Universidad de Valladolid

ESCUELA DE INGENIERÍA INFORMÁTICA
DE SEGOVIA

Grado en Ingeniería Informática
de Servicios y Aplicaciones

Uso de técnicas de aprendizaje profundo para la detección del glaucoma

---

Alumno: Carlos Jiménez Vaquero

Tutores: Aníbal Bregón Bregón
Diego García Álvarez

Fecha: 19 de junio de 2025

Uso de técnicas de aprendizaje profundo para la detección del glaucoma

Carlos Jiménez Vaquero

19 de junio de 2025

A mi madre, por su apoyo incondicional.

Resumen

El glaucoma es una de las principales causas de ceguera irreversible en todo el mundo. Un diagnóstico precoz de esta patología podría evitar la pérdida progresiva de visión. Así, surge la necesidad de desarrollar herramientas que faciliten esta labor.

A lo largo de este trabajo se exploran diversas técnicas de aprendizaje profundo con el objetivo de detectar el glaucoma a partir de una retinografía. Además, se construye una aplicación la cual integra todos los desarrollos conseguidos, de manera que permite utilizar el estudio realizado en el entorno médico.

Respecto a las técnicas empleadas, van desde algoritmos propios del campo del Aprendizaje Automático como *Support Vector Machine* o *Multi-Layer Perceptron* para el análisis de datos y la construcción de ensembles, hasta estrategias específicas del Aprendizaje Profundo para la visión por computador, como clasificación y segmentación, para lo que se utilizarán arquitecturas de tipo YOLO y U-Net.

En referencia a los resultados obtenidos, se verá cómo se ha conseguido detectar y segmentar de manera exitosa las distintas estructuras que se pueden reconocer en una retinografía, llegando a obtener más de un 95 % para la métrica escogida *accuracy_camvid*. Además, también se explicará cómo se llega a diagnosticar el glaucoma con un 97 % de *recall* y un 94.96 % de aciertos. Lo que implica superar el resto de soluciones propuestas para los datos utilizados.

Palabras claves: Detección del glaucoma, Aprendizaje Automático, Machine Learning, Aprendizaje profundo, Deep Learning, Segmentación de imágenes, Clasificación de retinografías, Visión por computador, Bioinformática.

Abstract

Glaucoma is one of the leading causes of irreversible blindness worldwide. Early diagnosis of this pathology could prevent progressive vision loss. Thus, the need arises to develop tools to facilitate this task.

Throughout this work, several Deep Learning techniques are explored with the aim of detecting glaucoma from a retinography. In addition, an application is built which integrates all the developments achieved, so that the study can be used in the medical environment.

About the techniques used, they include algorithms of Machine Learning, such as Support Vector Machine or Multi-Layer Perceptron for data analysis and ensemble construction, to specific strategies of Deep Learning for computer vision, such as classification and segmentation, for which YOLO and U-Net architectures will be used.

In reference to the results obtained, it will be shown how the different structures that can be recognised in a retinography have been successfully detected and segmented, obtaining more than 95 % for the chosen metric *accuracy_camvid*. In addition, it will also be explained how glaucoma is diagnosed with a 97 % of *recall* and a 94.96 % of successes. This means outperforming the rest of the solutions proposed for the data used.

Keywords: Glaucoma detection, Machine Learning, Deep Learning, Deep Learning, Image segmentation, Retinography classification, Computer vision, Bioinformatics.

Índice general

Lista de Figuras

Lista de Tablas

Parte I

Descripción del proyecto

Capítulo 1

Introducción

1.1. Planteamiento del problema

En las últimas décadas, el uso intensivo de dispositivos electrónicos como teléfonos móviles y ordenadores se ha convertido en una constante en la vida diaria. Esta exposición prolongada a pantallas se ha relacionado con un incremento en la prevalencia de miopía a nivel mundial [1]. Además, otros factores relacionados con el uso continuado de estas tecnologías, como la fatiga ocular o el enfoque continuo en distancias cortas, aumentan las posibilidades de padecer miopía.

La miopía no solo es una afección en sí misma, sino que resulta un factor de riesgo para sufrir otras patologías oculares como el glaucoma [4]. Según [3], el glaucoma es una enfermedad ocular que daña el nervio óptico, cuya principal causa es la tensión ocular elevada. Como consecuencia, genera puntos ciegos en la visión. Estas áreas ciegas van en aumento hasta la pérdida completa de la visión. Además, el glaucoma puede desencadenarse por otras circunstancias, como infecciones oculares [4].

Para evitar que se dé esta pérdida paulatina de visión, es de vital importancia un diagnóstico y seguimiento precoz de la enfermedad. En caso contrario, como la disminución de la visión se produce de manera gradual, el paciente no será consciente de los signos hasta que el glaucoma haya progresado a fases avanzadas y el daño al nervio óptico sea significativo. Por su parte, el diagnóstico del glaucoma requiere un examen ocular completo. Para llevarlo a cabo, como explica la Academia de Oftalmología Americana [2], se debe medir la presión ocular, examinar el nervio óptico, tomar una imagen del mismo (retinografía), hacer una prueba de visión periférica y medir el espesor de la córnea, entre otros.

Debido a la serie de pruebas que se deben realizar para poder detectar el glaucoma, junto con la necesidad de realizar un diagnóstico precoz, surge la posibilidad de desarrollar herramientas automáticas de apoyo al diagnóstico. Además, la revolución de la Inteligencia Artificial no ha dejado a ningún área de conocimiento indiferente, y ya se están produciendo los primeros avances mediante el uso de esta tecnología en el sector médico, y más concretamente, en la detección de patologías oculares [6]. Así, surge de manera natural la posibilidad de aplicar estas técnicas para desarrollar una herramienta

de diagnóstico del glaucoma rápida y eficaz.

No abundan conjuntos de datos públicos para la construcción de herramientas médicas mediante técnicas de Inteligencia Artificial dada las restricciones establecidas para la compartición de este tipo de información. En la actualidad se encuentran algunos *datasets* de referencia para el estudio del glaucoma mediante Inteligencia Artificial como los de *RIM-ONE DL* [13] o *EyePACS-AIROGS-light-V2* [9]. Estos bancos de datos públicos permiten el estudio de los mismos con fines de investigación.

Este trabajo se centra en el desarrollo de una herramienta de Inteligencia Artificial con el objetivo de realizar un diagnóstico rápido y eficaz del glaucoma a partir de imágenes oculares denominadas retinografías. De esta forma, se busca construir una herramienta capaz de asistir a los oftalmólogos en esta tarea de detección de la enfermedad del glaucoma, para contribuir al desarrollo de sistemas más accesibles, rápidos y precisos.

Con este objetivo, a lo largo de este trabajo se desarrollarán modelos basados en técnicas específicas dentro del campo de la Inteligencia Artificial, denominadas Aprendizaje Automático (*Machine Learning*), y a su vez dentro de las mismas, algoritmos de Aprendizaje Profundo (*Deep Learning*), para la detección del glaucoma. Con este propósito, se utilizarán técnicas de segmentación para imágenes, es decir, para señalar partes específicas, junto con otras para clasificar entre ojos sanos y glaucomatosos. Así, se persigue mejorar la precisión y la eficiencia en la detección temprana de la enfermedad.

1.2. Objetivos del trabajo

Los principales objetivos que abarca este proyecto y orientan su desarrollo son los siguientes:

- **OBJ-01**. Elaborar una herramienta para la detección del glaucoma en retinografías mediante técnicas de aprendizaje profundo.
- **OBJ-02**. Dotar a dicha herramienta de un módulo capaz de identificar las estructuras propias de las retinografías empleando métodos de segmentación.
- **OBJ-03**. Estudiar el rendimiento en la clasificación y segmentación de retinografías de ojos sanos y glaucomatosos mediante técnicas de aprendizaje profundo en base a las métricas más adecuadas para cada uno de ellos.

1.2.1. Restricciones

Además, dada la naturaleza del proyecto, surge el siguiente conjunto de restricciones:

- **R-01** Limitación temporal de entre 300 y 360 horas correspondientes a la carga de trabajo establecida de 12 ECTS.

1.3. Estructura de la memoria

A lo largo de esta memoria se exponen todas las etapas abordadas durante la elaboración del proyecto planteado. Como consecuencia, este documento se divide en tres partes fundamentales:

- **Descripción del proyecto**. En esta primera parte se establecen los objetivos del proyecto, se contempla la planificación de los recursos disponibles y se describen aquellos conceptos relacionados con el trabajo que se va a desarrollar. A su vez se divide en los siguientes capítulos:
  - **Introducción**: incluida en el Capítulo 1 expone el problema que se aborda en el proyecto sobre la detección del glaucoma, acompañándolo de los objetivos y restricciones planteados.
  - **Planificación**: incluida en el Capítulo 2 y detalla la estrategia elaborada para gestionar correctamente los recursos temporales y gestionar los riesgos presentados, se especifica la metodología de trabajo, y se explican el presupuesto y balance real.
  - **Antecedentes**: incluidos en el Capítulo 3 tratan los conceptos científico-técnicos y médicos que se abordan en el proyecto, además de analizar el estado del arte.
- **Desarrollo de propuestas y resultados**. Esta parte se divide en:
  - **Desarrollo de la propuesta y experimentación**: a lo largo del Capítulo 4 se trata el desarrollo de las propuestas que permiten la consecución de los objetivos planteados. Se detalla el proceso de entrenamiento de modelos de aprendizaje profundo, la extracción de las características relevantes en las predicciones hechas por los modelos, y el análisis de estos datos mediante algoritmos de *Machine Learning* para clasificar las retinografías entre pacientes sanos y con glaucoma.
  - **Evaluación de los resultados**: a lo largo del Capítulo 4 se analizan los resultados obtenidos tanto por los modelos de aprendizaje profundo para clasificación y segmentación, como de los algoritmos de *Machine Learning*. Este análisis se consigue estableciendo unas métricas adecuadas.
  - **Conclusiones y trabajo futuro**: en el Capítulo 6 se realiza una reflexión tanto a nivel técnico como personal sobre el desarrollo realizado, estudiando los objetivos planteados en un inicio y las propuestas de valor elaboradas; además, se detallan posibles vías de evolución del proyecto.
- **Apéndices**. En esta última parte se explica el procedimiento para poder hacer uso de los modelos construidos mediante los siguientes documentos:
  - **Manual de Instalación**.
  - **Contenido adjunto**.

Capítulo 2

Planificación

2.1. Metodología de trabajo

Durante el desarrollo del proyecto se ha utilizado una reinterpretación de la metodología Scrum [15] para la organización del mismo, la cual es una metodología ágil que promueve la colaboración, flexibilidad y entrega iterativa e incremental de productos que aporten valor. Su flujo de trabajo viene descrito en la Figura 2.1. Dadas las características propias de un Trabajo de Fin de Grado (TFG), se utiliza una adaptación de Scrum denominada SCORE [19]. Por otra parte, para el desarrollo de la propuesta se ha empleado una metodología complementaria a SCORE orientada a la realización de proyectos de análisis de datos. Se trata de la metodología CRISP-DM [17].

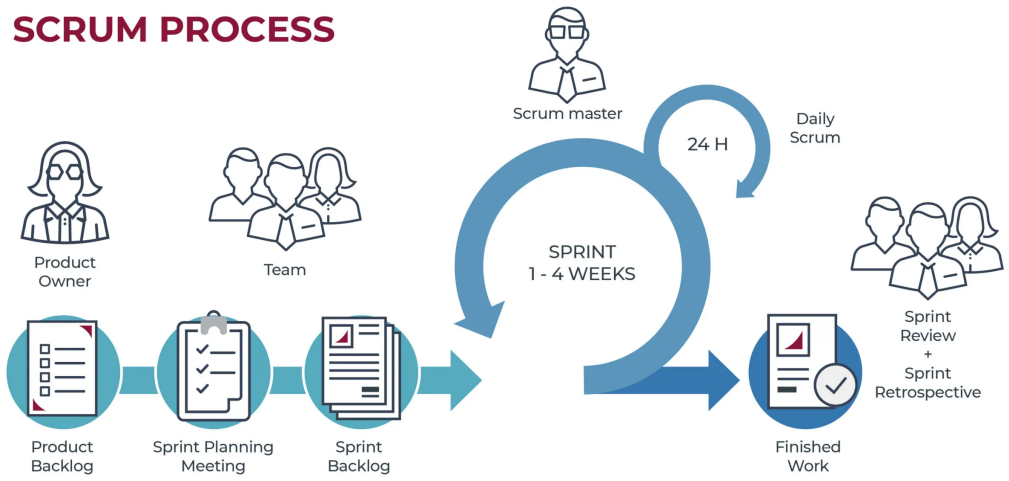


Figura 2.1: Flujo de trabajo de la metodología SCRUM [16].

2.1.1. Ciclo de vida: SCORE

La metodología SCORE [19] adopta prácticas de los marcos de trabajo ágiles que permiten organizar el ciclo de vida del TFG en torno a los objetivos definidos en el proyecto. Su objetivo principal es alcanzar las metas establecidas en el proyecto, garantizando la participación y comunicación efectiva de todos los involucrados para optimizar la calidad del producto final.

SCORE se puede utilizar para establecer una serie de objetivos centrados en estructurar cualquier TFG, facilitando el aprendizaje y aportando una visión de alto nivel sobre el trabajo que se pretende desarrollar. Además, estos objetivos se fundamentan en historias de aprendizaje. Cada historia se caracteriza por un conjunto de resultados alcanzables que funcionan como criterios de aceptación y favorecen la consecución del objetivo. Para que una historia se considere completa, cada uno de los resultados debe quedar satisfecho. Los objetivos que se contemplan para la consecución del Trabajo de Fin de Grado son los siguientes:

- **Proyecto**. El primer objetivo es plantear un proyecto con el fin de resolver un problema de la vida real. Para ello, se define el problema a resolver, se establecen los objetivos que se persiguen y se lleva a cabo una planificación que permita ajustarse a la carga de trabajo prevista.

- **Antecedentes**. Puesto que a lo largo del proyecto se tratan conceptos y temas que pueden resultar desconocidos, se realiza un trabajo previo de contextualización para comprender el planteamiento del problema y los objetivos del TFG. Así, se pretende obtener el conocimiento necesario para abordar el ámbito de negocio y científico-técnico.

- **Desarrollo**. Este objetivo se centra en la construcción de un producto de valor. Las historias de aprendizaje que aquí se tratan para conseguir el objetivo dependerán de la naturaleza del proyecto en función de si este es de desarrollo o de investigación.

- **Aceptación**. Se trata de evaluar si el producto se adecúa a los resultados que se pretende obtener. Al igual que el objetivo anterior, las historias en las que se fundamenta dependerán de si el proyecto es de desarrollo o de investigación. Por otra parte, el estudiante discute los resultados obtenidos, incluyendo una valoración de los métodos utilizados.

- **Comunicación**. Este objetivo combina habilidades de comunicación oral y escritas, y se materializa en las historias de aprendizaje que tratan de la redacción de una memoria técnica y del acto de defensa del proyecto. La memoria técnica condensará todo el trabajo realizado, desde las primeras fases de adquisición de conocimientos hasta el desarrollo del proyecto y los resultados obtenidos. Por su parte, el acto de defensa consistirá en explicar el trabajo realizado ante un tribunal.

Como se ha explicado anteriormente, esta metodología se fundamenta en Scrum, la cual define en [15] una serie de roles, eventos y artefactos. Estos materiales se adaptan al contexto académico de elaboración de un TFG.

Roles

Las metodologías tradicionales para el desarrollo de un Trabajo de Fin de Grado tan solo contemplan la relación entre tutor y estudiante. Por su parte, con la metodología utilizada, se tienen en cuenta un mayor número de roles y relaciones entre ellos, pues es un factor clave para el éxito en la ejecución del TFG.

- **Estudiante**. Es el rol fundamental dado que desarrolla cada tarea especificada en las historias de aprendizaje para la consecución de los objetivos. Además, el estudiante funciona como un rol central, pues se comunica con el resto de roles participantes en el proyecto, asumiendo el *feedback* proporcionado por los mismos para mejorar el producto construido.

- **Tutor**. Este rol lo desempeñan uno o más profesores que orientan el aprendizaje del estudiante a través de un plan de seguimiento establecido por ellos mismos. Durante el seguimiento, se encargan de aportar retroalimentación sobre el desarrollo del proyecto y orientan su avance. Además, colaboran en las primeras fases del proyecto, ayudando al planteamiento del problema, la definición de los objetivos y facilitando las referencias que consideren oportunas para consolidar el conocimiento necesario durante la realización del TFG.

- **Comunidad**. Este rol incluye a un amplio abanico de personas (estudiantes, profesores, expertos en la materia,...) capaces de aportar valor al proyecto. No tienen ninguna responsabilidad específica. Su función es la de aportar valor a través de comentarios objetivos acerca del producto, lo que incrementa las oportunidades de mejora.

- **Tribunal**. Comisión encargada de evaluar el TFG durante el acto de defensa del mismo, adecuándose a la satisfacción de los objetivos y la calidad tanto del producto construido como de la defensa del mismo.

Eventos

Aunque SCORE adopta los métodos de SCRUM, los eventos que se definen en ambos son diferentes, puesto que SCRUM es un marco más genérico para todo tipo de proyectos; mientras que SCORE se centra en identificar una serie de reuniones adaptadas a los procesos de investigación. Con esta premisa, define dos tipos principales de eventos que serán los mismos que se utilizarán para la realización del TFG, aunque debido a limitaciones temporales y de atención a otros compromisos, se modifican ligeramente los eventos de esta metodología para adecuarse a la situación personal de cada uno de los conformantes del proyecto:

- ***Status meetings***: son análogos a las reuniones de SCRUM denominadas *daily meetings*, que tienen lugar diariamente. Por su parte, SCORE recomienda tener tres de estas reuniones de manera semanal, manteniendo fijos los días de su realización, salvo causas justificadas. Para el caso concreto de este proyecto, dado que se debe realizar en conjunto con otro TFG y otras asignaturas, se ha considerado oportuno realizar una reunión de manera semanal para que las reuniones tengan contenido y aporten valor al proceso.

  Durante una *status meeting*, el estudiante se encarga de comunicar todas las tareas que se han llevado a cabo desde la anterior reunión, los resultados alcanzados y las posibles dudas u obstáculos encontrados en el proceso. Por su parte, los tutores del proyecto aportan la correspondiente retroalimentación de los avances que se les han comunicado. Para finalizar la reunión, se establecen las líneas de progreso que se seguirán hasta la siguiente reunión.

  Al igual que ocurre con las *daily meetings* en SCRUM, la metodología SCORE establece una duración para las *status meetings* de en torno a 15 o 20 minutos, evitando ahondar en detalles técnicos y dejando los mismos para el otro tipo de reunión: las *on-demand meetings*.

  Para este proyecto, se ha establecido que una de cada dos reuniones tenga esta duración, y se le denominará *weekly*. Las restantes serán de una hora, dado que los avances se producen de manera rápida y tan solo se cuenta con una reunión a la semana por motivos temporales y de otras obligaciones. A estas últimas se hará referencia con el propio nombre de reuniones".

- ***On-demand meetings***. Dado que las *status meetings* deben servir para actualizar el estado actual del proyecto evitando mencionar detalles técnicos, las *on-demand meetings* suplen esta necesidad de tratar aspectos técnicos. Como indica su nombre, las *on-demand meetings* se conciben bajo demanda expresa del estudiante. En estas se discuten aspectos sobre la investigación, resultados o métodos de una manera más profunda si fuese necesario.

Además de estos dos tipos de eventos, en ocasiones se comunica el estado del proyecto si se considera necesario vía correo electrónico, lo que la metodología SCORE define como *e-mail status reports*. También por este canal se harán consultas breves, por ejemplo, sobre aspectos técnicos del desarrollo. Esta comunicación se hará vía *Teams* y/o correo electrónico para tener un contacto directo en caso de necesitar algo más de ayuda o una respuesta más urgente.

Artefactos

Los artefactos son elementos clave durante el desarrollo del proyecto que permiten la transparencia y una visión clara del trabajo y el progreso del proyecto. Los artefactos propuestos por SCORE se fundamentan en los ya establecidos por SCRUM [15], recibiendo el mismo nombre:

- **Incremento**: resultado utilizable propio de cada sprint que aporta valor por sí mismo al proyecto satisfaciendo uno o más de los objetivos propuestos. La superposición de incrementos a lo largo de los sprints conformará el producto final.

- **Retroalimentación**: *feedback* recibido por el estudiante principalmente al final de cada *sprint*. La retroalimentación es un proceso continuo que se obtiene durante las reuniones entre el estudiante y el resto de roles partícipes. Este artefacto permite adaptarse a cambios propuestos y mejorar la calidad del producto.

2.1.2. Proceso de desarrollo: CRISP-DM

CRISP-DM [17] es una metodología ampliamente utilizada en proyectos de desarrollo enfocados en un análisis profundo de datos. En concreto, promueve un enfoque iterativo y cíclico que permite revisar y ajustar el desarrollo a medida que avanza el proyecto. Este proceso se resume en la ilustración de la Figura 2.2.

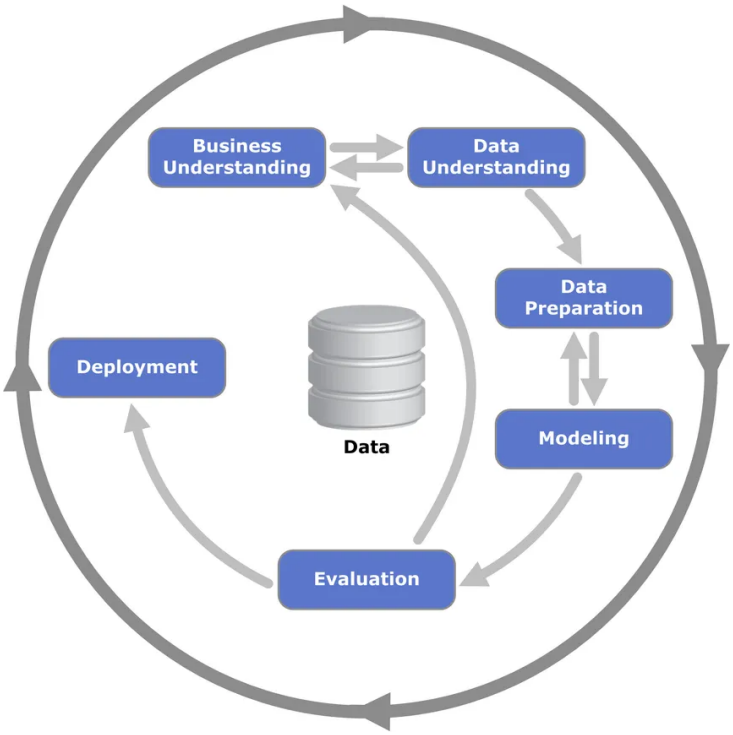


Figura 2.2: Esquema de desarrollo propuesto por la metodología CRISP-DM [18].

Como se determina en la misma imagen, CRISP-DM establece una serie de etapas de manera cíclica, que serán las que se lleven a cabo para cada sprint definido usando la metodología SCORE. A continuación se describen las fases en que se basa el desarrollo siguiendo la metodología CRISP-DM.

Etapas de desarrollo según CRISP-DM

A continuación se enumeran y explican cada una de las etapas que determina el proceso de desarrollo de CRSIP-DM.

1. **Comprensión del negocio** (*Business Understanding*). Esta fase se centra en definir el problema que se aborda con el proyecto, así como entender los objetivos del mismo. En esta etapa también se analizan los recursos necesarios para abordar el problema, así como de los que se dispone, se analizan los riesgos y se estiman los costes.

