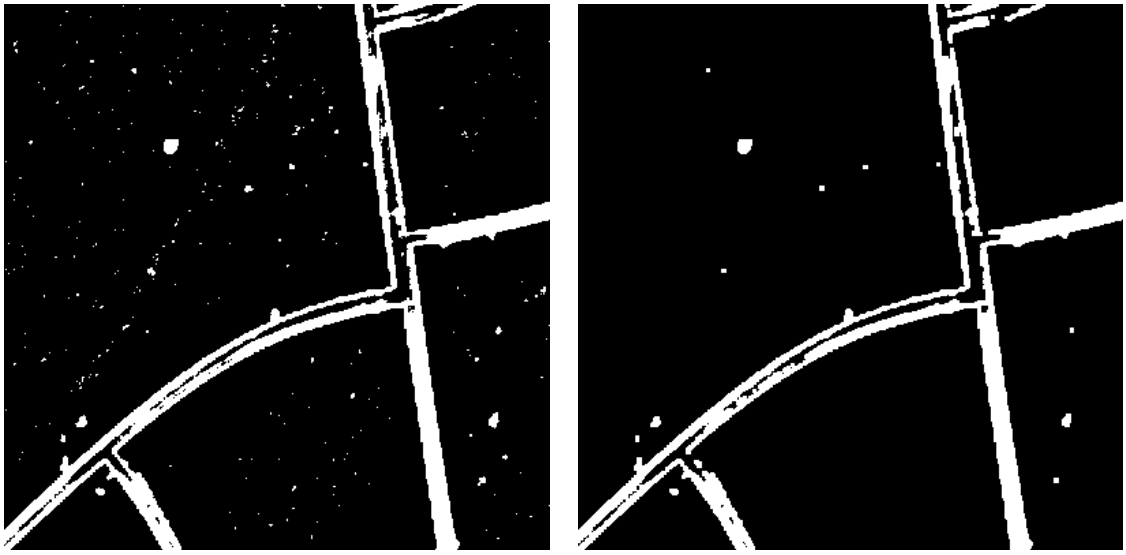
(a) Imagen binarizada. (b) Imagen tras aplicar la operación de dilatación.

Figura 2.8. Ejemplo de la operación morfológica de dilatación. Como se puede apreciar sobre la imagen, la mayoría de huecos dentro de las líneas se han cerrado, aunque el ruido de fondo también ha incrementado.

otra imagen con el nivel de variación entre píxeles adyacentes en cada zona. Posteriormente, para discriminar bordes de zonas con valor de brillo constante, se aplica un umbral que especifica a partir de qué nivel de variación se considera un píxel parte de un borde o no. Ejemplos conocidos de estos operadores son los filtros *Roberts*, *Sobel* o *Prewitt* [11]. Estos operadores básicos funcionan relativamente bien cuando la imagen en cuestión está libre de ruido, pero su desempeño es bastante malo cuando éste está presente, ya que el ruido, sobre todo en imágenes binarias, generalmente está compuesto por componentes de alta frecuencia que estos filtros resaltan.

Otro operador conocido es el Laplaciano, que a diferencia de los anteriores realiza una operación derivativa de segundo orden. Este operador es incluso más sensible al ruido que los mencionados anteriormente. Se diferencia de ellos en el sentido en que los operadores de gradiente, además de especificar variaciones grandes de brillo entre píxeles adyacentes, también especifican la dirección de cambio. El Laplaciano, en cambio, simplemente detecta zonas de la imagen en la que se presenta una variación significativa del valor de brillo, sin especificar su dirección. Este operador, al ser más sensible que los mencionados anteriormente, es capaz de detectar variaciones en zonas en las cuales los operadores de gradiente no podrían. No obstante, esta sensibilidad lo convierte en una mala elección ante imágenes ruidosas. Una solución común a este problema es realizar un filtrado de suavizado previo, como puede ser aplicar un filtro gaussiano. De esta forma se define el operador compuesto *Laplacian of Gaussian* (LoG) que es el resultado de aplicar en cascada un filtro gausiano y un operador Laplaciano.

Posteriormente, se han desarrollado algoritmos más completos de detección de contornos como *Canny* [12], el cual presenta más robustez frente al ruido y mayor precisión en cuanto a la



(a) Imagen binarizada. (b) Imagen tras aplicar la operación de apertura.

Figura 2.9. Ejemplo de la operación morfológica de la apertura. Como se puede apreciar sobre la imagen, el ruido de fondo ha desaparecido en gran medida mientras que el objeto resaltado mantiene su tamaño y estructura original.

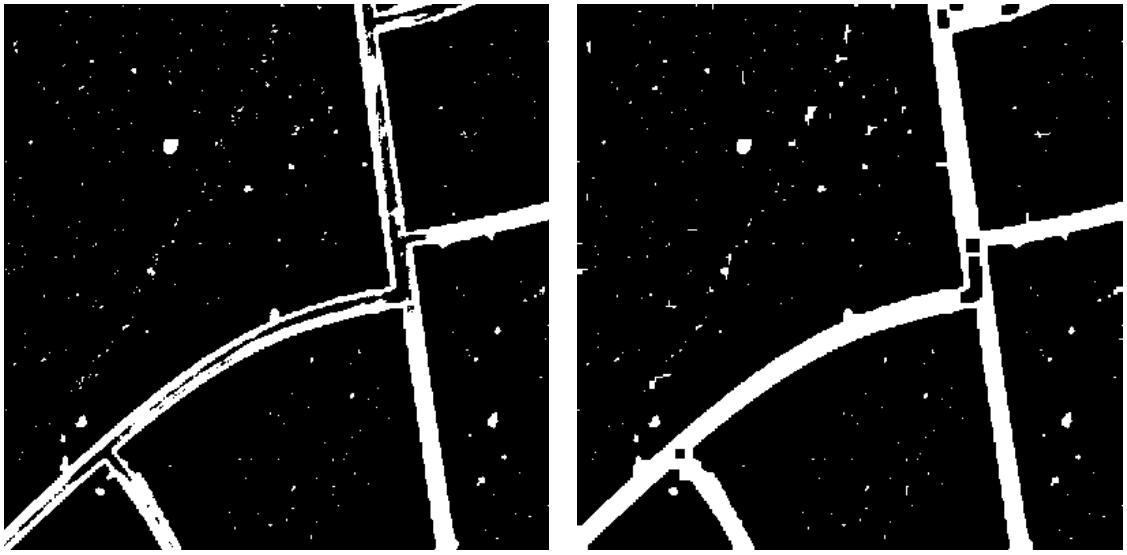
selección de los píxeles que conforman el contorno.

En el caso de este PFG, se ha hecho uso de la función `find_contours()` de OpenCV. La biblioteca OpenCV, disponible en los lenguajes de programación *C++* y *Python*, ofrece al usuario herramientas versátiles y prácticas para la detección, análisis y procesado de los contornos. En [13] se ofrece una guía detallada acerca de las herramientas implementadas por la biblioteca en este área. OpenCV ofrece al programador la capacidad de detectar los contornos y organizarlos por jerarquía. Esta jerarquía funciona de manera que para cada contorno presente en la imagen, se especifica quién es el contorno mayor que contiene a éste mismo, el primer contorno que se encuentra en el interior de éste, y los contornos que se encuentran en el mismo nivel jerárquico. Esta jerarquía puede ser de mucha utilidad para realizar filtrado de contornos en casos como, por ejemplo, en que los únicos contornos que interesan sean los exteriores.

2.4.4. Descriptores de formas

Dentro del ámbito de la visión por ordenador una función destacada es la de describir imágenes mediante datos numéricos. Estos datos pueden ser utilizados para tareas como reconocimiento o detección de objetos, clasificación o correspondencias entre imágenes con distintas perspectivas, escalado o rotaciones mediante técnicas como las de los *landmarks*[7] [14].

[7]Los *landmarks* en el ámbito de la visión por ordenador hacen referencia a puntos característicos que representan ubicaciones significativas o anatómicamente importantes. Estos puntos se utilizan comúnmente para tareas como detección facial, reconocimiento de poses, seguimiento de objetos o alineación de imágenes con distintas perspectivas, escalados, etc., entre otras. Para que dichos *landmarks* sean robustos deben cumplir características



(a) Imagen binarizada. (b) Imagen tras aplicar la operación de cerramiento.

Figura 2.10. Ejemplo de la operación morfológica *closing*. Como se puede apreciar sobre la imagen, se han unido, en gran medida, huecos que han resultado de la binarización en el interior del objeto, mientras que el ruido presente en la imagen mantiene su mismo tamaño.

Los descriptores más sencillos pueden ser simplemente valores como la intensidad media o contraste, aunque la utilidad de estos descriptores puede ser limitada a la hora de resolver tareas más complejas ya que imágenes completamente distintas pueden tener los mismos valores en estos descriptores.

Desarrollando más profundamente este campo, un ejemplo de descriptores más específicos son los momentos espaciales de una imagen. Estos descriptores caracterizan la distribución de píxeles en una imagen y proporcionan información acerca de las propiedades geométricas y estadísticas. Esta información se puede utilizar para resolver tareas más complejas como reconocimiento de patrones simples y análisis de formas. Los momentos se calculan a partir de la siguiente fórmula:

$$M_{pq} = \int_{-\infty}^{\infty} x^p y^q f(x,y) dx dy \tag{2.2}$$

donde x e y representan la posición en el eje horizontal y vertical de un píxel respectivamente, $f(x,y)$ es el valor de brillo en dicho píxel y M_{pq} es el momento para los índices pq.

Dado que las imágenes digitales son funciones discretas, los momentos de una imagen se calculan de la siguiente manera:

como ser detectables ante distintas transformaciones geométricas como traslaciones, escalado, cambios de perspectiva, etc., así como ser generalizables para tareas como, por ejemplo, la detección facial en la que se debe detectar la misma zona de la cara (por ejemplo, el ojo izquierdo) en imágenes pertenecientes a distintas personas.

$$M_{ij} = \sum_{i=0}^{x} \sum_{j=0}^{y} x^i y^j I(x, y) \tag{2.3}$$

donde i, j e $I(x, y)$ tienen su correspondencia con p, q y $f(x, y)$ en la versión continua, respectivamente. Esta fórmula se aplica sobre un objeto presente en una imagen que ha sido previamente segmentado. Por segmentado se hace referencia al acto de «separar» un objeto presente en una imagen del fondo de dicha imagen. Generalmente se realiza calculando una máscara binaria cuyos valores se multiplican punto a punto con los píxeles de una imagen, y donde las ubicaciones correspondientes a un objeto tienen un valor de 1 en dicha máscara mientras que los del fondo tienen un valor de 0.

Los momentos representan propiedades simples de un objeto en una imagen como pueden ser el área en píxeles cuadrados o su centroide[8]. Sin embargo, una problemática que presentan es que no son invariantes a traslaciones del objeto. Es por ello que a partir de estos momentos se definen los momentos centrales como:

$$\mu_{ij} = \sum_{i=0}^{x} \sum_{j=0}^{y} (x - \bar{x})^i (y - \bar{y})^j I(x, y) \tag{2.4}$$

siendo $\bar{x}$ e $\bar{y}$ las coordenadas del centroide de la figura que se está analizando. Aplicando este cambio se obtienen unos descriptores invariantes a traslaciones del objeto. A partir de estos momentos, además, se pueden obtener atributos de la figura como la excentricidad o la orientación.

Modificando las fórmulas de los momentos descritos en 2.2, 2.3 y 2.4, se pueden obtener momentos invariantes al escalado con la siguiente ecuación:

$$\eta_{ij} = \frac{\mu_{ij}}{\mu_{00}^{1+\frac{i+j}{2}}}, i + j \geq 2 \tag{2.5}$$

y operando todadvía más con estos últimos momentos se obtienen los *7 momentos de Hu* [15], los cuales se caracterizan por tener todas las propiedades antes mencionadas además de ser invariantes a la rotación. Los tres primeros momentos de Hu se calculan de la siguiente forma:

$$I_1 = \eta_{20} + \eta_{02} \tag{2.6}$$

$$I_2 = (\eta_{20} + \eta_{02})^2 + 4\eta_{11}^2 \tag{2.7}$$

$$I_3 = (\eta_{30} - 3\eta_{12})^2 + (3\eta_{21} - \eta_{03})^2 \tag{2.8}$$

Este descriptor de formas ha sido ampliamente utilizado en el ámbito de la visión por ordenador para tareas como clasificar o detectar formas y patrones. En [16] se muestra la utilización

[8]El centroide se calcula a partir de la distribución de los píxeles que conforman el objeto en una imagen y representa su centro geométrico o centro de masa.

de este descriptor para la identificación y clasificación automática de hojas de árboles, en [17] se muestra una aplicación de este descriptor para la detección de imágenes manipuladas o fraudulentas, y en [18] se muestra el uso de los momentos de Hu para la detección de patologías cerebrales.

Como resultado a extraer los descriptores de un determinado objeto en una imagen, se obtiene un vector de características o *features* el cual se puede utilizar para resolver tareas como determinar qué clase de objeto es o cuánto se parece a otro objeto presente en otra imagen. Comparando distintos vectores de características se puede calcular la similaridad[9] entre los objetos de los que se han extraído dichos vectores. Una manera de calcular esta similaridad es definiendo una función distancia, la cual recibe ambos vectores y devuelve un valor que evalúa cuánto se parece o se diferencian ambas instancias en la característica que se está mesurando. Esta función distancia puede ser simplemente la distancia euclidiana que existe entre ambos vectores. Otro ejemplo de función distancia es el caso de la distancia coseno, cuyo uso es bastante extendido. Esta función evalúa la similaridad entre dos instancias de la siguiente forma:

$$sim(a, b) = cos(\theta) = \frac{\vec{A}\vec{B}}{|\vec{A}||\vec{B}|} \tag{2.9}$$

donde A y B son los vectores de *features* de ambas instancias y θ es el ángulo entre ambos vectores. Esta función devuelve valores cercanos a 1 cuando ambos vectores son similares y valores cercanos a 0 cuando son ortogonales, lo que quiere decir que no están relacionados entre sí. Para resolver tareas más complejas de clasificación o de cómputo de la similaridad se puede recurrir a técnicas de aprendizaje automático mediante modelos que se entrenan utilizando dichos vectores de características. Algunas de estas técnicas se describen en la sección 2.5.

Existen muchos más descriptores de imágenes y formas, invariantes a diferentes transformaciones, para identificar patrones y formas presentes en imágenes. En [15], además de definir los mencionados *7 momentos de Hu* se presenta un estudio detallado sobre distintas propiedades de dichos momentos, así como su aplicación ante distintas transformaciones lineales y geométricas. Además, se presenta un caso de uso de un modelo de detección de patrones usando este descriptor.

Otro descriptor con propiedades invariantes muy utilizadas en el ámbito de la visión por ordenador son los *momentos de Zernike* [19]. Este descriptor, aunque no represente propiedades parecidas a las que representan los *7 momentos de Hu* se puede utilizar de forma similar que éste último. En [20] se muestra el uso de los *momentos de Zernike* para el reconocimiento de caras. En [21] y [22] se muestran ejemplos del uso de ambos descriptores al mismo tiempo. En el segundo de ellos se utilizan ambos descriptores como algoritmos de extracción de vectores de características junto con un modelo basado en una máquina de vectores soporte o *Supervised Vector Machine*

[9] La similaridad, en el campo de la visión por ordenador y el aprendizaje automático, se define como una métrica que evalúa cuánto se parecen dos instancias distintas en una determinada característica que se está evaluando.

(SVM), la cual se presenta en el apartado 2.5 de este capítulo, para la clasificación de imágenes médicas.

Existen otros descriptores como los descriptores de Fourier [23] [24], el *Curvature Scale Space* (CSS) [25] o el *Histogram of Gradients* (HoG) [26], entre otros. Estos descriptores han sido estudiados y analizados para valorar su utilización en este PFG, aunque al final se ha optado por limitarse a los *momentos de Hu* y los *momentos de Zernike* debido a que, incluyendo nuevos descriptores, la complejidad del sistema aumenta, así como la gestión de la información que se obtendría por muestra. Además, tras un análisis exhaustivo de distintos artículos académicos relacionados, entre los que se encuentran los referenciados anteriormente, se ha observado que utilizando estos dos descriptores es posible obtener resultados positivos en tareas relacionadas con la clasificación de patrones.

2.5. Técnicas de aprendizaje automático

El aprendizaje automático o *Machine Learning* (ML) es una rama de la inteligencia artificial que hace referencia a aquellas técnicas y algoritmos que permiten a la máquina aprender y mejorar a partir de los datos proporcionados, sin la necesidad de programar explícitamente el comportamiento de un determinado sistema. Estas técnicas se pueden utilizar para resolver problemas de regresión y clasificación, detectar patrones o tomar decisiones basadas en los datos proporcionados. En el área de las bioimágenes, estas técnicas están siendo estudiadas profundamente, ya que su uso puede mejorar el diagnóstico de una determinada enfermedad, ayudar a identificar y separar tejidos sanos de tejidos cancerosos o detectar correlaciones entre varias características que no se habían descubierto previamente, ya sea debido a la complejidad para hallar estas correlaciones o a que los algoritmos de computación clásicos no son lo suficientemente eficientes como para detectarlas.

Aunque la definición formal de ML incluye cualquier tipo de algoritmo capaz de «aprender por sí mismo», incluyendo en esta definición los modelos de *Deep Learning*, en esta sección se va a hacer una distinción entre algoritmos de ML y modelos de *Deep Learning*, los cuales se analizarán en la subsección 2.5.2. Se van a tomar como algoritmos de ML aquellos algoritmos que a priori tienen una carga computacional y unas arquitecturas más sencillas que los algoritmos de *Deep Learning* los cuales harán referencia principalmente a las redes neuronales. Esta separación se realiza porque facilita la redacción y estructura para definir el campo del aprendizaje automático.

2.5.1. Técnicas y algoritmos de *Machine Learning*

Existen diferentes técnicas de ML según el problema que se desea resolver. Estos problemas pueden ser de clasificación, regresión, clusterización, reducción de la dimensionalidad, etc. Según el tipo de problema se deberá emplear un tipo de modelo u otro. Además, también se pueden

clasificar estas técnicas en algoritmos supervisados y no supervisados. Los algoritmos supervisados son aquellos los cuales requieren disponer de un dataset previo de cuyos datos se conoce previamente el valor de salida real o *ground truth*. Con estos datos se puede realizar un entrenamiento del modelo de forma que éste puede aprender relaciones entre los datos de entrada y las salidas y, posteriormente, utilizarse con datos que representan instancias del mismo valor semántico que los datos de entrenamiento, pero para los que no se conoce previamente el valor de salida. Por otro lado, están los algoritmos no supervisados. Estos son aquellos algoritmos que aprenden a identificar patrones o estructuras en los datos sin la necesidad de que se conozca de un valor de salida previamente. Estos algoritmos se usan para agrupar datos similares, reducir la dimensionalidad o hallar relaciones ocultas entre los propios datos, y son una herramienta muy útil cuando se dispone de conjunto de datos sobre los que no se conoce una distribución previa.

A continuación, en los siguientes subapartados se detallan algunas de estas técnicas, clasificadas según el problema que resuelven:

2.5.1.1. Algoritmos de regresión

Por algoritmos de regresión se hace referencia a aquellos modelos cuyo valor de salida es un valor numérico. Algunos ejemplos de problemas en los que se pueden utilizar algoritmos de regresión son: predicción de precio o ventas o modelar el crecimiento económico, y en el campo de la medicina: estimación de la presión arterial o predicción de la dosis óptima de medicamentos.

Uno de los modelos de regresión más utilizados es el modelo de regresión lineal. Este modelo recibe como entrada un vector de características y a su salida devuelve un valor numérico que se ha calculado como la combinación lineal de los distintos elementos del vector de características multiplicados por un vector de pesos $\bar{w}$. La fórmula que describe este comportamiento es la siguiente:

$$y = \sum \bar{x}\bar{w} + w_o \tag{2.10}$$

donde $\bar{x}$ es el vector de características o *features* y w_o se define como el sesgo o *bias*. Si se considera $\bar{\mathbf{x}}$ como $[1, \bar{x}_1, \bar{x}_2, ...]$ y $\bar{\mathbf{w}}$ como $[1, \bar{w}_1, \bar{w}_2, ...]$, la ecuación se reduce a la siguiente:

$$y = \bar{\mathbf{w}}^T \bar{\mathbf{x}} \tag{2.11}$$

El vector de pesos se calcula a partir de minimizar una función de pérdida que, para este tipo de modelos, generalmente es el *Mean Squared Error*. Esta función de pérdida se calcula de la siguiente forma:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - y_{true,i})^2 \tag{2.12}$$

donde y_i es el valor el valor predicho por el modelo e $y_{true,i}$ el valor real correspondiente al mismo dato de entrenamiento.

Con esta definición y esta función de pérdida, el modelo tiene solución directa, conocida como *least-squares*:

$$\bar{\mathbf{w}}^* = (\mathbf{X}^T\mathbf{X})^{-1}\mathbf{X}^T\bar{y}_{true} \tag{2.13}$$

donde:

- $\bar{\mathbf{w}}^*$ es el vector de pesos del modelo final.
- $\mathbf{X}^T$ es la matriz de datos de entrenamiento. Cada fila i es una instancia $\bar{\mathbf{x}} = [1, \bar{x}_1, \bar{x}_2, ...]$
- $\bar{y}_{true}$ es el vector de valores reales correspondiente a cada fila de la matriz $\mathbf{X}^T$

Este modelo es útil ante datos cuyo valor de salida tiene una relación lineal con los datos de entrada, pero cuando la relación es no lineal pierde su efectividad. Una técnica común para abordar este problema de la «no linealidad» es hacer lo que se conoce como una extensión polinómica. Al aplicar esta técnica, el vector de características se extiende añadiendo todas las posibles combinaciones de multiplicaciones entre los valores del vector y entre sí, hasta un determinado grado. Por ejemplo, dado un vector de características (x_1, x_2), su extensión polinomial de grado dos sería $(1, x_1, x_2, x_1x_2, x_1^2, x_2^2)$. De esta manera se pueden encontrar relaciones no lineales para resolver un determinado problema.

Hay numerosas técnicas de optimización de los modelos de este tipo como las que se usan en los modelos de tipo *Ridge* [27] o *Lasso* [28] pero no se va a entrar en detalle en estas técnicas en este PFG debido a que estos modelos no han sido utilizados en el desarrollo del proyecto.

En [29] y [30] se muestran trabajos relacionados con el campo de la medicina en los que se usan modelos de este tipo. En el primero de ellos se utilizan modelos de regresión para la predicción de variables cognitivas a partir de imágenes médicas, y en el segundo se utilizan para la detección de anomalías en sensores remotos.

2.5.1.2. Algoritmos de clasificación

Los algoritmos de clasificación son aquellos algoritmos cuyo valor de salida no es un dato numérico sino un dato categórico, como puede ser una clase o una etiqueta. Algunos ejemplos de problemas que pueden solucionar este tipo de modelos son: diferenciar tejido sano de tejido

no sano, diagnóstico de arritmias o clasificación de imágenes médicas entre tumores benignos de tumores malignos.

Un modelo de este tipo es el *K-Nearest Neighbors Classifier*. Para un problema con M clases distintas, el funcionamiento de este modelo se describe mediante la siguiente fórmula:

$$y = \arg\max_{y \in \{0,1,\ldots,M-1\}} h(\bar{x}, y) \tag{2.14}$$

donde la función $h(\bar{x}, y)$ se define de la siguiente forma:

$$h(\bar{x}, y) = \frac{1}{K} \sum_{k \in \mathcal{S}_K(\bar{x})} \mathbb{I}[y^{(k)} == y], \tag{2.15}$$

En resumen, la clase de un nuevo dato de entrada será la clase mayoritaria de los K datos más cercanos a dicho valor nuevo en el espacio dimensional de los vectores de características.

Otro modelo muy extendido en la tarea de clasificación es la máquina de soporte vectorial o *Support Vector Machine* (SVM). El objetivo de estos modelos es encontrar el hiperplano que, en el espacio dimensional de los vectores de características, sea capaz de separar de manera óptima las distintas clases. El proceso para hallar este hiperplano es encontrar los *vectores de soporte*, que corresponden con aquellos vectores de datos más cercanos al hiperplano. El algoritmo de entrenamiento trata de maximizar la distancia entre dichos vectores de soporte y el hiperplano para aumentar la generalización del modelo todo lo posible. En la Figura 2.11 se puede apreciar el umbral de clasificación obtenido mediante un SVM para clasificar dos clases de un dataset de ejemplo. Además, se pueden apreciar los máximos márgenes que se han encontrado entre el hiperplano y los vectores de soporte.

Este modelo, en su forma más sencilla, tiene una problemática, y es que no es capaz de encontrar un hiperplano óptimo en un dataset cuya distribución de datos no permita una separación lineal. Un ejemplo de esto se visualiza en la Figura 2.12.

Para solucionar este problema existen las técnicas del *kernel*. Estas técnicas transforman los vectores de características a un espacio dimensional nuevo en el cual sí es posible encontrar una solución lineal. Un ejemplo de estos *kernels* es el *kernel* polinómico. Éste realiza una extensión polinómica de los vectores de características, de la misma forma que como se explicó en el apartado de regresores lineales, de manera que en este nuevo espacio dimensional se puede encontrar el hiperplano que separe las distintas clases de manera eficaz. Para ejemplificar esto, se aplica dicho *kernel* a los datos de la Figura 2.12 y se entrena el SVM con los datos en este nuevo espacio. En la Figura 2.13 se puede apreciar cómo se ha solventado el problema de la no linealidad y se ha encontrado un margen capaz de separar las distintas clases de forma eficaz.

Otros ejemplos de *kernel* son el gaussiano o *radial basis function* y el *kernel* sigmoide. El *kernel* gaussiano o *radial basis function* es uno de los más utilizados. Su desempeño en el entre-

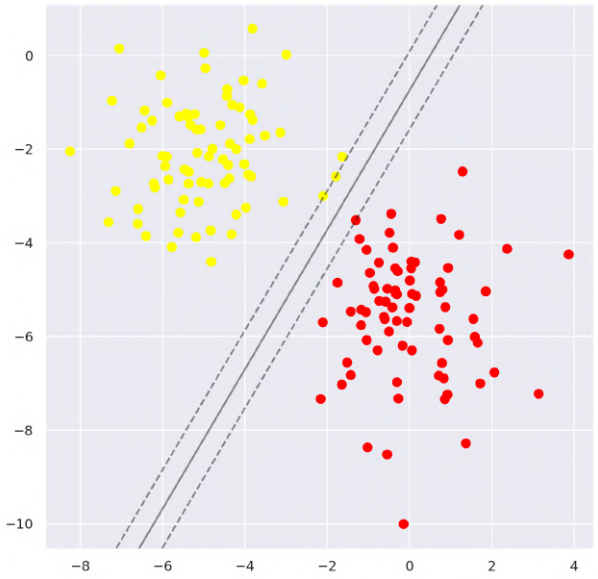


Figura 2.11. Utilización de un SVM de *kernel* lineal con un dataset de datos de ejemplo simple. Los vectores de soporte identificados en este caso están marcados con cruces.

namiento de SVMs sobresale sobre otros y esto, en parte se debe a que en su definición existe un hiperparámetro[10] llamado γ el cual permite ajustar el nivel de «flexibilidad» del entrenamiento. Un valor alto de γ ajustará el umbral de forma muy precisa a los datos de entrenamiento, pudiendo causar *overfitting*[11], mientras que un valor bajo de γ permitirá un margen de error mayor y, por ende, mayor «flexibilidad». Cuanto más bajo es γ, más se parecerá la solución obtenida por el SVM a la obtenida con un *kernel* lineal. En [31] se analiza más profundamente sobre los métodos de *kernel*, así como sus aplicaciones en los modelos SVM.

En [22] se muestra un ejemplo de clasificación de imágenes entrenando un SVM con descriptores de *Hu* y de *Zernike*. En [32] se evalúa la clasificación de distintas imágenes biomédicas mediante la utilización de un SVM y, además, se compara su eficiencia con la obtenida tras el entrenamiento de una red neuronal.

Estrategias de usos de SVM para problemas multiclase

Los SVM por defecto solo son capaces de solucionar problemas de clasificación de tipo binarios. Si se quieren utilizar en la resolución de problemas multiclase, es necesario realizar adaptaciones de la implementación de estos modelos. A continuación, se exponen dos estrategias para

[10] Por hiperparámetro se hace referencia a parámetros que intervienen en el entrenamiento de modelos y que, generalmente, no están definidos en un primer momento y su valor se suele obtener de manera heurística analizando el desempeño del modelo para distintos valores de estos hiperparámetros.

[11] El *overfitting* es un fenómeno relacionado con el uso y el entrenamiento de distintos modelos de ML y *Deep Learning* el cual se identifica por lograr una precisión muy elevada del modelo con respecto a los datos de entrenamiento, pero baja con respecto a los de prueba, incurriendo en una mala generalización del modelo. Generalmente, este fenómeno sucede por no disponer de datos suficientes, una mala regularización del modelo o un entrenamiento excesivamente largo, entre otros.

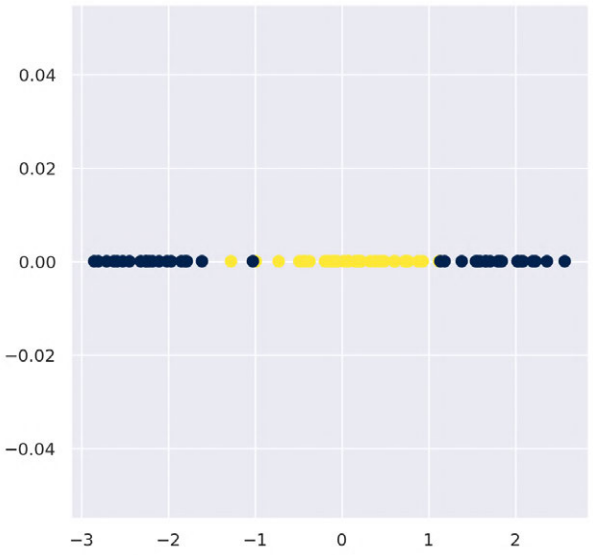


Figura 2.12. Ejemplo de un dataset cuya distribución de los datos no permite una separación óptima lineal.

utilizar SVMs en la resolución de problemas multiclase:

Estrategia *One vs. One* Esta estrategia consiste en entrenar un SVM por cada par de clases existentes en el dataset. Cada SVM se entrena con una partición de los datos que únicamente corresponden con instancias de las dos clases para las que se entrena dicho modelo, es decir, para cada par de clases A y B se toman todos los datos correspondientes a estas clases y se entrena a un SVM específico. Una vez se ha entrenado un SVM para cada par de clases, la predicción final se realizará tomando la clase más popular elegida por todos los modelos. Esta estrategia implica entrenar $\frac{k(k-1)}{2}$ SVMs, siendo k el número de clases en total.

Estrategia *One vs. All* Esta estrategia consiste en entrenar individualmente un SVM por cada clase. El objetivo de esta estrategia es que cada SVM aprenda a diferenciar una clase de todas las demás. Cada SVM se entrena con el dataset completo de entrenamiento pero modificando las etiquetas de los datos de manera que la clase objetivo tiene una etiqueta positiva y el resto de clases tienen una etiqueta negativa. La predicción final se realiza devolviendo el valor de salida del modelo con mayor margen o confianza. Esta estrategia implica entrenar k clasificadores.

Ambas estrategias consisten en convertir un problema multiclase en múltiples problemas binarios. La segunda estrategia requiere menos SVMs en total que la primera, por lo que generalmente es la preferida cuando el número de clases es elevado. Sin embargo, esta estrategia también tiene desventajas ya que puede derivar en muestras «inclasificables». Esto sucede cuando todos los SVMs clasifican la muestra como negativa, en cuyo caso no se puede asignar ninguna clase de forma fiable a dicha muestra. En [33] se presenta el uso de SVMs utilizando estrategias multiclase para diagnóstico médico.

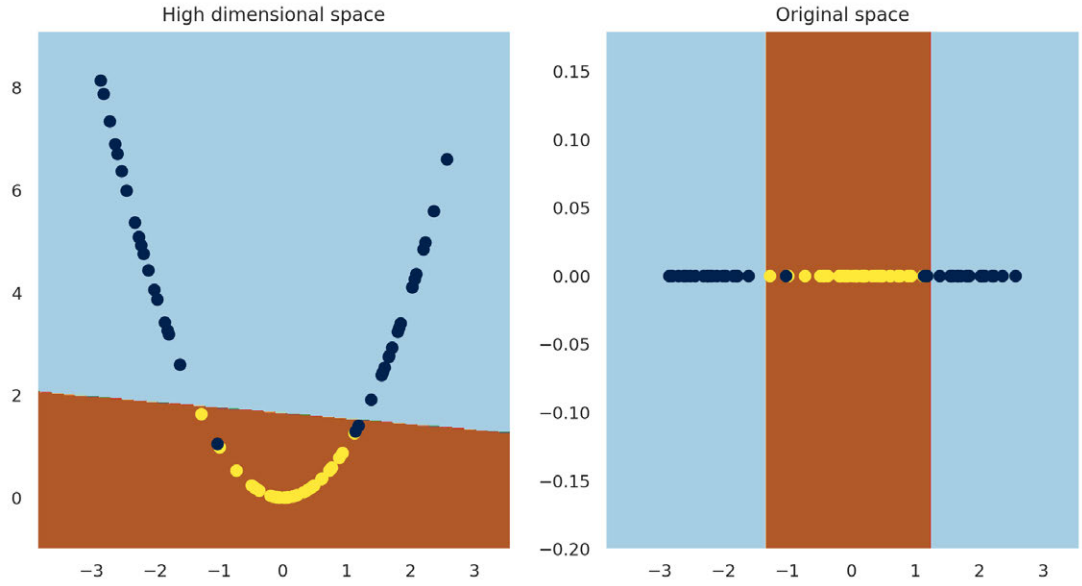


Figura 2.13. Ejemplo de cómo utilizando un SVM con *kernel* polinómico se ha logrado encontrar una separación eficiente entre los datos de distintas clases.

2.5.1.3. Algoritmos de clusterización

Los algoritmos de clusterización son algoritmos no supervisados cuyo objetivo es agrupar los datos de entrada al modelo en diferentes grupos o *clusters*. Estos modelos buscan patrones entre los datos de manera que puede agrupar en distintos conjuntos datos que tengan alguna característica similar. Los algoritmos de clusterización son útiles cuando se dispone de un gran número de datos no etiquetados para los cuales es posible hallar agrupaciones semánticas. Algunos ejemplos de problemas que pueden solucionar estos algoritmos son: agrupamiento de documentos o textos, agrupamiento de imágenes, detección de anomalías o compresión de datos. Estas agrupaciones se realizan en función de un criterio que, generalmente, son la distancia o similitud entre vectores, los cuales se definieron en el apartado 4.3.1.

Posiblemente, el algoritmo más extendido dentro de los modelos de este tipo sea el *K-Means*. El *K-Means* es un algoritmo iterativo cuyo objetivo es encontrar los K *clusters* que separan de forma más eficiente los datos del dataset. K es un hiperparámetro que, generalmente, se debe validar evaluando el desempeño del modelo para distintos valores.

K-Means es un algoritmo que funciona de la siguiente manera:

1. En el espacio dimensional de los vectores de características se inicializan de forma aleatoria K puntos, que posteriormente serán los centroides de los K clusters.

2. Se «asigna» cada vector del dataset a su centroide más cercano. Esto se hace generalmente utilizando la distancia euclídea. Tras esta asignación, en el espacio dimensional de los vectores de características se tienen K *clusters* formados por los K centroides y todos sus

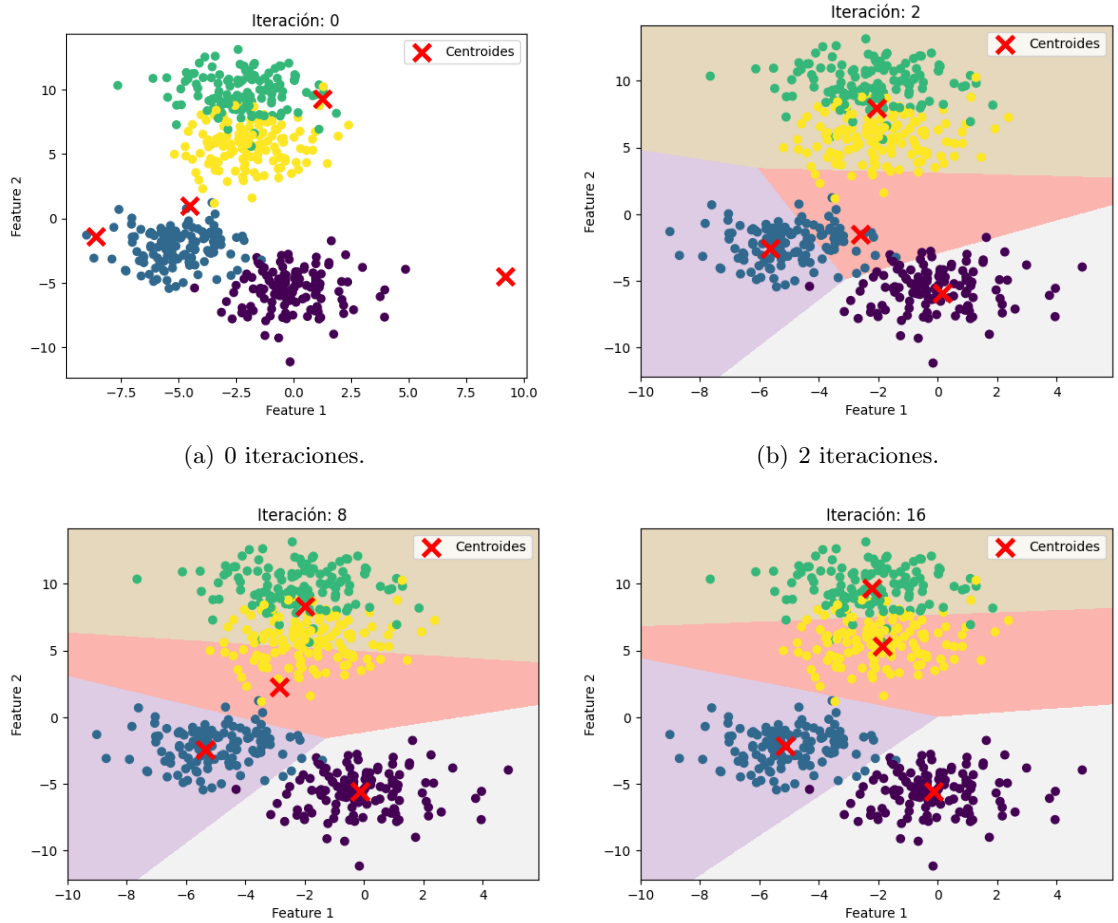
puntos asignados.

3. Se calcula el punto medio de cada uno de los K *clusters* y estos puntos se redefinen como los nuevos centroides de los K *clusters*. Tras esto, se vuelve a realizar una asignación de los datos a sus centroides más cercanos.

4. Estos pasos se repiten hasta llegar al número de iteraciones especificado previamente por el ingeniero o hasta que se cumpla una determinada condición, que debe ser especificada también previamente.

En la figura 2.14 se muestra un ejemplo del funcionamiento de este algoritmo a lo largo de las distintas iteraciones.

En [34] se presenta un sistema de segmentación de imágenes basado en el algoritmo de *K-Means* y en [35] se muestra una extensión del método tradicional de *K-Means* para aplicaciones de clusterizado en datos médicos.



(a) 0 iteraciones.

(b) 2 iteraciones.

(c) 8 iteraciones.

(d) 16 iteraciones.

Figura 2.14. Evolución de las agrupaciones realizadas y la posición de los centroides calculados por el algoritmo *K-Means* en sus distintas iteraciones y con un dataset de prueba.

2.5.1.4. Consideraciones sobre la aplicación de modelos de ML

Los modelos que se han mencionado en esta sección son unos pocos de todos los modelos que existen. Se ha decidido incluir concretamente estos porque sirven para introducir las soluciones propuestas en este PFG. A continuación se exponen algunos puntos importantes a tener en cuenta cuando se desarrollen aplicaciones basadas en ML:

- Identificar de forma precisa el problema que se desea solucionar y cómo se puede solucionar. Este primer paso es esencial para escoger eficazmente el modelo que se aplicará para hallar una solución. Por ejemplo, tendría sentido responder a preguntas como: ¿es un problema de agrupación de datos, de clasificación o de regresión?
- La importancia del dataset de entrenamiento. Para asegurar que un modelo de ML sea capaz de resolver un determinado problema es importante revisar la calidad de los datos de entrenamiento. Sin unos datos de entrenamiento de calidad, será muy difícil que el modelo converja a una solución eficiente. El dataset debe asegurar que se cumplen algunas condiciones como: estar en la medida de lo posible libre de artefactos, estar balanceado, lo que quiere decir que existan números parecidos de instancias de todas las clases en el dataset; debe estar libre de sesgos, ya sean introducidos por las personas que recopilaron información para crear el dataset como inducidos por la máquina; entre otros.
- Hacer un análisis exhaustivo de los datos de entrenamiento. Revisar y visualizar los datos puede ayudar también a la elección del modelo o de sus parámetros. Algunas características que se pueden revisar son:
    - Tendencias de los datos: ¿se pueden clasificar correctamente estos datos con un clasificador lineal o necesitan algún tipo de transformación?
    - Variabilidad de los elementos de los vectores de características: ¿hay elementos para los que se ve claramente que tienen más peso en el valor final que otros?
    - ¿Qué elementos son continuos y qué elementos son categóricos?
    - ¿Las características están en un mismo «nivel semántico» como para entrenar el modelo con todas a la vez? ¿O interesa separar por características en dos datasets para entrenar dos modelos distintos en paralelo?

2.5.2. Deep learning

Por *Deep Learning* se hace referencia a aquellos algoritmos de aprendizaje automático cuya arquitectura está basada en las redes neuronales. Las redes neuronales desarrollan relaciones matemáticas complejas entre unos valores de entrada y unos valores de salida. Están compuestas por distintas capas de nodos (o neuronas) sobre las cuales se van proyectando de distintas maneras los datos de entrada. De esta forma, cuando se introducen datos de entrada en una red neuronal,

cada capa representa una nueva transformación de los datos de un espacio N-dimensional a uno M-dimensional. Estas proyecciones extraen de forma jerárquica características sobre los datos de entrada, de forma que la información mostrada a la entrada o en las primeras capas es relativamente simple y, cuanto más cerca se está de las últimas capas, más abstracta se convierte esta información, pudiendo representar una clase, un valor o una decisión tomada por la red. En la Figura 2.15 se puede ver un ejemplo de red neuronal.

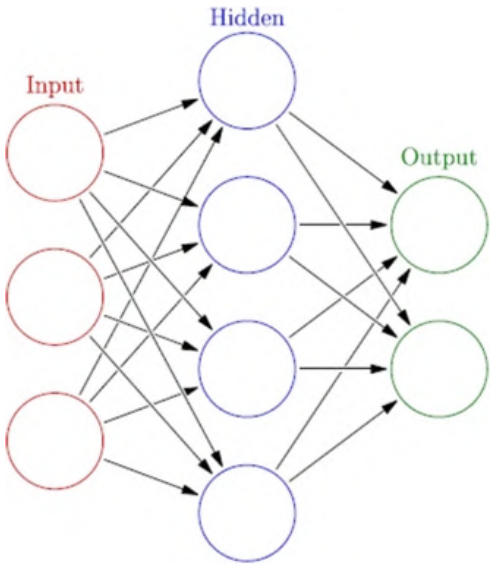


Figura 2.15. Ejemplo de la arquitectura de una red neuronal simple con una capa oculta. Imagen extraída de [36].

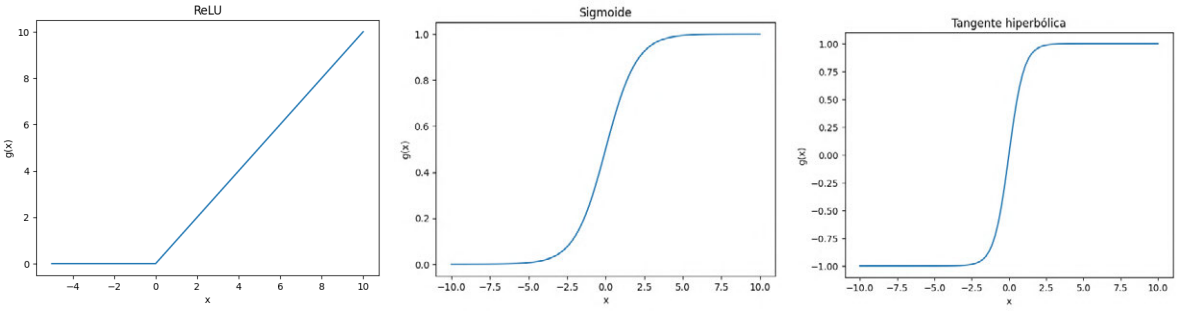
La salida de cada neurona se calcula como la combinación lineal de los datos de entrada multiplicados por sus correspondientes pesos más un sesgo, y todo ello transformado por una función no lineal denominada función de activación. Las capas formadas por neuronas que realizan este tipo de operaciones se denominan *Fully Connected*, y la operación que realiza cada neurona es la siguiente:

$$y = g(\sum_{i} x_i w_i + w_o) \tag{2.16}$$

donde x_i representa cada entrada del vector de datos entrante a la neurona, w_i el peso correspondiente a cada entrada de datos, w_o el sesgo y $g()$ representa dicha función de activación, la cual es importante para permitir al modelo converger mediante soluciones no lineales, ya que sin estas funciones de activación, la salida de la red no sería más que una combinación lineal sumamente compleja de los valores de entrada. En la Figura 2.16 se muestran algunas de las funciones de activación más utilizadas.

La función *Rectified Linear Unit* o *ReLU*, mostrada en 2.16 a) es una de las funciones más comunes utilizadas en las capas intermedias de una red neuronal. Se calcula de la siguiente forma:

$$y = max(0, x) \tag{2.17}$$



(a) Función de activación *ReLU*. (b) Función de activación *Sigmoide*. (c) Función de activación *tangente hiperbólica*.

Figura 2.16. Ejemplos de algunas de las funciones de activación más utilizadas para el desarrollo de redes neuronales.

Algunas de sus ventajas radican en que su carga computacional es muy baja y que, además, ayuda a evitar el problema del desvanecimiento de gradiente[12], ya que la función *ReLU* tiene una derivada constante de 1 cuando su salida es positiva. Esto ayuda a que el gradiente no decrezca conforme se aproxima a las primeras capas. Otra ventaja radica en que aumenta la diversidad en las activaciones de las capas. Cuando la entrada de la función es cero o menor que cero, la neurona no se activa, «obligando» de alguna manera a que las demás neuronas de la misma capa sean las que se activen para representar ese dato de entrada. Esto ayuda a que la red generalice mejor y de forma más eficiente.

Las funciones sigmoide y tangente hiperbólica, también mostradas en la Figura 2.16 son útiles para representar probabilidades, puesto que son funciones cuyas salidas están entre 0 y 1 para la sigmoide y -1 y 1 para la tangente hiperbólica. Estas funciones se suelen poner en la última capa de la red neuronal de manera que se puede evaluar la salida como la probabilidad de que la entrada sea un objeto concreto o la semejanza que tiene con dicho objeto concreto, por ejemplo. En un problema de clasificación binario estas funciones suelen ser las ideales para la última capa de la red. En el caso de problemas de clasificación multiclase, la opción más extendida es la función *softmax*, cuya fórmula se describe a continuación:

$$s(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{k} e^{x_j}} \tag{2.18}$$

siendo k el número de clases y x_i las salidas numéricas de la última capa de la red, la cual tendrá tantas neuronas como clases se está evaluando. La función *softmax* devuelve un vector del mismo tamaño que el vector de entradas, donde se muestran las probabilidades de acierto de cada una

[12]El desvanecimiento de gradiente es un problema que se da durante el entremiento del modelo y, concretamente, durante el paso del *backpropagation*, el cual consiste en propagar el gradiente de los pesos de las últimas capas hacia las primeras de la red. Un problema común que ocurre durante este paso es que conforme la propagación se acerca a las primeras capas de la red, el gradiente se va haciendo más pequeño, dificultando de esta manera el ajuste de los pesos de estas capas, ralentizando el entrenamiento o, incluso, incapacitando a la red de converger correctamente.

de las clases, calculando esta probabilidad como el peso que tiene la activación de una neurona de la capa sobre la suma total de las activaciones de dicha capa. Para extraer el índice de la clase se suele aplicar la función *argmax* a la salida de *softmax* la cual símplemente devuelve el índice de la clase con la mayor probabilidad.

Para el entrenamiento del modelo se debe especificar un criterio de error o función de pérdida, y en función de esta pérdida se ajustan los pesos (ecuación 2.16) del modelo utilizando la técnica del *backpropagation*. La función de pérdida evalúa cuánto se parece el valor predicho por la red neuronal al valor *ground truth*, estableciendo así el error entre ambos valores. Esta función de pérdida está en función de todos los pesos de la red, y sus mínimos se encuentran en aquellos puntos cuyos valores de los pesos producen una salida similar al valor de salida real. Para aplicar la técnica del *backpropagation* se calcula el gradiente del error conforme a los pesos de la última capa, y desde ahí se van propagando, utilizando la regla de la cadena, hacia los pesos de toda la red. De esta forma, con cada iteración los pesos se actualizan conforme la siguiente fórmula:

$$w_{i,j}^{(l)} = w_{i,j}^{(l-1)} - \alpha \frac{\partial \mathcal{L}}{\partial w_{i,j}} \tag{2.19}$$

siendo $w_{i,j}^{(l)}$ el peso j de la capa i tras la nueva iteración, $w_{i,j}^{(l-1)}$ el peso de la iteración anterior, $\mathcal{L}$ la función de pérdida y α un hiperparámetro conocido como *learning rate*. Este hiperparámetro define el nivel de ajuste que se realizarán a los pesos en cada iteración. Si es muy elevado es posible que la red no converja por no tener la precisión suficiente para encontrar soluciones óptimas, mientras que si es muy pequeño es posible que la red se quede estancada en mínimos locales, produciendo que no se encuentre una solución eficiente.

2.5.2.1. Redes neuronales convolucionales

Un caso concreto de redes neuronales son las redes neuronales convolucionales o *Convolutional Neural Networks* (CNNs). Estas redes sobresalen en tareas de reconocimiento y clasificación de imágenes y detección de patrones. Su arquitectura se puede ver en 2.17. Estas redes constan de dos partes principales:

1. La parte de extracción de las características visuales. Esta parte de la red se encarga de extraer vectores de características representativos de los datos de entrada y, generalmente, se denomina la parte de extracción de *features*. Esto se hace a partir de detectar patrones o estructuras que se repiten a lo largo del dataset. Los bloques que forman estas redes constan de una capa convolucional seguida de una capa de *pooling*. La primera capa convoluciona un *kernel* con la imagen de entrada. Este *kernel* generalmente representa algún tipo de patrón concreto, por ejemplo, una línea vertical u horizontal o un círculo. La salida de esta primera capa es un mapa bidimensional cuyas posiciones tendrán un valor mayor en aquellas ubicaciones donde la zona de la imagen se asemeja al *kernel*. Por cada capa

convolucional se debe especificar un número de n *kernels*, de manera que a la salida de la capa convolucional se obtienen n mapas bidimensionales, cada uno de ellos caracterizando cuánto y dónde se parece la imagen de entrada a los patrones caracterizados por cada uno de los *kernels*.

A estas capas convolucionales las siguen unas capas de *pooling* que reducen la dimensionalidad. Esto disminuye el número de parámetros a la vez que se extrae la información clave extraída por la capa convolucional.

2. La parte de clasificación. Una vez extraídos los vectores de características, se entrenan unas capas de tipo *Fully Connected* de la misma manera que se ha explicado previamente. La función de estas capas es la de clasificar y encontrar relaciones entre los vectores de características y una salida.

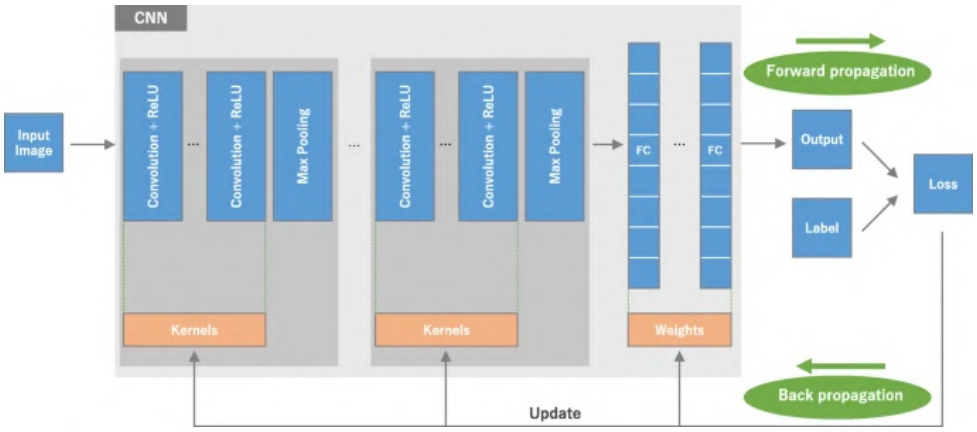


Figura 2.17. Arquitectura de una CNN. Imagen extraída de [37].

Este tipo de redes ofrecen una ventaja fundamental frente a los sistemas basados en un primer paso de extracción de características, como los descritos en 2.4.4, seguidos de una etapa de clasificación, como las descritas en 2.5.1. En este tipo de redes, ambos pasos están embebidos en el mismo sistema y, además, ambos pasos son optimizados, ajustados y adaptados a los datos de entrenamiento mediante la técnica del *backpropagation*. Esto permite conseguir un nivel de abstracción mayor, así como una detección de patrones y estructuras más complejos y específicos que si se utilizasen algoritmos de extracción de descriptores manuales como los comentados previamente. En [37] y [38] se analiza en detalle el funcionamiento y algunas aplicaciones de estas redes neuronales, y en [39] se realiza una comparativa y análisis de distintas CNNs en el ámbito de imágenes biomédicas.

La principal desventaja de las CNNs es el elevado número de datos y el coste computacional asociado a entrenarlas. La necesidad de que el dataset de entrenamiento sea grande y diverso es para lograr que la red generalice de forma correcta. Si se entrenase a la red con un número de datos pequeño, es posible que la red acabe «memorizando» las imágenes de entrada, incurriendo en el problema del *overfitting*. Es necesario prestar atención al dataset de entrenamiento. Cuanto

mayor sea y más diversidad haya en las imágenes que correspondan a las distintas clases, mejor generalizará el modelo.