   De esta manera, el resultado final de esta fase es un presupuesto de los costes que acarrea el proyecto, un análisis de los riesgos que presenta su construcción, y un estudio del negocio en el que se enmarca.

2. **Comprensión de los datos** (*Data Understanding*). Una vez entendido el negocio en el que se enmarca el proyecto, puesto que el mismo se centra en el uso de datos, se pasa a comprender los mismos. Por tanto, se recopilan los datos disponibles y se realiza un análisis inicial para evaluar su calidad, su relevancia y la relación con los objetivos del proyecto.

   Así, se hace un análisis de los datos que se tienen disponibles, analizando la calidad, patrones o problemas de los mismos. Además, se estudian las variables que hay, si faltan valores y qué tipo de datos son.

3. **Preparación de los datos** (*Data Preparation*). En esta fase se preparan los datos para ser utilizados en los modelos. Es una fase crítica que suele ocupar gran parte del tiempo. En esta, se lleva a cabo una limpieza de los datos eliminando valores nulos, duplicados, o que presenten algún otro problema en función del contexto del proyecto.

   Además, se seleccionan las variables que se van a tener en cuenta para el posterior entrenamiento y se realizan las transformaciones necesarias sobre los datos como normalización o escala de grises. Otro tipo de transformaciones incluyen la aplicación de filtros si los datos son imágenes. Además, si fuera necesario, en esta etapa se etiquetan los datos con sus correpondientes categorías.

4. **Modelado** (*Modeling*). Se trata de seleccionar y entrenar los modelos de Inteligencia Artificial que se consideren oportunos de acuerdo a los datos del problema. En concreto, durante este proyecto se emplearán tanto técnicas de *Machine Learning* como de *Deep Learning*, que más adelante se explicarán en las Secciones 3.2.1 y 3.2.3, respectivamente.

5. **Evaluación** (*Evaluation*). En esta etapa se evalúa el modelo de manera objetiva para comprobar su rendimiento. Esto se consigue estableciendo una serie de métricas como las que se definirán en la Sección 3.2.4, que como su nombre indica, miden la calidad del resultado obtenido.

En esta fase, se comprueba si el modelo construido durante la etapa de modelado es fiable y tiene utilidad para resolver el problema que se plantea; es decir, se debe analizar si se ajusta a los objetivos del proyecto.

6. **Despliegue** (*Deployment*). En esta última fase se implementa el modelo construido en un entorno real o se entrega de forma que los resultados puedan usarse en la práctica. Luego, el resultado del despliegue puede ser en forma de informe, API o de una aplicación, por ejemplo. En este proyecto el resultado del despliegue será una aplicación en la que se integren los modelos construidos, que se presentará en el Capítulo 5. Para su construcción, se debe tener en cuenta quién usará el modelo y cómo se interpretan los resultados.

Así, en este proyecto se realiza una planificación temporal en base a los fundamentos de SCORE, dividiendo el ciclo de vida en iteraciones. Por su parte, el proceso de desarrollo dentro de cada iteración viene determinado por la metodología CRISP-DM a la que se ha modificado ligeramente el esquema representado en la Figura 2.2. En vez de realizar un análisis del entorno del negocio y de los datos para cada iteración, tan solo se realizará una única vez al comienzo del desarrollo estas fases y cada iteración consistirá en la preparación de los datos, el modelado y la evaluación de los resultados.

2.2. Planificación temporal

La planificación temporal para este proyecto abarca el período comprendido entre el 16 de diciembre de 2024 y el 21 de mayo de 2025. Durante el mismo, se estructura el tiempo en base a un total de cinco *sprints*, de los que se supone que tendrán una duración cada uno de ellos de entre 60 y 72 horas de trabajo, aproximadamente, para completar las 300-360 horas que se supone que debería durar la ejecución de este proyecto.

Así, teniendo en cuenta situaciones como los períodos de exámenes, cada una de estas cinco iteraciones se enmarca en las siguientes fechas:

- ***Sprint* 1**: 16 de diciembre - 19 de febrero
- ***Sprint* 2**: 19 de febrero - 20 de marzo
- ***Sprint* 3**: 20 de marzo - 9 de abril
- ***Sprint* 4**: 9 de abril - 30 de abril
- ***Sprint* 5**: 30 de abril - 21 de mayo

Los estándares a abordar durante el trabajo en torno a los que se estructura la planificación temporal son: Proyecto, Antecedentes, Desarrollo y Comunicación, representados en la Estructura de Desglose del Trabajo (EDT) en tonos morados, naranjas, verdes y azules, respectivamente. Cada uno de estos estándares abarca un conjunto de tareas concretas:

- **Proyecto**.
  - **Planificación del *sprint***. Abarca las tareas relacionadas con la organización del *sprint*, definiendo el alcance y los plazos temporales de cada una de las tareas concretas existentes.
  - **Caracterización del proyecto**. Se refiere a la gestión del propio proyecto; es decir, a la definición de la motivación, los objetivos, la metodología, la planificación y las conclusiones del proyecto.
  - **Desarrollo del *sprint***. Se trata de las tareas relativas al seguimiento del *sprint*. Esto consiste en actualizar el tablero del proyecto y la previsión de las reuniones, que serán en su mayoría de manera semanal.
  - **Finalización del *sprint***. Se trata de comunicar los progresos realizados durante el *sprint* así como de la recepción del *feedback* generado por los tutores para orientar el avance del proyecto. En este caso, como ya se ha explicado anteriormente, el *feedback* se dará en cada reunión de manera semanal, por lo que este objetivo se incluye dentro del estándar de desarrollo del *sprint*.
- **Antecedentes**.
  - **Estado del arte**. Consiste en investigar a cerca de trabajos similares relacionados con el área de estudio. Con esto, se valorarán las ventajas y limitaciones de otras soluciones en el entorno de negocio para justificar la consecución del proyecto.
  - **Contexto científico-técnico**. Marco teórico sobre el que se inscribe el trabajo desarrollado. Consiste en desarrollar las tareas necesarias para entender los contenidos teóricos que involucra el proyecto.
- **Desarrollo**.
  - **Construcción**. Se trata de la implementación del producto en cuestión que se trata a lo largo del proyecto.
  - **Diseño experimental**. Se establece todo lo necesario para poder construir los modelos involucrados en la solución propuesta. Esto se refiere a la definición de los parámetros del experimento, de los conjuntos de prueba y de las métricas necesarias.
  - **Experimentación y Análisis de Resultados**. Implica el correspondiente estudio de los resultados obtenidos y su relación con el cumplimiento de los objetivos definidos.
- **Comunicación**.
  - **Presentación**. Construcción de una presentación orientada al acto de defensa del TFG. Se caracteriza por una contextualización del proyecto, seguida del

desarrollo realizado, para acabar con una valoración sobre el cumplimiento de la planificación y de los objetivos establecidos.

- **Memoria**. Se fundamenta en la elaboración de la documentación necesaria sobre el proyecto.

Teniendo todo lo anterior en cuenta, se han organizado una serie de reuniones como son las *Status meetings* y *on-demand meetings* ya tratadas. Estas han tenido lugar a lo largo de toda la realización del proyecto, según se muestra en la Tabla 2.3.

Sprint 1	16-dic.-24	*Status meeting*
	19-feb.-25	*Status meeting*

Sprint 2	19-feb.-25	*Status meeting*
	05-mar.-25	*Status meeting*
	20-mar.-25	*Status meeting*

Sprint 3	20-mar.-25	*Status meeting*
	26-mar.-25	*Status meeting*
	02-abr.-25	*Status meeting*
	09-abr.-25	*Status meeting*

Sprint 4	09-abr.-25	*Status meeting*
	16-abr.-25	*Status meeting*
	23-abr.-25	*Status meeting*
	30-abr.-25	*Status meeting*

Sprint 5	30-abr.-25	*Status meeting*
	07-may.-25	*On-demand meeting*
	14-may.-25	*Status meeting*
	21-may.-25	*Status meeting*

Figura 2.3: Planificación de las reuniones en cada *sprint*.

A continuación se detalla la planificación y el alcance individual de cada uno de los sprints haciendo uso de una EDT y reflejando la relación entre las actividades, su secuenciación y duración mediante un cronograma. En este caso, el cronograma se trata de un diagrama de Gantt.

2.2.1. *Sprint* 1

En este primer *sprint* se tratan las tareas pertenecientes a los objetivos: Proyecto, Antecedentes, Desarrollo y Comunicación. Para poder organizar de una manera adecuada el trabajo en esta primera iteración, se ha elaborado una Estructura de Desglose del Trabajo (EDT) que se puede observar en la Figura 2.4. Además, a partir de este mismo diagrama, se ha especificado la duración que debería tener cada una de las tareas dispuestas, construyendo así el diagrama de Gantt correspondiente que se recoge en la Figura 2.5.

En general, este primer *sprint* está enfocado en investigar el marco sobre el que se encuadra el proyecto, adquiriendo conocimiento sobre las soluciones existentes al problema que se plantea, además de comenzar con la construcción de los primeros modelos para la detección del glaucoma. Como se irá viendo, con la consecución de los *sprints* se irá

reduciendo la parte dedicada a la investigación del problema tratado y de los métodos que se pueden usar en favor de más trabajo en la parte de desarrollo.

2.2.2. *Sprint* 2

En este segundo *sprint* se aborda el problema de igual forma que en el primero. Se organiza el trabajo a desarrollar a partir del desglose de tareas que se encuentra representado en la EDT de la Figura 2.6. Por otra parte, a partir de la misma se planifica de manera temporal cada tarea, como se puede consultar en el diagrama de Gantt correspondiente a la Figura 2.7.

Este *sprint* ya presenta una aproximación más real al problema, pues en el primero, la parte técnica tan solo se encarga de construir modelos básicos a partir de los datos en posesión y no se manipulan los datos como se hará de aquí en adelante. En particular, en este sprint se resuelve el problema de la segmentación de las estructuras necesarias dentro de una retinografía, referido tanto a la parte de estudio e investigación de estas técnicas junto con el desarrollo de las mismas. Además, también se trata de abordar el problema de la clasificación a partir de los resultados obtenidos en la fase de segmentación.

Aquí se trata de abordar de una manera más contundente los métodos necesarios para alcanzar los objetivos propuestos, y esta será la tónica en las siguientes iteraciones, donde se disminuye la parte de investigación, porque ya ha sido completada, en favor de una mayor aportación al desarrollo de propuestas que mejoren los resultados buscados.

2.2.3. *Sprint* 3

A lo largo de este *sprint* se profundiza en las técnicas para la clasificación entre retinografías de ojos glaucomatosos y sanos. Además, se utilizarán los resultados obtenidos en la iteración anterior para tratar de mejorar los resultados. Tras esto, se realizará una comparativa con otras soluciones previas para comprobar si los nuevos métodos aportan más información en forma de mejores resultados.

Como se puede apreciar en la EDT de la Figura 2.8, una vez superadas las primeras iteraciones en las que se divide el proyecto, la parte de recogida de información de los antecedentes ocupa una menor parte del tiempo; mientras que la mayor parte del mismo se destina al desarrollo de la propuesta de solución.

Además, la organización temporal completa en base a las tareas establecidas se recoge en el diagrama de Gantt de la Figura 2.9.

2.2.4. *Sprint* 4

En este *sprint* se prevé el refinamiento de los métodos realizados en el *sprint* anterior para conseguir mejorar las métricas obtenidas. Las tareas que se van a desempeñar se encuentran recogidas en la EDT de la Figura 2.10 y consisten en una evolución de las de la etapa anterior.

Por su parte, la planificación derivada de la EDT de la Figura 2.10 se recoge en el diagrama de Gantt de la Figura 2.11.

2.2.5. *Sprint* 5

Este *sprint* contiene las tareas basadas en la agrupación de los resultados de las iteraciones anteriores. De este forma, se espera que combinando los resultados se obtengan otros aún mejores. Esto se hace mediante ensembles, que serán explicados en la Sección 3.2, para lo cual, se realiza un pequeño estudio de en qué consisten.

Junto con los antecedentes y la construcción de la solución, se desarrolla el objetivo Proyecto para planificar el *sprint*. Todo esto se recoge en la EDT de la Figura 2.12, completando la explicación de la planificación temporal de estas tareas en base al diagrama de Gantt de la Figura 2.13.

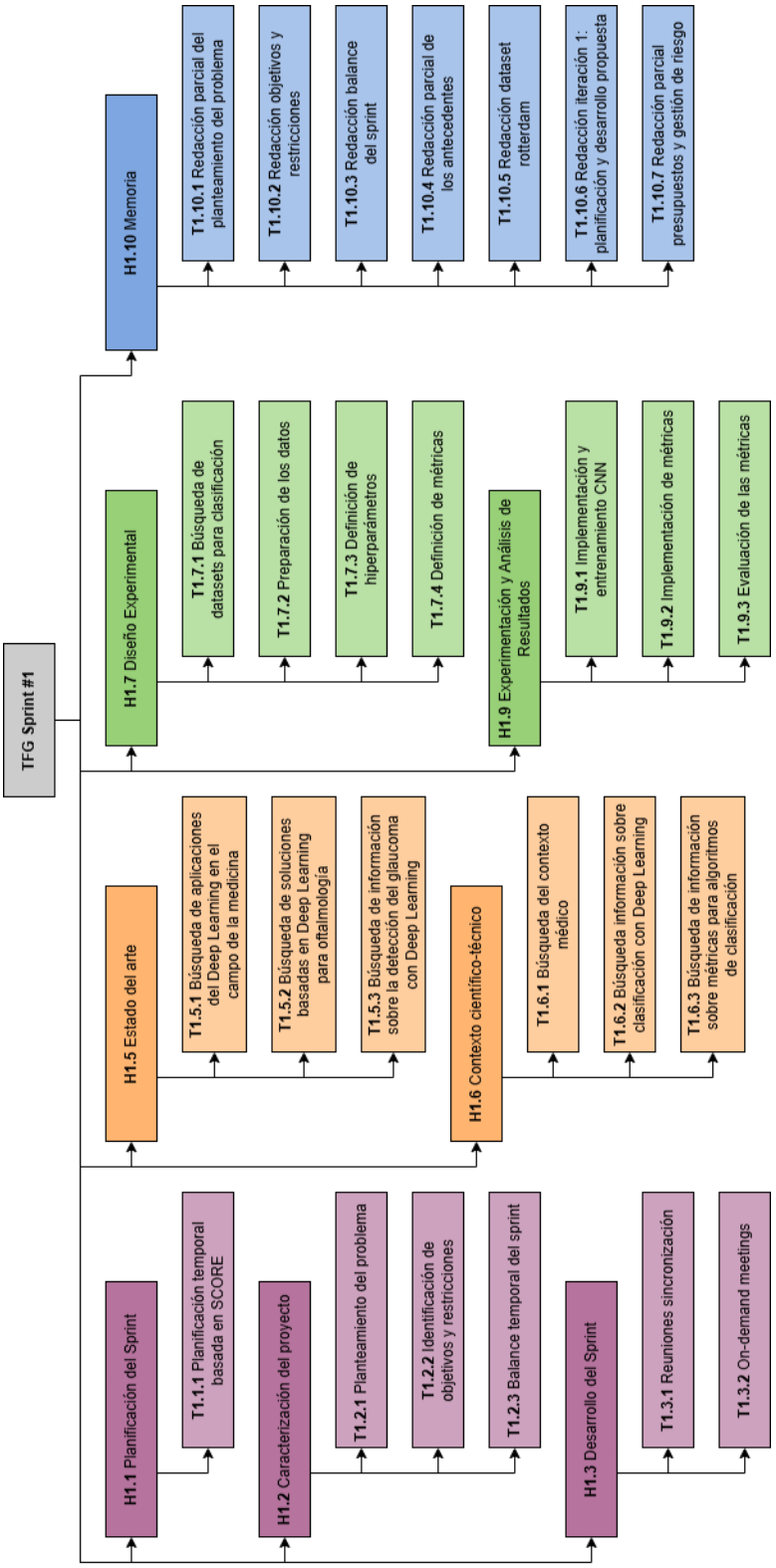


Figura 2.4: EDT del *Sprint* 1.

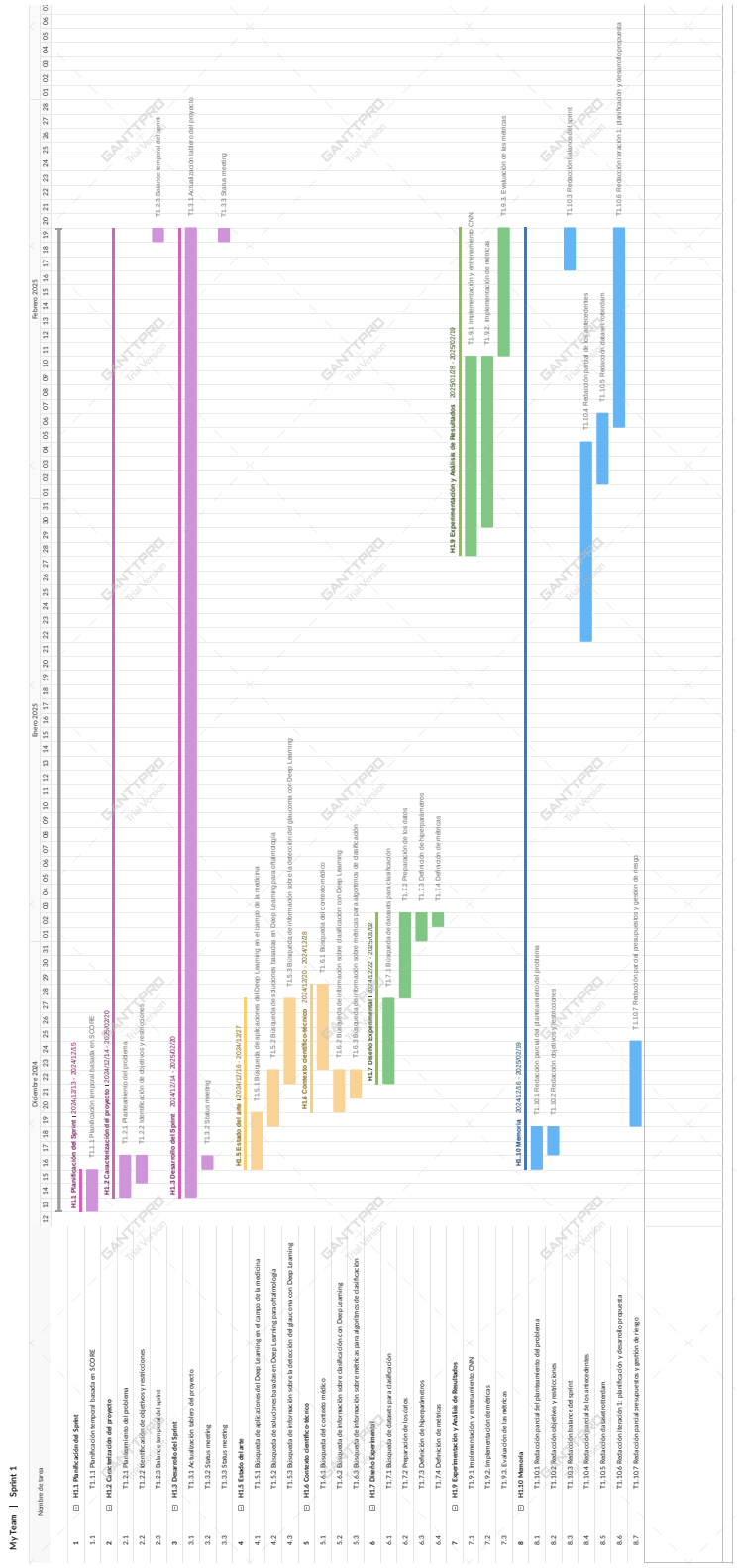


Figura 2.5: Diagrama de Gantt del *Sprint* 1.

Figura 2.6: EDT del *Sprint* 2.

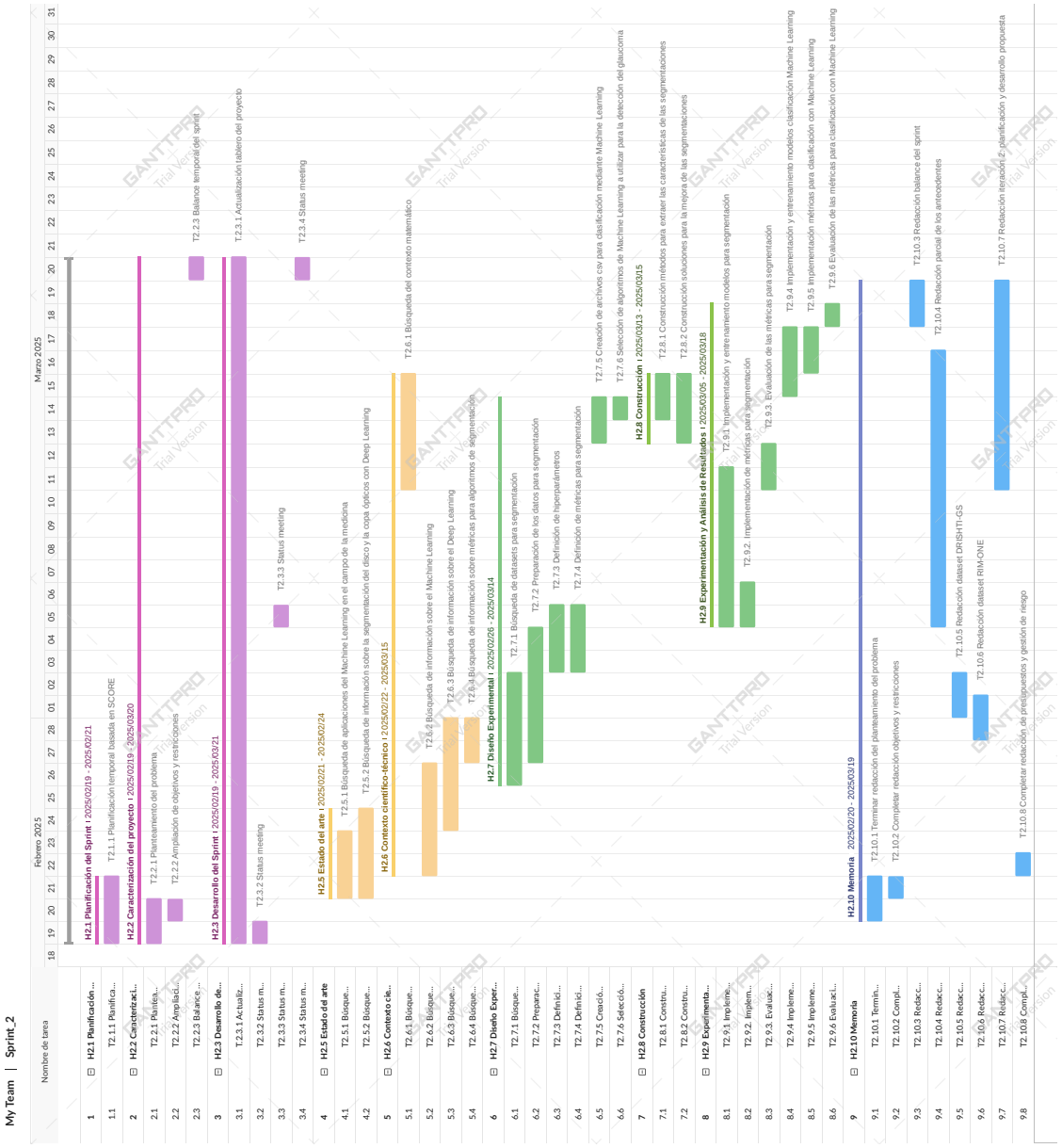


Figura 2.7: Diagrama de Gantt del *Sprint* 2.