Hay veces que, debido a limitaciones externas, el dataset de entrenamiento que se tiene es reducido y este no se puede ampliar de forma manual. Una técnica para ampliar el número de datos del dataset de forma programátaica es el *Data Augmentation*. Esta práctica consiste en generar nuevos datos de entrenamiento sintéticos a partir de los datos existentes. En el caso de las imágenes, algunos ejemplos comunes son aplicar distintas rotaciones o transformaciones, realizar recortes aleatorios o cambiar su color. En [40] se muestran algunos ejemplos de *Data Augmentation* aplicados en la clasificación de imágenes médicas utilizando CNNs.

Otra práctica muy común al utilizar CNNs es hacer lo que se conoce como *finetuning* de la red. El *finetuning* consiste en reentrenar una red que ha sido previamente entrenada con un dataset distinto al de la aplicación que se está desarrollando. Esta técnica permite a un modelo generalizar y lograr un buen desempeño aun con un dataset de tamaño reducido o con un tiempo de entrenamiento menor. En el caso de las CNNs, el *finetuning* generalmente aprovecha que la red preentrenada ya tiene adquiridas ciertas capacidades de abstracción, detección de patrones y clasificación aunque no estén especializadas en la tarea que se está tratando de resolver. Hacer *finetuning* consiste en inicializar esta red con los pesos obtenidos en el entrenamiento previo, y reentrenar la red con el dataset específico de la aplicación. En [41] se realiza un análisis detallado y una comparativa entre entrenar CNNs desde cero o realizar *finetuning*.

Un concepto relevante al hacer *finetuning* de un modelo es el concepto del *Transfer Learning*. El *Transfer Learning* hace referencia al fenómeno de «transferir» las capacidades previas de una red preentrenada a una tarea distinta. Cuanto más similar sean los datos con los que se está tratando de resolver la tarea nueva con los datos con los que se ha entrenado previamente el modelo, más facilidad tendrá el modelo de converger y obtener un buen rendimiento. Por ejemplo. será más sencillo hacer *finetuning* de una CNN para clasificar distintas razas de perros si ha sido entrenada previamente para clasificar distintos tipos de mamíferos que si ha sido entrenada previamente para detectar tumores cerebrales, ya que los patrones y contenidos de las imágenes serán más similares en el primer caso que en el segundo. En [42][43][44] se analiza en mayor profundidad este concepto de *Transfer Learning*.

2.5.2.2. Consideraciones sobre la aplicación de CNNs en tareas de visión por ordenador

A continuación se describen algunos puntos importantes a tener en cuenta cuando se utilicen modelos de este tipo:

- Las redes CNN son modelos con una alta carga computacional. Para ser entrenados de manera eficiente necesitan unidades de procesamiento con mucha VRAM y un tiempo de entrenamiento elevado. Antes de desarrollar un sistema basado en CNNs es necesario

analizar los datos y plantear posibles soluciones basados en otros sistemas menos computacionalmente costosos con los que quizás sea posible lograr un nivel de desempeño igual o incluso mejor.

- Es muy importante tener en cuenta el tamaño del dataset de entrenamiento. Como se ha descrito previamente, la buena generalización del modelo depende en gran medida del tamaño del dataset y la variabilidad de datos por cada clase. Si el dataset de entrenamiento es pequeño, es posible que merezca más la pena emplear tiempo en aumentar el dataset que en encontrar el modelo que pueda conseguir un desempeño bueno.

- Generalmente, es preferible hacer *finetuning* de un modelo entrenado previamente que entrenarlo desde cero [41]. Es importante valorar si existen modelos preentrenados que puedan ser utilizados para solucionar la tarea en cuestión antes de desarrollar el sistema. En el caso de que existan, también es importante valorar con qué datos se ha realizado el entrenamiento previo y valorar si se pueden aprovechar estas capacidades para la tarea que se está tratando de resolver.

Capítulo 3

Especificaciones y restricciones

En este capítulo se recogen las especificaciones y restricciones de diseño que aplican a al desarrollo a partir de los objetivos mostrados en el capítulo introductorio.

3.1. Especificaciones

Como especificaciones se establecen las siguientes:

- El dataset de imágenes debe ser capturado con buena calidad en razón de iluminación y enfoque.
- Se implementarán tres modelos distintos de aprendizaje automático para realizar la comparativa de rendimiento. Todos los modelos deben ser evaluados midiendo la tasa de aciertos obtenida sobre una partición de datos de test.
- La selección de datos de test se debe realizar asegurando la inexistencia de sesgos que puedan modificar los resultados en la evaluación de los distintos modelos.
- Se debe escoger una arquitectura de red neuronal convolucional y un sistema de preprocesado de imagen de manera que se pueda realizar el entrenamiento, evitando en la mayor medida posible la pérdida de información en las imágenes de la base de datos.

3.2. Restricciones

Como restricciones del proyecto se describen las siguientes:

- El desarrollo del proyecto, así como el entrenamiento de los modelos necesarios se hará en los recursos disponibles del GDEM.

- El sistema de captura para la base de datos es el disponibles en el GDEM. Las especificaciones de este sistema se muestran en la tabla 4.1.
- La red neuronal convolucional, la metodología de entrenamiento y el ajuste de hiperparámetros se debe escoger de manera que el entrenamiento se pueda ejecutar en una GPU *Nvidia Titan XP* de 12 GB de VRAM.
- El proyecto debe desarrollarse en Python como lenguaje de programación y utilizando bibliotecas y software de código abierto.
- La base de datos está limitada a la facilitada para este PFG por el hospital 12 de Octubre de Madrid, la cual consta de muestras de 117 pacientes.
- El desarrollo y ejecución de los distintos sistemas, a excepción del entrenamiento del modelo, se realizarán en una estación de trabajo con 16 GB de memoria RAM, procesador Intel® Core™ i5-2400 de 4 núcleos y una tarjeta gráfica Mesa Intel® HD Graphics 2000 (SNB GT1) de 1536 MB de VRAM.
- El tratamiento de las muestras y sus imágenes se hará respetando los estándares de privacidad y éticos relacionados con el manejo de datos clínicos. Sólo se utilizarán muestras previamente anonimizadas de manera que se asegure la protección de los datos del paciente.

Capítulo 4

Descripción de la solución propuesta

Como se adelantó en la introducción de este PFG, el objetivo de este proyecto es realizar una comparativa de distintos métodos de aprendizaje automático para el desarrollo de un sistema de predicción del grado de lesión craneoencefálica de un paciente a partir de muestras de suero sanguíneo secas. Como se explicó en el apartado 2.1 del Marco teórico y tecnológico, cuando las muestras de suero sanguíneo secan se forman distintos patrones en función de la composición química y los materiales disueltos en la sangre. Son estos patrones los que se van a utilizar como referencias sobre las que construir distintos modelos de clasificación. En este capítulo se presenta el trabajo realizado para la elaboración de los distintos sistemas de aprendizaje automático, así como el desarrollo de todos los pasos previos e intermedios que han sido necesarios.

4.1. Introducción a la solución propuesta

Para lograr los objetivos planteados en la introducción de este proyecto, se realizó un estudio inicial sobre los métodos y funcionalidades que se deben implementar para definir las bases de las distintas etapas del sistema de predicción del diagnóstico. En este capítulo se definen estas distintas etapas, desde la captura de las imágenes de las muestras hasta la obtención de la predicción por parte de los modelos de aprendizaje automático. El esquema general del flujo y las distintas soluciones diseñados se muestran en la Figura 4.1.

Este capítulo comienza con la captura de las imágenes y la creación de la base de datos. Posteriormente se desarrollan los sistemas de clasificación:

- El primero de ellos consiste en la implementación del sistema de extracción y caracterización de las propiedades visuales de las imágenes, basado en los contornos de los distintos patrones formados sobre las muestras. Este desarrollo se describe en la sección 4.3 de este capítulo. A este paso de extracción de la información le sigue la implementación de los distintos modelos de aprendizaje automático que aprenden de los datos extraídos por el sistema

anterior, cuyo desarrollo se presenta en el apartado 4.4 de este capítulo.

- El segundo consiste en el entrenamiento de una red neuronal convolucional directamente con las imágenes de la base de datos. Para el preprocesado de la base de datos y selección de las imágenes representativas se usa información extraída en la primera solución. Este desarrollo se describe en el apartado 4.5 de este capítulo.

En última instancia, se evalúan los resultados obtenidos mediante ambos métodos para elaborar las conclusiones que cierran este PFG.

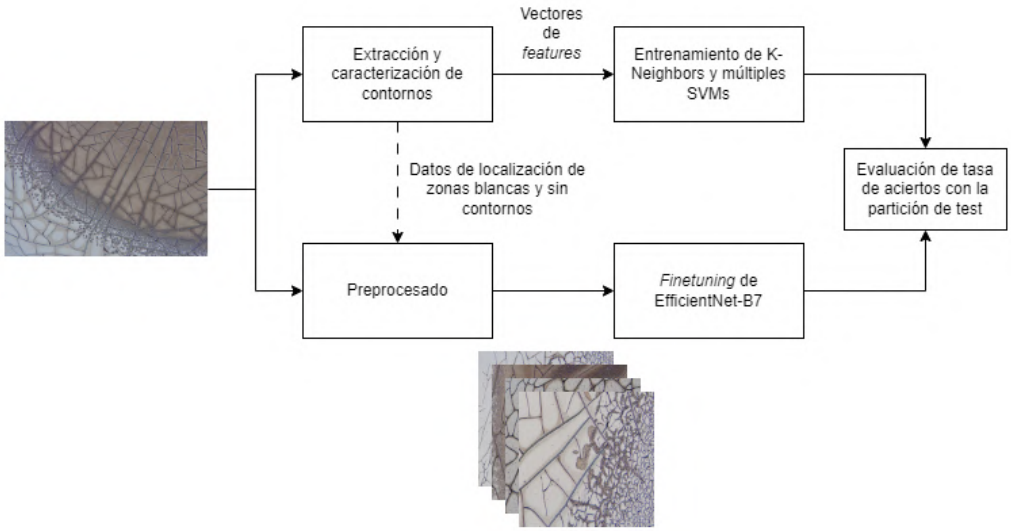


Figura 4.1. Diagrama de flujo de los distintos sistemas realizados en este PFG.

4.2. Captura de imágenes y creación de la base de datos

En primer lugar, es necesario construir, si no se dispone ya, una base de datos con imágenes RGB de un número suficiente de pacientes. Para la extracción de dichos datos ha sido necesario realizar una captura de suero seco dispuesto sobre un portamuestras por cada paciente. Estas capturas se han realizado en el laboratorio del CITSEM haciendo uso del microscopio de transmitancia de tipo *Bright Field* perteneciente al GDEM. Este microscopio cuenta, entre otros sistemas de captura, con una cámara RGB de alta resolución adjunta la cual ha sido adaptada y calibrada para el uso de microscopía de transmitancia. Las especificaciones de estos equipos se muestran sobre la tabla 4.1. Sobre la Figura 4.2 se muestra el microscopio y sus componentes. En [45] se especifica información más detallada sobre el sistema de captura utilizado.

Para realizar la captura de una única muestra, esta se debe acoplar en el portamuestras de la plataforma motorizada. A través de un software íntegramente desarrollado por el equipo del GDEM se puede especificar el área de la muestra que se desea capturar [45]. Dicho software calcula el número de capturas y la ruta óptima para cubrir el área especificada a capturar, tal

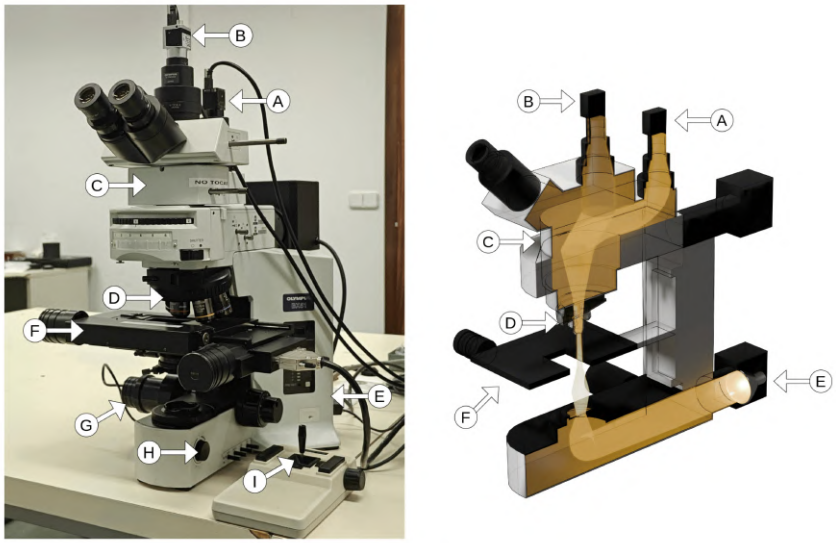


Figura 4.2. En la subfigura a) se muestra una imagen real del microscopio. En la subfigura b) un esquema del recorrido de la luz hasta llegar a la cámara. Sobre las imágenes se muestran los siguientes componentes: (A) cámara hiperespectral, (B) cámara RGB, (C) tubo trinocular, (D) lentes, (E) fuente de iluminación, (F) plataforma motorizada, (G) motor de movimiento en el eje Z, (H) controlador de la potencia lumínica (I) joystick para el control de la plataforma motorizada. Imagen e información extraídas de [45].

y como se especifica en [45]. Cada una de las secciones del área especificado será capturado por la cámara RGB. Estas capturas se denominarán *tiles* de ahora en adelante. Al final, por cada muestra resultan entre 40 y 70 *tiles* dependiendo del tamaño de estas. Cada uno de estos *tiles* tiene una resolución de 5472×3648 píxeles y ocupan 60 MB aproximadamente.

Durante la captura, se han ido recopilando distintas características visuales subjetivas observadas por muestra en una tabla de anotaciones como color, tamaño de patrones, transparencia, etc. Además, el equipo de investigación de neurocirugía del Hospital 12 de Octubre de Madrid ha facilitado información clínica sobre cada paciente capturado para generar una base de datos lo más completa posible. Entre la información facilitada se encuentra el grado de lesión cerebral. De esta forma se obtiene una visión general de las imágenes de la base de datos y sus características que, posteriormente, es muy útil para la selección de las propiedades que se desean caracterizar y los modelos que se utilizan. Esta tabla se realiza en una hoja de cálculo que posteriormente se exporta a un archivo `csv` y un archivo `json` que se utilizan para extraer información de cada paciente durante el entrenamiento de los distintos modelos.

El número de pacientes e imágenes así como el número de muestras correspondientes a cada clase se muestran sobre la tabla 5.1 del capítulo de resultados.

Tabla 4.1. Especificaciones del microscopio y la cámara RGB utilizados [45].

Componente	Parametro	Valor
Microscopio	Modelo	Olympus BX51 (Olympus, Tokyo, Japan)
	Lentes	UPlanSApo (4x) UPlanApo (10x) PlanApo (40x)
	Tipo de iluminación	Transmitancia
	Trinocular	U-TRU
Cámara RGB	Modelo	Basler ace acA5472-17uc (Exton, PA, USA)
	Tecnología de los sensores	CMOS
	Resolución	5472 x 3648 píxeles
	Profundidad de bit	10 ó 12 bits
	Tamaño de sensor	13.13 x 8.76 mm
	Tamaño de píxel	2.4 x 2.4 μm

4.3. Extracción de las características visuales

El sistema de extracción de las características visuales de las muestras se conforma de dos etapas: una primera en la que se aplica un algoritmo de detección y extracción de los contornos de los patrones formados sobre las muestras de suero, y una segunda etapa en la que se realiza el cálculo y extracción de descriptores que caractericen a los contornos extraídos. El resultado de este desarrollo es un sistema que recibe las imágenes de las muestras como entrada y a su salida devuelve lo siguiente:

- Una lista con los contornos detectados.
- Tres imágenes en las que se imprimen los contornos de distinta manera: una sobre fondo negro, otra sobre la imagen original de la muestra y una última sobre fondo negro en la que los contornos se muestran en gris y el interior de los contornos en blanco.
- Un archivo en formato `json` con las características extraídas por cada contorno y *tile*.

Estos objetos de entrada y salida del sistema así como sus formatos se muestran sobre la Figura 4.3. En los siguientes subapartados se describe en profundidad cada etapa.

4.3.1. Detección y extracción de contornos

El algoritmo de detección de contornos se ha desarrollado como un programa escrito en lenguaje Python. Para su desarrollo, se ha hecho uso principalmente de la librería de procesado de imagen OpenCV [8]. La estructura del algoritmo diseñado se puede ver en la Figura 4.4.

A continuación, se detallan cada uno de los bloques que integran el algoritmo de extracción de contornos:

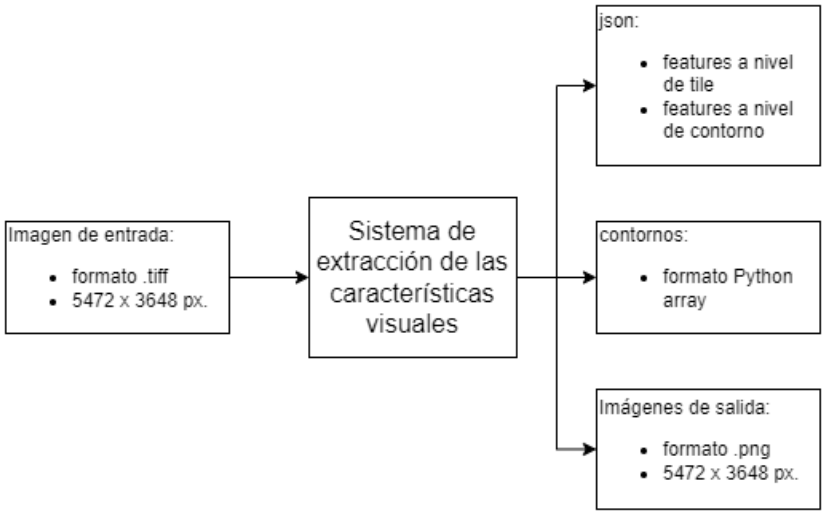


Figura 4.3. Esquema simplificado de los objetos de entrada y salida del sistema de extracción de características visuales, así como sus formatos.

1. **Filtrado de imágenes en blanco**. El primer paso es comprobar si el recorte de imagen o *tile* entrante al algoritmo corresponde con una de las secciones de las esquinas de la imagen, las cuales no tienen información útil ya que se corresponden con zonas sin suero. Para filtrar y descartar estas imágenes y ahorrar tiempo de procesado se calcula la varianza de la imagen y, si está por debajo de determinado umbral, se descarta. En este caso se descartan todas las imágenes con una varianza menor de 40. La función de filtrado, explicada en pseudocódigo, se realiza de la siguiente forma:

```
func es_imagen_blanca(img):
    si varianza(img) < 40 devuelve True, si no False
```

Listado 4.1. Función de filtrado de las imágenes en blanco correspondiente a zonas sin suero.

   Este umbral se ha obtenido de forma heurística, analizando imágenes de distintas secciones de las muestras y ofrece una aproximación suficiente para filtrar imágenes de secciones sin suero de imágenes con este, aunque el suero se capture solo de forma parcial en la imagen.

2. **Conversión a imagen monocromática**. El segundo paso es convertir la imagen RGB en una imagen monocromática. En un primer momento este paso se realizaba calculando la componente de luminancia de la imagen pero posteriormente se decidió utilizar el canal correspondiente al canal rojo como único canal. Esto es debido a que es el canal de color con mayor variabilidad en todas las muestras, lo que permite tener imágenes monocromáticas con mayor contraste sin necesitar ajustes adicionales.

3. **Umbralización adaptativa**. Este paso es crucial para la correcta extracción de contornos, ya que la calidad de los contornos extraídos posteriormente depende en gran medida del ruido presente en la imagen umbralizada, y de lograr la mayor ausencia posible de este. En un primer momento se intentó utilizar una umbralización con un valor fijo mediante

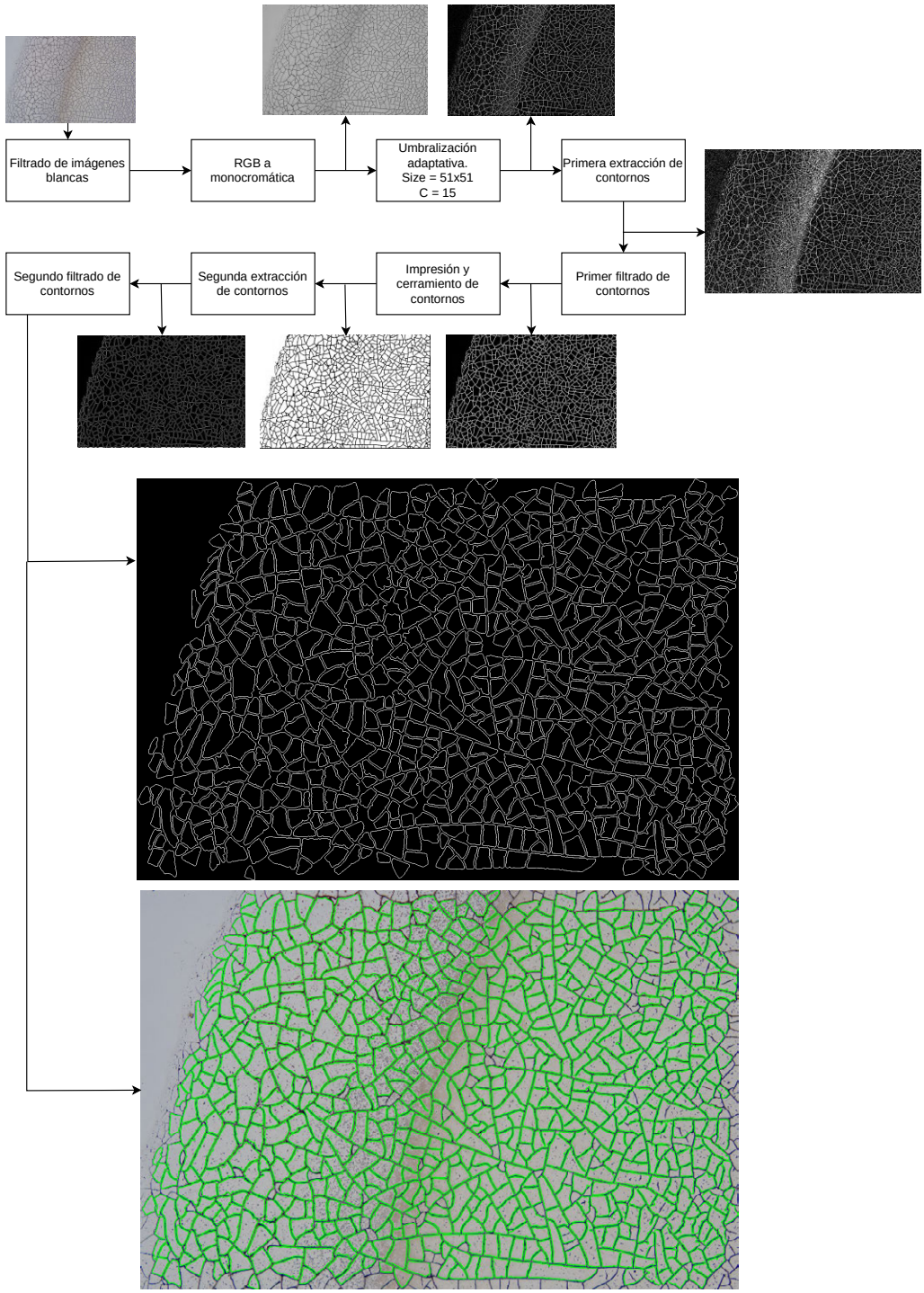


Figura 4.4. Diagrama simplificado del algoritmo de extracción de contornos.

el método de Otsu, algoritmo que se explicó previamente en el apartado 2.4.1 del marco teórico. Pero dada la naturaleza de las imágenes, las cuales presentan texturas complejas, superposición de elementos distintos y variaciones grandes del nivel de intensidad del fondo, este no proporcionaba buenos resultados. Se decidió entonces utilizar una umbralización adaptativa. El uso de una umbralización de este tipo propició una ventaja importante que consiste en que a pesar de producir grandes cantidades de ruido en regiones de la imagen más uniformes, en las zonas cercanas a los bordes se preservan los márgenes; el ruido derivado de la umbralización no está en contacto con los bordes, lo que permite un filtrado posterior más sencillo (ver Figura 2.6 del marco teórico).

La fórmula de la umbralización adaptativa se describe a continuación:

$$\alpha_{xy} = \frac{1}{NM}(\sum_i \sum_j p(i,j)) - C \quad \text{para } i \text{ y } j \text{ en la vecindad de } (x,y)$$
$$P(x,y) = \begin{cases} 1 & \text{si } p(x,y) > \alpha_{xy} \\ 0 & \text{si } p(x,y) < \alpha_{xy} \end{cases} \tag{4.1}$$

donde (i,j) representan la vecidad de píxeles del píxel (x,y) según el *kernel* especificado por el ingeniero, C es una constante también definida por el ingeniero cuyo valor especifica cuánta más intensidad debe presentar un píxel en comparación con su vecindad para ser tomado como blanco, y α_{xy} es la media de la intensidad de los píxeles dentro de la vecindad más la constante C. Por ejemplo, si el *kernel* es un cuadrado de tamaño 5×5, i y j irán desde $(x-2)$ e $(y-2)$ hasta $(x+2)$ e $(y+2)$.

Es por esto que en las zonas planas o de baja variabilidad de la imagen, al presentar los píxeles valores de intensidad muy similares, ligeras variaciones de esta pueden producir numerosos cruces del umbral, mientras que en las zonas cercanas a los bordes, dado que la variación en la intensidad de los píxeles es mayor, el valor de α_{xy} será un valor ubicado entre la intensidad del borde y del fondo, y el ruido derivado de la umbralización introducido será menor.

La función `cv2.AdaptiveThreshold()` disponible en la biblioteca de OpenCV realiza este tipo de umbralización. Usando esta función se define un tamaño de 51×51 píxeles y un valor de 15 para la constante C. Estos valores fueron fijados tras realizar múltiples pruebas sobre distintas imágenes. Se consideró que este tamaño de *kernel* es lo suficientemente grande como para capturar las diferencias de nivel en zonas cercanas a los bordes y aplicar los beneficios mencionados de la umbralización adaptativa en estas zonas críticas. El valor de C se definió de la misma manera, de forma que con un valor de 15 se puede diferenciar correctamente entre píxeles pertenecientes a los contornos y al fondo. La imagen resultante se invierte para que los contornos queden con el valor de brillo máximo y el fondo en negro.

4. **Primera extracción de contornos**. En este punto se aplica la primera extracción de contornos. Los bordes se definen como zonas de la imagen en las que la intensidad varía

de forma repentina, por lo que tanto los verdaderos bordes como el ruido presente en la imagen serán catalogados como bordes. Esta extracción de contornos se realiza con la siguiente función de OpenCV:

```
contours, hierarchy = cv2.findContours(image, mode=cv2.RETR_TREE, method=cv2.
    CHAIN_APPROX_NONE)
```

Listado 4.2. Función aplicada para realizar la primera extracción de contornos.

Los detalles de esta función se detallan en [13].

5. **Filtrado de contornos** El margen que se produce al aplicar la umbralización adaptativa, tal y como se comenta en el punto 3, es muy útil para limpiar la imagen de ruido y artefactos, ya que, una vez extraídos todos los contornos en el paso anterior, y estando estos separados de los contornos reales, aplicando un filtrado simple de área se puede eliminar la gran mayoría del ruido presente en la imagen. En este caso se filtran los contornos cuyo área es menor a 2000 píxeles cuadrados. Para detectar el área se hace uso de una función de OpenCV, de la siguiente forma:

```
area_filtered_contours = []
for cnt in contours:
    if cv2.contourArea(cnt) > FIRST_AREA_THRESHOLD: # FIRST_AREA_THRESHOLD = 2000
        area_filtered_contours.append(cnt)
```

Listado 4.3. Primer filtrado de área de contornos.

6. **Impresión de los contornos filtrados y cerramiento de los contornos** Para evitar falsos contornos, se reimprimen los contornos filtrados en el apartado anterior con una anchura de 5 píxeles. Posteriormente, se aplica el algoritmo de cerramiento dos veces: una primera vez con un *kernel* circular de diámetro 25 píxeles, y una segunda vez con un *kernel* circular de diámetro 7 píxeles con dos iteraciones. De esta forma se cierran posibles aperturas que hayan quedado entre contornos. Este paso se realiza para cerrar dichas aperturas y para evitar una detección posterior de falsos contornos como pueden ser contornos encerrados dentro de los verdaderos contornos, contornos que por la separación de unos pocos píxeles no son continuos y se detectan erróneamente, etc.

   Las funciones aplicadas para realizar la impresión y los cerramientos son las siguientes:

```
image=cv2.drawContours(np.zeros((img_original.shape), dtype=np.uint8), contours=
    area_filtered_contours, contourIdx=-1, color=255, thickness=5)
kern = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (25,25))
close_image_1 = cv2.morphologyEx(image, cv2.MORPH_CLOSE, kern, iterations=1)
kern = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (7,7))
close_image_2 = cv2.morphologyEx(close_image_1, cv2.MORPH_CLOSE, kern, iterations
    =2)
```

Listado 4.4. Impresión y cerramiento de los contornos.

7. **Inversión de la imagen**. Se aplica una inversión de los valores de intensidad de los píxeles debido a que, hasta el momento, los contornos se representan con el blanco y el fondo con negro. La función de detección de contornos utilizada devuelve como contornos regiones de píxeles cerradas con un valor de intensidad de blanco. Para caracterizar los patrones de las muestras es más útil representar los contornos con el color negro y el área interior de estos con blanco, de esta forma al volver a aplicar la función de detección de contornos, serán devueltos contornos que encierren dichos patrones en vez de los bordes:

```
invert_close_image = cv2.bitwise_not(close_image_2)
```

Listado 4.5. Inversión de los valores de los píxeles de la imagen.

8. **Segunda extracción de contornos**. Tras la impresión realizada en el paso anterior, se realiza una segunda extracción de los contornos presentes en la imagen, que esta vez serán mucho más precisos y coincidentes con los contornos reales:

```
inv_second_contours, _ = cv2.findContours(invert_close_image, cv2.RETR_EXTERNAL,
    method=cv2.CHAIN_APPROX_NONE)
```

Listado 4.6. Segunda detección de contornos.

9. **Segundo filtrado de contornos**. Generalmente, los contornos identificados en el centro de las muestras pierden su definición y la estructura cristalizada se vuelve mucho más caótica. Para evitar la detección de contornos en aquellas zonas de la muestra con estructuras más irregulares, lo que daría lugar a un número importante de falsas detecciones, se realiza un segundo filtrado por área y circularidad. Como umbral del filtro de área se establece 6000 píxeles cuadrados, y para el de circularidad un umbral de 0.1.

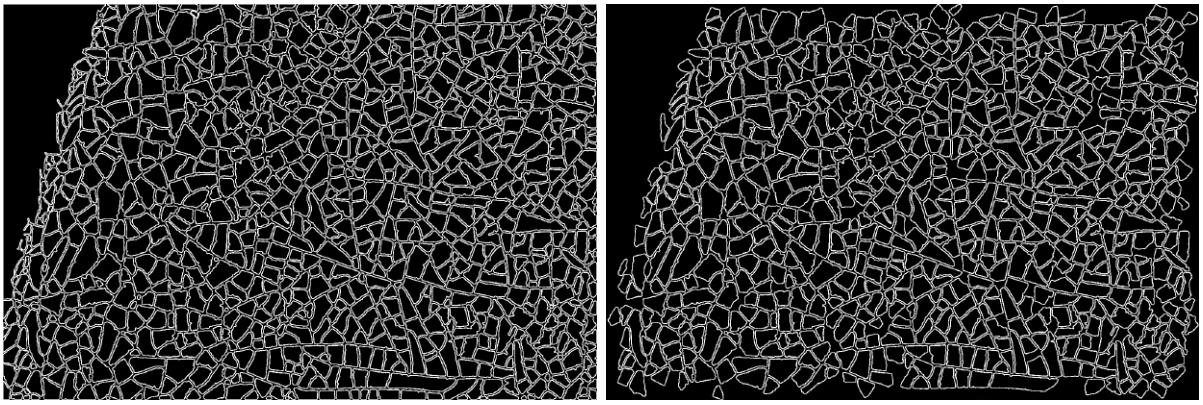
   El filtrado de área se realiza dado que en los bordes de las muestras los patrones tienen un tamaño mayor que los patrones presentes en el centro. Realizando este filtrado se puede reducir de manera muy simple la detección de contornos irregulares en el centro de las muestras. El filtrado de circularidad se realiza también, dado que pueden existir falsos contornos correspondientes con artefactos de la imagen que no se eliminan completamente solo con el filtrado por área. Además, se descartan los contornos que tienen algún píxel sobre alguno de los cuatro bordes de la imagen. Esto es debido a que la función de OpenCV no discrimina la detección de contornos aunque estos estén sobre alguno de los límites de la imagen y no salga el contornos completo. Es importante filtrar estos contornos ya que si no darían lugar a contornos cortados, introduciendo así información falsa. Las funciones utilizadas para realizar este segundo filtrado se detallan a continuación:

```
inv_area_filtered_second_contours = []
for idx, cnt in enumerate(inv_second_contours):
    isboundary = False
    for point in cnt:
        if any([coord == 0 for coord in point[0]]):
            isboundary = True
            break
```

```
        if point[0][0] == full_image.shape[1]-5:
            isboundary = True
            break
        if point[0][1] == full_image.shape[0]-5:
            isboundary = True
            break
    if not isboundary:
        # SECOND_AREA_THRESHOLD = 6000. CIRCULARITY_THRESHOLD = 0.1
        if cv2.contourArea(cnt) > SECOND_AREA_THRESHOLD and get_circularity(cnt) >
            CIRCULARITY_THRESHOLD:
            inv_area_filtered_second_contours.append(cnt)
```

Listado 4.7. Segundo filtrado de contornos.

donde la variable `point` representa cada uno de los puntos que conforma un contorno. Se comprueba si alguna de las coordenadas de estos puntos es 0, indicando que se encuentra sobre el margen superior o izquierdo, `full_image.shape[1]-5`, indicando que se encuentra sobre el margen derecho, o `full_image.shape[0]-5`, indicando que se encuentra sobre el margen inferior. La variable `full_image.shape` devuelve el tamaño de la imagen en formato $(filas, columnas)$ y el valor de -5 es necesario dado que las coordenadas de los píxeles limítrofes con el margen inferior y derecho se establecen con un margen de 5 píxeles. En la Figura 4.5 se muestra un ejemplo en el que se puede apreciar la presencia de contornos en los márgenes de la imagen y su resultado tras el filtrado.

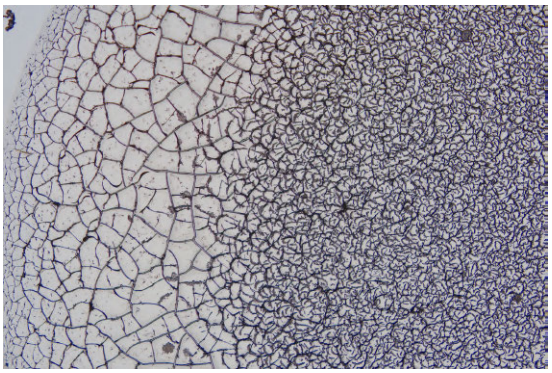


(a) Contornos antes del filtrado. (b) Contornos tras el filtrado.

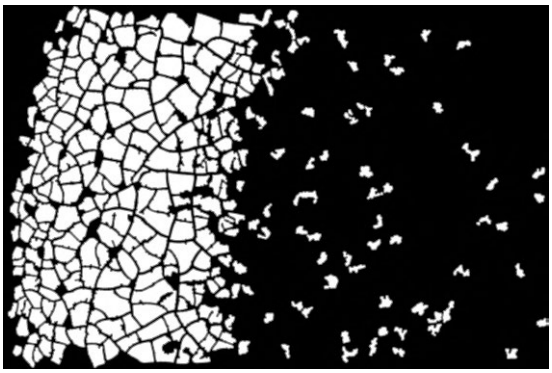
Figura 4.5. Ejemplo de la presencia de contornos en los bordes de la imagen y el resultado obtenido tras aplicar el segundo filtrado.

Una particularidad que presenta este sistema es que únicamente filtra contornos detectados en el centro de las muestras, y, excepcionalmente, en la corona si estos son debido a algún tipo de artefacto presente sobre las mismas muestras. Cabe mencionar que los contornos capturados en el centro de las muestras serán filtrados siempre y cuando su estructura sea caótica e irregular. En aquellas muestras en las que el centro de las muestras mantenga patrones regulares, estos contornos no serán filtrados y serán considerados también como

información válida, ya que al mantener estructuras regulares e identificables tras la cristalización, se estima que también aportan características representativas de la muestra. Este fenómeno se representa en la Figura 4.6 en la que se puede ver un ejemplo en el que en una muestra se ha discriminado la detección de contornos en el centro debido a su naturaleza caótica y otro ejemplo en el que, a pesar de corresponder con el centro de la muestra, la detección ha sido posible.



(a) Ejemplo que presenta cristalización caótica en el centro.



(b) Resultado de la extracción de contornos.

(c) Ejemplo que presenta cristalización regular en el centro.

(d) Resultado de la extracción de contornos.

Figura 4.6. En las subfiguras superiores se muestra cómo el algoritmo de detección de contornos ha descartado los contornos existentes en el centro de la muestra debido a su naturaleza caótica, puesto que no aportan información relevante. En las subfiguras inferiores la estructura de los patrones se mantiene regular por lo que el algoritmo sí es capaz de realizar esa extracción.

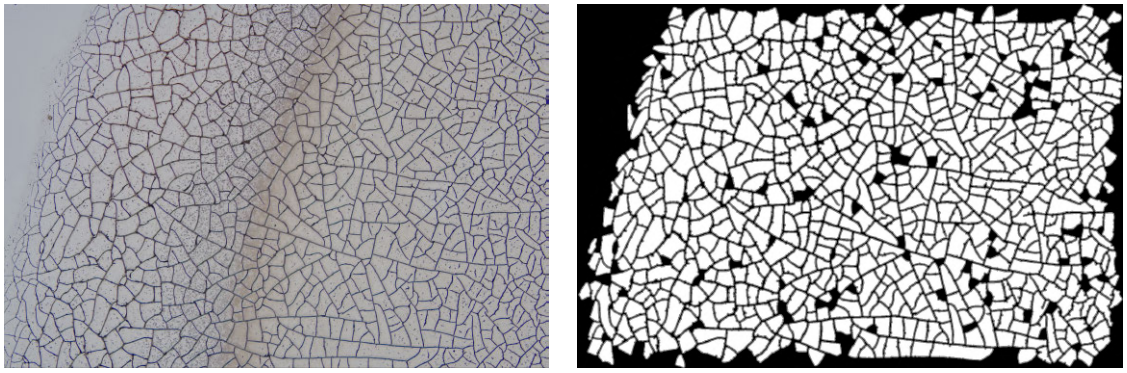
Además, se dan las siguientes particularidades respecto del algoritmo desarrollado:

- Las imágenes y contornos finales corresponden mayoritariamente con imágenes en las que únicamente existen contornos en la corona de las muestras, debido a que, generalmente, es en estas zonas donde los patrones formados tienen estructuras más regulares y, por ende, más representativas de la muestra.
- Debido a los distintos filtrados, es posible que se pierdan verdaderos contornos a lo largo

del algoritmo. Esto también es intencionado ya que se prioriza que no haya falsos contornos tras la aplicación del algoritmo a que estén presentes todos los verdaderos contornos de la muestra. Además, el número de contornos detectado por cada muestra sigue siendo lo suficientemente elevado como para tener un conjunto representativo de cada muestra.

- A pesar de los distintos filtrados, no es posible eliminar al 100 % los artefactos y falsos contornos detectados por cada muestra. Estos se han intentado minimizar lo máximo posible pero aún así, sobre todo en muestras con patrones más irregulares, existen algunos pocos contornos que corresponden con dichos artefactos que no se han conseguido eliminar completamente. En la Figura 5.1 del apartado 5.2 de resultados de este proyecto se ejemplifica este fenómeno con más detalle.

Tras la extracción final de los contornos, se imprime una última imagen para la visualización de los contornos de forma que estos se muestran en blanco, el interior de los contornos en gris y el fondo en negro.



(a) Recorte de la imagen original. (b) Contornos finales tras la aplicación del algoritmo.

Figura 4.7. Resultado de un *tile* de un paciente del dataset tras la impresión de los contornos finales y su relleno.

Dada la cantidad de operaciones que se deben realizar y el número de imágenes en el dataset, el proceso de extracción de contornos se paralelizó de manera que, por cada *tile*, se ejecuta un proceso en paralelo, hasta un máximo de ocho procesos concurrentes. Esto disminuyó notablemente el tiempo de ejecución necesario para extraer todos los contornos de 46 horas aproximadamente a 6, ejecutándose en el ordenador cuyas especificaciones se muestran en el capítulo 3 de especificaciones.

Para la gestión de errores y revisión de la ejecución se programó un sistema de `logs` de manera que se puede rastrear lo ocurrido con cada una de las imágenes. Concretamente, se busca guardar información de los siguientes casos: *tiles* detectados como imágenes en blanco (y, por ende, descartadas), imagen resultante sin contornos tras el filtrado (generalmente, imágenes del centro de las muestras con muchas irregularidades en las que el filtrado ha funcionado correctamente), fallo por no existir el archivo de la imagen o el directorio y, por último, fallo al procesar la imagen

en alguno de los distintos pasos. Por cada proceso ejecutado se escribe en distintos archivos cuando ha ocurrido alguno de los casos anteriores. Los dos primeros casos son para verificar que las imágenes detectadas como imágenes en blanco sean detecciones correctas y las imágenes sin contornos correspondan a zonas irregulares donde la información no es relevante, generalmente en el centro de las muestras. Los últimos dos casos son para la detección de errores y, concretamente, cuando sucede el último error, se imprime el registro de error en su correspondiente archivo de `log` para poder revisar el origen del fallo. El esquema de directorios de `logs` se muestra en la Figura 4.8.

```
logs/
├── file_directory_error.log
├── load_image_error.log
├── no_contours/
│   ├── pat_1/
│   │   └── no_contours.log
│   ├── pat_2/
│   │   └── no_contours.log
│   └── ...
└── white/
    ├── pat_1/
    │   └── white_images.log
    ├── pat_2/
    │   └── white_images.log
    └── ...
```

Figura 4.8. Estructura de directorios de los archivos log.

4.3.2. Extracción de descriptores de formas

Hay tres niveles distintos para representar la información relativa a cada paciente: el nivel global de paciente, el nivel de *tile* y el nivel de contorno. Dado que la base de datos está formada por *tiles*, se extrae información que caracterice a cada uno de estos *tiles* y, además, se extrae información también sobre cada uno de los contornos individualmente. De esta manera, se busca utilizar estos datos para representar a cada paciente:

1. A nivel de contorno se va a extraer la siguiente información: área, circularidad, momentos de Hu y momentos de Zernike, ya presentados en el apartado 2.4.4 del marco teórico.

2. A nivel de *tile* se va a extraer la siguiente información: color medio de la muestra en coordenadas CIELAB y HSV, número de contornos, media y desviación estándar del área

de los contornos presentes y media y desviación estándar de la circularidad de los contornos presentes.

3. Además, por cada *tile* se calculan valores atípicos de contornos mediante el *z-score*[1]. Como contornos con valores atípicos de área o circularidad se toman aquellos que tienen un *z-score* mayor que 2,5.

Una vez ha terminado el proceso de extracción de contornos para una única imagen, se ejecuta una función que recibe todos los contornos extraídos y genera un archivo de tipo `json` para almacenar dicha información. La función es la siguiente:

```
def get_patch_features(contours, shape):
    patch_features = {}
    patch_features['n_instances'] = len(contours)
    patch_features['instances'] = {}
    for idx, cnt in enumerate(contours):
        patch_features['instances'][idx] = get_instance_features(cnt, shape)
    areas = [instance['area'] for instance in patch_features['instances'].values()]
    circularities = [instance['circularity'] for instance in patch_features['instances'].
        values()]
    patch_features['mean_area'] = np.mean(areas)
    patch_features['std_area'] = np.std(areas)
    patch_features['median_area'] = np.median(areas)
    patch_features['mean_circularity'] = np.mean(circularities)
    patch_features['std_circularity'] = np.std(circularities)
    patch_features['median_circularity'] = np.median(circularities)
    patch_features['outliers'] = {}
    patch_features['outliers']['area'] = []
    patch_features['outliers']['circularity'] = []
    # Outliers
    for idx, instance in patch_features['instances'].items():
        # Outliers will be computed with z scores
        # 2 is for 95 % of the data
        if abs((instance['area'] - patch_features['mean_area']) / patch_features['
            std_area']) > 2:
            patch_features['outliers']['area'].append({idx : instance['area']})
        if abs((instance['circularity'] - patch_features['mean_circularity']) /
            patch_features['std_circularity']) > 2:
            patch_features['outliers']['circularity'].append({idx : instance['circularity
                ']})
    return patch_features
```

Listado 4.8. Función que extrae los distintos descriptores para los contornos de un *tile* concreto.

donde la función que extrae los descriptores individuales de cada contorno (línea 6 de la función), es la siguiente:

```
def get_instance_features(cnt, shape):
    instance_features = {}
```

[1]El *z-score* se calcula de la siguiente forma: $z_i = |x_i - \mu_x|/\sigma_x$ donde μ_x es la media y σ_x la desviación estándar. Cuanto más elevado sea este valor, más atípico es el dato que se está evaluando. Usando el *z-score* se pueden establecer unos umbrales tal que si $z > 2$ el valor se encuentra fuera del 95 % de la distribución de los datos y para $z > 3$ se encuentra fuera del 99.7 %, suponiendo que dichos datos siguen una distribución normal.

```
    instance_features['circularity'] = get_circularity(cnt)
    instance_features['area'] = cv2.contourArea(cnt)
    instance_features['HuM'] = get_HU_moments(cnt).tolist()
    instance_features['Zernike'] = get_Zernike_moments(cnt, shape)
    return instance_features
```

Listado 4.9. Función que extrae los distintos descriptores para un contorno concreto.

De esta forma por cada *tile* resulta un archivo `json` en el que se ha registrado el valor de todos los parámetros que se desea analizar, como el siguiente:

```
{
    "n_instances": 741,
    "instances": {
        "0": {
            "circularity": 0.38515048808548014,
            "area": 70359.5,
            "HuM": [
                    ...
            ],
            "Zernike": [
                ...
            ]
        },
        "1": {
            ...
        },
        ...
    },
    "mean_area": 16715.275978407557,
    "std_area": 8267.079085486286,
    "median_area": 14995.0,
    "mean_circularity": 0.6331410249734867,
    "std_circularity": 0.08838788059297671,
    "median_circularity": 0.6501828393415718,
    "outliers": {
        "area": [
            {
                "0": 70359.5
            },
            ...
        ],
        "circularity": [
            {
                "5": 0.4076581359256458
            },
            ...
        ]
    },
    "avg_colour": {
        "hue": 77.01013522991049,
        "A": 130.74154302084938,
        "B": 125.9175017092599
    }
}
```

Listado 4.10. JSON extraído por cada *tile* con los valores de los descriptores extraídos.

4.4. Categorización mediante técnicas de machine learning

Una vez realizada la detección y caracterización de los contornos de las muestras, se procede a implementar y entrenar los modelos de *Machine Learning* con los datos extraídos. Estos modelos se entrenaron con los vectores de características o *features* extraídos y descritos en el apartado 4.3.2 de la sección anterior.

4.4.1. Selección de los modelos

El primer modelo de *Machine Learning* seleccionado es *K-Neighbors*. La selección de este modelo se justifica por los siguientes motivos:

- Es un modelo no paramétrico, fácil de implementar y compatible con problemas multiclase.
- Este modelo puede ofrecer resultados positivos cuando se tiene un conjunto de datos sobre los que no se conoce una distribución previa. Utilizando este modelo se puede comprobar si existen relaciones o agrupaciones entre los datos extraídos mediante un modelo computacionalmente muy sencillo y eficiente.

Para el segundo experimento se seleccionó SVM adaptado a problemas multiclase mediante la estrategia de *One vs. All*, ya explicada previamente en el apartado 2.5.1 del marco teórico. La selección de este modelo se justifica por los siguientes motivos:

- Las SVM, mediante la técnica del *kernel*, explicada previamente también en el marco teórico, son capaces de adaptar no linealidades y relaciones entre los datos de entrada más complejos para establecer los márgenes óptimos de separación entre las clases.
- El *kernel* seleccionado es el gaussiano o *radial basis function*. El uso de este *kernel* introduce un nuevo hiperparámetro γ que, junto con el hiperparámetro de regularización C, ofrecen mayores posibilidades de encontrar una configuración que converja sobre los datos de entrenamiento, y cuya configuración se puede hallar a través de aplicar métodos de validación cruzada.
- A través de la estrategia *One vs. All* se puede adaptar este modelo a problemas multiclase de forma relativamente sencilla.

Tras la selección de los modelos se procede a realizar su entrenamiento para comprobar si, con la información extraída, se puede desarrollar un sistema de clasificación eficiente de las muestras.

4.4.2. Entrenamiento de los modelos

Con los datos obtenidos se procede a entrenar distintos modelos de ML. Se realizaron varias pruebas utilizando como datos vectores de características con información a nivel de *tile* y con

información a nivel de contornos individuales.

4.4.2.1. Entrenamiento a nivel de *tile*

A partir de los datos obtenidos siguiendo el procedimiento mostrado en el apartado anterior, se genera un dataset el cual cada instancia corresponde con el vector de características de un único *tile* o fragmento de imagen individual. Por cada paciente existen entre 40 y 60 *tiles* dependiendo del tamaño de la muestra, es por ello que en el dataset creado existen instancias con valores distintos, pero que pertenecen a un mismo paciente. En la tabla 4.2 se ven los distintos valores que conforman el vector de características de cada instancia del dataset, donde A y B representan las componentes de color medio en el espacio *CIELAB*.

Tabla 4.2. Vector de características del dataset de entrenamiento a nivel de *tile* para los modelos de ML.

X							Y
Id (paciente, tile)	Área media	Desv. est. del área	Circularidad media	Desv. est. de la circularidad	A	B	Clase

Para el entrenamiento se realizan una partición de datos de entrenamiento y una de datos de test. Dado que en el dataset generado existen múltiples instancias para cada paciente, como se ha explicado previamente, para evitar sesgos en la evaluación del modelo, se asegura que las instancias que conforman los datos de entrenamiento y las instancias que forman los datos de test están formados *tiles* de pacientes distintos, es decir, que no hay instancias correspondientes a varios fragmentos de imagen de un mismo paciente compartidas entre ambas particiones.

Una vez hecha la partición se realizarán los siguientes experimentos:

1. Entrenamiento de un clasificador de tipo *K-Neighbors*. Para entrenar este modelo se realiza una validación cruzada con diez particiones (*10-fold cross validation*) de los datos de entrenamiento. El modelo se evalúa para valores del parámetro $K = 2, 3, ..., 10$ donde K es el número de datos que utiliza el modelo para la predicción.

2. Entrenamiento de múltiples SVMs utilizando la estrategia *One vs. All*. En el entrenamiento de los SVM se utiliza un *kernel* de tipo gaussiano o *radial basis function*. De forma similar que con el modelo anterior, se realiza una validación cruzada con cinco particiones de los datos de entrenamiento. El modelo se evalúa para valores de $C = \{0{,}001, 0{,}01, 0{,}1, 1, 10, 100, 1000\}$ y $\gamma = \frac{1}{M}\{0.125,\ 0.25,\ 0.5,\ 1,\ 2,\ 4,\ 8,\ 16\}$ donde M es el número de características por vector y C es el factor de regularización del modelo. A mayor valor de C menos regularización es aplicada. La estrategia del *kernel* y los parámetros asociados al entrenamiento de SVMs se presentaron previamente en el apartado 2.5.1 del marco teórico.

Como procedimiento de predicción final del paciente se extraen todas las instancias correspondientes a este mismo, es decir, todos los vectores de características correspondientes a los distintos fragmentos de imagen que componen la muestra, y se obtiene el valor de predicción para todos ellos. La predicción final del nivel de lesión del paciente se calcula como el valor más común que se ha obtenido entre todas las instancias.

Tanto para realizar el entrenamiento como la inferencia, los elementos de los vectores de características se normalizan de forma estándar, esto es siguiendo la siguiente fórmula:

$$f_i^{'(j)} = \frac{f_i^{(j)} - \mu_{f^{(j)}}}{\sigma_{f^{(j)}}} \tag{4.2}$$

donde $f_j^{'(i)}$ es el elemento j del vector correspondiente a la instancia i normalizado, $\mu_{f^{(j)}}$ es la media de todos los elementos j y $\sigma_{f^{(j)}}$ la desviación estándar de todos los elementos j de los vectores de características.

Los resultados correspondientes a estos dos últimos experimentos se muestran en el apartado 5.3.1 del capítulo de resultados.

4.4.2.2. Entrenamiento a nivel de contorno

En este caso se genera un dataset en el cual cada instancia representa un contorno extraído de las imágenes de muestra. Estas instancias están formadas por las características extraídas por cada contorno y el grado de lesión del paciente como etiqueta. En la tabla 4.3 se puede ver el vector de características correspondiente a cada instancia de este dataset donde **Hu* y **Zernike* hace referencia a los valores de los vectores correspondientes a los momentos de *Hu* y *Zernike*. El dataset final consta de 595633 instancias correspondientes a 117 pacientes distintos. Cada instancia está formada por un vector de 34 características.

Tabla 4.3. Vector de características del dataset de entrenamiento a nivel de contorno para los modelos de ML.

X					Y
Id (paciente, tile)	Área	Circularidad	*Hu	*Zernike	Clase

De manera similar al entrenamiento a nivel de *tile*, se realiza una separación entre datos de entrenamiento y datos de test de manera que no haya instancias pertenecientes a un mismo paciente en ambas particiones para evitar sesgos en la evaluación del modelo. La partición se realiza de forma que los datos pertenecientes al 45 % de los pacientes pertenecen a la partición de test y el resto a la partición de entrenamiento. El número de instancias obtenidas para realizar este entrenamiento se muestra en la tabla 5.5. Como se puede observar en ella, el número de datos de entrenamiento es de 293.056 instancias, correspondientes a todos los contornos extraídos de

los pacientes de la partición de entrenamiento. Este número es tan elevado que el entrenamiento de un SVM con todos los datos supondría un tiempo de computación excesivo. Para abordar esta problemática, se realizan distintos entrenamientos con distintos tamaños de subparticiones, formados por instancias seleccionadas de forma aleatoria de los datos de entrenamiento.