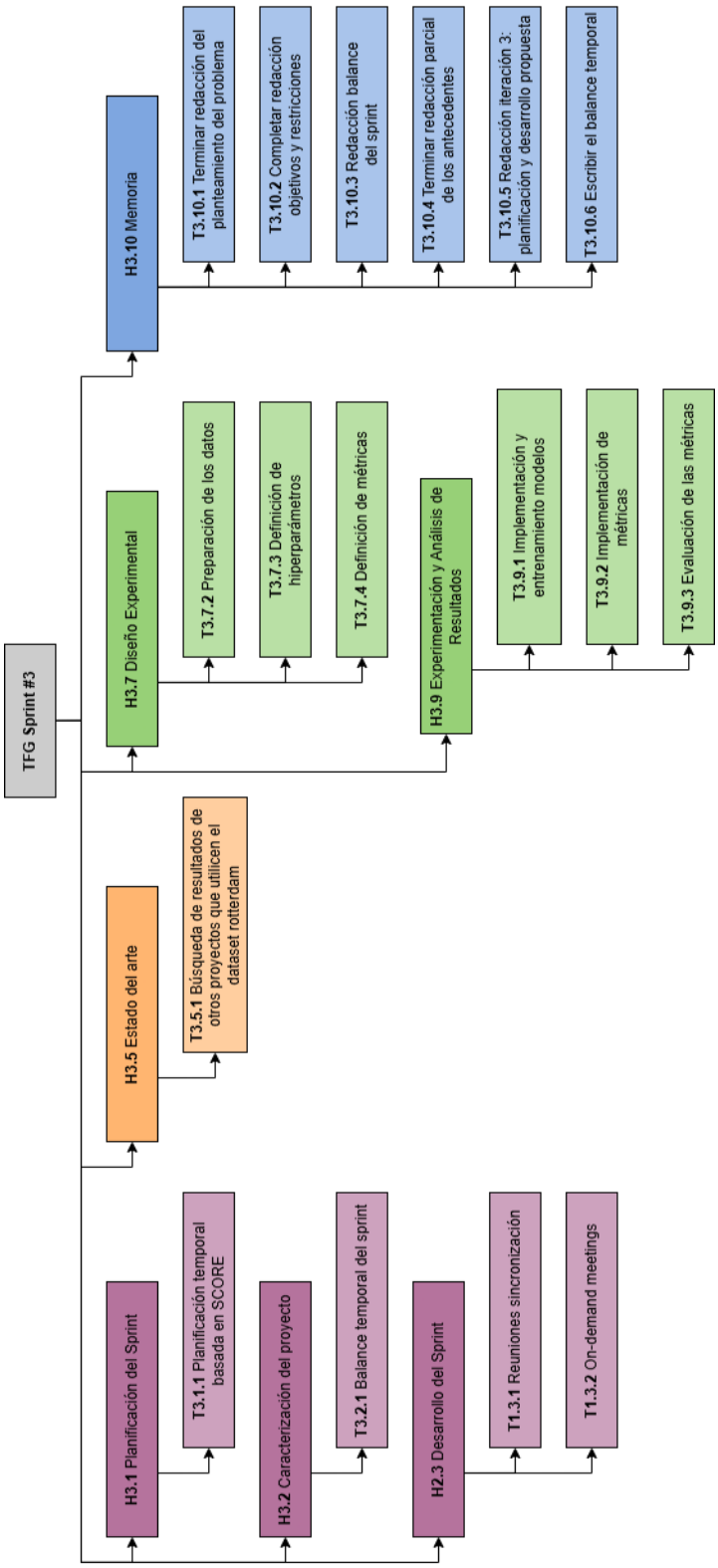


Figura 2.8: EDT del *Sprint* 3.

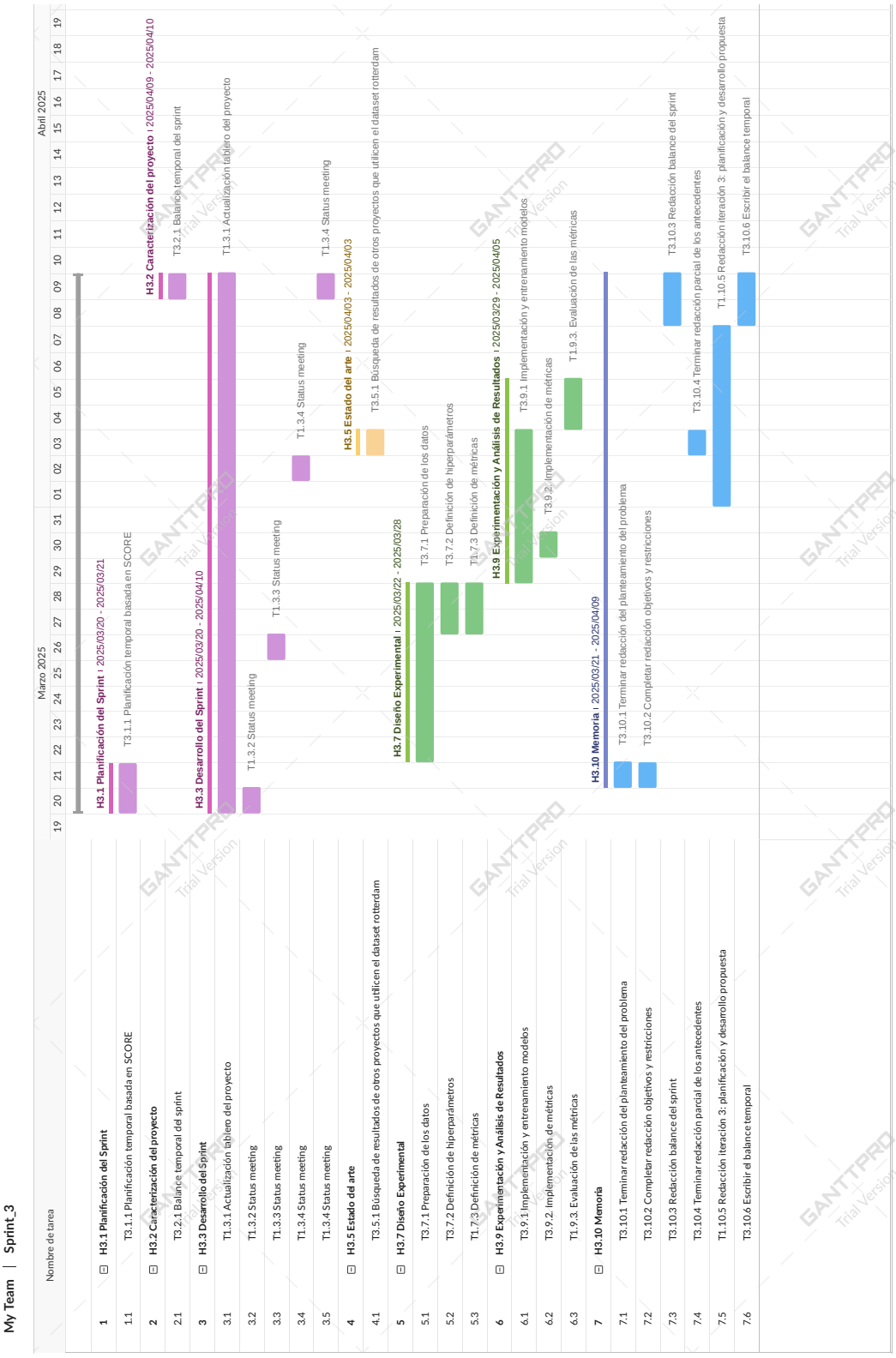


Figura 2.9: Diagrama de Gantt del *Sprint* 3.

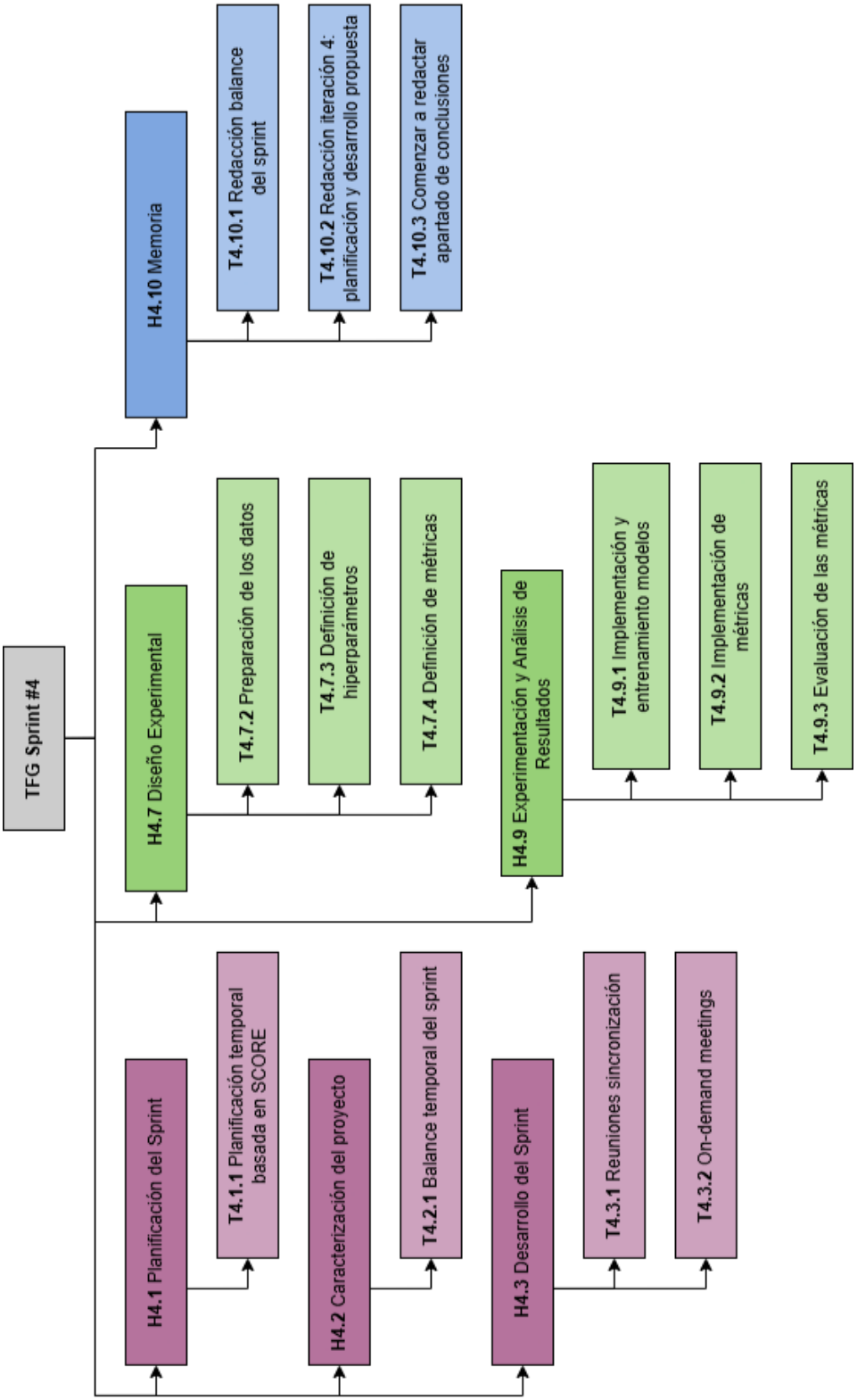


Figura 2.10: EDT del *Sprint* 4.

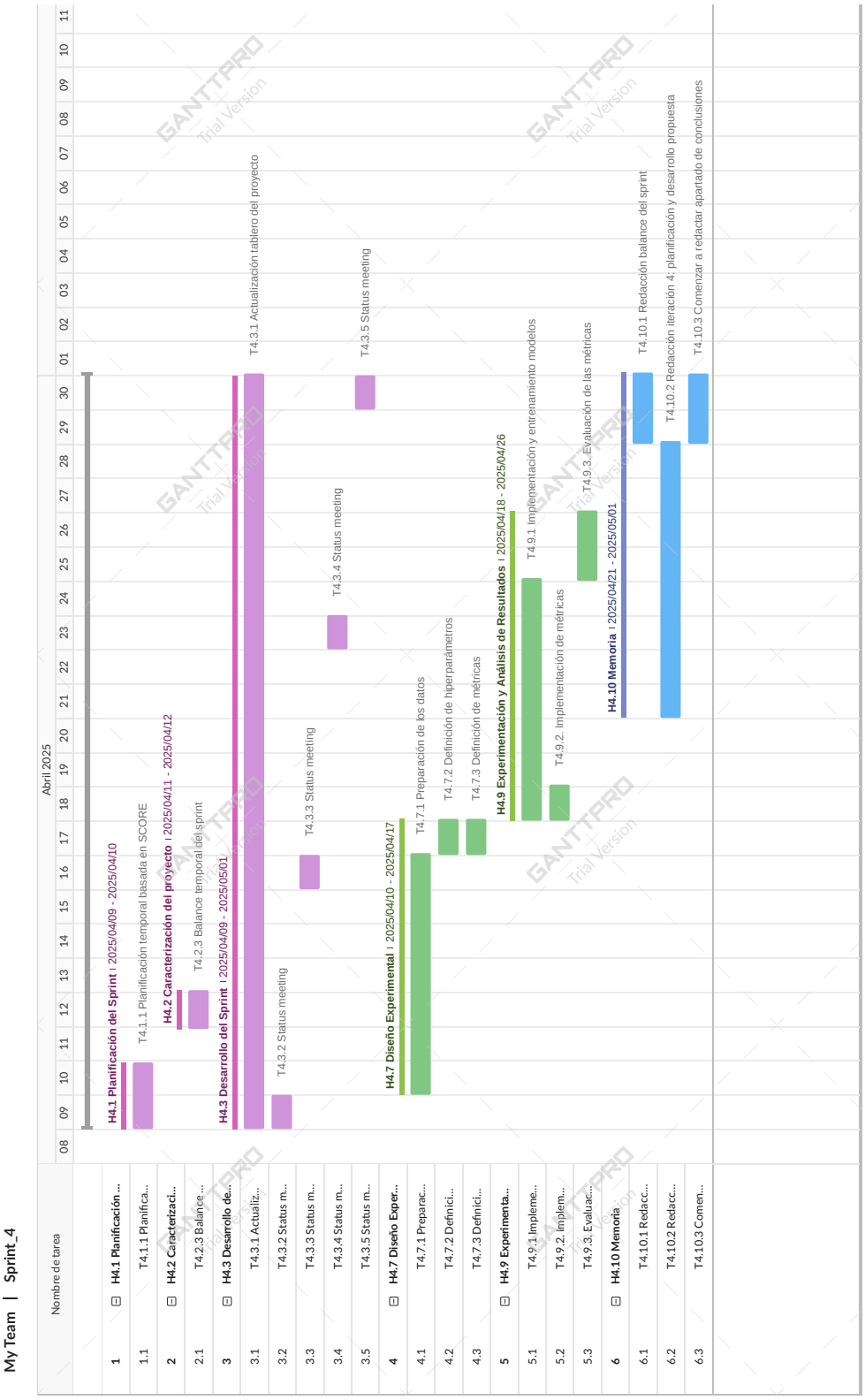


Figura 2.11: Diagrama de Gantt del *Sprint* 4.

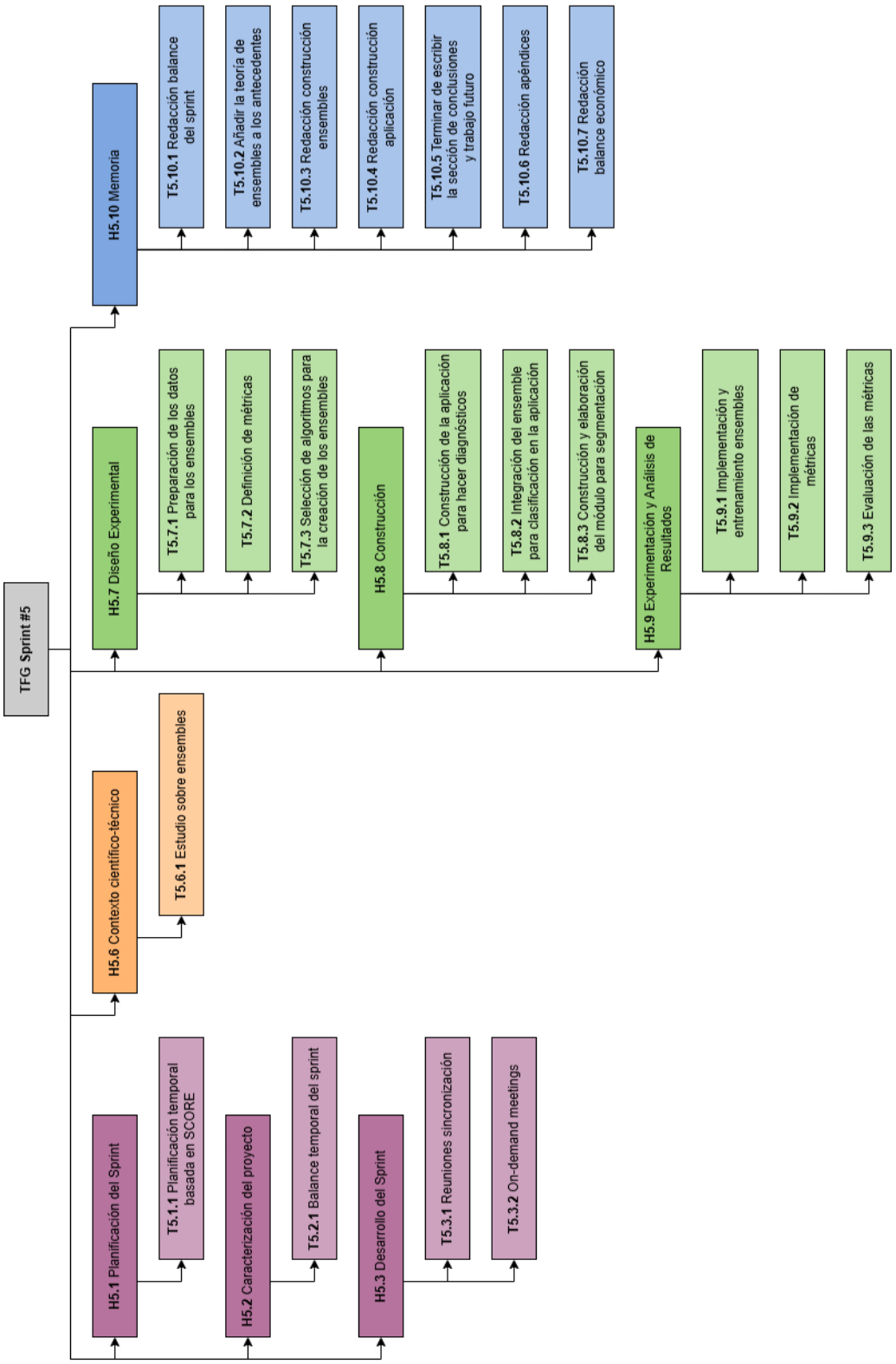


Figura 2.12: EDT del *Sprint* 5.

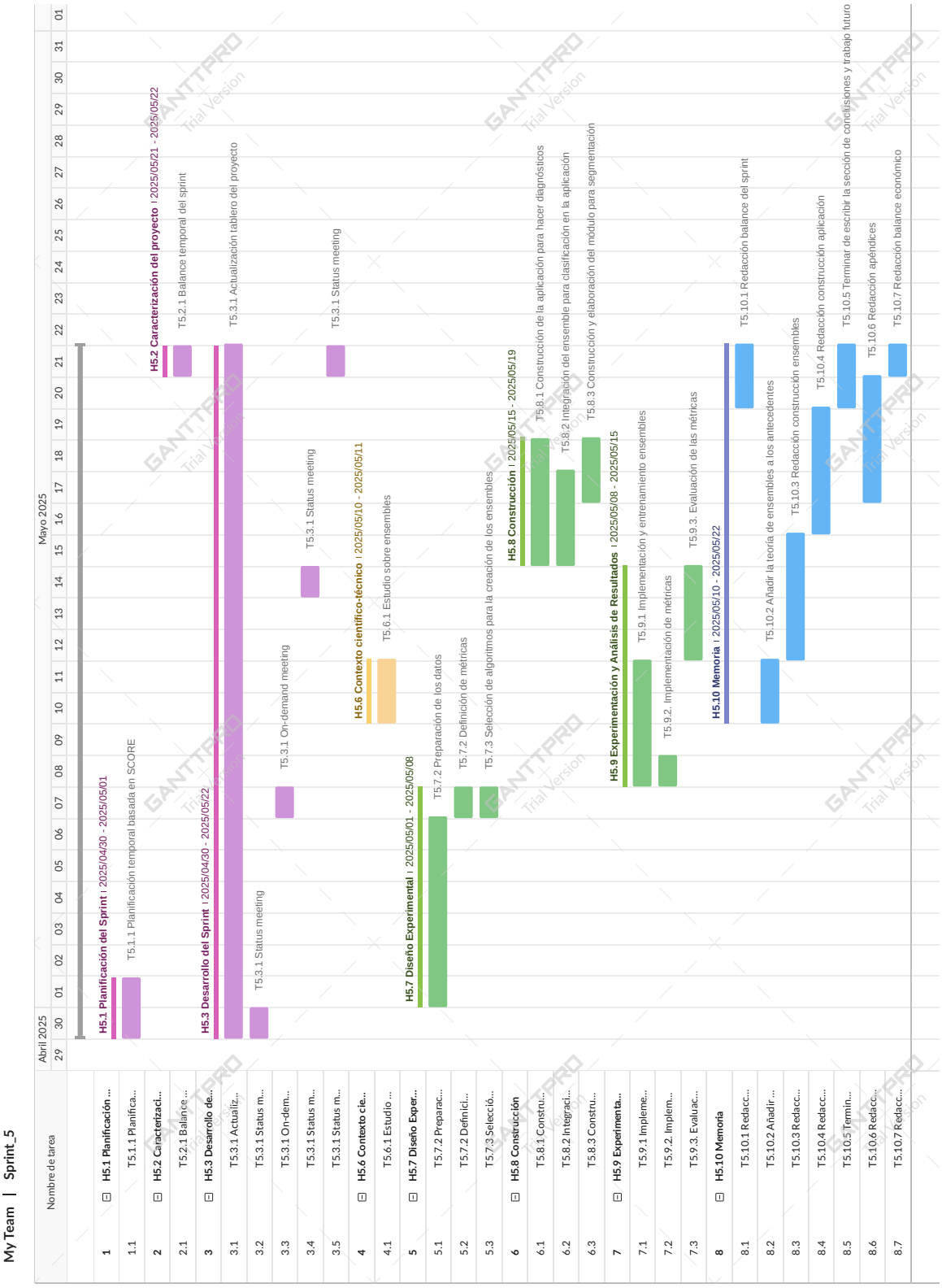


Figura 2.13: Diagrama de Gantt del *Sprint* 5.

2.3. Presupuestos

En esta sección se muestra un análisis detallado de los presupuestos que se deben afrontar para la realización del trabajo. El presupuesto desempeña un papel clave en la planificación y ejecución de cualquier proyecto. Esto se debe a que facilita la estimación de los recursos necesarios para alcanzar los objetivos establecidos. Por ello, a continuación se detallan los costes en materia humana, software y hardware.