Se han realizado dos experimentos:

1. Entrenamiento de un clasificador de tipo *K-Neighbors*. Para entrenar este modelo se realiza una validación cruzada con diez particiones de los datos de entrenamiento (*10-fold cross validation*). El modelo se evalúa para valores de $K = 2, 3, ..., 14$, donde K representa el número de datos que utiliza *K-Neighbors* para realizar la predicción, y se utilizan todos los datos de entrenamiento.

2. Entrenamiento de varios clasificadores SVM con *kernel* gaussiano o *radial basis function* utilizando la estrategia *One vs. All*. Se realiza una validación cruzada con cinco particiones de los datos de entrenamiento (*5-fold cross validation*) para los valores de $C = \{0.001, 0.01, 0.1, 1, 10, 100, 1000\}$ y $\gamma = \frac{1}{M}\{0.125, 0.25, 0.5, 1, 2, 4, 8, 16\}$ donde M es el número de características por vector. El entrenamiento se repite con distintos tamaños de subparticiones del dataset de entrenamiento: 4000, 10000 y 40000 instancias.

Tanto para el entrenamiento como para la inferencia se normalizan los datos utilizando la fórmula mostrada en la ecuación 4.2. La inferencia y predicción del paciente, se calcula de manera similar al caso de los *tiles*. Esta se hace extrayendo todas las instancias de un sólo paciente, es decir, todos los vectores de características correspondientes a todos los contornos detectados sobre la muestra de un paciente, y devolviendo como predicción final del paciente la clasificación obtenida un número mayor de veces entre todas las instancias.

Los resultados correspondientes a estos experimentos se muestran en el apartado 5.3.2 del capítulo de resultados.

4.5. Clasificación mediante una red neuronal convolucional

En esta sección se va a describir el procedimiento usado para entrenar una CNN con el objetivo de clasificar las muestras de los pacientes.

4.5.1. Primeros pasos

Dada la complejidad de entrenar un modelo desde cero y el tamaño relativamente reducido del dataset para el entrenamiento de una CNN se ha decidido optar por la opción de aplicar *finetuning* (ver apartado 2.5.2) a un modelo preentrenado. El dataset está formado por *tiles* correspondientes a las imágenes de las muestras de cada paciente. Estas imágenes son de una resolución muy

elevada, dado que es importante tener registro de detalles pequeños en los contornos presentes en las imágenes. Concretamente, cada *tile* tiene una resolución de 5472×3648 píxeles. La mayoría de CNNs aceptan resoluciones de imágenes pequeñas. 224×224 píxeles suele ser una de las resoluciones de imagen de entrada más comunes. Existen algunas técnicas para adaptar el tamaño de las imágenes del dataset al tamaño de entrada de las CNNs como puede ser aplicar un reescalado o un recorte aleatorio, no obstante, si se aplicase alguna de estas técnicas a las imágenes se perdería gran parte de la información y el detalle.

Teniendo en cuenta estas condiciones, se han tenido presentes las siguientes consideraciones:

- Por un lado, la elección del modelo se ha seguido teniendo en cuenta, como aspecto prioritario, que la resolución de las imágenes de entrada fuera de un tamaño grande y suficiente. Por ello, tras un proceso de búsqueda, se ha seleccionado el modelo *EfficientNet B7* [46] ya que permite un tamaño de imágenes de entrada de 600×600 píxeles y con esta resolución la información presente en las imágenes no se verá tan alterada como con modelos de resoluciones menores.
- Se ha aplicado un preprocesado a las imágenes del dataset. Este preprocesado consta de dos pasos: primero, los *tiles* del dataset se han dividido en cuatro subimágenes de forma que cada una de ellas tenga una resolución de 2736×1824 píxeles. Segundo, se ha aplicado un reescalado a estas imágenes a un tamaño de 600×600 píxeles, de manera que su resolución coincida con la de entrada de la CNN.

Otra condición a tener presente es que la clasificación se quiere hacer basándose en aquellas zonas de las imágenes en las que los patrones formados son regulares, los cuales generalmente coinciden con los patrones formados sobre la corona de las muestras. Además, en el dataset también están presentes las imágenes en blanco correspondientes a las secciones donde no hay suero, de las que se habló en las secciones anteriores. Para evitar incluir estas imágenes en el entrenamiento, se realizó un filtrado del dataset utilizando la información extraída en el apartado 4.3.1. Por un lado, las imágenes que han sido detectadas como blancas se descartan directamente del dataset de entrenamiento. Por otro lado, las imágenes las cuales resultaron en imágenes sin contornos fueron descartadas también ya que estas imágenes, generalmente, coinciden con zonas que no han tenido patrones detectables, mayoritariamente ubicadas en el centro de las muestras, debido a la presencia de artefactos en la imagen o a la irregularidad de las estructuras, de manera que los contornos detectados son descartados por los pasos de filtrado del algoritmo. Tras aplicar estas transformaciones y filtrados, se obtiene el dataset con el que se entrena y evalúa el modelo. En la tabla 5.8 se muestra el número de pacientes e imágenes final que conforman las distintas particiones del dataset. Un ejemplo de las imágenes de este dataset se muestran en la Figura 4.9.

Se realizan dos experimentos con dicho modelo:

1. En el primer experimento se congelan todas las capas del modelo relacionadas con la extracción de *features* y se entrena únicamente la parte de clasificación. Esta parte de

clasificación consta únicamente de una capa de *dropout*[2] con probabilidad $p = 0{,}5$ y una capa lineal. En este experimento se entrena el modelo con los datos extraídos sin ningún tipo de transformación o técnica de *data augmentation*.

2. En el segundo experimento se congelan durante el entrenamiento los módulos 1, 2, 3 y 4 de extracción de *features* y se entrenan los módulos 5, 6, 7 y 8 de extracción de *features* así como la capa de clasificación. Además, en este segundo experimento se aplican técnicas de *data augmentation* a los datos de entrada, concretamente volteos aleatorios horizontales y verticales y giros aleatorios en pasos de 90º.

4.5.2. Preparación del modelo y los datasets de entrenamiento, validación y test

El entrenamiento del modelo se va a realizar utilizando el *framework* de *PyTorch* [47], el cual ofrece una serie de módulos, herramientas y facilidades que permite entrenar modelos y preparar datasets de forma rápida y sencilla, así como por tener definidos distintos modelos preentrenados preparados para su uso, entre ellos, el seleccionado *EfficientNet B7*.

4.5.2.1. Preparación de los datasets de entrenamiento, validación y test

Para preparar los datasets se aprovechan las funcionalidades ya programadas de la clase `Dataset` de *PyTorch*. Se crea una clase personalizada que hereda de esta clase `Dataset` para implementar algunas funcionalidades concretas necesarias para llevar a cabo el experimento. Una de estas funcionalidades es la de separar el dataset de prueba en datasets de entrenamiento, validación y test. Estas particiones, de forma similar a como se realizaron en el apartado 4.4.2, se harán separando por pacientes, es decir, asegurando que no haya instancias de un mismo paciente repartidas entre distintos datasets para evitar sesgos en la evaluación del modelo. El ratio de datos de test será del 20 % del total y el ratio de validación será del 15 % de los datos de entrenamiento.

A estos datasets se les ha añadido funcionalidades que aplican transformaciones a las imágenes de entrada cada vez que se extrae un *batch* de imágenes. Estas transformaciones se muestran en los listados de código 4.11 y 4.12. En el primero se muestran las transformaciones que se aplican a las imágenes de test en ambos experimentos y a las imágenes de entrenamiento en el

[2]Las capas de *dropout* son capas de regularización cuyo funcionamiento consiste en desactivar neuronas de la misma capa de forma aleatoria en cada iteración durante el entrenamiento. Estas capas tienen como parámetro la probabilidad de que una neurona sea desactivada. De esta manera, con un valor de p, cada neurona de la capa será desactivada con probabilidad p. El objetivo de estas capas es ampliar «forzadamente» la diversidad en la activación de las neuronas de una misma capa, ya que es posible que la red neuronal aprenda a representar la información en una capa utilizando únicamente ciertas neuronas, lo que aumenta la posibilidad de *overfitting*. Aplicando esta capa se fuerza a que la red neuronal aprenda a representar un mismo dato de entrada con un patrón de activación distinto cada vez que alcanza dicha capa, aumentando la capacidad de generalización del modelo.

primero de ellos. En el segundo se muestra las transformaciones que se aplican a las imágenes de entrenamiento para utilizar la técnica de *Data Augmentation* en el segundo experimento.

```
img = F.pil_to_tensor(img)
img = img/torch.max(img)
img = F.normalize(img, mean=self.mean, std=self.std)
```

Listado 4.11. Transformaciones que se aplican a las imágenes de entrenamiento en el primer experimento y a las de test en ambos.

```
img = F.pil_to_tensor(img)
img = RandomChoice([
    RandomRotation([0,0]),
    RandomRotation([90,90]),
    RandomRotation([180,180]),
    RandomRotation([270,270]),
])(img)
img = RandomVerticalFlip()(img)
img = RandomHorizontalFlip()(img)
img = img/torch.max(img)
img = F.normalize(img, mean=self.mean, std=self.std)
```

Listado 4.12. Transformaciones que se aplican a las imágenes de entrenamiento en el segundo experimento.

Estos datasets están formados por instancias de pares imagen-etiqueta del paciente. Cada imagen corresponde con un recorte de una sección distinta de las muestras de cada paciente.

4.5.2.2. Preparación del modelo

Como se comentó anteriormente, el modelo seleccionado para realizar estos experimentos es *EfficientNet B7*. Éste se puede utilizar directamente desde el *framework* de *PyTorch*, concretamente desde el módulo `torchvision` el cual implementa modelos, transformaciones y herramientas útiles relacionadas con las redes neuronales aplicadas a imágenes.

Este modelo se instancia inicialmente con los pesos de un entrenamiento previo realizado con un dataset denominado *ImageNet-1k*[3] [48]. Se cambia la dimensión de salida de la última capa del modelo de 1000 a 4, ya que es el número de clases presente en los datos de entrenamiento: control, leve, moderado y grave. Por último, se congelan las capas necesarias según se especifica en la descripción de los dos experimentos.

[3]ImageNet es una base de datos en las que se pueden encontrar distintos datasets de imágenes agrupadas en distintas clases. Estos son de uso libre siempre que sean para fines no comerciales. Esta base de datos es famosa y ampliamente utilizada por investigadores para desarrollar modelos de *deep learning* y visión por ordenador. Entre los datasets ofrecidos en esta base de datos está el mencionado *ImageNet-1k* el cual contiene 1.281.167 datos de entrenamiento, 50.000 de validación y 100.000 de test, y está formado por imágenes de mil clases distintas.

4.5.2.3. Entrenamiento del modelo

El modelo se entrenó en una tarjeta gráfica Nvidia Titan XP de 12 GB de VRAM. Como parámetros del entrenamiento se definen los siguientes:

- Se escoge *Adam*[49] como optimizador con un *learning rate* de 0,001.
- Como función de pérdida se utiliza la función *cross entropy loss* [50].
- Se define un número de *epochs*[4] máximo de 10 para el primer experimento y 50 para el segundo.
- Se define un número de *batch*[5] de 24 para el primer experimento y 8 para el segundo. Esto se debe a que en el primer experimento únicamente se entrena la última capa, lo que libera de computación a la unidad de procesamiento y permite un tamaño de *batch* mayor.

Con estos parámetros se entrenó el modelo, aplicando la técnica de *finetuning*. Al finalizar cada *epoch* se evaluó el rendimiento del mismo con los datos de validación y se guardaron aquellas configuraciones de pesos que sobrepasaron la mejor configuración hasta el momento.

4.5.2.4. Inferencia y evaluación del modelo

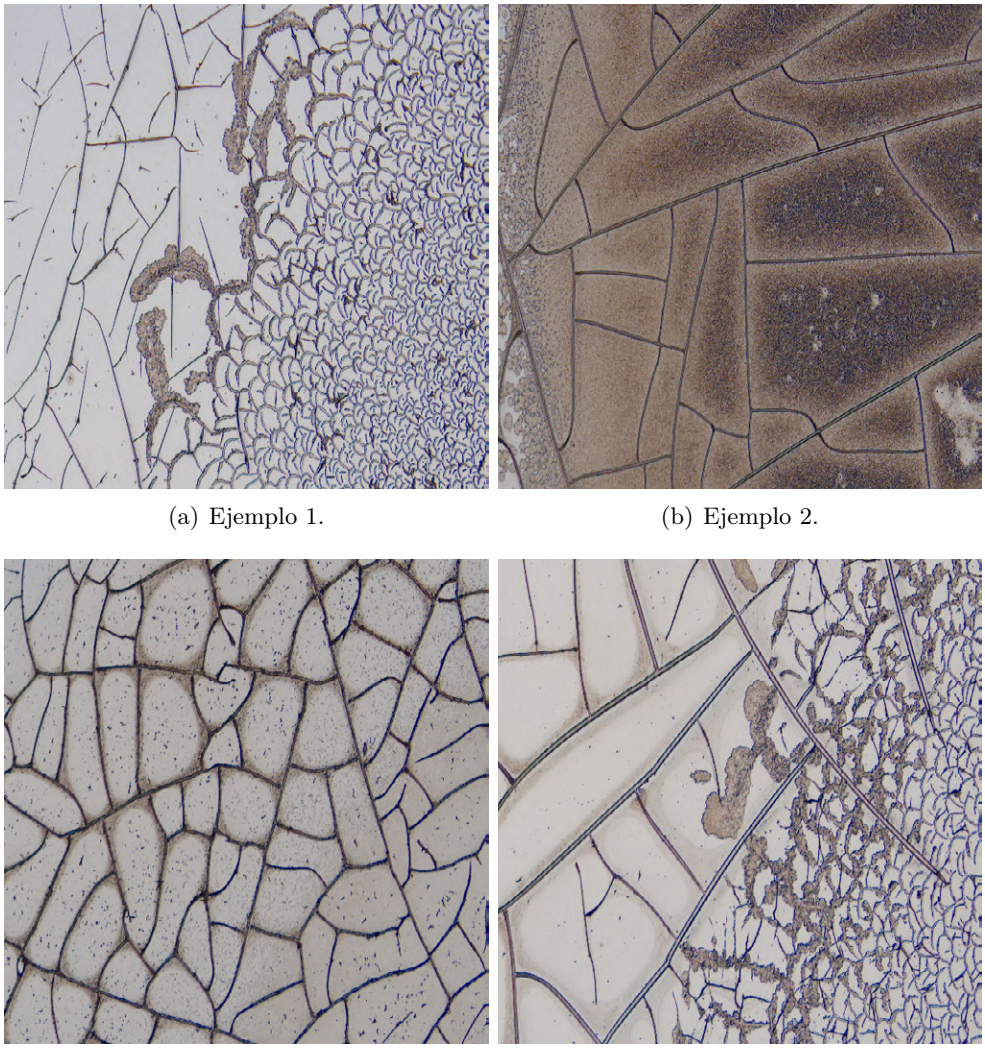
Para realizar inferencia con el modelo una vez entrenado se aplica una lógica similar a la utilizada con los modelos de ML comentados en las secciones anteriores. La predicción final se obtiene de la siguiente forma:

1. Se extraen todos los recortes de imágenes del dataset que correspondan a un único paciente.
2. Se predicen los valores de salida para cada una de las imágenes.
3. Como predicción final del paciente se determina aquella clase que haya sido la más numerosa entre todas las obtenidas.

Los resultados tras la evaluación del modelo se muestran en el apartado 5.4 del capítulo de resultados.

[4]El número de *epochs* define cuántas veces pasarán todos los datos del dataset por el modelo durante el entrenamiento.

[5]El número de *batch* es el número de datos que se meten a la vez en el modelo y sobre los que se calcula el gradiente y aplica la técnica de *backpropagation* en cada iteración.



(a) Ejemplo 1. (b) Ejemplo 2.

(c) Ejemplo 3. (d) Ejemplo 4.

Figura 4.9. Ejemplos de las imágenes del dataset tras el filtrado y las transformaciones de las imágenes con el que se entrena y evalúa el CNN. En las distintas subfiguras se pueden ver fragmentos reescalados a 600×600 píxeles correspondientes a muestras de distintos pacientes.

Capítulo 5

Resultados

En este capítulo se presentan los resultados obtenidos para los distintos procedimientos empleados en este proyecto. En los apartados 5.1 y 5.2 se muestran los resultados con respecto a la adquisición de datos. En el primero de ellos se presenta información acerca de la base de datos de imágenes generada tras el proceso de captura, y en el segundo se muestran los resultados obtenidos mediante el sistema de extracción de características visuales aplicado a las imágenes obtenidas en el primer apartado.

Posteriormente, en el apartado 5.3 se muestran los resultados obtenidos tras aplicar los modelos de aprendizaje automático descritos en 4.4 a los datos extraídos por el sistema de extracción de características visuales. Por último, en el apartado 5.4 se muestran los resultados obtenidos mediante el *finetuning* de la CNN, tal y como se describe en la sección 4.5.

5.1. Resultados de la adquisición de datos y presentación del dataset

La base de datos de imágenes se creó mediante un sistema de captura, cuyas especificaciones se describieron anteriormente en el apartado 4.2 del capítulo de descripción de la solución propuesta. En la tabla 5.1 se muestra la información relativa a la base de datos creada a partir de la captura de imágenes de las muestras disponibles para este PFG. Las diferentes clases presentes sobre la tabla corresponden con los siguientes niveles de lesión cerebral: 0 control, 1 leve, 2 moderado y 3 grave.

En el Anexo I de este informe se muestran distintos ejemplos de capturas de muestras de cada uno de los niveles de lesión tratados en este proyecto.

Tabla 5.1. Información sobre el dataset creado a partir de las imágenes de muestras de plasma sanguíneo secas. El número de clase coincide con los siguientes niveles de lesión: 0 control, 1 leve, 2 moderado y 3 grave.

Nº pacientes	117
Nº clases	4
Nº imágenes totales	7932
Nº muestras clase 0 (control)	33
Nº muestras clase 1 (leve)	47
Nº muestras clase 2 (moderado)	9
Nº muestras clase 3 (grave)	28

5.2. Resultados de la extracción de características visuales

En la Figura 5.1 se muestran algunos ejemplos de haber aplicado el algoritmo de detección de contornos desarrollado. En la parte izquierda están las imágenes de entrada al algoritmo y en la derecha se encuentran imágenes con los contornos detectados. En gris se pinta el propio contorno, en blanco el interior del propio contorno y en negro el fondo de la imagen.

Se puede apreciar cómo en la primera y última fila, se han descartado los contornos que pertenecen al centro de las muestras. Esto es debido a que en el centro de dichas muestras las estructuras formadas siguen unos patrones más caóticos y, como se explicó en la sección 4.3.1, estos se eliminan debido a los filtrados realizados. En la tercera fila se puede apreciar que no se han descartado los contornos del centro de la muestra. Esto es porque la manera de cristalizar en el centro de dicha muestra ha mantenido una estructura formada por patrones fácilmente identificables, de la misma forma que en la corona. De esta manera, los contornos se han mantenido tras los pasos de filtrado del algoritmo. Como se explicó también en la sección 4.3.1 este fenómeno no se considera como algo problemático ya que el objetivo de la extracción de contornos es identificar aquella información de las muestras que sea característica de las mismas.

Análisis de los resultados del sistema

El desarrollo de este sistema y la selección final de los diferentes parámetros y funciones se hizo a base de aplicar de forma iterativa el algoritmo a distintas imágenes y observar la calidad de los resultados, considerando como criterio el número de contornos bien detectados, extraídos de forma que se ajustan de manera precisa a los patrones de las imágenes, y el filtrado correcto de artefactos y falsos contornos. De esta forma se acabó optando por la aplicación de la umbralización adaptativa en vez de una con umbral fijo, la extracción de contornos en dos pasos y se determinaron los umbrales de filtrado de área y circularidad, que corresponden con un umbral de 2000 píxeles cuadrados en el primer filtrado de área y unos umbrales de 6000 píxeles cuadrados en el segundo filtrado de área y 0.1 como umbral de circularidad tras la segunda extracción de contornos. Tras la realización de estas múltiples pruebas y modificaciones y el establecimiento de

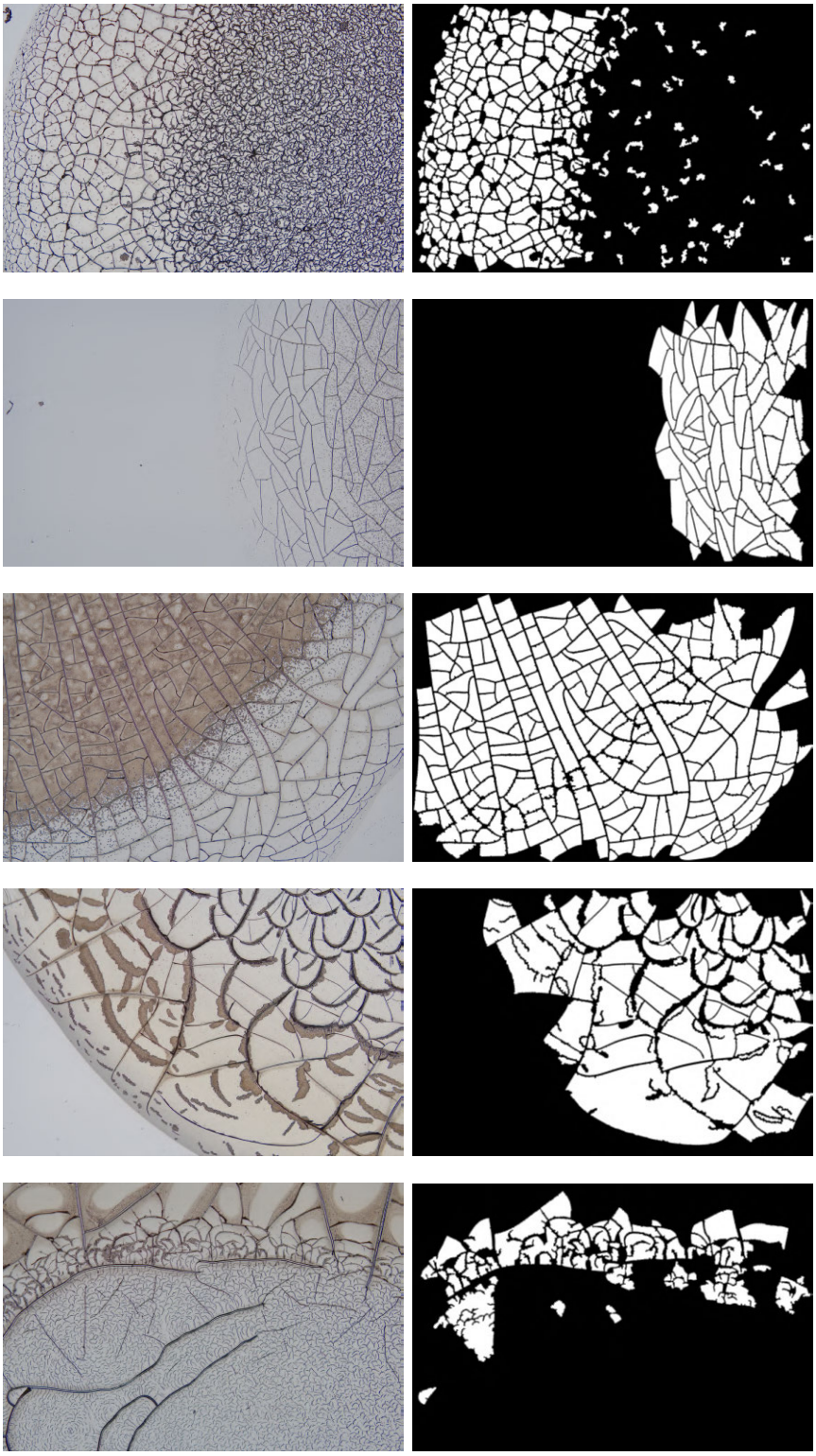


Figura 5.1. Ejemplos de los resultados obtenidos al aplicar el algoritmo de detección de contornos imágenes del dataset. Las imágenes de la izquierda corresponden a las imágenes del dataset, y las de la derecha, a los contornos detectados.

dichos parámetros y funciones finales, se destacan las siguientes características sobre el algoritmo de detección y extracción de contornos.

Como características positivas destacan las siguientes:

- Aplicando la umbralización adaptativa ha sido posible aumentar la precisión de detección a pesar de que la diferencia de intensidad contorno-fondo varíe de forma significativa en distintas ubicaciones de la imagen.
- Aplicando la detección de contornos en dos pasos distintos ha sido posible filtrar la detección errónea de contornos, por ejemplo, detectar el contorno como el exterior en vez del interior del patrón o que se unan dos contornos de patrones contiguos. De esta manera, y analizando los resultados, se puede apreciar que cuando un contorno está bien definido sobre la imagen de origen, su contorno es detectado correctamente por el algoritmo.
- Aplicando los distintos filtrados ha sido posible eliminar una gran cantidad de imágenes que no aportan información útil, como las imágenes en blanco en la que no hay muestra presente, y falsos contornos, mejorando de esta forma la eficiencia computacional y reduciendo el almacenamiento necesario.

Como características negativas, destacan las siguientes:

- Como se puede apreciar en las filas cuatro y cinco de la Figura 5.1, sobre las muestras existen formaciones que deforman, dificultan e, incluso, imposibilitan en algunos casos la detección de los patrones que subyacen a estas formaciones, dando lugar a detecciones que no son del todo correctas. Estas detecciones que no son del todo correctas afectan al siguiente experimento de este PFG que corresponde con la extracción de *features*. Aun así, se puede apreciar que la mayoría de contornos sí son bien identificados.
- Como se puede apreciar en la primera fila de la Figura 5.1, se puede ver la existencia de contornos correspondientes a zonas irregulares de la muestra que han permanecido tras aplicar el filtrado del algoritmo. No se puede escoger un valor óptimo para los umbrales de los filtrados de área y circularidad puesto que a mayor umbral, menos falsas detecciones pero también se descartan detecciones correctas; y a menor umbral, menos contornos correctos se descartan pero aumentan las falsas detecciones. La elección de estos umbrales se ha realizado de manera heurística e intentando compensar las pérdidas que se producen por ambas partes.

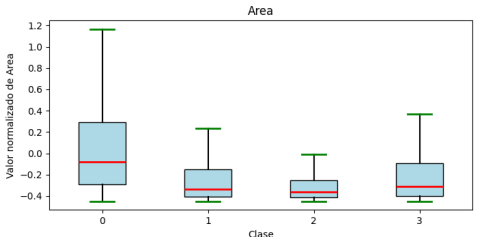
También se analizan los resultados obtenidos sobre la extracción de características visuales y de los contornos y la caracterización de estos mediante vectores de *features* o características. Dichas características se analizan y visualizan para observar tendencias y distribuciones en los datos. Sobre la Figura 5.2 y la Figura 5.3 se puede observar la distribución normalizada según la ecuación 4.2 de algunos elementos de los vectores extraídos para las distintas clases.