2.3.1. Hardware

Respecto a los recursos hardware, es fundamental para la realización de este trabajo el uso de un ordenador y de una conexión Wi-Fi. Respecto al primer componente, se ha hecho uso de un ordenador portátil OMEN con procesador Intel i7-7700HQ, tarjeta gráfica NVIDIA GeForce GTX 1050 y memoria RAM de 16 GB. Además, respecto a la capacidad de almacenamiento, cuenta con 128 GB de disco HDD y 1 TB de SSD. Este portátil se ha completado con un monitor AOC de 144 Hz y 27 pulgadas, junto con un teclado Razer Huntsman Mini y un ratón inalámbrico.

Respecto al Wi-Fi, se dispone de una tarifa plana mensual de la que se emplea para la realización de este proyecto en torno a un 70 %. En la Tabla 2.1 se muestra el porcentaje de uso de cada uno de los elementos descritos hasta el momento, además del coste que representa para la realización del proyecto.

Componente	Coste total	Porcentaje de uso	Total
Ordenador portátil	1000 €	8 %	80 €
Teclado	65 €	15 %	9.75 €
Monitor	165 €	18 %	29.70 €
Ratón inalámbrico	12 €	7 %	0.84 €
Conexión Wi-Fi	20 €/mes	70 % (7 meses)	98 €
			218.29 €

Tabla 2.1: Costes hardware previstos

2.3.2. Software

Los componentes software que se han empleado en el desarrollo del proyecto son todos software libre sin coste, aunque en caso de que se tratase de una empresa, los costes serían distintos. Para la redacción de la memoria se ha utilizado Overleaf, mientras que Trello y Microsoft Teams han facilitado la gestión del trabajo colaborativo, el seguimiento del proyecto y la comunicación.

Por su parte, Google Colab ha sido la plataforma principal para el desarrollo y entrenamiento de los modelos de inteligencia artificial, dado que proporciona un entorno

de ejecución con GPU sin coste. Por otro lado, el entorno Spyder ha sido empleado para realizar pruebas con filtros de imágenes u organizar los conjuntos de datos, por ejemplo. Para la elaboración de diagramas y esquemas representativos, se ha utilizado Draw.io, y para la planificación temporal se ha recurrido a Gantt PRO.

Todo este software será ejecutado a través del mismo ordenador portátil descrito en la Sección 2.3.1 con sistema operativo Windows 10. La Tabla 2.2 recoge los componentes software que se usan a lo largo del proyecto, así como sus respectivos costes.

Componente	Coste	Total
Google Colab	0 €	0 €
Spyder	0 €	0 €
Overleaf	0 €	0 €
Trello	0 €	0 €
Microsoft Teams	0 €	0 €
Draw.io	0 €	0 €
Gantt PRO	0 €	0 €
Windows 10	0 €	0 €
		0€

Tabla 2.2: Costes software previstos

2.3.3. Recursos humanos

Los recursos humanos han sido distribuidos para el desarrollo de un proyecto con una duración prevista de 360 horas. En este caso, como se trata de un TFG personal, todos los roles son encarnados por una misma persona. En total, se han distinguido los siguientes roles en función de la tarea que se desarrolla:

- **Gestor de proyectos**. Planifica y organiza el proyecto. En particular, se encarga de los plazos y los recursos disponibles. Además, será el medio de comunicación entre el equipo y los interesados. Su salario por hora es de 19.23 €/hora, según recoge [73].
- **Analista y Diseñador**. Recoge los requisitos del proyecto, define las funcionalidades que se deben desarrollar para satisfacerlos y diseña la arquitectura de la aplicación antes del desarrollo. Su salario por hora es de 14.23 €/hora, como se observa en [74].
- **Científico de datos**. Se encarga de procesar y analizar los datos, entrenando modelos de Inteligencia Artificial y extrayendo la información que sea útil de los mismos. Como se menciona en [75], su salario por hora es de 18.61 €/hora.

- **Programador y Tester**. El programador implementa el código según los diseños, integra los modelos y construye la aplicación o sistema funcional. Además, para este proyecto también hará las labores de tester. Para ello, se encarga de verificar que la aplicación construida funciona correctamente, detectando los errores de la misma. Su salario por hora es de 14.33 €/hora, como se menciona en [76].

En la Tabla 2.3 se calcula el coste total en materia de recursos humanos que conlleva la realización de este proyecto, teniendo en cuenta las horas de trabajo desempeñadas por cada rol y el sueldo por hora indicado anteriormente. Como se puede comprobar, en las referencias dadas aparece el sueldo anual, por lo que se ha usado [77] para calcular el sueldo por hora.

Cargo	Nº. puestos	Sueldo (por hora)	Horas totales	Salario total
Gestor de proyectos	1	19.23 €/hora	20 horas	384.60 €
Analista/Diseñador	1	14.23 €/hora	65 horas	924.95 €
Científico de datos	1	18.61 €/hora	145 horas	2698.45 €
Programador/Tester	1	14.33 €/hora	130 horas	1862.90 €
				5870.90 €

Tabla 2.3: Costes en materia de recursos humanos previstos

Adicionalmente al coste total en materia de recursos humanos, se debe asumir el coste que supone dar de alta a un trabajador en la Seguridad Social (SS). Esto supone el 28.30 % del salario bruto [55]. Luego, el coste total para afrontar los recursos humanos necesarios para la realización del proyecto es

$$Coste_{RRHH} = Total + SS = 5870.90€ + 28.30\,\% \cdot 5870.90€ = 7532.36€$$

2.3.4. Costes totales

En las Secciones 2.3.1, 2.3.2 y 2.3.3 se analiza detalladamente cada uno de los tipos fundamentales de recursos que interaccionan durante el proyecto. Una vez se ha desglosado cada parte fundamental del presupuesto, se puede hacer un cómputo de los costes totales que presenta el trabajo como se sigue en la Tabla 2.4.

Concepto	Coste
Hardware	218.29 €
Software	0 €
Recursos humanos	7532.36 €
Total	7750.65 €

Tabla 2.4: Costes totales previstos.

2.4. Gestión de riesgos

La gestión de riesgos es un proceso fundamental para identificar, analizar y mitigar los posibles eventos que podrían afectar al desarrollo del proyecto. Lo que se busca al realizar una gestión de riesgos es garantizar que los objetivos del proyecto se cumplan dentro de los plazos, costos y calidad establecidos.

Con este propósito, se identifican los riesgos que pueden hacer peligrar el proyecto. Además, se analizan y evalúan para clasificar según su probabilidad de ocurrencia e impacto para priorizar la atención según su importancia, y se planifica un plan de respuesta para solucionar el problema ocasionado si el riesgo se materializa.

2.4.1. Identificación factores de riesgo

El primer paso para poder gestionar un riesgo es tener constancia de él. La identificación de riesgos consiste en detectar y describir los riesgos potenciales que pueden afectar al desarrollo del proyecto. Un estudio profundo sobre los riesgos que se pueden producir a lo largo del proyecto permite anticiparse a ellos, pudiendo reducir el impacto que tienen sobre el proyecto. En la Tabla 2.5 se recogen los posibles riesgos identificados para este trabajo.

Riesgo	
R-01	Retraso del proyecto respecto a la planificación inicial
R-02	Falta de balanceo en los datos de entrenamiento
R-03	Limitación temporal de uso de GPU en Google Colab
R-04	Falta de experiencia con técnicas de segmentación de imágenes
R-05	Retraso y/o denegación en la obtención de permisos y datos
R-06	Las características como el brillo, el contraste o la saturación disminuyen significativamente el rendimiento del modelo
R-07	Diferencias entre los datos de entrenamiento y clínicos reales
R-08	Modelos de segmentación no detectan con alta precisión las estructuras
R-09	Dificultad para validar los resultados del modelo con expertos médicos
R-10	Las métricas de clasificación no alcanzan valores satisfactorios

Tabla 2.5: Identificación de riesgos.

2.4.2. Estimación de los riesgos

Para cada riesgo identificado se lleva a cabo su debido análisis. En él, se estima la probabilidad de que ocurra el evento estudiado, y el impacto que tiene en caso de ocurrencia. Para poder cuantificar estas estimaciones, se utiliza un valor entero entre 0 y 5.

En la Tabla 2.6 se recoge la estimación sobre la probabilidad que tiene un riesgo de producirse. En concreto, la probabilidad de ocurrencia es la probabilidad de que se produzca el evento que identifica el riesgo si no se lleva a cabo una acción preventiva. Además, en esta tabla, junto con el valor entero del rango $[0, 5]$, se precisará el valor de la probabilidad de ocurrencia entre 0 % y 100 %. Esta probabilidad tiene relación directa con el valor numérico dado previamente.

Riesgo	Probabilidad de ocurrencia	Valor Probabilidad	Justificación
R-01	20 %	2	El proyecto se realiza simultáneamente con otras asignaturas, incluido otro TFG
R-02	40 %	3	Es habitual en investigaciones médicas que los datos contengan muchos ejemplos sanos y pocos casos con la patología en cuestión
R-03	80 %	5	*Google Colab* restringe el uso temporal de la CPU, y los modelos a entrenar requieren de mucho tiempo de entrenamiento
R-04	50 %	3	No se ha trabajado nunca con técnicas de segmentación, tan solo clasificación
R-05	40 %	3	Los datos utilizados para la investigación se solicitan al Hospital Clínico de Valladolid y requieren de autorización previa
R-06	20 %	2	Las imágenes pueden presentar zonas con brillos que afecten a la segmentación y detección
R-07	35 %	2	Dependiendo de la diversidad de ejemplos puede que los datos no cubran todos los casos reales
R-08	30 %	2	Los modelos entrenados pueden no cumplir con las expectativas de rendimiento

Riesgo	Probabilidad de ocurrencia	Valor Probabilidad	Justificación
R-09	10 %	1	Se conocen expertos en la materia a los que habría que contactar para evaluar el producto final
R-10	15 %	1	Puesto que se trata de una investigación, es posible no obtener resutados relevantes

Tabla 2.6: Análisis probabilidad de ocurrencia de cada riesgo.

Por su parte, en la Tabla 2.8 se analiza el impacto que tendrá cada uno de los riesgos identificados en caso de que se produzcan. Además del valor numérico entre $[0, 5]$, el estudio del impacto recoge la pérdida operacional y el impacto en los costes. La pérdida operacional se refiere a las pérdidas económicas que supondría no hacer frente a un riesgo; mientras que el impacto en los costes mide el incremento sobre los costes totales si el riesgo se produce.

Riesgo	Pérdida operacional	Impacto en los costes	Impacto	Justificación
R-01	95.30 €por cada día de retraso	5.8 %	3	Como consecuencia no se entrega en la fecha prevista el proyecto
R-02	-	-	1	Repercute en los resultados, pero no tiene impacto en los costes
R-03	95.30 €por cada día de retraso	11.6 %	5	No tener capacidad suficiente de procesamiento puede retrasar el desarrollo del proyecto
R-04	95.30 €por cada día de retraso	3.6 %	2	El período de aprendizaje puede retrasar el proyecto
R-05	95.30 €por cada día de retraso	4.3 %	3	La búsqueda de otras fuentes de datos puede retrasar el proyecto
R-06	95.30 €por cada día de retraso	2.9 %	1	La construcción de filtros que mitiguen los brillos de las imágenes puede retrasar el proyecto
R-07	-	-	1	Repercute en la utilidad de la investigación, pero no tiene impacto en los costes
R-08	-	-	1	Repercute en los resultados, pero no tiene impacto en los costes
R-09	-	-	1	El producto final no tendrá el mismo respaldo, sin afectar a los costes

Riesgo	Pérdida operacional	Impacto en los costes	Impacto	Justificación
R-10	-	-	1	Repercute en los resultados, pero no tiene impacto en los costes

Tabla 2.7: Análisis impacto de cada riesgo.

2.4.3. Matriz de Probabilidad × Impacto

La matriz de Probabilidad × Impacto es un artefacto que ayuda a organizar la prioridad que se debe dar a un riesgo. Para ello, se calcula la exposición, que viene dada por

$$Exposición = Probabilidad \times Impacto.$$

De esta forma, se asociada cada riesgo a un único valor, denominado exposición, que contempla tanto la probabilidad de ocurrencia como el impacto. Este valor permite clasificar los riesgos según la prioridad con la que se deben tratar:

- Prioridad alta (en color rojo): $Exposición \geq 10$.
- Prioridad media (en color amarillo): $5 \leq Exposición \leq 10$.
- Prioridad baja (en color verde): $Exposición \leq 5$.

En la Tabla 2.8 se muestra la probabilidad e impacto de cada riesgo identificado, así como su valor de exposición.

Riesgo	Probabilidad	Impacto	Exposición
R-01	2	3	6
R-02	3	1	3
R-03	5	5	25
R-04	3	2	6
R-05	3	3	9
R-06	2	1	2
R-07	2	1	2
R-08	2	1	2
R-09	1	1	1
R-10	1	1	1

Tabla 2.8: Matriz de Probabilidad × Impacto

Atendiendo al valor de exposición presentado en la Tabla 2.8, se observa que el único riesgo con prioridad alta es R-03. Este riesgo necesita de una actuación inmediata para que no interfiera con la actividad del proyecto. A continuación se presentará un plan de contingencia tanto para los riesgos con más prioridad como para el resto.

2.4.4. Plan de contingencia

Tras analizar en detalle cada uno de los riesgos identificados, se plantea el plan de contingencia para cada uno de ellos recogido en la Tabla 2.9. Además, se presentan en orden de prioridad atendiendo a su valor de exposición.

Riesgo	Plan de contingencia
R-01	Establecer una planificación realista teniendo en cuenta períodos de exámenes y ritmo de trabajo
R-02	Usar técnicas de data-augmentation
R-03	Tener disponible más de un entorno de Google Colab
R-04	Dedicar tiempo de investigación y lectura a la adquisición de los conocimientos necesarios
R-05	Explorar distintas vías de obtención de los datos
R-06	Experimentar con distintos filtros en el preprocesado de imágenes y analizar los resultados obtenidos con cada uno
R-07	Controlar una amplia variedad de imágenes de prueba para comprobar la correcta funcionalidad independientemente de la imagen analizada
R-08	Plantear distintas soluciones en la construcción de los modelos y realizar un postprocesado para mejorar la segmentación
R-09	Controlar una amplia variedad de imágenes de prueba para comprobar la correcta funcionalidad independientemente de la imagen analizada
R-10	Explorar distintas vías de clasificación desde Machine Learning a partir de los parámetros extraídos en las segmentaciones como clasificación con *Deep Learning*. Mostrar un análisis detallado con idependencia de los resultados.

Tabla 2.9: Plan de contingencia de riesgos

2.5. Balance temporal y económico

Esta sección tiene como objetivo analizar dos aspectos fundamentales del desarrollo de un proyecto: el tiempo y los recursos económicos que se han utilizado para su realización.

2.5.1. Balance temporal

En el balance temporal del proyecto se presentan los recursos temporales requeridos para realizar las distintas fases de trabajo. De esta forma, se especificarán a continuación las desviaciones ocurridas respecto a la planificación prevista que se describió en la Sección 2.2.

La primera de las desviaciones respecto a la planificación temporal repercute en el estándar Comunicación. Las tareas que se recopilaron en las EDT de cada sprint se referían a la elaboración de la memoria. Sin embargo, no se reservó tiempo para la redacción de la memoria. En consecuencia, se retrasa la finalización del proyecto completo y de la preparación para la defensa dos días respecto de la planificación inicial.

Por otra parte, aunque en la planificación temporal no quedó reflejado, dada la potencial aplicación en el campo de la medicina para apoyar el diagnóstico del glaucoma a través de una solución como la que se propone desarrollar, se consideró oportuno realizar una solicitud de acceso a un conjunto de retinografías en posesión del Hospital Clínico Universitario de Valladolid, con el objetivo de contar con datos clínicos reales y representativos. El objetivo de esta petición es poder colaborar con el sistema de sanidad pública, desarrollando una herramienta de manera gratuita para su mejora o evolución.

Sin embargo, tras un total de tres meses de trámites burocráticos, la autorización para la concesión del dataset fue rechazada por el Comité de Ética de Investigación Clínica del Área de Salud de Valladolid, quedando supeditada a una posible revisión y aprobación por parte de la Consejería de Sanidad. Como se pretendía acabar con la realización de este proyecto durante el curso 2024-2025, frente al aplazamiento y demora para la autorización de acceso a las imágenes, se optó por paralizar esta vía y comenzar este estudio a partir de conjuntos de datos públicos.

Luego, como se ha mencionado con anterioridad, en la planificación temporal no quedó reflejado este hecho porque no se realizó la misma, ni se comenzó el proyecto, hasta que no se tuvo la resolución de la solicitud. Aún así, se ha optado por explicar este hecho como parte del balance temporal, ya que tiene repercusiones en cuanto al retraso en el comienzo del proyecto.

2.5.2. Balance económico

Aquí se detalla el coste que acarrea este proyecto y su desviación respecto al presupuesto estimado descrito en la Sección 2.3. Igual que se hizo para el presupuesto, para el balance económico se describen los costes en recursos humanos, software y hardware.

Respecto al software y hardware utilizados para el desarrollo del proyecto, no se ha requerido del uso de ningún recurso adicional. Por otro lado, como se ha indicado en

el balance temporal de la Sección 2.5.1, la no reserva de tiempo para la elaboración de la presentación que se utilizará en el evento de defensa del TFG retrasa en dos días la finalización del proyecto, que, como se describió en la Sección 2.4, conlleva un coste de 95.30 €por cada día de retraso. Por tanto, esto eleva los costes finales a un total de

$$7750.65\ € + 2 \cdot 95.30\ € = 7941.25\ €.$$

Capítulo 3

Antecedentes

En este capítulo se describe en detalle el contexto científico-técnico del proyecto; es decir, la presente sección trata de ubicar el marco teórico general sobre el cual se asienta la solución propuesta al problema planteado de detección del glaucoma. Para ello, se dividen los antecedentes en tres partes claramente diferenciadas en función del área de conocimiento en el que se incluyen:

- **Área médica**. Para entender con claridad el objetivo que se persigue con este proyecto y las soluciones definidas, es importante conocer el contexto médico sobre el que se trabaja. Con este propósito, en la Sección 3.1 se ilustra la anatomía y fisiología del sistema ocular. Tras esto, se presenta la patología que se pretende diagnosticar, el glaucoma.

  Además, se explica qué es una retinografía, puesto que es el tipo de dato que se usa para construir la solución que se pretende que detecte el glaucoma. Junto con la retinografía, se explican las estructuras que se identifican en la misma, pues serán de utilidad a lo largo del proyecto.

- **Área informática y de ciencia de datos**. A la vista de los objetivos del proyecto descritos en la Sección 1.2, este es el campo de conocimiento principal sobre el que se desarrolla este trabajo. A lo largo de la Sección 3.2 se tratan las técnicas de *Machine Learning*, y *Deep Learning* que pueden ser de interés para la persecución de los objetivos. Junto con el concepto de *Deep Learning*, se trata el de redes neuronales sobre el que se sustenta.

  Por otra parte, para evaluar los modelos de predicción construidos a través de las técnicas de *Machine Learning* y *Deep Learning*, es necesario el uso de métricas que permitan identificar cuál es la mejor solución. Por este motivo, en la Sección 3.2.4 se explican las métricas utilizadas para este trabajo. Además, en la construcción de los modelos se contemplan distintas variaciones de los datos aplicando una serie de filtros que se describen en la Sección 3.2.5.

- **Área matemática**. Tras encontrar las estructuras propias de una retinografía, se lleva a cabo un postprocesado de la imagen. Con este fin, se explotan técnicas

que emplean conceptos matemáticos como el de envolvente convexa o componentes conexas, definidos a lo largo de la Sección 3.3.

3.1. Contexto médico

A lo largo de esta sección se detalla la anatomía y fisiología del sistema ocular. Tras entender esto, se estará en disposición de presentar qué es el glaucoma, sus causas y sus consecuencias. Finalmente, se explica en qué consiste una retinografía, así como las estructuras que se distinguen en la misma.

3.1.1. Anatomía y fisiología ocular

A grandes rasgos, el funcionamiento del ojo es similar al de una cámara fotográfica. Su funcionamiento consiste en capturar los rayos de luz entrantes en el globo ocular para que el cerebro los interprete. A continuación se explica este proceso en detalle, describiendo cada uno de los elementos que conforman el sistema óptico representado en la Figura 3.1.

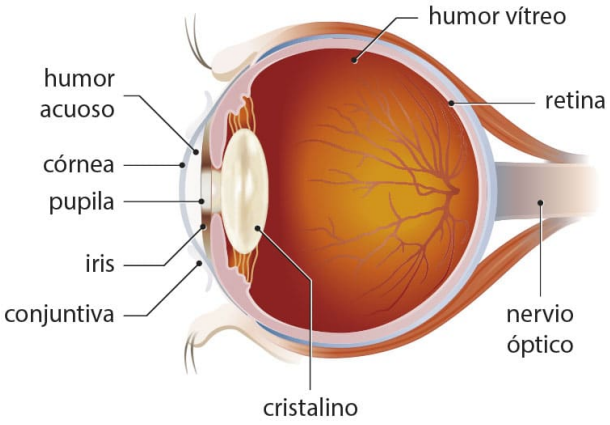


Figura 3.1: Anatomía del sistema óptico [82].

En primera instancia, cuando la luz llega a la parte frontal del globo ocular, se topa con dos lentes. Estas lentes son la córnea y, tras esta, el cristalino. Además, entre estas dos estructuras se localiza el iris, y la cavidad que genera, la pupila. La función de la pupila es regular la cantidad de luz entrante en el globo ocular; mientras que el iris es quien controla el tamaño de la pupila para que esto suceda.

Por tanto, los rayos de luz atraviesan una primera lente, que es la córnea. A continuación, pasan por la pupila para llegar al cristalino. Este último es una lente flexible que se encarga de enfocar la luz para proyectarla contra las paredes del fondo del globo ocular, como se representa en la Figura 3.2. Estas paredes son la retina.

La retina es una capa de tejido nervioso que recubre la parte interna y posterior del ojo. Siguiendo con la analogía de la cámara, la retina es el sensor de esta. Como se ha explicado, su función principal es captar la luz y convertirla en señales eléctricas para

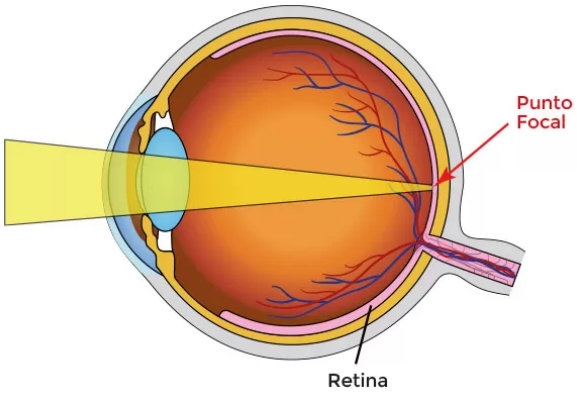


Figura 3.2: Diagrama representativo del funcionamiento del ojo [83].

que viajen hasta el cerebro a través del nervio óptico. Con este fin, la retina contiene una serie de subcapas, recogidas en la Figura 3.3, cada una con una función específica, para recoger los distintos colores que se proyectan sobre la retina y reportar esta información al cerebro a través del nervio óptico.

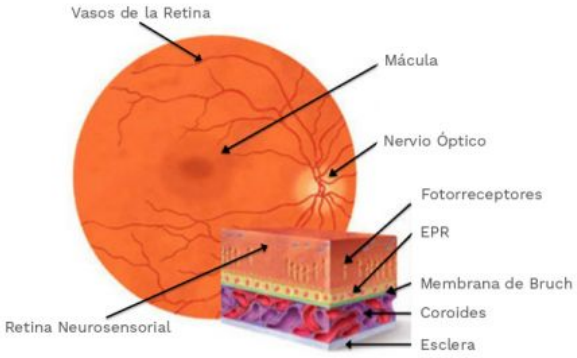


Figura 3.3: Diagrama representativo del funcionamiento del ojo [84].

Esta última estructura mencionada, el nervio óptico, está compuesto por más de un millón de fibras nerviosas que son los axones de las neuronas que recogen la información proyectada sobre la retina [36]. Para entenderlo con una analogía propuesta por la Academia Americana de Oftalmología [2], el nervio óptico es similar a un cable eléctrico compuesto por muchos alambres pequeños que conducen la información al cerebro.

Una vez entendido a grandes rasgos cómo funciona el sentido de la vista, queda destacar un elemento importante en este sistema previo a la introducción del glaucoma, el humor acuoso. Este es un líquido transparente que se encarga de lubricar las dos lentes del ojo: la córnea y el cristalino. Como relata la Academia Americana de Oftalmología [2], el ojo produce humor acuoso constantemente, y a medida que fluye nuevo humor acuoso en el ojo, debe drenarse la misma cantidad. Además, cabe mencionar que la transparencia de este líquido permite el paso de la luz.

3.1.2. Glaucoma

El glaucoma es una enfermedad ocular que daña el nervio óptico. Generalmente, se produce cuando aumenta la presión en el ojo. Como se ha explicado en la sección anterior, el sistema ocular genera humor acuoso de manera sostenida. Si no se drena por completo, se acumula este fluido, aumentando la presión ocular. Este aumento en la presión ocular puede dañar al nervio óptico, lo que conlleva el surgimiento del glaucoma [2].

Como consecuencia al aumento de la presión, las fibras nerviosas del nervio óptico mueren. Debido a la pérdida de las fibras nerviosas, no se recoge la luz proyectada en los lugares donde han muerto; de esta forma, se desarrollan puntos ciegos en la visión. En general, las fibras que se pierden primero son las más alejadas del nervio óptico, lo que hace que el campo de visión se vaya cerrando. Puesto que este proceso es paulatino, puede que el paciente no note los puntos ciegos hasta que hayan muerto la mayoría de las fibras del nervio óptico. En caso de que todas las fibras mueran, el usuario que padezca glaucoma se quedará ciego.

Estadísticas

El glaucoma es una enfermedad relevante por su frecuencia y potencial gravedad, ya que, como asegura la Sociedad Española del Glaucoma (SEG) [3], esta enfermedad es, junto con la diabetes (retinopatía diabética), la principal causa evitable de ceguera en España; además de afectar a más del 3 % de la población. Como reportaba la Cadena SER el pasado mes de abril de 2025 [37], en un solo hospital, en este caso el Hospital Universitario de Elche, se diagnostican más de 400 casos de glaucoma anualmente.

Según el Instituto Catalán de la Retina (ICR) [4] y la SEG [3], esta enfermedad es crónica, progresiva e irreversible. Luego, la ceguera podría evitarse diagnosticando y tratando la enfermedad de manera adecuada, cobrando especial relevancia facilitar el diagnóstico precoz de la enfermedad.

Ampliando el foco al contexto internacional, según la *Glaucoma Research Foundation* de Estados Unidos [38], la prevalencia en el país es de alrededor de 4.22 millones de personas. Globalmente, esta cifra alcanzó en 2020 los 80 millones de personas. Reforzando la proposición de que es una causa importante de ceguera evitable junto con la diabetes, según este mismo organismo, el glaucoma encabeza las causas de ceguera globales, solo por detrás de las cataratas, siendo responsable de entre el 9 % y el 12 % de los casos, lo que representa 5.9 millones de personas.

3.1.3. Retinografía

Una retinografía es una prueba diagnóstica no invasiva que permite capturar imágenes detalladas de la parte posterior del ojo; es decir, del fondo del ojo [39]. Para llevar a cabo una retinografía, se emplea un microscopio adaptado junto con una cámara de manera que proporcionan una vista de en torno a 50^{o} en torno al nervio óptico para una retinografía central, o si se trata de una retinografía de campo amplio, más de 200^{o} de amplitud [40].

Para llevar a cabo esta prueba, si se trata de una retinografía central, se deben aplicar unas gotas de un colirio ciclopéjico en la superficie del ojo para dilatar la pupila. Como indica el Instituto de Microcirugía Ocular [40], en ambos casos se tarda entre 5 y 10 minutos en tomar la imagen, aunque para el caso de la retinografía central hay que esperar previamente en torno a 15 minutos para que el colirio dilatador haga efecto.

Una retinografía es de utilidad para la detección de enfermedades que afectan a la parte posterior del ojo; en particular, el glaucoma. Además, presenta un aspecto como el que se muestra en la Figura 3.3, pudiendo identificar una serie de estructuras clave:

- **Mácula**. Es la parte central de la retina y aparece de color más oscuro en las retinografías. En la Figura 3.4, aparece una parte con mayor brillo, y a su izquierda otra región con mayor sombreado. Esta segunda es la mácula. Esta estructura se puede identificar en la representación de la Figura 3.2 como el punto focal. La mácula es la zona con más conos; es decir, fotorreceptores especializados en el color y el detalle. Además, a la parte central de la mácula se le denomina fóvea.

Figura 3.4: Diagrama identificativo del disco y la copa en una retinografía [9].

- **Nervio óptico**. Esta parte se ha explicado en la Sección 3.1.1. Además, en la retinografía es fácilmente distinguible por tener tonalidades más claras y presentar un mayor brillo, como en el ejemplo de la Figura 3.4. Por otra parte, la región del nervio óptico se identifica cuando se localizan las dos estructuras que se tratan a continuación, y se denomina comúnmente zona o región ONH.

- **Disco óptico**. Punto de unión entre el nervio óptico y el ojo. Es una estructura redonda y clara, visible en las retinografías como se muestra en la Figura 3.5. Además, por esta región del ojo es por donde entran y salen los vasos sanguíneos.

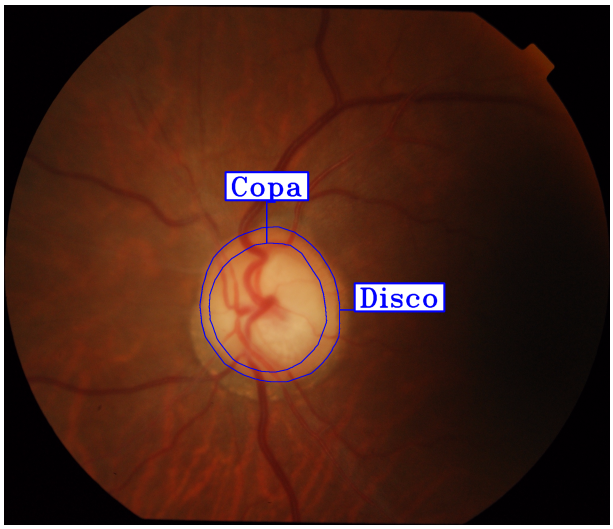


Figura 3.5: Diagrama identificativo del disco y la copa en una retinografía [11].

- **Copa óptica**. Parte central del disco óptico. Su disposición frente a la del disco viene identificada en la Figura 3.5. Su tamaño varía dependiendo de la persona. Además, en presencia de glaucoma, la copa puede agrandarse. Para la detección de esta patología, cuanto más grande es la copa respecto al disco, es más probable que el nervio óptico esté dañado.

3.2. Contexto teórico

Este proyecto se sustenta sobre el área de conocimiento de la Inteligencia Artificial. Su objetivo es la construcción de modelos. En el campo de la Inteligencia Artificial, un modelo es una representación computacional que, por lo general, aprende a realizar una tarea específica a partir de los datos. También puede hacerlo a partir de reglas si se trata de sistemas expertos. En concreto, es un sistema que aprende patrones de los datos para predecir, clasificar, detectar o generar información nueva.

En la propuesta de solución del proyecto, se emplearán distintos procesos para la construcción de modelos. A lo largo de esta sección, se repasará de manera teórica cada una de estas técnicas, desde los algoritmos típicos de *Machine Learning*, pasando por las redes neuronales, hasta la construcción de modelos propios del *Deep Learning*.

3.2.1. Machine Learning

El *Machine Learning* o Aprendizaje Automático es la rama de la Inteligencia Artificial que se centra en el diseño y desarrollo de algoritmos que permiten a los ordenadores mejorar su desempeño en la realización de una tarea a partir de la experiencia [20]. Esta experiencia viene dada en forma de datos. Luego, como se explica en [21], el *Machine Learning* permite a los sistemas aprender y mejorar de forma autónoma a partir de datos.

Siguiendo [21], según la forma en la que se utilicen los datos para adquirir conocimiento, podemos dividir los algoritmos de Machine Learning en distintas categorías: aprendizaje supervisado, no supervisado, semisupervisado, autosupervisado y por refuerzo [28]. A continuación se explica con más detalle cada una de ellas, prestando más atención a las dos primeras, pues son las que aparecen en el desarrollo de la propuesta de solución del problema que se explica en la parte II de este documento.

Tipos de algoritmos de Machine Learning

- **Aprendizaje supervisado**: Este tipo de *Machine Learning* se basa en entrenar modelos con datos etiquetados; es decir, de los que se conoce el resultado esperado [23]. El objetivo es aprender un cierto concepto a partir de ejemplos de los que se conocen un conjunto de atributos y los resultados esperados (variable objetivo). Como se distingue en [28], a su vez el aprendizaje supervisado comprende dos variantes principales:

  - **Algoritmos de regresión**: la variable objetivo es una variable numérica continua [23]. Estos algoritmos tratan de encontrar una función que lleve un conjunto de atributos conocidos en el valor de la variable objetivo. Luego, tratan de ajustar los datos a una función minimizando el error cometido, surgiendo así algoritmos como los de regresión lineal o regresión logística.
  - **Algoritmos de clasificación**: la variable objetivo es discreta [23]; es decir, se busca clasificar cada elemento en una de las categorías definidas. Este tipo de algoritmos se basan en la búsqueda de patrones que sigan las características de cada clase y usar ese conocimiento para determinar la clase a la que pertenecen nuevos elementos.

  Entre los algoritmos de aprendizaje supervisado que se emplearán en este proyecto, se incluyen algunos como K-NN (*K-Nearest Neighbors* - K vecinos más cercanos) [30], SVM (*Support Vector Machines* - Máquinas de Vectores Soporte) [31], árboles de decisión [24] o aprendizaje bayesiano [23]. Todos ellos se revisitarán más adelante para dar una explicación detallada de cada uno.

- **Aprendizaje no supervisado**: frente a los algoritmos de aprendizaje supervisado, se tienen los algoritmos de aprendizaje no supervisado. Estos se emplean cuando no existe ningún tipo de etiquetado en los datos y no se necesita ningún conocimiento previo [28]. Además, como se explica en [29], se pueden dar tres enfoques principales para los algoritmos de aprendizaje no supervisado:

  - **Algoritmos de clustering**: la agrupación en clústeres es una técnica que agrupa datos no etiquetados en función de sus similitudes o diferencias [29]. Entre los algoritmos de *clustering* destaca el de *k-means* o k-medias [32], del que se hace uso para la solución propuesta al problema planteado.

  - **Algoritmos de asociación**: método basado en una serie de reglas que se emplean para detectar y abstraer las relaciones existentes entre un conjunto de datos determinado.
  - **Algoritmos de reducción de dimensionalidad**. En ocasiones el conjunto de atributos o características que se tiene sobre los datos es demasiado elevado. Esto incide en el rendimiento y dificulta la visualización de los datos. Para solucionar este problema se lleva a cabo la técnica de reducción de dimensionalidad, que consiste en disminuir el número de entradas de datos a un tamaño manejable, tratando de perder la menor cantidad de información posible. Como se describe en [29], los principales algoritmos de reducción de la dimensionalidad son: el análisis de componentes principales (PCA) [33], la descomposición en valores singulares [35], y los codificadores automáticos [34].

- **Aprendizaje por refuerzo**: entrenamiento de los algoritmos a través de un sistema de recompensa y castigo [23]. Típicamente, se define un agente que realiza acciones en un entorno específico para alcanzar un objetivo determinado. Por otra parte, se contempla una métrica para recompensar o penalizar las acciones que el agente toma para lograr el objetivo. Se usan con frecuencia para enseñar a los robots a reproducir tareas humanas.

- **Aprendizaje semisupervisado**: combina el aprendizaje supervisado y el no supervisado; es decir, dentro del conjunto de datos con el que se construye el modelo, se tiene un conjunto de datos no etiquetados, y otro etiquetado [23] que guía el proceso de aprendizaje. Usualmente, el conjunto de datos sin etiquetar es mucho mayor que el etiquetado [28].

- **Aprendizaje autosupervisado**: estos modelos, también llamados de aprendizaje predictivo, usan datos no etiquetados. Emplean parte de la entrada para aprender de la otra parte, generando así etiquetas y transformando los problemas no supervisados en supervisados [28]. Este tipo de algoritmos es usado para problemas de *computer visión* o procesamiento del lenguaje natural.

Algoritmos Machine Learning

A continuación se van a definir brevemente ciertos algoritmos de Machine Learning, que típicamente son de los más utilizados, puesto que se hará uso de ellos en el desarrollo de la propuesta del Capítulo 4. Estas explicaciones están extraídas en su mayoría de [23]. La mayor parte de los algoritmos que se mencionan son de aprendizaje supervisado, puesto que conocemos los resultados esperados; es decir, el valor objetivo. Aunque también se menciona algún algoritmo de aprendizaje no supervisado, más concretamente, de clustering.

K-Nearest Neighbors (K-NN). Este algoritmo de aprendizaje supervisado almacena los ejemplos de entrenamiento. Para clasificar una nueva instancia, utiliza una función de distancia para determinar los K elementos más cercanos. Por ejemplo, supongamos

que la nueva instancia es $x = (x_1, \ldots, x_n)$ donde n es el número de atributos conocidos. Si usamos $K = 1$ y tomamos la distancia euclídea, clasificaremos el nuevo elemento igual que el elemento conocido $y = (y_1, \ldots, y_n)$ que minimice

$$d(x, y) = \sqrt{(x_1 - y_1)^2 + \cdots + (x_n - y_n)^2}.$$

En resumen, para $K = 1$, el nuevo elemento que se pretende clasificar se encasilla en la misma clase del elemento conocido más cercano, como se representa en la Figura 3.6. En general, se clasifica a partir de los K elementos más próximos de manera análoga a partir de la clase a la que pertenecen.

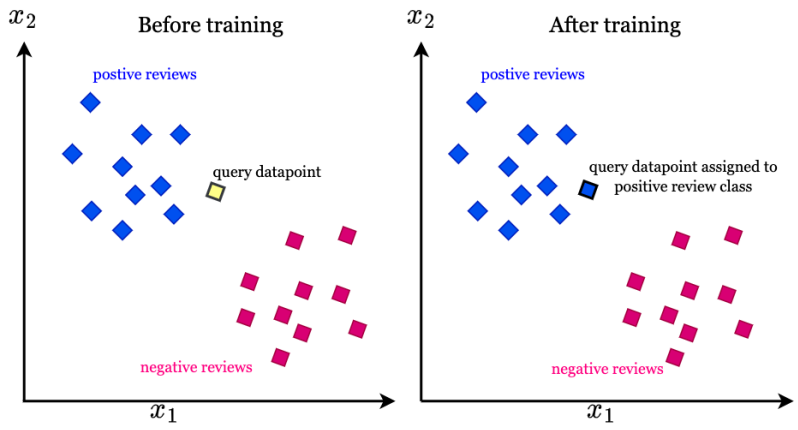


Figura 3.6: Representación de una predicción con el algoritmo K-NN [85].

Aprendizaje bayesiano. Dado un conjunto de atributos, permite determinar la hipótesis más probable (h) para un conjunto de entrenamiento (D) a partir del teorema de Bayes:

$$P(h|D) = \frac{P(D|h)P(h)}{P(D)}.$$

Naive Bayes es el algoritmo principal, aunque existen otras técnicas como Bernoulli-Naive Bayes o Redes Bayesianas.

Árboles de decisión. Esta técnica consiste en la construcción de un árbol en función de los valores de los distintos atributos que forman el problema a resolver. Los árboles van dividiendo el espacio de características de los datos de entrenamiento en rectángulos paralelos a los ejes como se ilustra en la Figura 3.7, siendo útiles tanto para clasificación como para regresión.

De esta forma, en los nodos interiores del árbol construido aparecen los atributos sobre los que se pregunta; en los arcos, los posibles valores que pueden tomar los atributos del nodo interior donde salen; y en las hojas del árbol se recoge el valor objetivo.

Regresión lineal. Este algoritmo trata de aproximar la distribución de los datos a una recta para calcular la variable de salida y. Para el caso en el que solo tengamos una

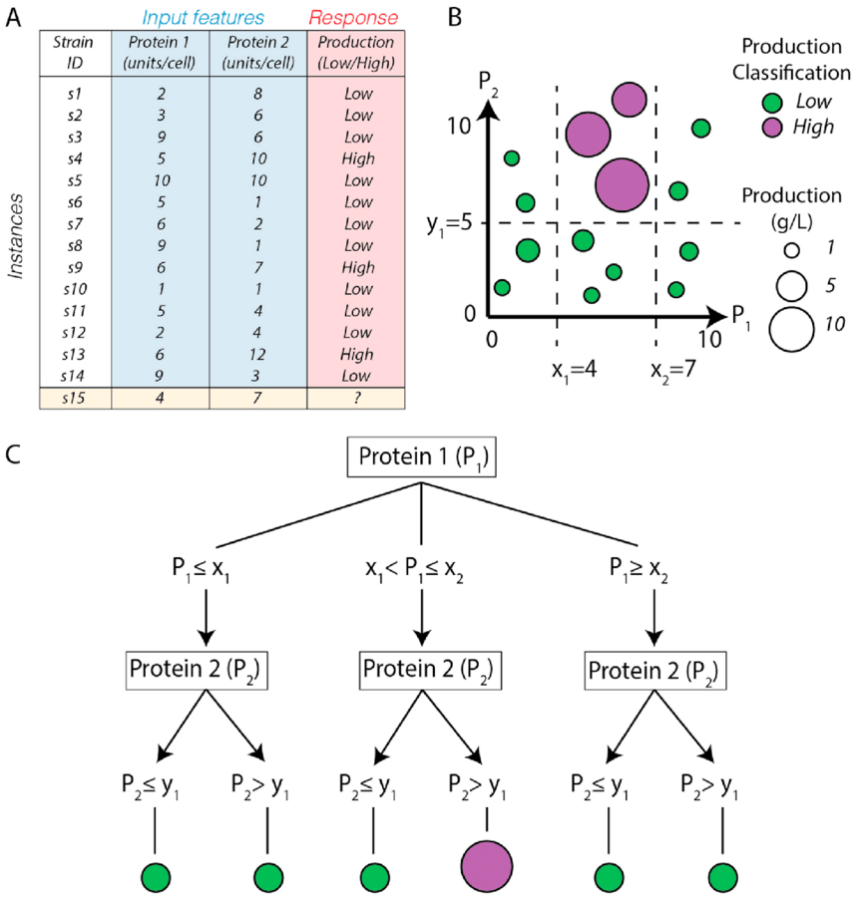


Figura 3.7: Representación de un árbol de decisión y su división del espacio de características [86].

variable de entrada x (regresión lineal simple), la recta tiene la forma $y = mx + b$, y el aprendizaje trata de encontrar los parámetros b y m que mejor se ajustan a los datos de entrenamiento.

Por otra parte, es usual que tengamos más de una variable de entrada. En este caso, diremos que el algoritmo es de regresión lineal múltiple. Suponiendo que tenemos las variables de entrada $x_1, x_2, \ldots, x_n$, y siguiendo el caso de la regresión lineal simple, se busca aproximar los datos con una función de la forma $y = \alpha_1 x_1 + \ldots \alpha_n x_n + b$. El proceso de aprendizaje tratará de determinar los valores b y α_i con $1 \leq i \leq n$.

Regresión polinómica. De igual forma que surgen los algoritmos de regresión lineal para ajustar los datos a rectas, se pueden usar otras funciones para modelar la relación entre los datos de entrada y salida. En caso de que la función sea un polinomio sobre las variables de entrada, diremos que la regresión es polinómica. No obstante, podemos caer en problemas de sobreajuste si escogemos polinomios de grado muy alto.

Regresión logística. La regresión logística es otra de las posibles variantes de regresión que se tienen en función de la aplicación elegida. Se utiliza para clasificación. Para llevar a cabo este algoritmo, se emplea una función logística; es decir, una sigmoide, que viene dada por la ecuación

$$f(x) = \frac{1}{1 + e^{-x}}, \tag{3.1}$$

cuya gráfica se representa en la Figura 3.8. De esta forma, a partir de la función sigmoidea, que se puede probar que su imagen es el intervalo $(0, 1)$, se determina la probabilidad de pertenecer a una u otra clase.

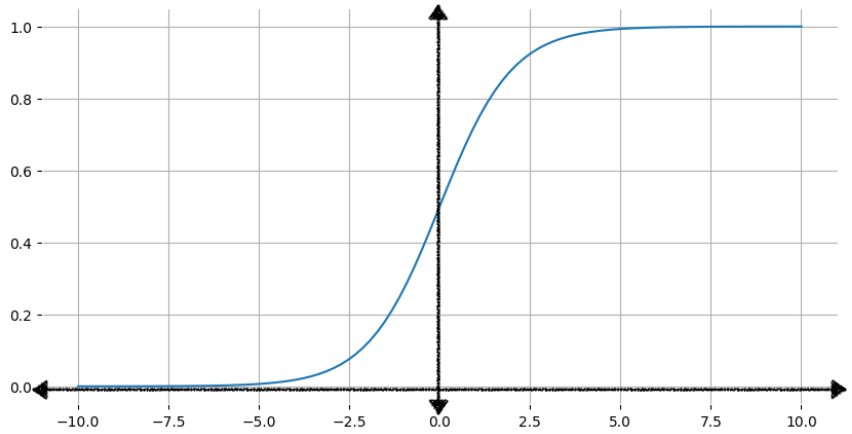


Figura 3.8: Gráfica de la función sigmoidea dada por la ecuación (3.1) [87].

SVM - Máquinas de Vectores Soporte (*Support Vector Machines*). Las SVMs tratan de encontrar la mejor separación entre clases. Consisten en encontrar una función que deje a cada lado elementos de clases diferentes. La solución óptima será la que maximice la anchura de la "calle" entre las clases como se representa en la Figura 3.9.