En la Figura 5.2 se pueden visualizar las distribuciones de algunas características extraídas para los contornos detectados en las diferentes secciones de imágenes que componen la base de datos de pacientes. Estas características son: área del contorno, circularidad del contorno, momentos primero y quinto de Hu del contorno y momentos primero, cuarto, décimo y decimosexto de Zernike del contorno. Estas características, tras su normalización, se muestran para cada una de las clases que componen el dataset: 0 control (sano), 1 leve, 2 moderado y 3 grave. Sobre las gráficas se muestra la mediana como una línea roja, la distribución de datos que se encuentran desde el cuartil Q_1 hasta el cuartil Q_3 como una rectángulo azul y la distribución de datos que se encuentran entre $Q_1 - 1,5 \times IRQ$[1] hasta Q_1 y entre Q_3 hasta $Q_3 + 1,5 \times IRQ$ como dos líneas negras que se encuentran por debajo y por encima del rectángulo azul respectivamente. Sobre estas gráficas se pueden apreciar unas distribuciones de los datos similares y poco distintivas entre las diferentes clases a excepción del área, en la que se puede apreciar que la clase 0 (control) presenta un número más grande de contornos con áreas mayores al resto de las clases; y el primer momento de Zernike, para el que se puede apreciar una mayor dispersión en el caso de contornos pertenecientes a la clase de control. No obstante, esta última gráfica está escalada por 10^{-12}, lo que indica que la mayoría de valores normalizados son prácticamente 0 y, por ende, los valores sin normalizar son prácticamente iguales para todos los contornos. Por lo que este elemento del vector de características no aporta ningún tipo de información.

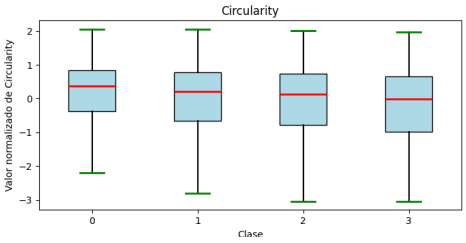
En la Figura 5.3 se pueden visualizar las distribuciones de algunas características extraídas a nivel de *tile*. Los elementos de los gráficos y las clases mostradas son las mismas que en el caso de los contornos. Las características mostradas son: el número de contornos capturados por *tile*, el área media de los contornos capturados por *tile*, la media y desviación estándar de la circularidad de los contornos capturados por *tile* y las componentes de color medias A y B de los distintos *tiles*, todos ellos mostrados para cada clase. De forma similar a las características extraídas por contorno, las distribuciones son muy similares. Como excepciones se puede visualizar el área media de los contornos capturados por *tile*, el cual es mayor para los pacientes de control, caso que resulta coherente con las gráficas extraídas para los contornos; el número de instancias por *tile* que es ligeramente mayor para los pacientes de clase 2 y menor para los de clase 3, y la componente de color media B, la cual es ligeramente mayor para los pacientes de clase 2.

En ambas figuras se puede apreciar que la distribución de los datos extraídos es muy similar entre clases y no existe una tendencia especialmente clara que ofrezca una distinción entre los distintos niveles de lesión. Esta distribución de los datos sugiere que alcanzar resultados positivos mediante el entrenamiento de los modelos de clasificación supone una gran dificultad, ya que estos no proporcionan información diferenciada que puedan representar y caracterizar eficazmente a cada clase. Al no ofrecer variabilidad los datos, los modelos de clasificación apenas tienen información diferenciada de la que pueden aprender para encontrar una correlación con la etiqueta de salida

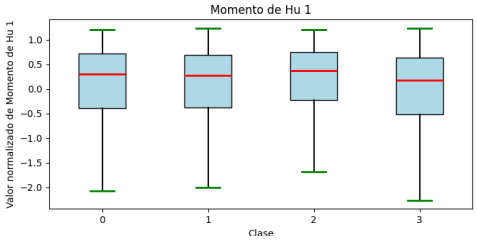
[1] El Rango Inter Cuartil (IRQ) se calcula como la diferencia entre el tercer cuartil Q_3 menos el primer cuartil Q_1.



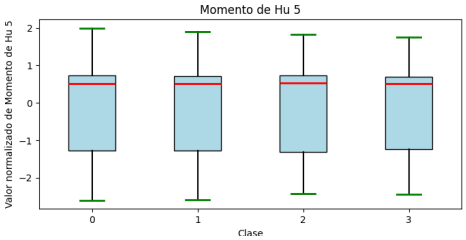
(a) Distribución del área.



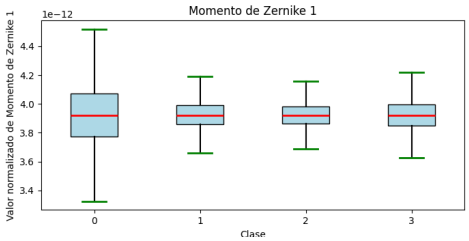
(b) Distribución de la circularidad.



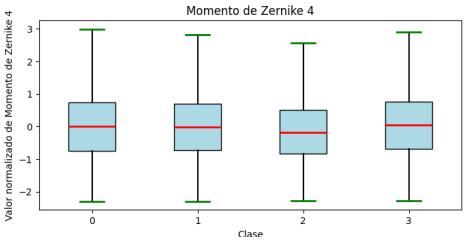
(c) Distribución del momento de Hu 1.



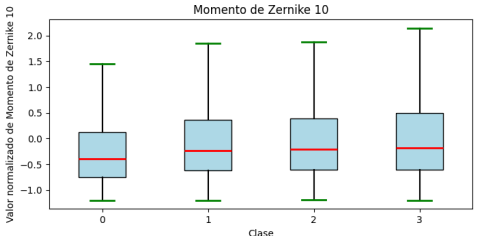
(d) Distribución del momento de Hu 5.



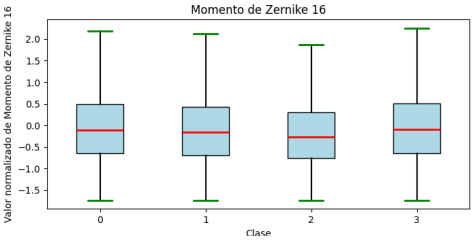
(e) Distribución del momento de Zernike 1.



(f) Distribución del momento de Zernike 4.

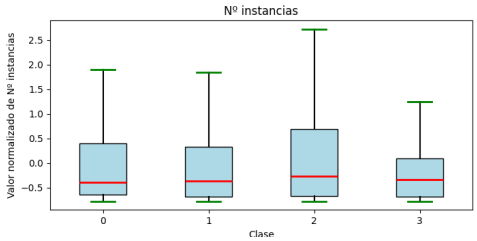


(g) Distribución del momento de Zernike 10.

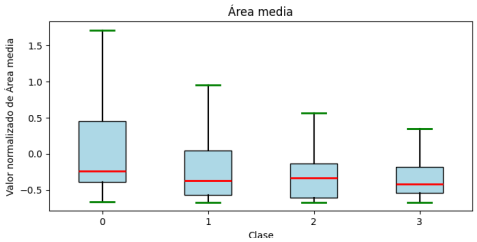


(h) Distribución del momento de Zernike 16.

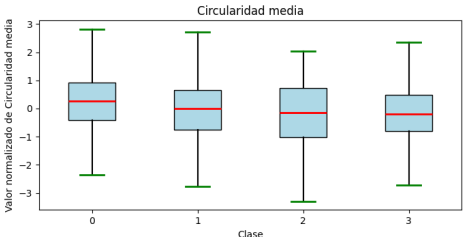
Figura 5.2. Distribución en diagramas de cajas de distintos elementos normalizados de los vectores de características extraídos a nivel de contornos individuales. En el eje horizontal se muestran las distintas clases de los datos: 0 control, 1 leve, 2 moderado y 3 grave; y sobre el eje vertical la escala de los valores normalizados de cada elemento del vector de características. Las subfiguras mostradas corresponden con las distribuciones de: a) área, b) circularidad, c) primer elemento de los momentos de Hu, d) quinto elemento de los momentos de Hu, e) primer elemento de los momentos de Zernike, f) cuarto elemento de los momentos de Zernike, g) décimo elemento de los momentos de Zernike, y h) decimosexto elemento de los momentos de Zernike.



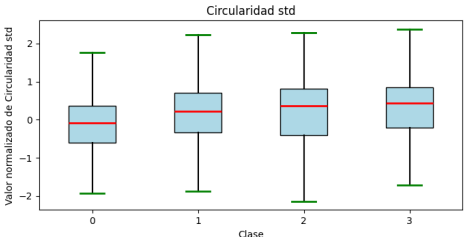
(a) Distribución del número de instancias.



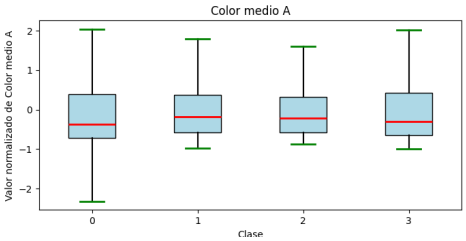
(b) Distribución del área media.



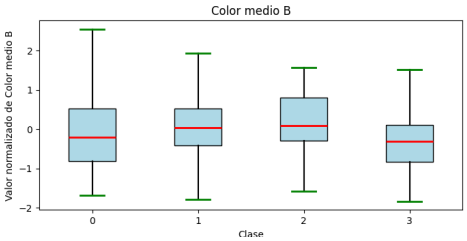
(c) Distribución de la circularidad media.



(d) Distribución de la desviación estándar de la circularidad.



(e) Distribución de A.



(f) Distribución de B.

Figura 5.3. Distribución en diagramas de cajas de distintos elementos normalizados de los vectores de características extraídos a nivel de *tiles*. Sobre el eje horizontal se muestran las distintas clases de los datos: 0 control, 1 leve, 2 moderado y 3 grave. El eje vertical muestra la escala de los valores normalizados para cada elemento. Las subfiguras muestran las distribuciones de: a) número de contornos detectados por sección de imagen, b) área media de los contornos por sección de imagen, c) circularidad media de los contornos por sección de imagen, d) desviación estándar de la circularidad de los contornos por sección de imagen, e) componente de color A media por sección de imagen, y f) componente de color B media por sección de imagen.

Aun así, se procede a realizar los experimentos dado que sigue existiendo la posibilidad de que existan relaciones internas entre las distintas características de las que alguno de los dos modelos pueda aprender. Estos resultados se presentan en la sección siguiente.

5.3. Resultados de los modelos de clasificación

En esta sección se describen los resultados obtenidos a partir del entrenamiento del algoritmo *K-Neighbors* y el SVM con los datos mostrados en la sección anterior. En primer lugar se

presentan los resultados obtenidos para el entrenamiento con los datos a nivel de *tile*, y posteriormente para los datos obtenidos a nivel de contorno, tal y como se describió en los apartados 4.4.2.1 y 4.4.2.2 del capítulo de descripción de la solución propuesta. Finalmente, se muestra una valoración general de los resultados obtenidos para todos los experimentos presentados en esta sección.

5.3.1. Resultados utilizando el dataset de *tiles*

En este apartado se presentan los resultados obtenidos al entrenar los modelos mencionados con los datos obtenidos a nivel de *tile*. La información sobre los datos con los que se realizan los entrenamientos se muestra en la tabla 5.2. La partición de datos de test está conformada por las características de los *tiles* correspondientes al 45 % de pacientes.

Tabla 5.2. Información sobre el dataset generado a nivel de *tile* para el entrenamiento.

Nº pacientes	117
Nº *tiles* totales	5.795
Nº instancias de entrenamiento	3.064
Nº instancias test	2.731
Nº *features*	7

5.3.1.1. Resultados del entrenamiento mediante el algoritmo de *K-Neighbors*

Los resultados obtenidos tras aplicar este experimento se muestran sobre la tabla 5.3.

Tabla 5.3. Resultados de los experimentos hechos a partir del entrenamiento de un clasificador *K-Neighbors* para la clasificación de *tiles*.

k	Acc. muestras de test	Acc. pacientes	Acc. clase 0	Acc. clase 1	Acc. clase 2	Acc. clase 3
2	45,0 %	27/53	15/17	12/19	0/4	0/13

Se puede apreciar que el valor seleccionado en la validación cruzada ha sido $k = 2$. La celda de *Acc. muestras de test* hace referencia a la tasa de aciertos total sobre todas las imágenes de la partición de test. El resto de celdas hacen referencia a la tasa de aciertos utilizando el método de inferencia descrito en el apartado 4.4.2.

Los resultados son relativamente negativos. Mientras que, por ejemplo, para los casos de pacientes de clase 0 (control) y 1 (leves) las tasas de acierto son del 88,2 % y 63,2 % respectivamente, para los casos de pacientes de clase 2 (moderado) y 3 (grave) la tasa de aciertos es de 0, produciendo una tasa de aciertos total del 50,1 %. Esto implica que el clasificador no ha sido capaz de aprender a detectar los pacientes con lesiones moderadas y graves y que no ha adquirido

la capacidad de detectar entre los distintos niveles de lesión. Estos resultados son especialmente negativos para los pacientes de clase 3 los cuales, en un caso real, son los que se encuentran en una situación más crítica y probablemente los que requieren de una precisión de detección mayor.

5.3.1.2. Resultados del entrenamiento de múltiples SVM para la clasificación de *tiles*

La tabla 5.4 muestra los resultados obtenidos para los mejores parámetros obtenidos al realizar la validación cruzada. Los parámetros que se validaron fueron el parámetro de regularización C y el parámetro relativo al *kernel* aplicado γ, tal y como se muestra en la sección 4.4.2 del capítulo de la descripción de la solución propuesta.

Tabla 5.4. Resultados de los experimentos hechos a partir del entrenamiento de múltiples SVM para la clasificación de *tiles*.

C	γ	Acc. muestras de test	Acc. pacientes	Acc. clase 0	Acc. clase 1	Acc. clase 2	Acc. clase 3
1000	$8/M$	$46,3\,\%$	27/53	14/17	13/19	0/4	0/13

Se puede apreciar que los valores seleccionados en el proceso de la validación cruzada han sido $C = 1000$ y $\gamma = 8/M$ siendo M el número de *features* de los datos, que en este caso corresponde con un valor de 7.

De forma similar que en el experimento anterior, los resultados nuevamente son relativamente negativos. Las tasas de acierto para los pacientes de clase 0 y 1 son del 82,3 % y 68,4 % respectivamente, mientras que para los pacientes de clase 2 y 3 la tasa de aciertos es de 0. De nuevo, el modelo no está adquiriendo la capacidad de detección de los pacientes con lesiones moderadas y graves, y, en general, no ha logrado aprender correctamente.

5.3.2. Resultados utilizando el dataset de contornos

En este apartado se muestran los resultados obtenidos tras el entrenamiento de modelos de clasificación con el dataset generado a nivel de contorno. En la tabla 5.5 se muestra información acerca del dataset generado que se usa para los entrenamientos. La partición de datos de test está conformada por los datos relativos al 45 % de los pacientes. En este caso el número de instancias de la partición de test es mayor que el de la partición de entrenamiento. Esto se debe a que el número de contornos por imagen varía de forma significativa de un paciente a otro, resultando que aunque el número de pacientes que formen la partición de test sea menor, el número de contornos total de estos no lo es.

Tabla 5.5. Información sobre el dataset generado a nivel de contorno para el entrenamiento.

Nº pacientes	117
Nº contornos totales	595.633
Nº instancias entrenamiento	293.056
Nº instancias test	302.577
Nº *features*	34

Resultados del entrenamiento mediante el algoritmo de *K-Neighbors*

Los resultados obtenidos tras aplicar este experimento se muestran en la tabla 5.6.

Tabla 5.6. Resultados de los experimentos hechos a partir del entrenamiento de un clasificador *K-Neighbors* para la clasificación de contornos.

k	Acc. muestras de test	Acc. pacientes	Acc. clase 0	Acc. clase 1	Acc. clase 2	Acc. clase 3
9	52,8 %	18/53	13/17	5/19	0/4	0/13

El parámetro k escogido tras la validación cruzada resultó ser de 9. La celda *Acc. muestras de test* hace referencia a la tasa de aciertos sobre las contornos individuales, tratándolos como datos independientes. Las siguientes casillas hacen referencia a las tasas de aciertos totales y de cada clase utilizando la estrategia de inferencia descrita en 4.4.2.

En este caso, los resultados son peores incluso que para el entrenamiento de un *K-Neighbors* utilizando los datos a nivel de *tiles*. La tasa de aciertos total ha disminuido del 50,1 % al 34 %. Esto implica que el algoritmo *K-Neighbors* apenas está encontrando relaciones entre los datos de entrenamiento con los que realizar estimaciones válidas. De forma similar que con los experimentos realizados a nivel de *tile*, la tasa de aciertos para los pacientes de clase 2 y 3 es de 0, y la mayor tasa de aciertos es para los pacientes de clase 0 (sanos), con un valor de 76,5 %, implicando que el 76,5 % de pacientes sanos será detectado pero que la gran mayoría de pacientes con lesiones serán identificados erróneamente.

5.3.2.1. Resultados del entrenamiento de múltiples SVM para la clasificación de contornos

Sobre la tabla 5.7 se muestran los resultados obtenidos para los mejores parámetros obtenidos al realizar la validación cruzada y para los tres experimentos.

En la tabla 5.7 se pueden ver los parámetros seleccionados para C y γ tras aplicar validación cruzada con los distintos tamaños de dataset. El valor M corresponde con el número de *features* que en este caso es de 34. La celda *Acc. muestras de test* hace referencia a la tasa de aciertos sobre los contornos individuales, tratándolos como datos independientes. Las siguientes casillas hacen

Tabla 5.7. Resultados de los experimentos hechos a partir del entrenamiento de múltiples SVM para la clasificación de contornos.

Tamaño subset	C	γ	Acc. muestras de test	Acc. pacientes	Acc. clase 0	Acc. clase 1	Acc. clase 2	Acc. clase 3
4000	10	$\frac{1}{8}/M$	52,8 %	28/53	12/17	16/19	0/4	0/13
10000	1	$2/M$	46,5 %	26/53	12/17	14/19	0/4	0/13
40000	1	$2/M$	45,8 %	26/53	12/17	14/19	0/4	0/13

referencia a las tasas de aciertos totales y de cada clase utilizando la estrategia de inferencia descrita en 4.4.2.

En este experimento se ha obtenido la tasa de aciertos total mayor de los cuatro realizados utilizando esta metodología. Esta tasa de aciertos es del 52,8 % y se ha obtenido para el caso del entrenamiento con una subpartición de los datos de entrenamiento de 4000 instancias, un valor del parámetro de regularización C de 10 y un valor de γ de $\frac{1}{8}/M$. En este caso para los pacientes de clase 0 (sano) y 1 (leve) las tasas de acierto son del 70,6 % y 84,2 % respectivamente. Aun así, para los pacientes de clase 2 (moderado) y 3 (grave) las tasas de acierto son de 0. Esto implica que ninguno de los cuatro experimentos realizados con esta metodología ofrecen resultados positivos, y ninguno de ellos ha sido capaz de obtener una tasa de aciertos para pacientes con lesiones moderadas o graves mayor de 0.

5.3.3. Análisis de los resultados

Como se puede apreciar en las tablas 5.3, 5.4, 5.6 y 5.7, los resultados son relativamente negativos utilizando esta metodología. La mayor tasa de aciertos en los pacientes de test es de 28/52 y se ha obtenido para el caso del entrenamiento de múltiples SVMs aplicados al dataset conformado por los vectores de características extraídos de los contornos. Esto corresponde con el 52,8 % de los pacientes. Si se compara este rendimiento con el que se obtendría usando un *dummy classifier*[2], en este caso sería de 19/53 o 35,8 %. Se puede observar que el rendimiento obtenido es mejor, lo que implica que el modelo sí ha conseguido aprender ligeramente de los datos. Aun así, los resultados obtenidos muestran que esta técnica no es eficiente, por si sola, para lograr el sistema de predicción del diagnóstico planteado en la introducción de este PFG.

Analizando estos resultados, se estima que algunas de las causas de su bajo rendimiento pueden ser las siguientes:

- Como se explicó en la sección de resultados 5.2, el algoritmo de detección de contornos

[2]Por *dummy classifier* se hace referencia a un tipo de clasificador que siempre devuelve la etiqueta de la clase mayoritaria en los datos. El nombre que recibe se debe a que este clasificador no aprende de los datos que se le ingesta y se suele usar como referencia del peor rendimiento posible obtenido por un modelo (sin contar el caso de un modelo que siempre responda aleatoriamente).

obtuvo unos resultados subjetiva y visualmente positivos, pero no resultan buenos para la representación precisa de todos y cada uno de los contornos. Aunque, tras analizar sus resultados, se puede observar que la mayoría de contornos han sido correctamente extraídos y representados, sigue existiendo un número elevado de detecciones erróneas. Esto se puede deber principalmente a dos razones: La primera, que es debido a la naturaleza de las muestras que de por sí presentan estructuras y formaciones superpuestas a los patrones de cristalización, dificultando el proceso de extracción de contornos. La segunda, que este resultado es debido a los métodos utilizados para la extracción de contornos, los cuales no son lo suficientemente eficientes para hacer esta extracción precisa y detectar aquellos contornos erróneos. Es posible que para lograr resolver esta tarea sea necesario generar modelos más avanzados de segmentación y estimación de formas, que puedan ser entrenados y evaluados con el mismo dataset de imágenes.

- Aun realizando una detección y extracción óptima de los contornos y su representación, no está claro que éstas sean las propiedades visuales de las imágenes que caractericen de forma distintiva los distintos grados de lesión cerebral. Esta conclusión se extrae tras el análisis de la distribución de los distintos elementos de las características mostradas sobre la Figura 5.2 y la Figura 5.3 de la sección 5.2.

- La correspondencia características extraídas-paciente es difusa y compleja. De media, se han extraído 5,091 contornos aproximadamente por cada paciente, presentándose variaciones muy significativas en los tamaños, circularidad y formas que presentan dichos contornos para un solo paciente. Debido a este gran número de instancias por muestra y a su variabilidad, es muy complejo hallar una forma óptima de correlacionar todas las instancias extraídas y sus características asociadas con un único paciente.

Los resultados obtenidos en esta sección fueron los que motivaron a investigar y desarrollar un método alternativo para hallar una clasificación eficiente, y a descartar esta metodología de clasificación. Este método alternativo es el descrito en la sección 4.5, y sus resultados se muestran en la siguiente sección.

5.4. Resultados obtenidos con la CNN *EfficientNet-B7*

En este apartado se describen los resultados obtenidos con una versión ajustada de la red *EfficientNet-B7* con las imágenes del dataset generado para este PFG. Se han realizado dos experimentos. El primero consiste en el entrenamiento de únicamente las capas de clasificación de la CNN, mientras el resto de capas dedicadas a la extracción de *features* se mantienen congeladas. En el segundo experimento se entrenan, además de las capas de clasificación, los módulos 5, 6, 7 y 8 de la red relativos a la extracción de *features* a la vez que se aplican técnicas de *Data Augmentation* a las imágenes de entrenamiento. Estos procedimientos se describieron en la sección 4.5 del capítulo de descripción de la solución propuestas.

Los datos con los que se realizaron estos experimentos se muestran sobre la tabla 5.8.

Tabla 5.8. Información de los datos con los que se realizaron los experimentos relativos a aplicar *finetuning* a la red *EfficientNet-B7*.

Nº de pacientes totales	117
Nº imágenes totales	13.980
Nº pacientes entrenamiento	79
Nº imágenes entrenamiento	9.290
Nº pacientes evaluación	14
Nº imágenes evaluación	1.887
Nº pacientes test	24
Nº imágenes test	2.803

Primer experimento

El primer experimento consiste en entrenar únicamente la última capa lineal de la CNN, correspondiente a la capa de clasificación. Se define un número de *epochs* de 10 y un tamaño de *batch* de 24.

En la tabla 5.9 se muestran las tasas de acierto finales obtenidas con el modelo sobre la partición de test tras las 10 épocas de entrenamiento.

Tabla 5.9. Tasas de acierto obtenidos con el modelo tras el primer experimento.

T.A. total	[%]	T.A. clase 0	T.A. clase 1	T.A. clase 2	T.A. clase 3
13/24	52,2 %	7/8	3/7	0/2	3/7

Como se puede apreciar, las tasas de acierto son bastante bajas. Esto puede ser debido a que entrenar únicamente la capa de clasificación puede no ser una buena estrategia en este caso, puesto que, como se explicó acerca del *Transfer Learning* en el apartado 2.5.2, se obtendrá un mejor rendimiento al aplicar *finetuning* a un modelo cuanto más se parezcan el dataset con el que se va a entrenar y el dataset con el que se entrenó originalmente. En este caso, el modelo *EfficientNet-B7* fue entrenado con el dataset *ImageNet-1K*. Este dataset está conformado por imágenes de objetos y animales generales de internet. Al aplicar esta técnica en este caso no se están aprovechando del todo las capacidades adquiridas por la CNN en el entrenamiento previo, por lo que es necesario entrenar un número mayor de capas que permitan a la red aprender las características de nuevos datos.

Segundo experimento

En este segundo experimento se entrenan la última capa lineal y las capas pertenecientes a los bloques *features.5*, *features.6*, *features.7* y *features.8*. Se define un número de *epochs* máximo de 50 y un tamaño de *batch* de 8. Dada la posibilidad de que ocurra *overfitting* por el número reducido

de datos con el que se está entrenando el modelo, al finalizar cada época del entrenamiento se compara el rendimiento del modelo con los pesos que tiene en ese preciso instante con los rendimientos obtenidos por los modelos en épocas anteriores y, si mejora, se registra el estado del modelo en diferentes archivos.

En la tabla 5.10 se muestran las tasas de acierto finales obtenidas con los pesos del modelo en la época de entrenamiento que mejor rendimiento ha obtenido.

Tabla 5.10. Tasas de acierto obtenidos con el modelo tras el segundo experimento.

T.A. total	[%]	T.A. clase 0	T.A. clase 1	T.A. clase 2	T.A. clase 3
18/24	75 %	8/8	5/7	0/2	5/7

Como se puede apreciar, el rendimiento del modelo ha mejorado significativamente. En este caso se están entrenando la capa de clasificación y los últimos cuatro módulos de extracción de *features* del modelo. De esta forma, se están aprovechando las capacidades aprendidas por la CNN en el entrenamiento previo en las primeras capas convolucionales, las cuales detectan patrones sencillos como líneas, rectas o ángulos, mientras que las últimas capas del modelo son las que detectan patrones más complejos y abstractos a partir de las detecciones de las capas previas. Al entrenar estas capas se está capacitando al modelo de aprender a representar la información contenida por las imágenes del dataset, obteniendo así un rendimiento mayor que en el caso de entrenar únicamente la última capa del modelo.

Analizando los resultados se puede afirmar que el entrenamiento de una CNN con imágenes de muestras de suero sanguíneo secas puede ser una metodología mucho más acertada para desarrollar el sistema de predicción del diagnóstico descrito en la introducción de este proyecto. Las estructuras formadas sobre las muestras son numerosas y complejas, haciéndose difícil caracterizarlas mediante el método descrito en el apartado 4.3 del capítulo de la descripción propuesta, basado principalmente en la detección y caracterización de los contornos. Sin embargo, una CNN ofrece una funcionalidad más robusta para elaborar el sistema de clasificación ya que elimina la dependencia a dicha extracción de contornos y caracterización previa de las propiedades visuales de las imágenes de las muestras. La arquitectura de la CNN ofrece una versatilidad tal que aprende a representar las estructuras de las imágenes por sí misma, pudiendo caracterizar y clasificar información más complejas. Además, estas redes son capaces de aprender patrones y estructuras a nivel global de la imagen mientras que con las técnicas de extracción de las características a nivel de contorno la caracterización se hace individualmente sobre cada uno de los contornos presentes en la imagen, perdiendo de esta forma el análisis de estructuras más amplias que pueden ser útiles para elaborar la predicción.