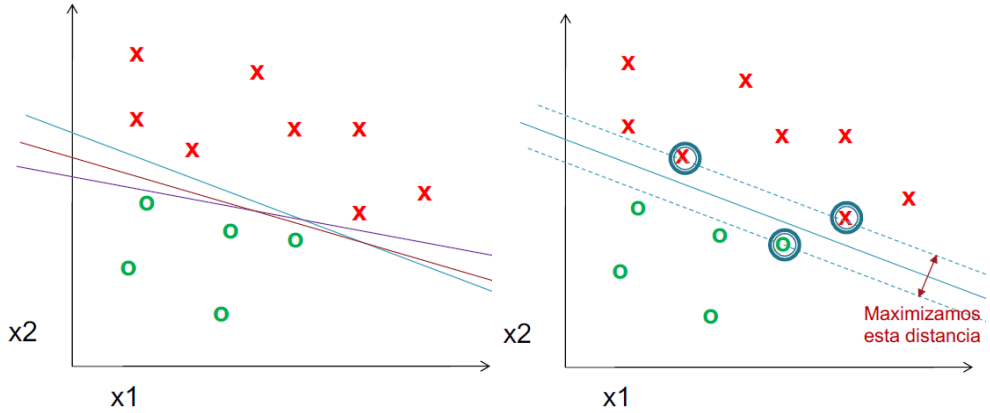


Figura 3.9: Elección de la solución óptica en un SVM de tipo lineal [88].

Existen distintos tipos de *Support Vector Machine* dependiendo de la función que separa las clases. Si es lineal, como la de la Figura 3.9, el algoritmo se denomina SVM lineal. En caso de que los datos no sean linealmente separables, se pueden llevar a dimensiones superiores en las que el conjunto de entrenamiento sea separable, surgiendo así subtipos del algoritmo SVM como el SVM polinómico o el SVM radial.

K-medias (*K-means*). Este es un algoritmo propio del clustering [25]. Los problemas de clustering, como ya se explicó en la Sección 3.2.1, buscan el particionado óptimo de los datos en N *clústers* independientes. Cada uno de estos *clústers* tiene asociado un centroide, que es el centro geométrico de la nube de datos que contiene. Los puntos se asignan al clúster cuyo centroide esté más próximo.

Ensembles

Un ensemble consiste en combinar múltiples modelos en uno solo para resolver un problema a partir de las salidas de los mismos [24]. Por lo general, mejoran la precisión y la robustez frente a la utilización de modelos individuales. El inconveniente que presentan es que computacionalmente son muy exigentes, aunque esto no impide que sea de las soluciones más utilizadas de *Machine Learning* hoy en día. Las principales aproximaciones son:

- **Votación**. Para cada nuevo dato, se pasa a todos los algoritmos que conforman el ensemble. De cada modelo se obtiene una salida y se escoge como resultado final la más votada; es decir, la que más veces aparezca.

- ***Bagging - Bootstrap AGGregatING***. Partiendo de un mismo algoritmo de *Machine Learning* se entrenan distintos modelos usando un subconjunto del conjunto de datos de entrenamiento. El subconjunto de entrenamiento se selecciona eligiendo muestras aleatorias con repetición (bootstrap). La salida de todos los modelos se combina entre sí para dar el resultado final. Este proceso se puede realizar por votación o cálculo de medias aritméticas entre otros. Se implementa con el Random Forest.

  Los ensembles de árboles de decisión mediante bagging se conocen como Random Forest. La gran ventaja de hacer ensembles únicamente con árboles de decisión es su eficiencia. Además permiten que se pueda paralelizar su ejecución (algo que con boosting no podemos hacer). Para problemas de clasificación los resultados se combinan con soft-voting. Para problemas de regresión se usa la media aritmética.

- ***Boosting***. Los algoritmos se entrenan de manera secuencial. El siguiente modelo se centra en los datos mal clasificados. Los datos de entrenamiento son distintos ya que cada algoritmo usa un subconjunto del total, pero no es aleatorio, se centra en los mal clasificados por el modelo anterior dando más peso a los mal clasificados. La precisión que obtiene Boosting suele ser muy buena, aunque es más lento que Bagging. Algunas implementaciones de este tipo de ensemble son AdaBoost, XGBoost o CatBoost.

- ***Stacking***. En esta forma de ensemble, se tienen una serie de modelos apilados. A partir de distintos modelos, todos entrenados con los mismos datos, se obtiene una salida de todos ellos, que se usa como entrada para otro modelo de ML. La salida de este último modelo de ML es el que toma la decisión final.

3.2.2. Redes Neuronales

Las redes neuronales [26] son un modelo computacional sobre el que se cimentan potentes algoritmos de aprendizaje automático. En concreto, en la Sección 3.2.3 se explica el uso de las redes neuronales en el campo del *Deep Learning*.

La unidad fundamental de toda red neuronal es la neurona artificial. Esta construcción recibe el nombre de neurona artificial puesto que se inspira en la estructura de una neurona biológica como la de la Figura 3.10. A continuación se explica su funcionamiento a través de la analogía con una neurona biológica:

- Cada neurona biológica consta de un conjunto de dendritas que reciben información; mientras que, para la neurona artificial, cada conexión tiene un valor de entrada y está caracterizada por un peso; es decir, un factor que modifica la entrada.
- La información que llega a la neurona a través de las dendritas se reúne en el cuerpo de la neurona, al igual que en la neurona artificial se tiene una función sumatoria que calcula la suma de todas las entradas.
- Al final de la neurona biológica se generan impulsos eléctricos. Por su parte, al final de la neurona artificial se contempla una función de activación que limita la amplitud de la salida de la neurona.

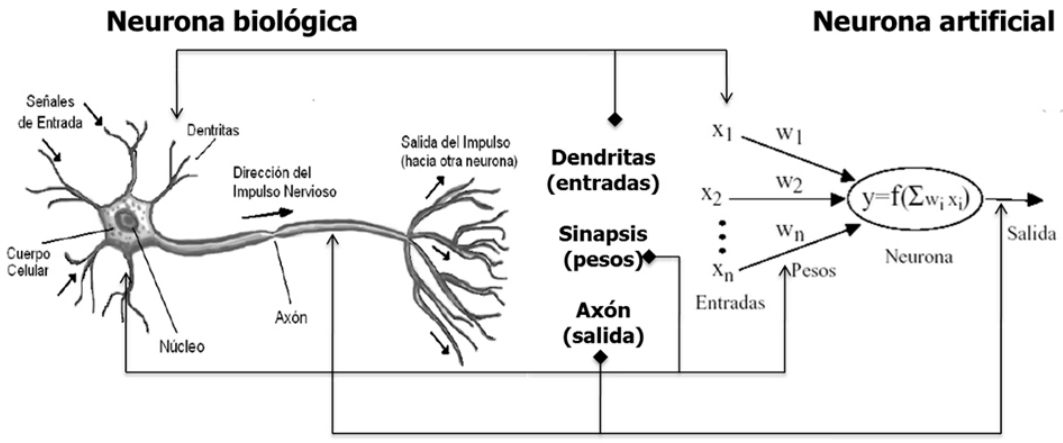


Figura 3.10: Comparación entre una neurona natural y una artificial [89].

El inicio de esta revolución parte del modelo de McCulloch y Pitts. Esta es la base sobre la que se comenzaron a construir redes neuronales y permitía simular comportamientos booleanos. Una neurona artificial típica consta de las siguientes componentes que vienen representadas en la Figura 3.11:

- **Vector de entrada** x. Conjunto de datos que se van a procesar en la neurona arificial. Lo constituye un vector $x = [x_1, x_2, \ldots, x_n]^T$ donde cada elemento es un dato de entrada.

- **Vector de pesos** w. Factor por el que se modifica cada uno de los datos de entrada. Significa una ponderación de cada elemento de entrada. A partir del vector $w = [w_1, w_2 \ldots, w_n]^T$, se multiplica el elemento x_i por el factor w_i.

- **Sesgo** b. Umbral de activación de la neurona. Representa la facilidad con la que se dispara una neurona.

- **Función de propagación**. Representa la suma de los elementos de entrada ponderadas junto con el sesgo.

- **Función de activación** f. Función encargada de transformar el resultado de la función de propagación para limitar la amplitud de sus valores. Típicamente, se utilizan funciones continuas monótonas crecientes, siendo la más sencilla la función identidad que no representa ningún cambio en los datos. Algunas de las más usuales se recogen en la Tabla 3.1.

- **Salida**. Dato final obtenido mediante las transformaciones oportunas de los datos del vector de entrada.

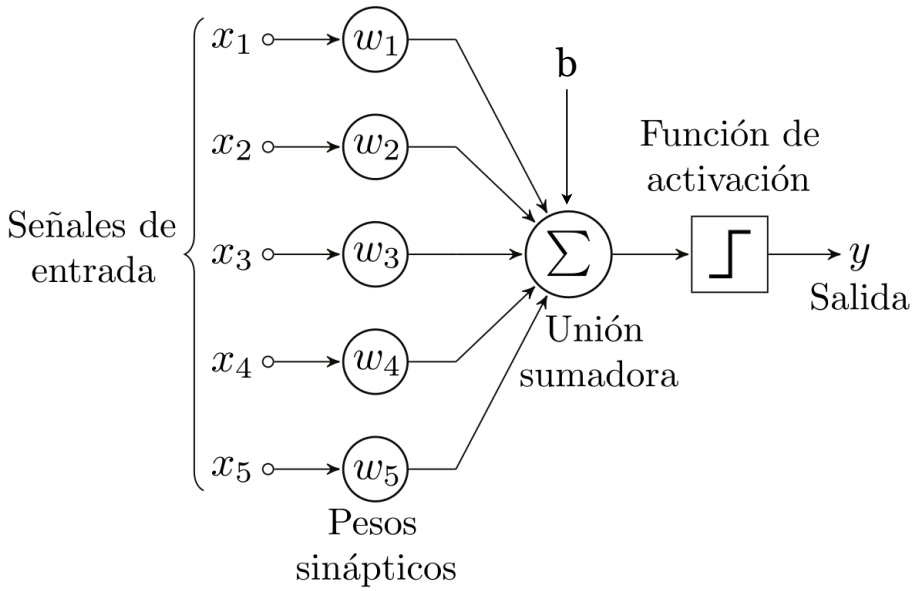


Figura 3.11: Modelo neurona artificial [90].

Luego, una red neuronal se forma a partir de un conjunto de estas neuronas artificiales. Para ello, se conectan unas a otras, estructurándolas en capas.

La necesidad de construir redes neuronales surge dado que usar una única neurona es una práctica muy limitada en cuanto a capacidad. De esta forma, una red neuronal posee una capa de entrada, una o varias capas ocultas y una capa de salida. Cada una de

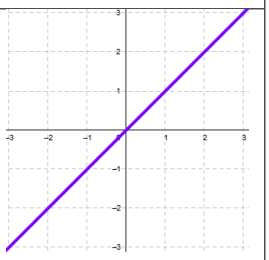
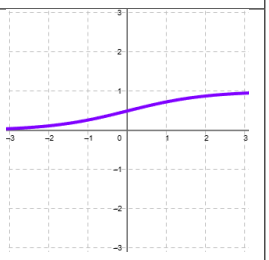
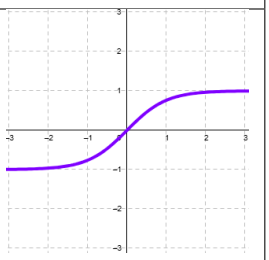
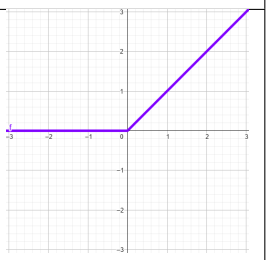
Nombre	Función	Gráfica
Lineal	$f(x) = x$	
Sigmoidea	$f(x) = \frac{1}{1+e^{-x}}$	
Tangente hiperbólica	$f(x) = \frac{e^{x}-e^{-x}}{e^{x}+e^{-x}}$	
ReLU	$f(x) = \text{máx}\{0, x\}$	

Tabla 3.1: Resumen funciones de activación comunes

estas capas tendrá, por lo general, un gran número de neuronas. Además, cada una de las mismas posee su propio vector de pesos y su sesgo.

Por otra parte, como se verá en la sección siguiente, surge el concepto de *Deep Learning*. El requisito para que una red neuronal se considere un modelo de Deep Learning reside en el número de capas ocultas que posea. Si la red posee más de una capa oculta, un mayor número de neuronas por capa o neuronas de diverso tipo, entonces el modelo generado es considerado un algoritmo dentro del ámbito del Deep Learning.

Por lo general, la capa de entrada está formada por tantas neuronas como atributos tengan los datos que se utilizarán como entrada para el modelo. Posteriormente, esta capa de entrada se comunica con una o más capas ocultas; es decir, aquellas que no son capas de entrada ni de salida. Por último, se tiene la capa de salida, que es la responsable de

producir el resultado final.

El proceso de aprendizaje consiste en ajustar los pesos y el sesgo de cada una de las neuronas que componen la red neuronal. De esta forma, se pretende abstraer la distribución de los datos para generar la salida deseada a partir de ajustes en los parámetros de la red. El proceso de entrenamiento de estas redes en el caso concreto del aprendizaje supervisado, que será el empleado a lo largo del proyecto, sigue los siguientes pasos:

- Se introduce el conjunto de datos de entrada en la capa de entrada de la red. La red procesa los datos y produce una salida para cada elemento.
- Se calcula una función de pérdida atendiendo al valor real y al valor obtenido por la red.
- Se emplean algoritmos mediante los que se pretende optimizar los parámetros de la red de forma que minimicen el error arrojado por la función de pérdida.

Este proceso se repite de manera iterativa para tratar de abstraer los datos. Además, cada iteración recibe el nombre de época.

3.2.3. Deep Learning

El *Deep Learning* [27] o aprendizaje profundo es el campo del *Machine Learning* que trata de extraer conocimiento a partir de un conjunto de datos mediante una jerarquía de múltiples capas de neuronas artificiales. Los datos de entrada pueden ser de tipo muy diverso, como imágenes, audios o texto. En cada capa, los datos de entrada se transformarán en representaciones más abstractas que se combinan a medida que se profundiza en la red.

La clave en las técnicas de *Deep Learning* es aportar una gran cantidad de datos con los que entrenar los modelos. Así, se pueden crear modelos flexibles capaces de abstraer las características de los conjuntos de datos de manera más eficiente de lo que lo hacen las técnicas clásicas de *Machine Learning* como las descritas en la Sección 3.2.1. Entre las aplicaciones del *Deep Learning* se encuentran la clasificación de imágenes, la detección de objetos, la segmentación o la generación de voz e imágenes. Las tres primeras se explotarán en la propuesta de solución en el Capítulo 4.

Instrumentos Redes Neuronales Profundas

- **Función de activación**. Este concepto ya ha sido introducido en la Sección 3.2.2. Pero existe una problemática, pues algunas de estas funciones presentan un problema para el entrenamiento de algoritmos basados en *Deep Learning*, como la función sigmoidea y la tangente hiperbólica. Como se observa en sus gráficas de la Tabla 3.1, el gradiente de la función varía. En el intervalo $[-2, 2]$ no existe ningún problema; sin embargo, fuera de él, el gradiente es muy pequeño y la función apenas varía. Este problema recibe el nombre de *vanishing gradient*.

El *vanishing gradient* ocasiona que la red no aprenda. Para solucionar esta casuística, surge la función ReLU, también tratada en la Tabla 3.1. Esta es la más utilizada para *Deep Learning* puesto que no presenta *vanishing gradient* para valores positivos. Además, cualquier función puede aproximarse como combinación de ReLUs y como no tiene límite superior, es mucho más rápida de entrenar.

- **Softmax**. Esta es una capa que se suele utilizar en la salida de la red. Si en la salida hubiera una capa normal, dada su construcción, el resultado sería un valor numérico para cada una de las clases posibles. Esto dificulta su interpretación. Para solucionarlo, se implementa una función softmax que transforma los valores numéricos en el rango $[0, 1]$, identificando la salida con la probabilidad de ocurrencia de cada una de las posibles predicciones.

- **Loss function o cost function**. La función de pérdida o *loss function* tiene como objetivo medir lo bien que modela la red los datos de entrenamiento, siendo una pieza fundamental en el entrenamiento de redes neuronales con *Deep Learning*. La función de pérdida mide la diferencia entre las estimaciones de la red y el resultado real. Luego, si la red no funciona correctamente, la *loss function* tendrá un valor alto. Por tanto, el entrenamiento consiste en minimizar el valor de la función de pérdida en cada etapa. Las principales funciones de pérdida son:

  - **Error cuadrático medio (*mean squared error*, MSE)**. Usado típicamente para predicción de escalares.
  - **Cosine similarity**. Se suele aplicar para problemas que tratan el procesamiento del lenguaje natural (NLP).
  - **Cross-entropy**. Mide la distancia entre dos distribuciones de probabilidad. Por tanto, sólo se puede aplicar sobre distribuciones de probabilidad; es decir, para datos en el rango $[0, 1]$. Por ejemplo, después de aplicar una capa softmax.

- **Optimizador**. Como se ha explicado, el objetivo a lo largo de los entrenamientos es el de minimizar la función de pérdida. Esta tarea la desarrolla el optimizador de manera iterativa. Para ello, el optimizador ajusta los pesos en cada etapa, buscando siempre hacer menor el valor de la función de pérdida. Existen numerosos algoritmos de optimización para crear redes, siendo el del descenso del gradiente el algoritmo básico.

Sobreaprendizaje

Al comienzo de esta sección se explicó que el propósito fundamental de las técnicas de *Deep Learning* es generalizar el conocimiento extraído de los datos de entrenamiento. El *overfitting* o sobreajuste [41] ocurre cuando un algoritmo se ajusta demasiado a los datos de entrenamiento y no puede generalizar los nuevos datos. En consecuencia, de todo el conjunto de datos de entrenamiento, se suele reservar una parte para validación; es decir, para comprobar el sobreajuste.

El problema del *overfitting* no es único del *Deep Learning*, sino que se da en el *Machine Learning* en general, aunque tiene especial relevancia para redes neuronales profundas debido a la gran cantidad de parámetros a entrenar. La regularización es un proceso que tiene como propósito evitar el sobreajuste. Luego, se encarga de realizar pequeños cambios sobre el algoritmo de aprendizaje para que generalice mejor. Los principales tipos de regularización son:

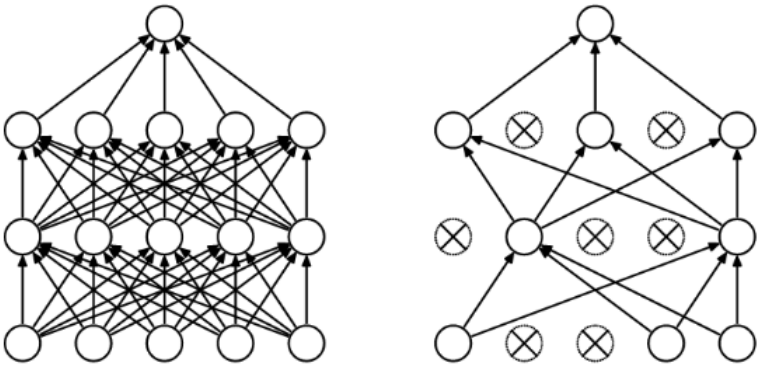
- **L2 y L1**. Agregan un valor de regularización a la función de pérdida consiguiendo que los pesos disminuyan en mayor medida. De esta forma, se consiguen modelos más sencillos, lo que ocasiona un menor sobreajuste. Siguiendo [42] y [43], el término de regularización viene dado por

$$\begin{aligned} &\textit{regularización } L_2 = |w_1|^2 + \cdots + |w_n|^2, \\ &\textit{regularización } L_1 = |w_1| + \cdots + |w_n|. \end{aligned} \tag{3.2}$$

  Además, ese término se escala con una tasa de regularización λ para ajustar el impacto que tiene en el modelo. Luego, el objetivo final es minimizar una función de coste de la forma

$$\textit{coste}\,(loss + \lambda * r),$$

  donde el término de regularización r es uno de los dados en la ecuación (3.2) en función del método utilizado L_2 o L_1.



(a) Standard Neural Net (b) After applying dropout.

Figura 3.12: Comparación entre una red neuronal antes y después de aplicar dropout [91].

- **Dropout**. Este término se refiere a la eliminación temporal de algunos nodos de la red neuronal [44]. Se pueden eliminar tanto nodos de las capas ocultas como de la entrada. Además, cuando se elimina una neurona también se desechan de manera temporal todas las conexiones con ella, creando así una nueva estructura de red como se ilustra en la Figura 3.12. Los nodos que se eliminarán se determinan con una probabilidad que viene dada como un hiperparámetro p.

- **Early Stopping**. Consiste en parar el entrenamiento antes de tiempo. Cuando el conjunto de validación empeora o no mejora tras un determinado número de épocas se detiene el entrenamiento.
- **Data Augmentation**. Esta técnica de regulación surge ante la posibilidad de un mayor sobreajuste cuanto menos datos tengamos para entrenar. Luego, trata de ampliar el conjunto de datos de entrenamiento. Por ejemplo, si se trata de imágenes, que es cuando más se utiliza, se aplican rotaciones o escalados para tener mayor variedad de imágenes.

Redes Neuronales Convolucionales - CNN

Una Red Neuronal Convolucional o CNN (*Convolucional Neural Network*) es un tipo de red neuronal con una serie de capas especializadas en el tratamiento de imágenes. Las CNN operan de manera jerárquica, de forma que en las primeras capas se identifican elementos básicos y estos se van combinando para formar objetos en las capas más profundas.

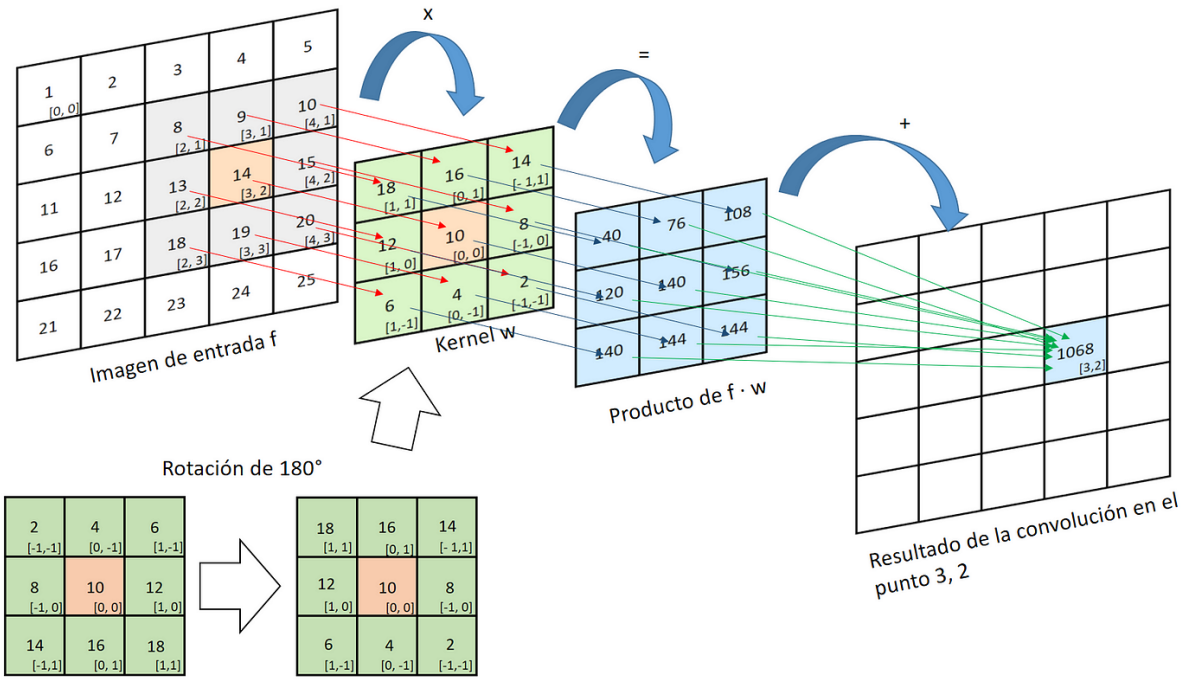


Figura 3.13: Ejemplificación del proceso de convolución [92].

Para tratar con imágenes en una red neuronal, estas se representan como una matriz con el valor de los píxeles. Si la imagen es a color (RGB), entonces se utilizan tres canales; es decir, tres matrices apiladas. Por otra parte, si la imagen es en escala de grises, tan solo es necesario un único canal. Además, es adecuado que los datos estén normalizados (basta dividir el valor del píxel por 255). A partir de este tratamiento de las imágenes se puede construir de manera sencilla una CNN integrando los siguientes componentes principales:

- **Capa de Convolución**: es la capa encargada de extraer las características de la imagen. La operación de convolución trata de aplicar un filtro o *kernel* sobre la imagen para construir un mapa de características. El *kernel* es una matriz, típicamente de tamaño 3×3 o 5×5, con valores numéricos que llamaremos pesos. La aplicación de convolución utiliza un *kernel* y consiste en multiplicar los píxeles de la imagen por los pesos definidos en el filtro y sumar el total como se representa en la Figura 3.13.

  La capa de convolución puede configurarse a partir de tres parámetros fundamentales:

  - **Profundidad**: número de filtros que se aplican sobre la imagen de entrada. Una mayor profundidad indicará un mayor número de filtros, luego se crearán más mapas de características.
  - ***Stride***: Número de píxeles que se desplaza el *kernel* sobre la imagen. Un mayor *stride* producirá mapas de características más pequeños.

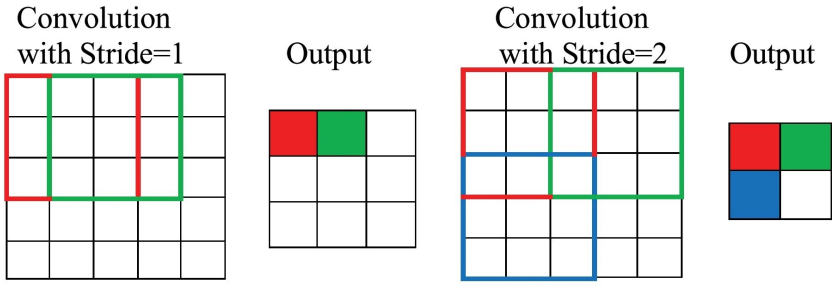


Figura 3.14: Ejemplificación del proceso de convolución - *stride* [93].

  - ***Padding***: se refiere a un marco de ceros que se añade alrededor de la imagen. El *padding* indica el número de filas y columnas que enmarcan la imagen con píxeles nulos. El objetivo que se busca con esta técnica es el de aplicar mejor los filtros sobre los elementos de los bordes.

- **Función de Activación (ReLU)**. Se aplica la función de activación ReLU para anular los valores negativos que se hayan obtenido al aplicar los filtros de la capa de convolución, manteniendo los valores positivos.

- **Capa de Pooling**: utilizada para reducir el tamaño de la matriz. Se trata de aplicar un filtro sobre la salida de la capa anterior y seleccionar un único número como salida. Como se verá a continuación, el siguiente componente de la CNN es una red neuronal multicapa normal. Luego, esta capa busca disminuir el tamaño de la matriz para que la red entrene a mayor velocidad. Los tipos principales de *pooling* son:

  - **Max Pooling**: se selecciona el valor máximo.

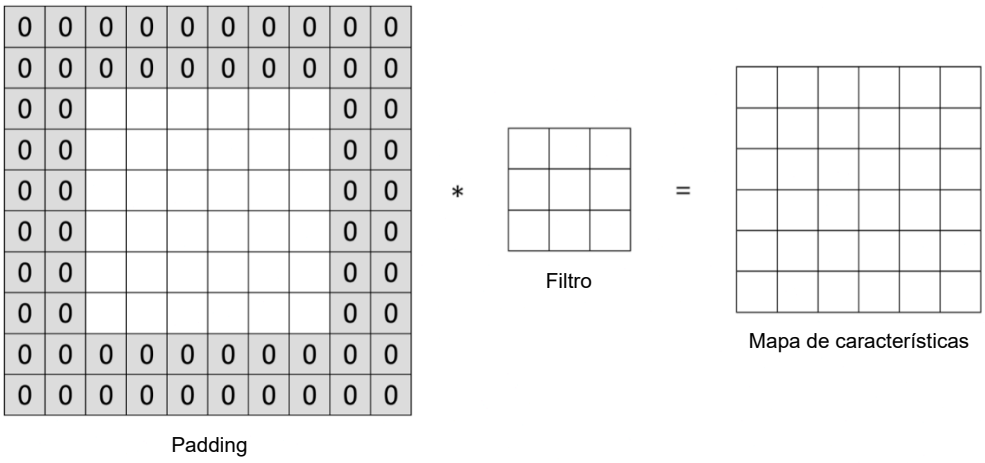


Figura 3.15: Ejemplificación del proceso de convolución - *padding*.

  - **Average Pooling**: se calcula el valor medio.
  - **Sum Pooling**: se escoge la suma de los valores.

- **Capa Fully Connected**. Tras iterar sobre las anteriores etapas, se pasa finalmente a la capa *fully connected*. Esta capa consiste en una red neuronal multicapa normal. El objetivo es combinar de manera eficiente todas las características extraídas en las capas anteriores, transformando la matriz de características que se genera en las convoluciones en un vector unidimensional. Para concluir, a este vector unidimensional se le aplica posteriormente Softmax para obtener la probabilidad de que la salida sea de una clase o de otra.

Objetivos principales de las Redes Neuronales Convolucionales

Gracias a la capacidad de las CNNs para extraer características y patrones dentro de las imágenes, estas se utilizan en una amplia variedad de tareas dentro del campo de la visión por computadora. Los objetivos que pueden abordarse con este tipo de redes neuronales varían según el tipo de salida deseada. De esta forma, las CNNs varian según la naturaleza del problema. Así, se pueden utilizar para resolver los siguientes problemas recogidos en [72]:

- **Clasificación**. El objetivo más básico. Consiste en tomar una imagen como dato de entrada y predecir a que clase pertenece dentro de unas preestablecidas. De esta forma, para el entrenamiento se tiene una única etiqueta asociada a cada imagen que indicará la clase a la que pertenece la misma.

- **Detección**. Además de identificar la clase del objeto esta técnica señala la ubicación del mismo mediante una caja que delimita la zona donde aparece. La detección permite no solo encontrar un objeto, sino varios dentro de la misma imagen.

La detección implica dos procesos claramente separadados. Por una parte, la red neuronal localiza un objeto. Por la otra, clasifica el objeto encontrado dentro de las clases que hay predefinidas. Así, en ocasiones se contempla la localización como un problema aparte que trata de buscar objetos de una misma clase, y la detección se explica como una combinación de las técnicas de localización y clasificación.

- **Segmentación**. La segmentación trata de señalar los píxeles que corresponden a un objeto. Luego, supone una mayor precisión que la detección. Mientras que los problemas de detección marcan la zona de aparición, la segmentación busca los píxeles exactos que corresponden al objeto buscado.

  Por otra parte, se pueden distinguir dos tipos de segmentación: semántica y por instancias. Para el primer caso, se clasifican directamente los píxeles según la clase a la que pertenecen. Para la segmentación por instancias, además de hacer esto mismo, se distingue si los píxeles pertenecen a un objeto u otro del mismo tipo.

Estos tres objetivos son los más usados comúnmente. De hecho, las técnicas usadas a lo largo de este proyecto se pueden enmarcar en cada uno de los mismos. Sin embargo, existen otras formas de usar redes neuronales convolucionales como la regresión, que consiste en una clasificación donde el propósito no es conocer un dato discreto como es la clase del objeto, sino obtener un dato continuo del mismo. Por ejemplo, predecir la edad de una persona a partir de una imagen sería una forma de regresión. Se puede considerar esto como una particularización de la clasificación.

Luego, las CNNs son muy útiles en el campo del *Deep Learning* y permiten abordar diferentes problemas cuando el dato de entrada es una imagen. De hecho, existen otras técnicas que emplean capas convolucionales en su arquitectura como la estimación de poses (*pose estimation*) o la generación de imágenes.

Amplicación sobre Redes Neuronales Convolucionales

En este trabajo se tratan, para una mayor completitud, tanto con modelos YOLO como con otros proporcionados por la biblioteca FastAI. Cada uno de ellos se entrena con una arquitectura distinta; mientras que YOLO tiene su propia arquitectura interna, los modelos de FastAI se utilizan a partir de la arquitectura U-Net. Por este motivo, a continuación se explica cada una de estas dos situaciones:

- **Arquitectura U-Net**. Esta arquitectura se enfoca en resolver problemas de segmentación de imágenes [64]. Es un tipo especial de red neuronal convolucional que tiene una estructura como la que se muestra en la Figura 3.16. Se puede dividir el proceso que se sigue en esta en dos partes bien diferenciadas: *encoder* y *decoder*; esto es, fases de codificación y decodificación o de contracción y expansión:

  - **Fase de codificación**. Durante la etapa de contracción se aplican de manera secuencial capas de convolución con ReLU como función y una capa de *pooling*,

en concreto, de *max-pooling*. De esta forma se va decreciendo la dimensión espacial del mapa de características mientras que se extrae la información relevante de la imagen.

- **Fase de decodificación**. Esta etapa tiene como objetivo adaptar el mapa de características a la imagen y producir un mapa de segmentación utilizando los patrones aprendidos en la fase de contracción [65].

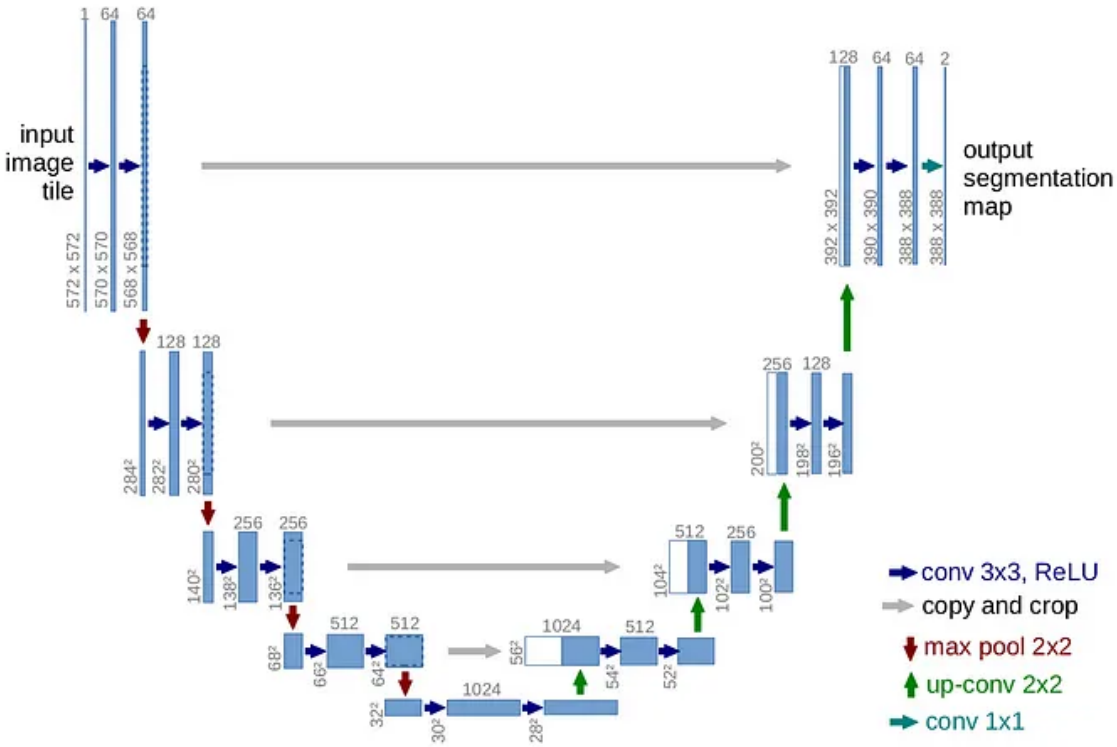


Figura 3.16: Diagrama representativo de la arquitectura U-Net [65].

- **YOLO**. Se basa en el paradigma "*You Only Look Once*". Es un modelo de detección de objetos y segmentación de imágenes en tiempo real [66]. Además, presenta un muy buen rendimiento en cuanto a velocidad se refiere.

  Respecto a su arquitectura, esta se divide en tres partes, las cuales se explican a continuación siguiendo [67] y [68]:

  - ***Backbone***. Como su traducción al español significa, esta parte es la columna vertebral de la arquitectura; es decir, la parte fundamental de la red. Su diseño consiste en la estructura proporcionada por la arquitectura *Darknet* [69], que no deja de ser una serie de capas convolucionales. Además, con cada versión de YOLO se actualiza y aumentan estas capas, haciendo la red más profunda y poderosa. Este incremento en la profundidad permite extraer características

más complejas de las imágenes, lo que repercute en un mejor rendimiento de la red.

- ***Neck***. Esta componente sirve de intermediaria entre las otras dos: *backbone* y *head*. Además, utiliza la arquitecturas de SPPF (*Spatial Pyramid Pooling - Fast*) [70], que es un conjunto de capas de *pooling*, y la de PANet *Path Aggregation Network* [71], que combina los mapas de características de las fases del *backbone*.
- ***Head***. Esta es la parte final, la cual se encarga de generar la salida de la red.

3.2.4. Métricas

Hasta ahora, a lo largo de las secciones 3.2.1, 3.2.2 y 3.2.3, se explican los algoritmos tanto de *Deep Learning* como de *Machine Learning* que se valorarán para la elaboración de la propuesta de solución, así como sus fundamentos. No obstante, se necesita una manera de cuantificar qué técnica funciona mejor en el problema a estudiar. Así es como surgen las métricas.

Una métrica de evaluación sirve para valorar el rendimiento de un modelo de aprendizaje automático y su capacidad para generalizar con precisión los datos [45]. Puesto que en la construcción de la solución se utilizan clasificadores y segmentación de imágenes, a continuación se presentan métricas utilizadas en el estudio de estas técnicas.

Métricas para clasificadores

Para evaluar el rendimiento de un clasificador se atiende a su matriz de confusión. Una matriz de confusión es cuadrada y tiene la dimensión del número de clases existentes. A continuación vamos a definir una matriz binaria; es decir, de dos dimensiones, pues son la base para el resto de matrices de confusión y serán las utilizadas en la propuesta construida.

Una matriz de confusión binaria, como la que se muestra en la Tabla 3.2, es una representación matricial de los resultados de las predicciones de cualquier prueba binaria que se utiliza para describir el rendimiento del modelo de clasificación [45]. Cada predicción puede ser de uno y solo uno de los cuatro tipos siguientes:

- **Verdadero Positivo (VP)**: El modelo predice el dato como verdadero y ciertamente es verdaero.
- **Verdadero Negativo (VN)**: se predice el dato como negativo y su valor real es negativo.
- **Falso Positivo (FP)**: el valor predicho es verdadero, frente al real, que es falso.
- **Falso Negativo (FN)**: la predicción establece que es falso pero el valor real es verdadero

	Clasificados positivos	Clasificados negativos
Ejemplos positivos	Verdadero Positivo (VP)	Falso Negativo (FN)
Ejemplos negativos	Falsos Positivos (FP)	Verdadero Negativo (VN)

Tabla 3.2: Representación Matriz de Confusión

A cada uno de estos tipos le corresponderá un lugar en la matriz de confusión 2×2 como se muestra en la Tabla 3.2.

Como se aprecia en la Figura 3.2 y como se explicó anteriormente, existen dos tipos de errores. Estos son los falsos positivos y falsos negativos. En función del objetivo que tenga el clasificador y del área de negocio para el que se construye, un error será más importante que otro. Por este motivo, surgen distintas medidas de evaluación en función de los errores que se consideren. Siguiendo la webgrafía existente [46], las métricas más importantes y usuales son:

- ***Accuracy* o Exactitud**: mide la proporción de predicciones correctas. Su expresión viene dada por

$$accuracy = \frac{VP + VN}{VP + FP + VN + FN} = \frac{\text{clasificados correctamente}}{\text{todos los ejemplos}}.$$

  Esta métrica tendrá un valor entre 0 y 1, siendo preferible un valor alto; puesto que indicará mayor proporción de predicciones correctas.

- ***False positive rate* o tasa de falsos positivos (FPR)**: proporción de todos los negativos reales que se clasificaron incorrectamente como positivos. Se define de manera matemática de la siguiente manera:

$$fpr = \frac{FP}{FP + VN} = \frac{\text{clasificados erróneamente como positivos}}{\text{todos los negativos}}.$$

  Esta métrica también tomará valores entre 0 y 1; sin embargo, al contrario de lo que ocurre con el *accuracy*, será preferible un valor próximo al 0. Esto es, cuanto menor sea la tasa de falsos positivos, mejor será el rendimiento del algoritmo entrenado.

- **Precisión**: proporción de todas las clasificaciones positivas del modelo que realmente son positivas. La precisión se expresa como

$$precision = \frac{VP}{VP + FP} = \frac{\text{clasificados correctamente como positivos}}{\text{todos los clasificados como positivos}}.$$

  Una vez más, puesto que el numerador es mayor que el denominador, la precisión será un valor entre 0 y 1. Además, cuanto mayor sea este valor, mejor será el modelo

entrenado. Esto se debe a que una mayor precisión indica que más valores predichos como positivos son realmente positivos.

- ***Recall* o tasa de verdaderos positivos (TPR)**: es la proporción de todos los positivos reales que se clasifican correctamente como positivos; es decir, se expresa como

$$recall = \frac{VP}{VP + FN} = \frac{\text{clasificados correctamente como positivos}}{\text{todos los positivos}}.$$

  El *recall* en ocasiones también se denomina probabilidad de detección. De hecho, esta métrica es muy útil en aplicaciones médicas, pues como su nombre indica, identifica la probabilidad de detectar una patología cuando un paciente la presenta. De igual forma que con el resto de métricas comentadas, como el valor del numerador es menor que el del denominador, el *recall* está entre un rango de 0 y 1.

- ***F1 Score***: promedio calculado como la media armónica entre la precisión y el *recall*; es decir,

$$F1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall}.$$

  Esta métrica toma valores entre 0 y 1, siendo preferibles valores cercanos a 1. Además, como se explica en [46], esta es una métrica muy utilizada en problemas en los que el conjunto de datos a analizar está desbalanceado.

  Esta métrica combina el precision y el *recall*, para obtener un valor mucho más objetivo. En el caso en el que estas dos métricas sean similares, F1 también tendrá un valor parecido. Por otra parte, si la precisión y el *recall* están muy separadas, entonces F1 será similar a la métrica que sea peor.

- ***F-β Score***. Generalización de la métrica *F1-Score*. Viene dada por

$$F_\beta = \frac{(1 + \beta^2) \cdot Precision \cdot Recall}{\beta^2 \cdot Precision + Recall}.$$

  A partir de esta métrica, introduciendo el parámetro β, se puede dar más peso a la *precision* o al *recall* según se desee. Así, para $\beta > 1$, se da mayor peso al *recall*; mientras que para $\beta < 1$, la *precision* cobra más importancia. La elección de β mayor o menor que 1 dependerá de la importancia que se le dé a detectar todos los positivos y a no tener falsos positivos, respectivamente.

  En el uso de aplicaciones médicas, donde lo importante es detectar todos los casos positivos de una enfermedad, cobrará mayor importancia el *recall* con el objetivo de minimizar el número de pacientes que presentan la patología en los que no ha sido detectada. En caso de que no se quiera favorecer o destacar una métrica por encima de la otra, se utilizará $\beta = 1$, coincidiendo la métrica F_β con $F1$, siendo esta última un caso particular de F_β.

- **Curva ROC**. Representación gráfica del rendimiento de un modelo de clasificación. Esta curva relaciona la métrica *recall* (TPR) en el eje Y con la tasa de falsos positivos (FPR) en el eje X. Como el objetivo es tener un TPR alto y un FPR bajo, cuanto mayor sea la pendiente de la curva para que se acerque con valores pequeños de FPR a valores altos de TPR, mejor será el entrenamiento. Esto es justo lo que muestra la Figura 3.17.

  Es posible establecer una medida objetivo de lo buena que es la curva ROC. Cuanto mayor pendiente tenga la curva, mayor área abarcará. Como lo que se pretende es que la curva crezca rápido, el objetivo es que el área sea mayor. A la métrica que mide el área bajo la curva ROC se le denomina AUC (área bajo la curva - *area under curve*). Por otra parte, como la curva está representada en $[0, 1] \times [0, 1]$, el área máxima y por tanto el valor de AUC está limitado entre 0 y 1.

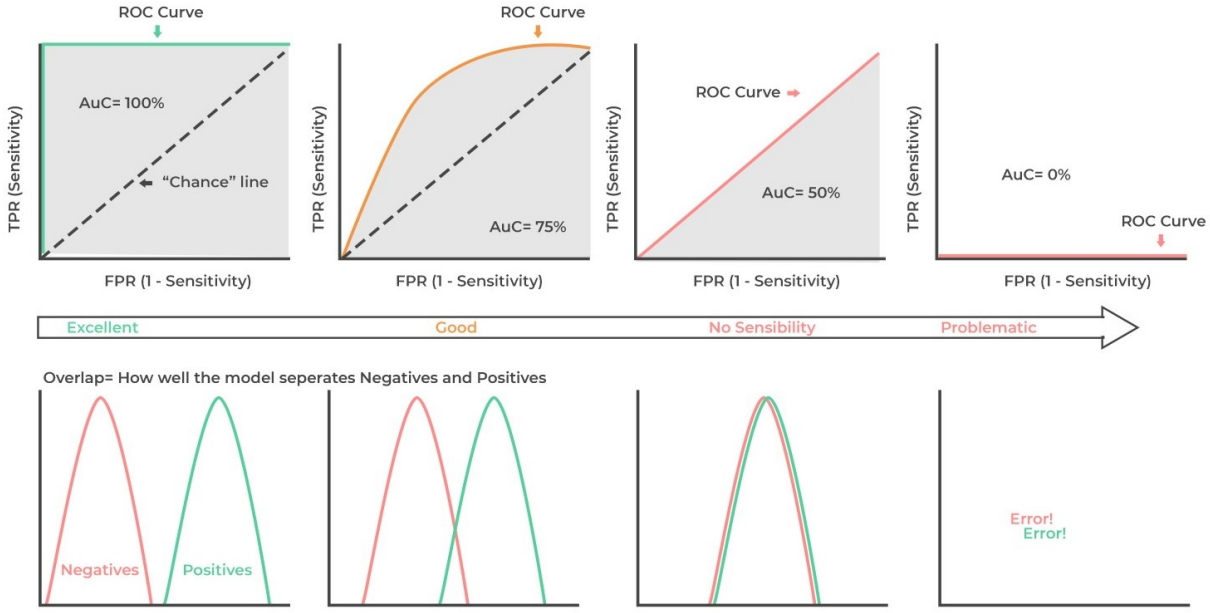


Figura 3.17: Relación curvas ROC con la separabilidad de los datos [94].