Sin embargo, a pesar de estos resultados positivos, el número de datos con el que se entrena y valida la red es muy reducido. Dadas las limitaciones del proyecto, la base de datos consta únicamente de 117 pacientes distintos. Para elaborar un sistema de predicción basado en el entrenamiento de una CNN se necesita un número mucho mayor de datos. De esta manera la red generalizaría más al tener un número mayor de datos de entrenamiento. También es importante

tener un número mayor de datos de evaluación, ya que, aunque en este caso la partición de test constase de 2.803 imágenes, el número de pacientes es de 24. Una diferencia en la tasa de aciertos de un paciente implica una diferencia del 4 % de la precisión obtenida. Conforme mayor es el número de datos de evaluación, menos sensibilidad hay a estos pequeños cambios.

Capítulo 6

Presupuesto

El presupuesto general de este proyecto se divide entre gastos de personal y gastos asociados a los recursos utilizados. En la tabla 6.1 se muestra la estimación del coste de personal asociado al desarrollo de este PFG. La estimación se realiza en función del sueldo medio que recibe un ingeniero con la experiencia que tiene cada una de las personas involucradas. En la tabla 6.2 se muestra el gasto asociado a los recursos utilizados para realizar este PFG. El precio de algunos materiales utilizados se estima mediante el coste aproximado por hora que tienen actualmente si se recurre al alquiler a terceros. Por último, la tabla 6.3 recoge el gasto total.

Tabla 6.1. Estimación del coste de personal asociado al desarrollo de este PFG.

	Nº total de horas	Estimación €/h	Subtotal
Desarrollador	360	17,4	6.264,00 €
Director del proyecto	60	20,9	1.254,00 €
Supervisor del proyecto	35	24,3	850,50 €
		Gasto total:	8.368,50 €

Tabla 6.2. Estimación del coste asociado al uso de recursos para el desarrollo de este PFG.

	Coste por unidad	Unidades	Subtotal
Puesto de trabajo	1.000 €	1	1.000,00 €
GPU	1,20 €/h	60	72,0 €
Sistema de captura	40 €/h	40	1.600,00 €
		Gasto total:	2.672,00 €

Tabla 6.3. Estimación total del presupuesto asociado a este PFG.

Gasto personal	8.368,50 €
Gasto material	2.672,00 €
Total:	11.040,50 €

El gasto total del proyecto asciende a 11.040,50 €.

Capítulo 7

Conclusiones

En este capítulo se analizan los resultados obtenidos para los diferentes desarrollos realizados en este PFG evaluando si se han cumplido los objetivos planteados. Posteriormente, se proponen líneas de trabajo futuras relacionadas con el desarrollo de este PFG.

7.1. Conclusiones del proyecto

El objetivo inicialmente planteado en la introducción de este PFG consiste en el desarrollo e implementación de distintas técnicas experimentales con el fin de desarrollar un sistema de diagnóstico del daño cerebral que sufre un paciente, contrastando los resultados y desempeño de las técnicas utilizadas. En este proyecto se han planteado dos propuestas para conseguir dicho fin:

- Un clasificador basado en las características visuales de las imágenes extraídas por medio de un sistema de detección y extracción de contornos.
- Un clasificador basado en una red neuronal convolucional, entrenado mediante la técnica de *finetuning* con secciones de las imágenes capturadas de las muestras.

Analizando ambas implementaciones, se puede observar que mediante el sistema basado en una CNN se ha obtenido una tasa de acierto del 75 % para las cuatro clases del dataset: control, leve, moderado y grave; mientras que con el primer sistema, la mayor tasa de aciertos obtenida es del 52,8 %, resultado que no ofrece una proyección muy positiva hacia el desarrollo del sistema de diagnóstico planteado. A continuación, se discute sobre los aspectos que pueden haber estado involucrados en los resultados obtenidos con ambos sistemas:

El primer sistema requiere de una etapa de extracción de características visuales que debe estar diseñado de forma que sus resultados representen de forma precisa las propiedades visuales de las muestras. Debido a la naturaleza de las muestras, sobre las que existen distintas estructuras,

formaciones y artefactos, la extracción precisa de los contornos se ve dificultada considerablemente, de forma que los datos extraídos no son fieles al 100 % de la verdadera naturaleza de las imágenes.

Otra razón que puede estar relacionada con los resultados obtenidos en este primer sistema es si las características extraídas por cada contorno realmente son capaces de representar y diferenciar el grado de lesión cerebral. Dada la investigación realizada acerca de las propiedades de las muestras de plasma sanguíneo seco y su posible relación con diferentes estados de salud del paciente, ámbito el cual se presentó en el apartado 2.1 del marco teórico, el objetivo de este sistema es el de extraer datos numéricos que caractericen las propiedades de las estructuras formadas sobre las distintas muestras. No obstante, sobre la Figura 5.2 y la Figura 5.3 se puede apreciar que los datos extraídos apenas aportan variabilidad entre las distintas clases del dataset, lo que dificulta gravemente el entrenamiento exitoso de alguna técnica o modelo basado en aprendizaje automático. Analizando estos resultados se estima que los relativamente bajos resultados obtenidos del 52,8 % se deben más bien a la etapa de extracción de características que a la aplicación de los modelos de aprendizaje automático. El sistema de detección y extracción de contornos se considera que puede servir como herramienta para extracción y clasificación de patrones más simples o como un complemento de ayuda visual, pero, al menos en este punto, no para conseguir el objetivo planteado en este proyecto con suficiente confizana.

Con respecto al segundo entrenamiento, basado en aplicar *finetuning* a la CNN *EfficientNet-B7*, los resultados obtenidos son razonablemente positivos. La tasa de aciertos fue del 75 %, la cual supera por un 22,2 % a la obtenida mediante el primer sistema. Estos resultados se estiman que pueden ser debido a que las CNN son capaces de extraer y representar patrones mucho más complejos que los caracterizados por medio del uso de descriptores. Otra diferencia clave es que mediante el primer sistema las características extraídas caracterizan a los contornos de manera individual, mientras que la CNN también es capaz de extraer características a una escala mayor e, incluso, global de la imagen.

No obstante, los experimentos se han realizado sobre una base de datos de 117 pacientes, de los cuales únicamente 24 se han utilizado para la evaluación del modelo. Aunque un resultado en la tasa de aciertos del 75 % ofrece proyecciones positivas hacia la efectividad de este método, el número de pacientes de prueba y, en general, de la base de datos de pacientes es limitado como para extrapolar estos resultados a un ámbito más general. Sería necesario repetir la evaluación y, posiblemente, el entrenamiento del modelo con una base de datos mayor, para contrastar si los resultados obtenidos son estables y replicables, y minimizar así el posible margen de error que se pueda deber a predicciones aleatorias.

Por otro lado, es necesario añadir algunas consideraciones con respecto al despliegue e implementación de este sistema en procesos reales. El entrenamiento del modelo puede ser un procedimiento largo y costoso computacionalmente que, además, requiere la captura de un gran número de imágenes de muestras de pacientes para realizar un entrenamiento exitoso. Estos

factores, sobre todo el último, pueden hacer que el entrenamiento del modelo sea un proceso laborioso. Una vez entrenado, para desplegar e implementar el modelo en un sistema real se necesitaría de una unidad de procesamiento con la suficiente memoria como para almacenar los parámetros con los que se realizará la inferencia, aunque, para hacer de éste un sistema versátil y portable, existen opciones como la de realizar inferencia en remoto, de manera que el modelo es albergado en un servidor que puede estar ubicado en una sala concreta del hospital o, incluso, mediante servicios *cloud* que permitan albergar el modelo y utilizarlo a cambio de un precio estipulado.

7.2. Posibles líneas futuras

En este apartado se proponen dos posibles líneas de investigación futuras que puedan mejorar el desempeño del sistema basado en la CNN, puesto que es el que ha obtenido mejor desempeño entre ambos sistemas. Estas líneas de investigación futuras van en relación con las problemáticas desecritas en el apartado anterior y en la sección 5.4.

- Realizar un entrenamiento de la CNN con una base de datos de muestras de pacientes mayor. De esta forma se podrían obtener unos resultados con un margen de error menor, hecho que es una problemática en este proyecto y viene derivada de una de las restricciones del mismo que es el tamaño del dataset de pacientes. Sería necesario evaluar el modelo con una partición de datos de evaluación mayor que la que ha sido utilizada en este proyecto para contrastar los resultados obtenidos y comprobar su verdadera efectividad.
- Contrastar con otras arquitecturas y tamaños de imágenes menores. Aunque pueda parecer contraproducente, hay ocasiones en los que modelos con arquitecturas más simples pueden generalizar mejor ante problemas de clasificación debido a que la capacidad de ajuste fino a los datos de entrenamiento, hecho que generalmente deriva en *overfitting*, es menor. *EfficientNet-B7* es un modelo grande y pesado, formado por una arquitectura compleja, el cual fue escogido según las condiciones mostradas en el apartado 4.5.1 del capítulo de descripción de la solución propuesta. Realizar un experimento con un tamaño de imágenes de entrada menor y una red más simple podría ofrecer resultados interesantes.

Capítulo 8

Impacto social y ambiental

Este proyecto tiene la pretensión de crear un impacto positivo en temas sociales y sanitarios. Aunque en él no se haya diseñado un sistema completamente robusto y utilizable, pretende arrojar luz sobre la investigación referente al desarrollo de nuevas pruebas de diagnóstico alternativas del grado de lesión cerebral. El desarrollo de estas pruebas tendría un impacto positivo en el ámbito socioeconómico, debido a la creación de pruebas de diagnóstico más económicas y accesibles en aquellos lugares en los que no existen los recursos necesarios debido a limitaciones tecnológicas, económicas o geográficas. También implicaría una mejora en el sistema sanitario y concretamente sobre el proceso y protocolo de evaluación del grado de lesión cerebral, creándose de esta forma una prueba de triaje que permitiría reducir costes económicos y evitar problemas potenciales de salud.

En relación con el impacto medioambiental, se considera que el desarrollo de las tecnologías en las que se encuentra enmarcado este proyecto podría producir un impacto positivo a largo plazo debido a la reducción en el número de pruebas TAC o de otras tecnologías de imagen médica que se produciría, reduciendo el impacto ecológico asociado a la realización de estas pruebas. No obstante, en el periodo de desarrollo de estas pruebas se puede ver involucrado el entrenamiento de modelos de inteligencia artificial, los cuales generalmente requieren utilizar muchos recursos de computación y unidades de procesamiento gráficos (GPUs) de alta gama. La utilización de estos recursos de computación de alto rendimiento tiene asociado un gran impacto en la huella de carbono, por ende, el desarrollo de estas tecnologías se debe hacer de forma sostenible y responsable, intentando reducir al máximo y siempre que sea posible el uso prolongado de estas unidades de alto rendimiento, y valorando en todo momento si el uso de estos recursos compensa el impacto medioambiental asociado.

El trabajo en áreas relacionadas con este proyecto tiene como consecuencias el desarrollo de nuevas tecnologías e infraestructuras, necesarias para llevar a cabo la labor de investigación. Los avances logrados en este ámbito, no sólo implicarán mejoras el ámbito sanitario, sino que también tendrá un impacto positivo sobre otros campos, dado que en la creación de pruebas de este tipo

hay asociados desarrollos en el área de procesado de imagen, visión por ordenador e inteligencia artificial que pueden ser usados para otros proyectos como, por ejemplo, adaptación sensorial de personas con discapacidad o mejora en la seguridad a través de biomarcadores capturados por imagen.

Por estas razones, se considera que este PFG está alineado con los Objetivos de Desarrollo Sostenible (ODS) definidos por la Asamblea General de las Naciones Unidas. Entre los distintos objetivos planteados, se puede destacar la aportación a los siguientes:

- **Objetivo 3. Salud y bienestar.** El desarrollo de este proyecto y la investigación en el marco en el que se encuentra tiene un impacto directo sobre la salud y bienestar a través de la mejora de los procesos de diagnóstico de los sistemas sanitarios y la creación de pruebas con mayor accesibilidad.
- **Objetivo 9. Industria, innovación e infraestructura.** El desarrollo de este proyecto y otros enmarcados en el mismo contexto tienen como consecuencia una mejora de las tecnologías de imagen y las tecnologías relacionadas con el aprendizaje automático, generando nuevas herramientas e infraestructuras reutilizables en otros ámbitos.
- **Objetivo 10. Reducción de las desigualdades.** El desarrollo de una prueba de diagnóstico alternativa, tal y como se plantea en la introducción de este proyecto, puede aumentar la accesibilidad a pruebas de evaluación médicas en zonas, ciudades y países que sufran limitaciones en los recursos disponibles.
- **Objetivo 12. Producción y consumo responsable.** El desarrollo de una prueba de diagnóstico como la planteada en este proyecto tendría como consecuencia la reducción de numerosas pruebas de diagnóstico médico como el TAC o la RM, reduciendo de esta forma la huella de carbono asociada y el consumo de recursos, así como el coste económico, que implica la realización de estas pruebas.

Figura 8.1. Objetivos definidos en los ODS en los que se considera que este proyecto y similares pueden aportar un impacto positivo.

Bibliografía

[1] R. Chen, L. Zhang, D. Zang y W. Shen, "Blood drop patterns: Formation and applications," *Advances in Colloid and Interface Science*, vol. 231, págs. 1-14, 2016, ISSN: 0001-8686. DOI: https://doi.org/10.1016/j.cis.2016.01.008. dirección: https://www.sciencedirect.com/science/article/pii/S000186861530018X.

[2] L. Hamadeh, S. Imran, M. Bencsik, G. R. Sharpe, M. A. Johnson y D. J. Fairhurst, "Machine Learning Analysis for Quantitative Discrimination of Dried Blood Droplets," *Scientific Reports*, vol. 10, n.º 1, pág. 3313, feb. de 2020, ISSN: 2045-2322. DOI: 10.1038/s41598-020-59847-x. dirección: https://doi.org/10.1038/s41598-020-59847-x.

[3] A. G. Theakstone, P. M. Brennan, K. Ashton et al., "Vibrational Spectroscopy for the Triage of Traumatic Brain Injury Computed Tomography Priority and Hospital Admissions," *Journal of Neurotrauma*, vol. 39, n.º 11-12, págs. 773-783, 2022, PMID: 35236121. DOI: 10.1089/neu.2021.0410. eprint: https://doi.org/10.1089/neu.2021.0410. dirección: https://doi.org/10.1089/neu.2021.0410.

[4] S. Ortega, M. Halicek, H. Fabelo, G. M. Callico y B. Fei, "Hyperspectral and multispectral imaging in digital and computational pathology: a systematic review [Invited]," *Biomed. Opt. Express*, vol. 11, n.º 6, págs. 3195-3233, jun. de 2020. DOI: 10.1364/BOE.386338. dirección: https://opg.optica.org/boe/abstract.cfm?URI=boe-11-6-3195.

[5] E. Gomez-Gonzalez, B. Fernández Muñoz, A. Barriga-Rivera et al., "Hyperspectral image processing for the identification and quantification of lentiviral particles in fluid samples," *Scientific Reports*, vol. 11, ago. de 2021. DOI: 10.1038/s41598-021-95756-3.

[6] Wikipedia, *Microscopio óptico — Wikipedia, La enciclopedia libre*, [Internet; descargado 15-octubre-2024], 2024. dirección: https://es.wikipedia.org/w/index.php?title=Microscopio_%C3%B3ptico&oldid=163030473.

[7] Keyence, *Iluminacion transmitida | Tecnicas de iluminacion de los microscopios | Glosario de microscopios | KEYENCE Mexico — keyence.com.mx*, https://www.keyence.com.mx/ss/products/microscope/microscope_glossary/lighting/transmitted_illumination.jsp, [Accessed 20-10-2024].

[8] Itseez, *Open Source Computer Vision Library*, https://github.com/itseez/opencv, 2015.

[9] W. Li, Q. Huang y S. Gautam, "Contour Feature Extraction of Medical Image Based on Multi-Threshold Optimization," English, *Mobile Networks and Applications*, vol. 26, n.º 1, págs. 381-389, feb. de 2021, Copyright - © Springer Science+Business Media, LLC, part of Springer Nature 2020; Última actualización - 2023-11-29. dirección: https://www.proquest.com/scholarly-journals/contour-feature-extraction-medical-image-based-on/docview/2505799776/se-2.

[10] *OpenCV: Morphological Transformations — docs.opencv.org*, https://docs.opencv.org/4.x/d9/d61/tutorial_py_morphological_ops.html, [Accessed 24-08-2024].

[11] G. Shrivakshan y C. Chandrasekar, "A comparison of various edge detection techniques used in image processing," *International Journal of Computer Science Issues (IJCSI)*, vol. 9, n.º 5, pág. 269, 2012.

[12] Wikipedia contributors, *Canny edge detector — Wikipedia, The Free Encyclopedia*, [Online; accessed 27-August-2024], 2024. dirección: https://en.wikipedia.org/w/index.php?title=Canny_edge_detector&oldid=1235653606.

[13] *OpenCV: Contours : Getting Started — docs.opencv.org*, https://docs.opencv.org/3.4/d4/d73/tutorial_py_contours_begin.html, [Accessed 28-08-2024].

[14] D. Han, Y. Gao, G. Wu, P.-T. Yap y D. Shen, "Robust Anatomical Landmark Detection for MR Brain Image Registration," en *Medical Image Computing and Computer-Assisted Intervention – MICCAI 2014*, P. Golland, N. Hata, C. Barillot, J. Hornegger y R. Howe, eds., Cham: Springer International Publishing, 2014, págs. 186-193, ISBN: 978-3-319-10404-1.

[15] M.-K. Hu, "Visual pattern recognition by moment invariants," *IRE Transactions on Information Theory*, vol. 8, n.º 2, págs. 179-187, 1962. DOI: 10.1109/TIT.1962.1057692.

[16] R. R. Isnanto, A. A. Zahra y P. Julietta, "Pattern recognition on herbs leaves using region-based invariants feature extraction," en *2016 3rd International Conference on Information Technology, Computer, and Electrical Engineering (ICITACEE)*, 2016, págs. 455-459. DOI: 10.1109/ICITACEE.2016.7892491.

[17] G. Duan, X. Zhao, A. Chen e Y. Liu, "An improved Hu moment invariants based classfication method for watermarking algorithm," en *ICINS 2014 - 2014 International Conference on Information and Network Security*, 2014, págs. 205-209. DOI: 10.1049/cp.2014.1287.

[18] Y. Zhang, S. Wang, P. Sun y P. Phillips, "Pathological brain detection based on wavelet entropy and Hu moment invariants," *Bio-Medical Materials and Engineering*, vol. 26, S1283-S1290, 2015, s1, ISSN: 1878-3619. DOI: 10.3233/BME-151426. dirección: https://doi.org/10.3233/BME-151426.

[19] A. Khotanzad e Y. Hong, "Invariant image recognition by Zernike moments," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 12, n.º 5, págs. 489-497, 1990. DOI: 10.1109/34.55109.

[20] C. Singh, N. Mittal y E. Walia, "Face recognition using Zernike and complex Zernike moment features," *Pattern Recognition and Image Analysis*, vol. 21, págs. 71-81, 2011.

[21] K. Otiniano-Rodríguez, G. Camara-Chavez y D. Menotti, "Hu and Zernike Moments for Sign Language Recognition," en *Computer Science*, 2012. dirección: `https://api.semanticscholar.org/CorpusID:207809630`.

[22] D. Patil, S. Krishnan y S. Gharge, "Medical Image Retrieval by Region Based Shape Feature For CT Images," en *2019 International Conference on Machine Learning, Big Data, Cloud and Parallel Computing (COMITCon)*, 2019, págs. 155-159. DOI: `10.1109/COMITCon.2019.8862446`.

[23] C. T. Zahn y R. Z. Roskies, "Fourier descriptors for plane closed curves," *IEEE Transactions on computers*, vol. 100, n.º 3, págs. 269-281, 1972.

[24] E. Persoon y K.-S. Fu, "Shape discrimination using Fourier descriptors," *IEEE Transactions on systems, man, and cybernetics*, vol. 7, n.º 3, págs. 170-179, 1977.

[25] A. BenKhlifa y F. Ghorbel, "An almost complete curvature scale space representation: Euclidean case," *Signal Processing: Image Communication*, vol. 75, págs. 32-43, 2019, ISSN: 0923-5965. DOI: `https://doi.org/10.1016/j.image.2019.03.009`. dirección: `https://www.sciencedirect.com/science/article/pii/S0923596518306921`.

[26] R. Hu y J. Collomosse, "A performance evaluation of gradient field hog descriptor for sketch based image retrieval," *Computer Vision and Image Understanding*, vol. 117, n.º 7, págs. 790-806, 2013.

[27] G. C. McDonald, "Ridge regression," *Wiley Interdisciplinary Reviews: Computational Statistics*, vol. 1, n.º 1, págs. 93-100, 2009.

[28] J. Ranstam y J. A. Cook, "LASSO regression," *Journal of British Surgery*, vol. 105, n.º 10, págs. 1348-1348, 2018.

[29] B. M. Kandel, D. A. Wolk, J. C. Gee y B. Avants, "Predicting Cognitive Data from Medical Images Using Sparse Linear Regression," en *Information Processing in Medical Imaging*, J. C. Gee, S. Joshi, K. M. Pohl, W. M. Wells y L. Zöllei, eds., Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, págs. 86-97, ISBN: 978-3-642-38868-2.

[30] O. Salem, A. Guerassimov, A. Mehaoua, A. Marcus y B. Furht, "Anomaly detection in medical wireless sensor networks using SVM and linear regression models," *International Journal of E-Health and Medical Communications (IJEHMC)*, vol. 5, n.º 1, págs. 20-45, 2014.

[31] J.-P. Vert, K. Tsuda y B. Schölkopf, "A primer on kernel methods," 2004. DOI: `https://doi.org/10.7551/mitpress/4057.003.0004`.

[32] C. Tchito Tchapga, T. A. Mih, A. Tchagna Kouanou et al., "Biomedical Image Classification in a Big Data Architecture Using Machine Learning Algorithms," *Journal of Healthcare Engineering*, vol. 2021, n.º 1, pág. 9 998 819, 2021. DOI: https://doi.org/10.1155/2021/9998819. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1155/2021/9998819. dirección: https://onlinelibrary.wiley.com/doi/abs/10.1155/2021/9998819.

[33] P. Tiwari, D. Upadhyay, B. Pant y N. Mohd, "Multiclass classification of disease using cnn and svm of medical imaging," en *International Conference on Advances in Computing and Data Sciences*, Springer, 2022, págs. 88-99.

[34] H. Ng, S. Ong, K. Foong, P.-S. Goh y W. Nowinski, "Medical image segmentation using k-means clustering and improved watershed algorithm," en *2006 IEEE southwest symposium on image analysis and interpretation*, IEEE, 2006, págs. 61-65.

[35] S. Khanmohammadi, N. Adibeig y S. Shanehbandy, "An improved overlapping k-means clustering method for medical applications," *Expert Systems with Applications*, vol. 67, págs. 12-18, 2017.

[36] Y.-c. Wu y J.-w. Feng, "Development and application of artificial neural network," *Wireless Personal Communications*, vol. 102, págs. 1645-1656, 2018.

[37] R. Yamashita, M. Nishio, R. K. G. Do y K. Togashi, "Convolutional neural networks: an overview and application in radiology," *Insights into Imaging*, vol. 9, n.º 4, págs. 611-629, ago. de 2018, ISSN: 1869-4101. DOI: 10.1007/s13244-018-0639-9. dirección: https://doi.org/10.1007/s13244-018-0639-9.

[38] Z. Li, F. Liu, W. Yang, S. Peng y J. Zhou, "A Survey of Convolutional Neural Networks: Analysis, Applications, and Prospects," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 33, n.º 12, págs. 6999-7019, 2022. DOI: 10.1109/TNNLS.2021.3084827.

[39] H. Yu, L. T. Yang, Q. Zhang, D. Armstrong y M. J. Deen, "Convolutional neural networks for medical image analysis: State-of-the-art, comparisons, improvement and perspectives," *Neurocomputing*, vol. 444, págs. 92-110, 2021, ISSN: 0925-2312. DOI: https://doi.org/10.1016/j.neucom.2020.04.157. dirección: https://www.sciencedirect.com/science/article/pii/S0925231221001314.

[40] Y. Ren, Z. He, Y. Deng y B. Huang, "Data Augmentation for Improving CNNs in Medical Image Classification," en *2023 8th International Conference on Intelligent Computing and Signal Processing (ICSP)*, 2023, págs. 1174-1180. DOI: 10.1109/ICSP58490.2023.10248857.

[41] N. Tajbakhsh, J. Y. Shin, S. R. Gurudu et al., "Convolutional Neural Networks for Medical Image Analysis: Full Training or Fine Tuning?" *IEEE Transactions on Medical Imaging*, vol. 35, n.º 5, págs. 1299-1312, 2016. DOI: 10.1109/TMI.2016.2535302.

[42] H. E. Kim, A. Cosa-Linan, N. Santhanam, M. Jannesari, M. E. Maros y T. Ganslandt, "Transfer learning for medical image classification: a literature review," *BMC Medical Imaging*, vol. 22, n.º 1, pág. 69, abr. de 2022, ISSN: 1471-2342. DOI: 10.1186/s12880-022-00793-7. dirección: https://doi.org/10.1186/s12880-022-00793-7.

[43] Z. Shi, H. Hao, M. Zhao et al., "A deep CNN based transfer learning method for false positive reduction," *Multimedia Tools and Applications*, vol. 78, n.º 1, págs. 1017-1033, ene. de 2019, ISSN: 1573-7721. DOI: 10.1007/s11042-018-6082-6. dirección: https://doi.org/10.1007/s11042-018-6082-6.

[44] H. Azizpour, A. S. Razavian, J. Sullivan, A. Maki y S. Carlsson, "Factors of Transferability for a Generic ConvNet Representation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 38, n.º 9, págs. 1790-1802, 2016. DOI: 10.1109/TPAMI.2015.2500224.

[45] G. Rosa-Olmeda, M. Villa, S. Hiller-Vallina, M. Chavarrías, F. Pescador y R. Gargini, "A Microscope Setup and Methodology for Capturing Hyperspectral and RGB Histopathological Imaging Databases," *Sensors*, vol. 24, n.º 17, 2024, ISSN: 1424-8220. DOI: 10.3390/s24175654. dirección: https://www.mdpi.com/1424-8220/24/17/5654.

[46] M. Tan y Q. V. Le, *EfficientNet: Rethinking Model Scaling for Convolutional Neural Networks*, 2020. arXiv: 1905.11946 [cs.LG]. dirección: https://arxiv.org/abs/1905.11946.

[47] *PyTorch — pytorch.org*, https://pytorch.org/, [Accessed 12-10-2024].

[48] *ImageNet — image-net.org*, https://www.image-net.org/, [Accessed 12-10-2024].

[49] D. P. Kingma y J. Ba, *Adam: A Method for Stochastic Optimization*, 2017. arXiv: 1412.6980 [cs.LG]. dirección: https://arxiv.org/abs/1412.6980.

[50] A. Mao, M. Mohri e Y. Zhong, "Cross-entropy loss functions: Theoretical analysis and applications," en *International conference on Machine learning*, PMLR, 2023, págs. 23 803-23 828.

Anexo I

Sobre este anexo se muestran algunos ejemplos de las imágenes capturadas de distintas muestras de suero sanguíneo secas. Con esto se pretende ofrecer una visión general de las distintas muestras de las que se compone la base de datos de imágenes en relación al grado de lesión cerebral del paciente del cual han sido extraídas.

Las imágenes han sido elaboradas mediante un algoritmo ajeno a la realización de este PFG que compone las imágenes completas de las muestras a partir de los distintos *tiles* capturados mediante el sistema de captura.

Ejemplos de muestras de control

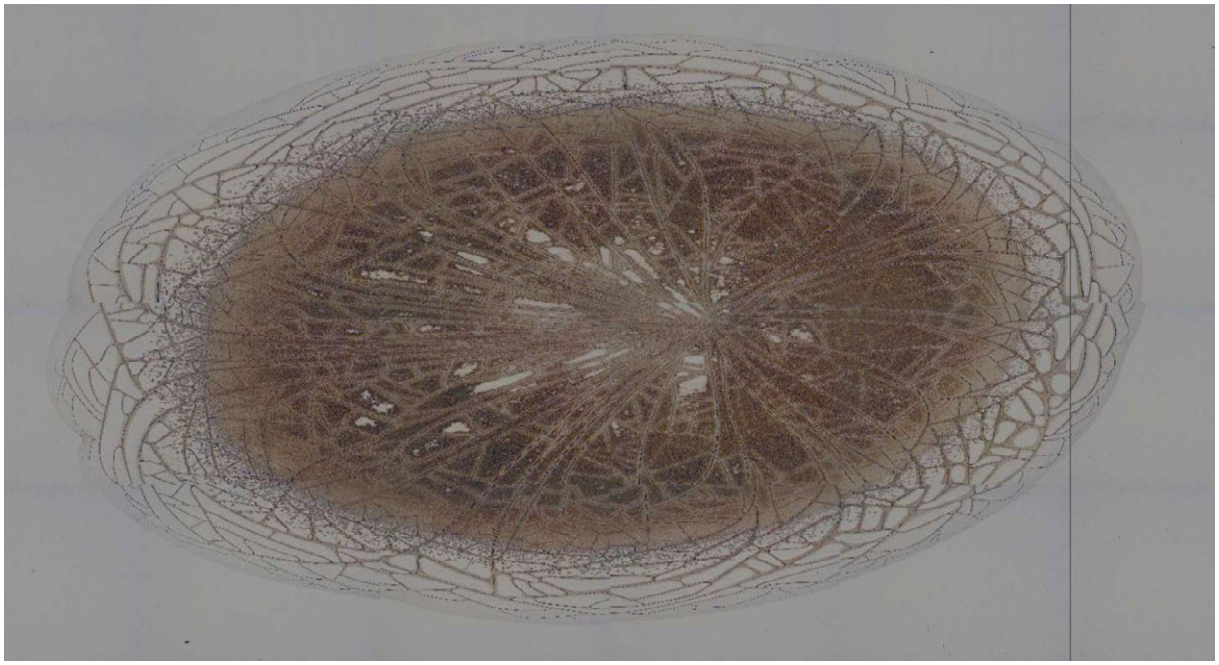


Figura A1. Ejemplo 1 de muestra de control.

Figura A2. Ejemplo 2 de muestra de control.

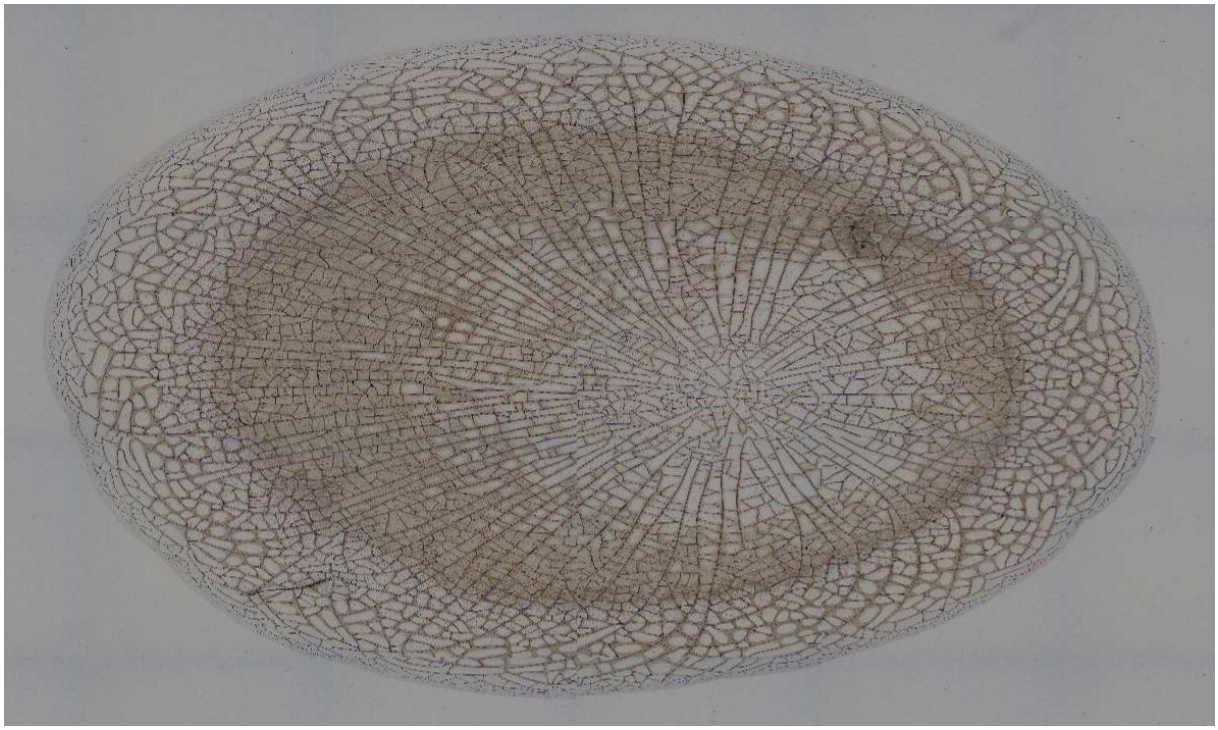


Figura A3. Ejemplo 3 de muestra de control.

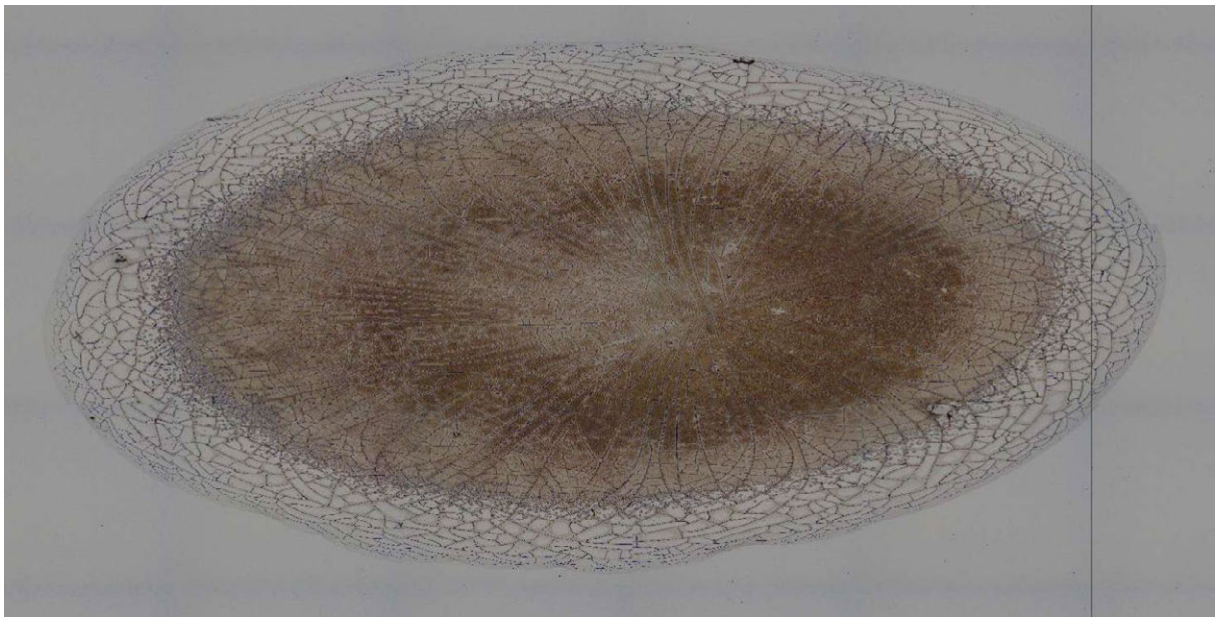


Figura A4. Ejemplo 4 de muestra de control.

Ejemplos de muestras de pacientes con daño leve

Figura A5. Ejemplo 1 de muestra de paciente con daño leve.

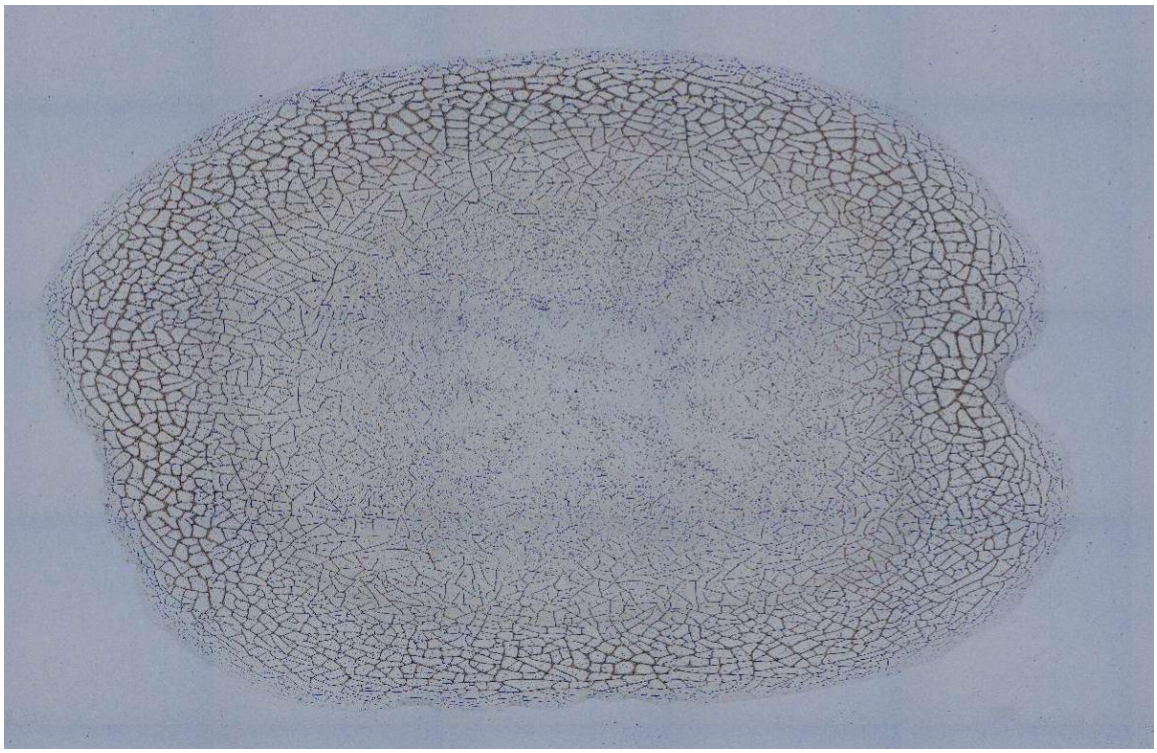


Figura A6. Ejemplo 2 de muestra de paciente con daño leve.

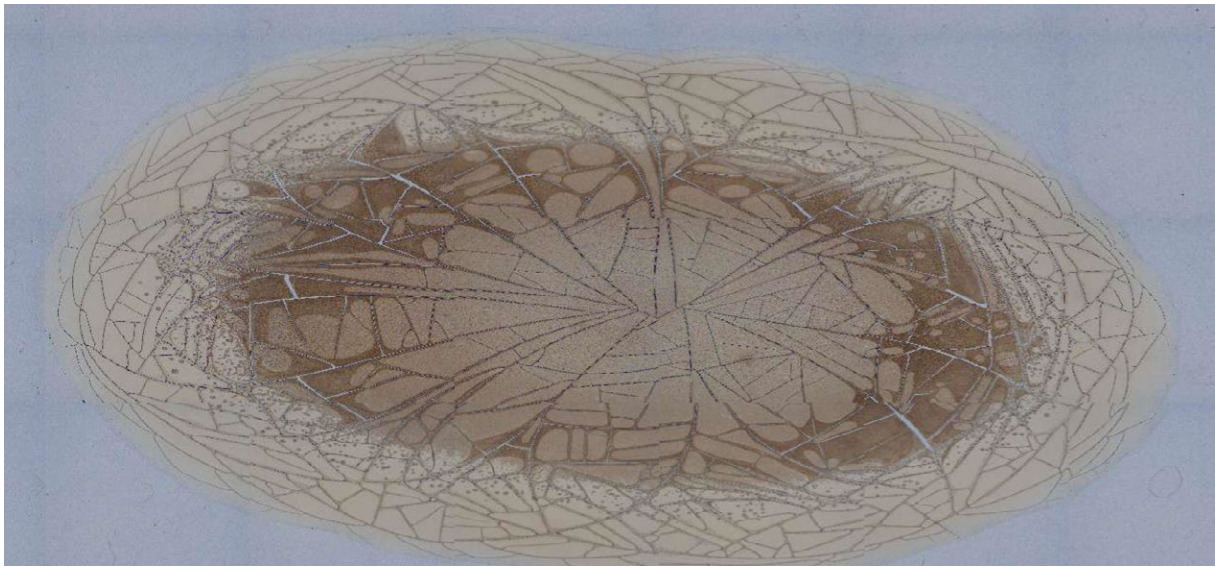


Figura A7. Ejemplo 3 de muestra de paciente con daño leve.

Figura A8. Ejemplo 4 de muestra de paciente con daño leve.

Ejemplos de muestras de pacientes con daño moderado

Figura A9. Ejemplo 1 de muestra de paciente con daño moderado.

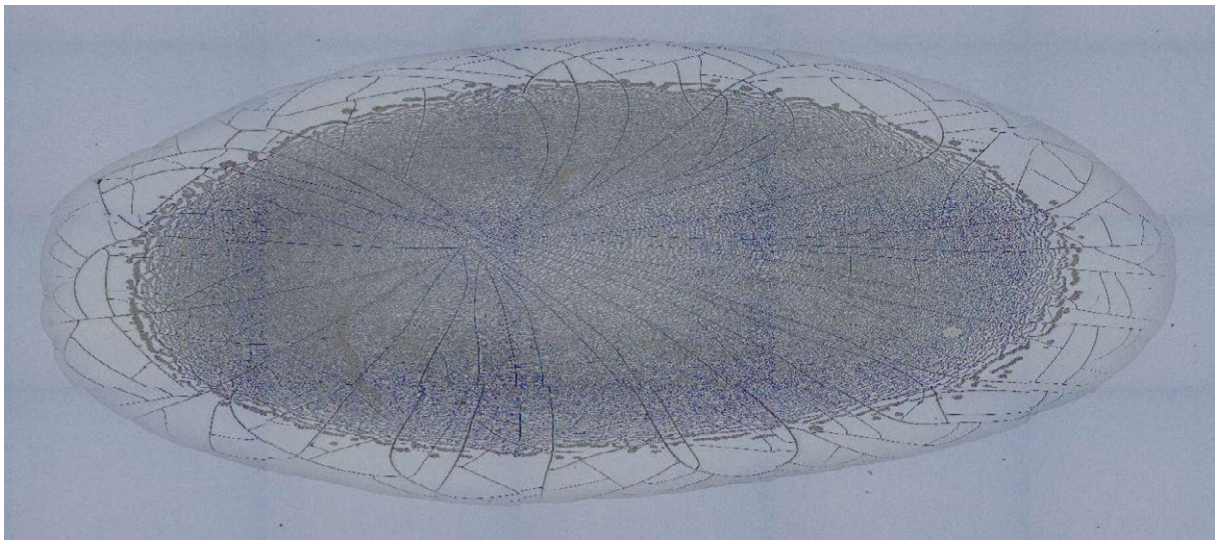


Figura A10. Ejemplo 2 de muestra de paciente con daño moderado.

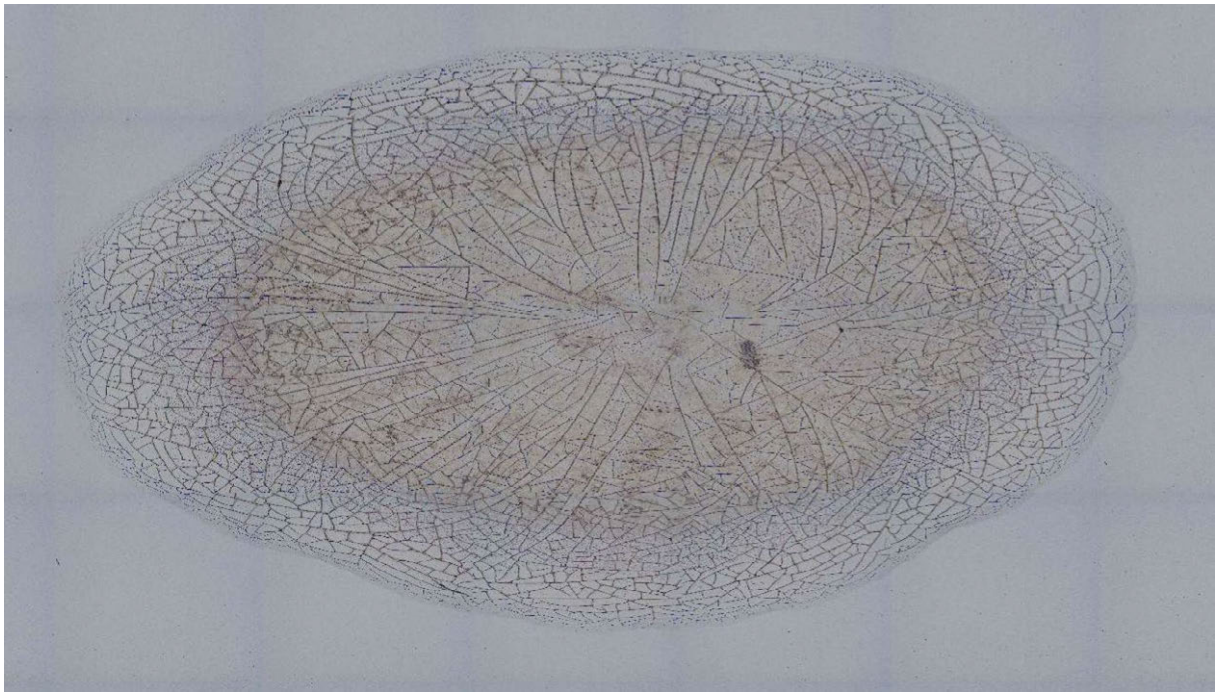


Figura A11. Ejemplo 3 de muestra de paciente con daño moderado.

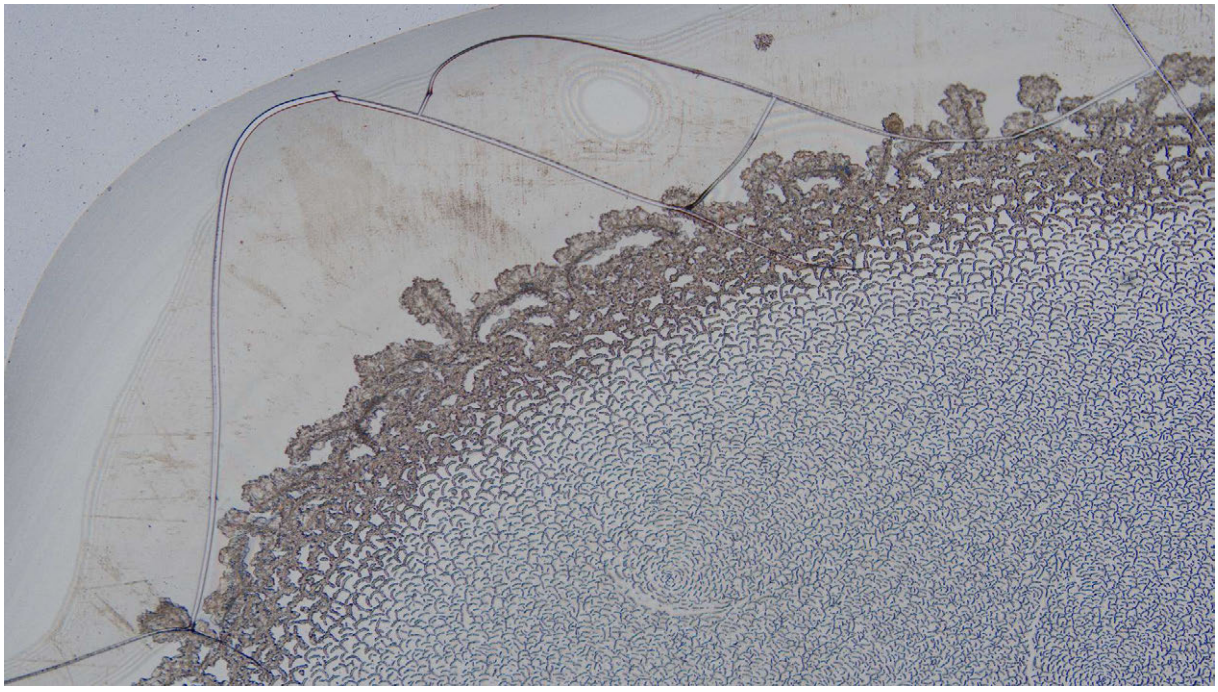


Figura A12. Ejemplo 4 de muestra de paciente con daño moderado.

Ejemplos de muestras de pacientes con daño grave

Figura A13. Ejemplo 1 de muestra de paciente con daño grave.

Figura A14. Ejemplo 2 de muestra de paciente con daño grave.

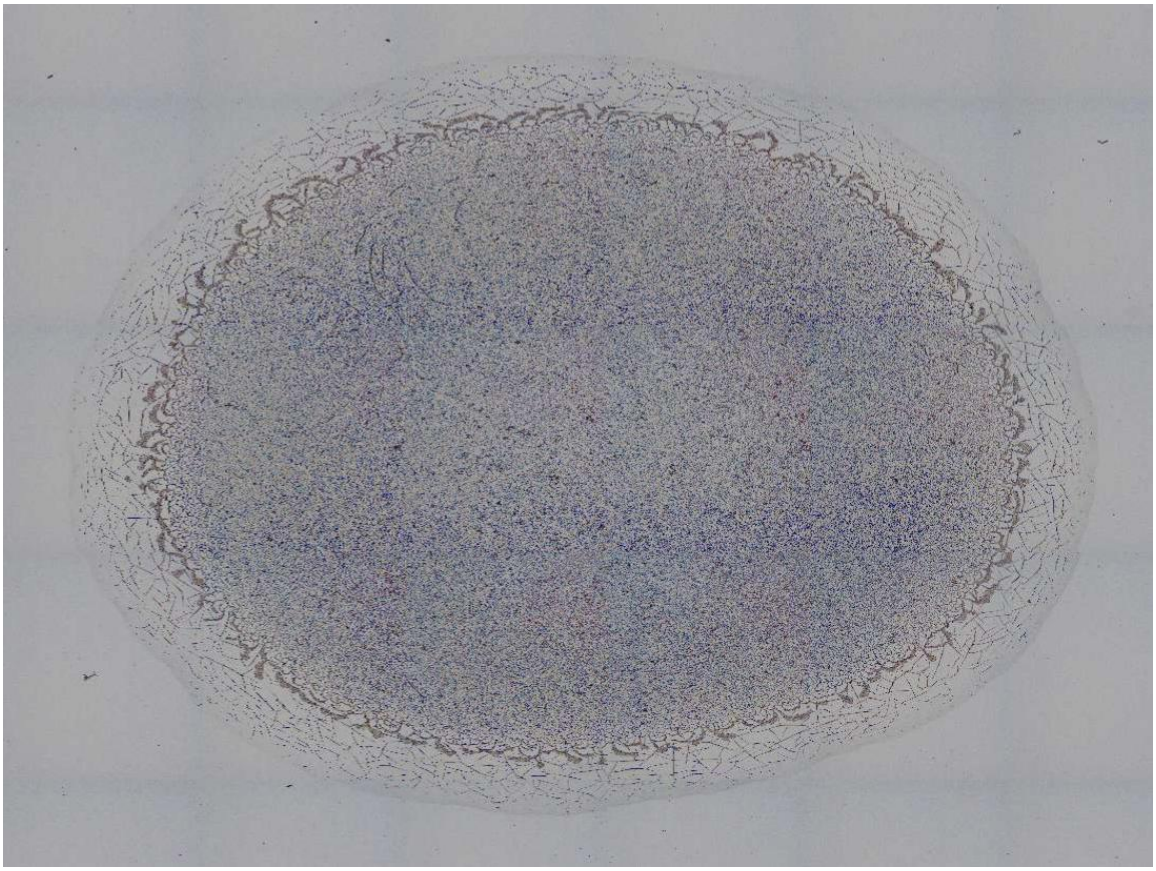


Figura A15. Ejemplo 3 de muestra de paciente con daño grave.

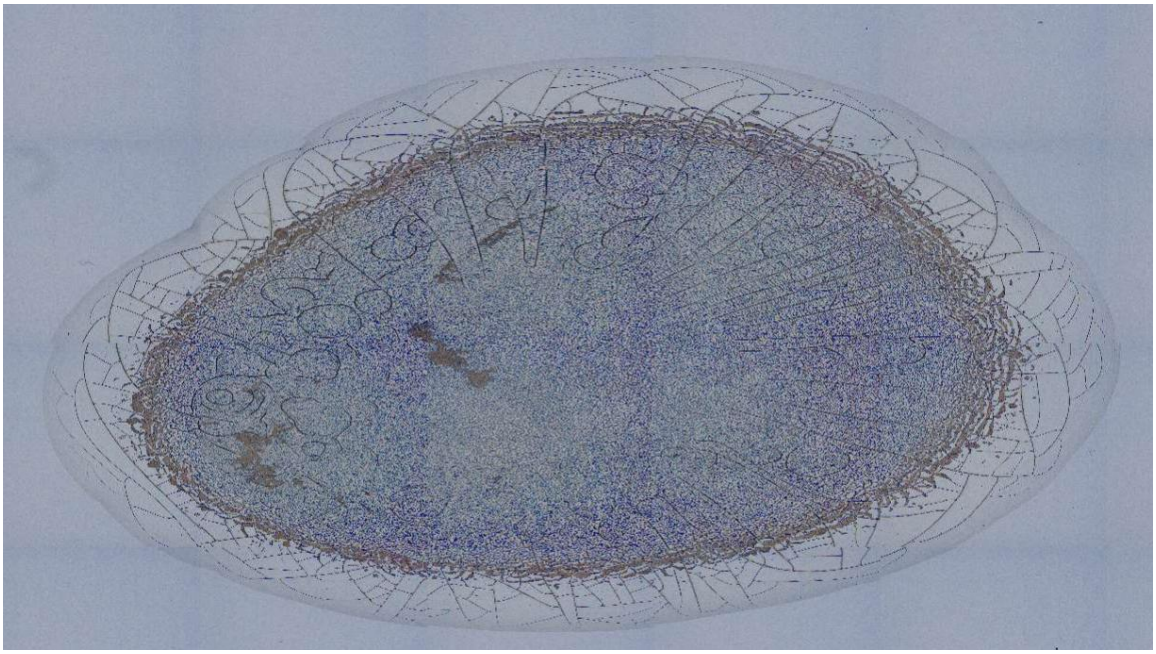


Figura A16. Ejemplo 4 de muestra de paciente con daño grave.