Métricas para localización y segmentación

La salida para clasificación de dos clases será un valor binario en función de si el valor predicho es uno u otro. Para este tipo de salidas se han definido unas métricas. Sin embargo, para problemas de segmentación la salida no es la clase predicha, sino una máscara que indica dónde se sitúa el objeto buscado. Por tanto, como la salida es diferente, no podemos usar las mismas métricas y será necesario definir otras nuevas; o al menos, adaptarlas para el caso de uso de la segmentación. A continuación se define las principales métricas utilizadas en problemas de segmentación:

- **Coeficiente Dice**: se emplea para medir la similitud entre dos conjuntos. Si A es la máscara real y B es la máscara predicha, el coeficiente Dice se define como

$$Dice = \frac{2|A \cap B|}{|A| + |B|},$$

donde $|C|$ representa el número de píxeles del conjunto C. Este coeficiente toma valores entre 0 y 1. Además, un valor mayor indica mejor coincidencia entre la segmentación predicha y la real.

- **Índice de Jaccard**: evalúa la superposición entre la predicción y el valor real de la máscara. Esta métrica, también denominada *Intersection over Union* (IoU) se calcula como sigue:

$$IoU = \frac{|A \cap B|}{|A \cup B|}.$$

El cálculo permite conocer la proporción de la segmentación o localización real que forma parte de la predicción.

- **mAP**. Métrica que combina la precisión y el *recall* entre diferentes clases para comprobar el rendimiento del modelo construido. Se usa para la detección de objetos. La métrica tomará valores entre 0 y 1. Cuanto más cercana sea la métrica a 1, mejor será el rendimiento. Para calcular el valor mAP, *mean Average Precision*, se siguen los siguientes pasos descritos en [48]:

  1. Se calcula la precisión y el *recall* para cada clase.
  2. Se construye la curva precisión-*recall* para cada clase (véase la referencia [47]).
  3. Se calcula el valor AP, que se corresponde con el área bajo la curva construida en el paso anterior.
  4. Una vez se tiene el valor AP calculado para cada clase, *mean Average Precision* se obtiene a partir de la media de estos valores; es decir, si se tienen N clases y AP_i es el área bajo la curva precisión-*recall* de la clase i,

$$mAP = \frac{1}{N} \sum_{i=1}^{n} AP_i.$$

- ***Accuracy Camvid***. Métrica proveniente de un dataset de segmentación semántica que recibe el mismo nombre. Establece la proporción de píxeles clasificados correctamente una vez descartados los píxeles del fondo; esto es,

$$Accuracy\ Camvid = \frac{\text{píxeles clasificados correctamente}}{\text{total píxeles}}.$$

3.2.5. Procesamiento y representación de imágenes

En muchos casos en los que el objetivo es desarrollar algoritmos de *Deep Learning* sobre imágenes, la manipulación de las mismas permite obtener mejores resultados durante el proceso. De esta forma, surge la necesidad de entender cómo se representan las imágenes computacionalmente y los distintos tipos de procesamiento que se pueden realizar sobre las mismas.

Entre el procesamiento que se puede realizar para obtener variaciones de interés sobre las imágenes, se encuentra tomar solo la parte de la imagen con mayor cantidad de información o la transformación a través de la aplicación de filtros.

Representación de imágenes

Las imágenes digitales se representan como matrices de píxeles. Dada una imagen, esta se divide en forma de cuadrícula, de manera que cada cuadro representa un píxel. Así, cada elemento de la matriz representa el valor de cada píxel en esa posición de la cuadrícula en la que se divide la imagen. No obstante, no es suficiente con un solo valor, pues no se puede representar de esta forma todos los colores. Para ello, se toma una base de colores donde se puede obtener el resto de colores a partir de una combinación de los de la base. Para cada elemento de la base, se toma una capa; es decir, una matriz, y se almacenan los valores entre 0 y 255 para ese color de cada píxel. Finalmente, se puede generar la imagen teniendo en cuenta todas las capas de la base. A continuación, se van a comentar diferentes estilos o formatos de una imagen en cuanto a color se refiere.

RGB. RGB es la base más empleada para representar una imagen. Almacena tres capas de colores o canales, rojo (R), verde (G) y azul (B), que combinados forman el color final de cada píxel. Cada canal puede verse como una matriz bidimensional donde cada valor representa la intensidad de ese color.

Una imagen en formato RGB contiene tres matrices superpuestas, una por cada componente de color. No obstante, se puede reducir la imagen a un solo canal si los modelos no requieren información de color, como se verá a continuación.

Escala de grises. Convertir una imagen a escala de grises implica reducir los tres canales RGB a uno solo. Una forma de obtener un resultado en escala de grises a partir de la base RGB es calcular la media de los tres valores. Sin embargo, esta solución no es acorde con la realidad debido a que cada canal no aporta lo mismo. Para ello, existen distintas recomendaciones de qué valores usar para ponderar cada canal [49]. Algunas de estas opciones se recogen en la Tabla 3.3.

Nombre ponderación	Factor rojo	Factor verde	Factor azul
CIE 1931	0.2126	0.7152	0.0722
rec601	0.299	0.587	0.114
ITU-R BT.2100	0.2627	0.6780	0.593

Tabla 3.3: Métodos de conversión de RGB a escala de grises con sus respectivas ponderaciones.

Imágenes binarias. Este tipo de imágenes tiene un solo canal al igual que las imágenes en escala de grises. Sin embargo, son más simples. Como su nombre indica, una imagen binaria tan solo toma dos valores, que serán el 0 y 255. Luego, se puede considerar como una particularización de las imágenes en escala de grises donde solo se contempla el blanco y el negro. Su uso para el desarrollo de técnicas de *Deep Learning* está muy extendido. Típicamente, se utilizan para localizar con uno de los colores la región de interés de otra imagen a la que acompañan; mientras que la parte que presenta el otro color no tiene relevancia.

Filtro de Gabor

Los filtros de Gabor son filtros lineales cuya respuesta es una función sinusoidal multiplicada por una función gaussiana [50]. A colación de lo anterior, se pueden construir una infinidad de este tipo de filtros dado que cabe la posibilidad de modificar parámetros como la amplitud de la función sinusoidal o la orientación de la misma. En la Figura 3.18 se muestra una variedad de filtros de Gabor, así como su resultado sobre una imagen.

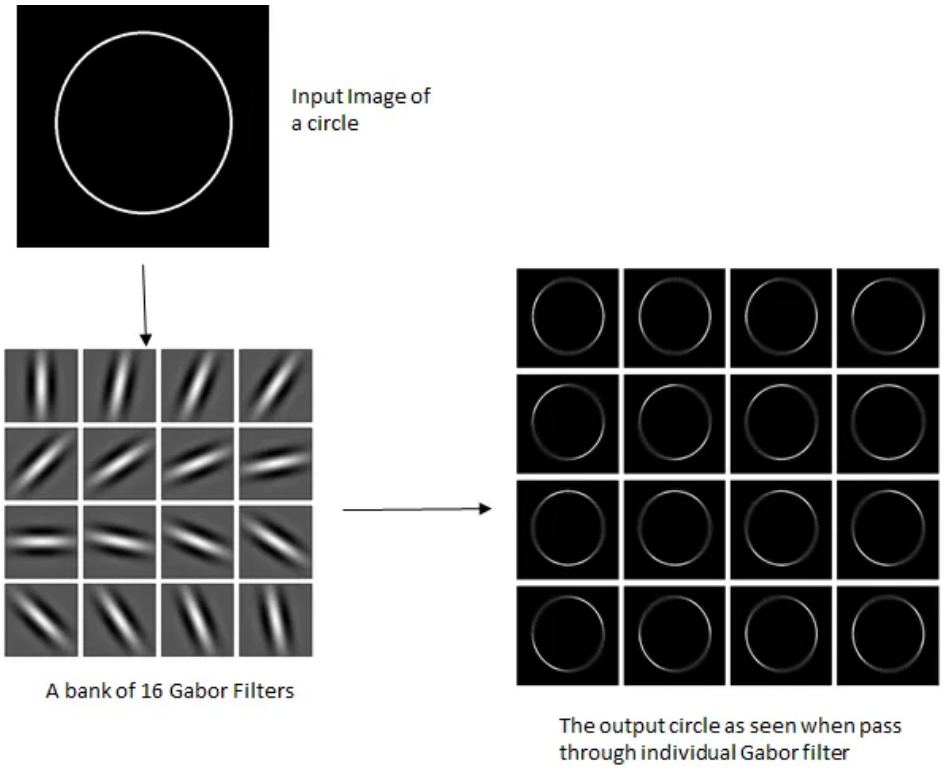


Figura 3.18: Resultados obtenidos a partir de filtros de Gabor sobre una circunferencia [95].

Los filtros de Gabor son de especial utilidad para remarcar estructuras dentro de las imágenes. Eso se debe a que pueden resaltar, por ejemplo, contornos. En la Figura 3.19 se muestra un ejemplo de los resultados que se pueden obtener a través de filtros de Gabor.

Figura 3.19: Resultado obtenido tras la aplicación de un filtro de Gabor [95].

3.3. Conceptos matemáticos

Motivación

El objetivo que se persigue con la introducción de conceptos matemáticos en el desarrollo de la propuesta será el perfeccionamiento de las detecciones. En algunos casos, en la detección del objeto pueden aparecer ciertas vesículas que no representan al mismo. También sucede que, queriendo buscar un objeto único en la imagen, se detecta un mayor número de elementos. Así, para solucionar estos problemas puede ser útil la aplicación de algunos métodos matemáticos.

Cada proceso de segmentación produce una máscara binaria indicando dónde se encuentra la estructura buscada. Esta máscara consiste en una matriz de unos y ceros donde el 1 indica que ese píxel forma parte del objeto buscado y el 0 implica que el píxel no forma parte del objeto. Así, podemos suponer el conjunto de píxeles como el conjunto total V, y la parte de píxeles que se identifican con unos en la máscara como un subconjunto S. De esta forma, surgen una serie de conceptos para tratar y perfeccionar las máscaras predichas.

Envolvente convexa

Previo a conocer qué es una envolvente convexa, se introduce el término de conjunto convexo[56]. Se dice que el subconjunto S es convexo si para cualquier par de puntos de S, el segmento que los une está contenido en S; es decir, para cada $x, y \in S$,

$$\lambda x + (1 - \lambda)y \in S \quad \forall \lambda \in [0, 1].$$

Conociendo esta definición se puede introducir el concepto de envolvente convexa [56]. La envolvente convexa de un conjunto S en un espacio V es el conjunto convexo más pequeño que contiene a S. Formalmente, la envolvente convexa de S, denotada como conv(S), es el conjunto de todas las combinaciones convexas de puntos en S; es decir,

$$\text{conv}(S) = \left\{ \sum_{i=1}^{n} \lambda_i x_i \mid x_i \in S, \lambda_i \geq 0, \sum_{i=1}^{n} \lambda_i = 1, n \in \mathbb{N} \right\}$$

En la Figura 3.20 se ejemplifica cómo se obtiene la componente convexa de una máscara.

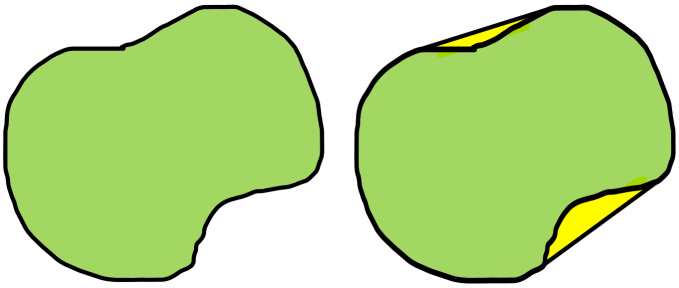


Figura 3.20: Diagrama de ejemplo del cálculo de la envolvente convexa.

Diámetro de un conjunto

Llamaremos diámetro de un conjunto S a la mayor distancia entre dos puntos de ese mismo conjunto. Formalmente, si d es una distancia (véase [56]) se define el diámetro de S [56] como

$$diametro(S) = \sup\{d(x, y) \mid x, y \in S\}.$$

Considerando la máscara obtenida en una predicción como el conjunto de los píxeles cuyo valor asociado en la máscara es 1, el cálculo de la distancia de ese conjunto permitirá extraer características sobre el objeto detectado.

Componente conexa

Para introducir formalmente lo que es una componente conexa [56] se necesitan conocer muchos conocimientos previos fuera del alcance de este proyecto. Por ello, en este apartado se busca dar una idea de lo que este concepto significa y cómo se ha usado para tratar de mejorar los resultados obtenidos por los modelos de segmentación entrenados.

Para entenderlo de manera sencilla, se dice que las componentes conexas son las partes de un conjunto que están completamente unidas. Se establece ahora una analogía para aclarar este concepto siguiendo la representación de la Figura 3.21.

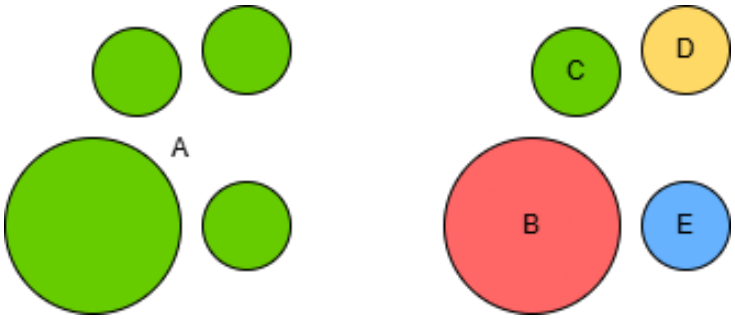


Figura 3.21: Diagrama separación en componentes conexas de un conjunto.

Si se tiene un archipiélago de islas, que sería nuestro conjunto, cada isla representa una

región conectada independiente, lo que es comparable con cada una de las componentes conexas de un conjunto. Cada isla sería una componente conexa porque dentro de ella todo está conectado, pero no hay conexión entre diferentes islas.

3.4. Estado del arte

El diagnóstico del glaucoma es una necesidad creciente en el ámbito de la oftalmología, dado que tiene un pronóstico más favorable si se diagnostica en etapas tempranas, como ya se ha explicado en la Sección 1.1. Sin embargo, en la actualidad siguen sin utilizarse herramientas automatizadas que asistan en la detección de esta patología, lo que representa una oportunidad para el desarrollo de modelos basados en aprendizaje profundo, capaces de identificar patrones a partir de retinografías que ayuden con el diagnóstico del glaucoma.

Como se ha planteado en la Sección 1.2, el objetivo principal es tratar las imágenes tomadas sobre el fondo del ojo para detectar el glaucoma mediante técnicas de aprendizaje profundo. En lo siguiente, se verán soluciones similares para este tipo de problemas en el campo de la medicina, para centrarse a continuación en las aplicaciones existentes para el diagnóstico de otras enfermedades oculares, así como del glaucoma.

Avances en el área médica

Además de la oftalmología, las redes neuronales han sido aplicadas exitosamente para la construcción de diversas herramientas en el área médica. Como se recoge en la revisión de [6], en el año 2017 ya se habían realizado más de 300 contribuciones en el análisis de imágenes médicas mediante *Deep Learning*. Además, en este mismo artículo se repasan soluciones diversas que atañen a las zonas pulmonares, cardíaca, abdominal o cerebral. Así, el uso de técnicas de aprendizaje profundo representa una herramienta real de apoyo para los especialistas de la salud.

Concretando con algunos casos de éxito en el uso de estas técnicas, se encuentra la detección del cáncer de piel, que también se trata en [6]. Para este tipo de problemáticas se han desarrollado soluciones como las que se presentan en [52] y [53]. Para este segundo artículo, se ha alcanzado un *accuracy* del 84.4 % y una sensibilidad del 92.8 % para algunos de los modelos contemplados. Estas métricas indican la posibilidad de crear modelos de aprendizaje profundo con un buen rendimiento en el campo médico.

Otro caso de estudio interesante se trata en [51], donde se hace uso de redes neuronales para detectar la presencia de tumores cerebrales y poder realizar una clasificación de los mismos. En este caso se ha logrado un *accuracy* de 96.7 % y 88.25 % en los datos de validación y test, respectivamente. Estos avances subrayan el potencial del aprendizaje profundo para diagnosticar patologías, lo que convierte a la oftalmología en un área potencial de desarrollo de herramientas similares.

Avances previos en el área oftalmológica

Como se ha introducido previamente con ejemplos de otras áreas médicas, el uso de técnicas basadas en aprendizaje profundo ha mostrado un rendimiento sobresaliente en la interpretación de imágenes médicas. En particular, las redes neuronales convolucionales han demostrado su capacidad para identificar características en las imágenes médicas y realizar predicciones con precisión. Estas tecnologías ofrecen una promesa significativa para mejorar los métodos no invasivos y complementarios de diagnóstico en oftalmología.

Los sistemas de aprendizaje profundo han logrado resultados destacados en la detección de patologías oculares, como se evidencia en algunos trabajos como los recogidos en [8], donde se revisitan y resumen nuevos sistemas de *Deep Learning* en aplicaciones oftalmológicas. Por otra parte, existen diversos ejemplos de aplicación, como ocurre en [7], donde se prueba que una CNN entrenada para la detección de retinopatía diabética puede obtener resultados comparables a los de especialistas. Estos trabajos han establecido una base sólida para el uso de CNN en imágenes oftalmológicas. Sin embargo, la detección de glaucomas oculares sigue siendo un área de investigación en desarrollo.

Avances previos en la detección del glaucoma

Recientemente, el periódico Huffington Post [57] publicaba un artículo que hacía referencia a otro artículo de prensa publicado por la Universidad de Tohoku [58], en Japón. En este, se hace eco de una investigación [59] para la detección del glaucoma, en la que se obtuvo un método con un 93.52 % de sensibilidad y un 95 % de especificidad. El proceso que se sigue para la detección del glaucoma según esta solución se representa en la Figura 3.22, que comparte ciertas analogías con la solución que se propondrá más adelante.

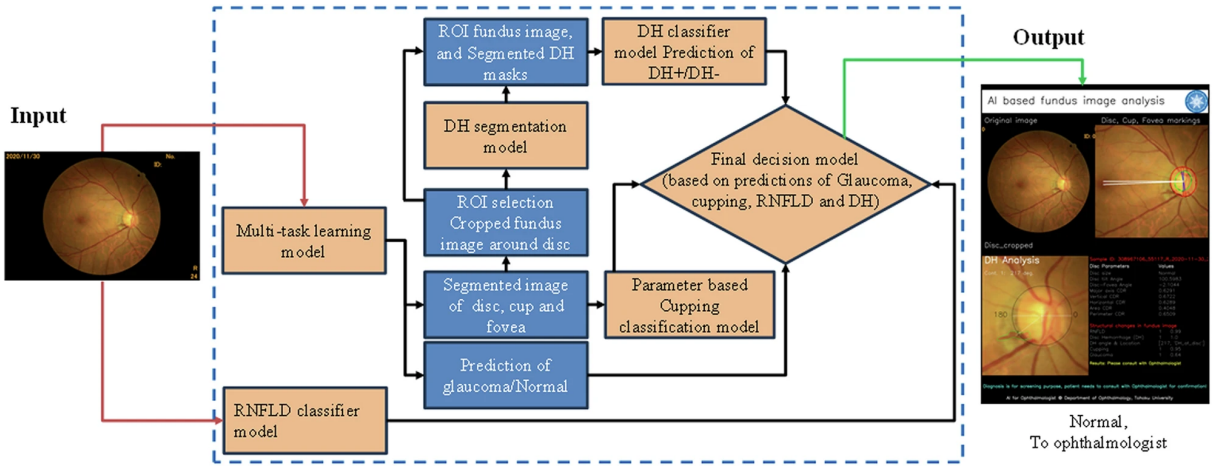


Figura 3.22: Diagrama de flujo para la detección del glaucoma según la Universidad de Tohoku [59].

Por otra parte, en plataformas como Kaggle, se encuentran diversos trabajos y conjuntos de datos en los que se emplea *Deep Learning* para clasificación de imágenes, separando conjuntos de retinografías según representan un ojo sano o con glaucoma. En particular,

[9] es uno de estos *datasets*. Una segunda vía de estudio en estos proyectos es la segmentación de imágenes para delimitar dos áreas fundamentales para la detección del glaucoma, como son la copa y el disco ópticos. En este segundo marco se tienen los conjuntos de datos [11] y [13] Sin embargo, analizando estos trabajos resaltan algunos problemas. Para el primer enfoque, el de clasificación de imágenes, surgen las siguientes dificultades:

- **Falta de interpretabilidad**. Para la clasificación con *Deep Learning*, no se indica qué regiones de la imagen contribuyen al diagnóstico al trabajar como una "caja negra".
- **Dependencia de los datos**. Requiere una gran cantidad de imágenes bien etiquetadas, lo cual puede ser un desafío en medicina, dado que, en ocasiones, es complicada incluso la obtención de *datasets* públicos como los de la Sección 4.2 por motivos de protección de datos.
- **Poca sensibilidad a estructuras específicas**. Puede ignorar detalles importantes, como el tamaño de la copa óptica, que son cruciales para la detección del glaucoma.

Por su parte, el enfoque basado en segmentación de las retinografías representa principalmente un problema al no tener continuidad en el proceso de diagnóstico; es decir, los modelos segmentan las imágenes y esto no se utiliza con ningún propósito. De este modo, surge una oportunidad de elaborar una herramienta que combine ambos enfoques, tomando la mejor parte de cada uno, mejorando las soluciones existentes. Además, para el *dataset* contenido en [9], que como se verá a lo largo del Capítulo 4 será sobre el que se desarrollen los modelos de clasificación, se han encontrado otras soluciones, de entre las cuales los mejores resultados se muestran en la Tabla 3.4.

Modelo	Accuracy	Precision	Recall	Conjunto de datos
[78]	0.9416	0.9315	0.9532	Test
[79]	0.9391	0.8756	0.8791	Validación
[79]	0.9313	0.8518	0.9028	Test
[80]	0.9376	0.9079	0.9740	Test
[80]	0.9493	0.9458	0.9532	Test
[81]	0.8889	0.8662	0.92	Test

Tabla 3.4: Modelos estado del arte del *dataset* contenido en [9].

Parte II

Desarrollo de las propuestas y resultados

Capítulo 4

Desarrollo de la propuesta y resultados

En este capítulo se explican todas las técnicas que se han probado para la construcción de la propuesta final a lo largo de todas las iteraciones que comprende el proyecto. Como se indicó en la Sección 2.1.2, el proceso de desarrollo se efectúa siguiendo CRISP-DM, luego para la explicación de este capítulo se seguirán las mismas fases que allí se trataron.

4.1. Entendimiento del negocio

Esta fase se corresponde con el estudio del estado del arte tratado en la Sección 3.4.

4.2. Comprensión de los datos

Esta sección se corresponde con la etapa *Data Understanding* de la metodología CRISP-DM. Se trata de llevar a cabo un análisis sobre los datos que se van a utilizar en el posterior entrenamiento de modelos. Este estudio consiste en identificar los conjuntos de datos que se van a emplear, el tipo de datos de cada conjunto y comprobar la distribución de los mismos.

4.2.1. Dataset Rotterdam

El *dataset EyePACS-AIROGS-light-V2* [9] será el principal conjunto de datos sobre el que se desarrollarán algoritmos de clasificación. A este *dataset* se hará referencia por el nombre de Rotterdam, dado que es un subconjunto balanceado de imágenes estandarizadas de retinografías del conjunto de datos *Rotterdam EyePACS AIROGS*.

Descripción de los datos

El *dataset rotterdam* contiene más de 9500 imágenes retinográficas entre pacientes que presentan glaucoma y otros en los que no se ha detectado esta patología. Algunas de las

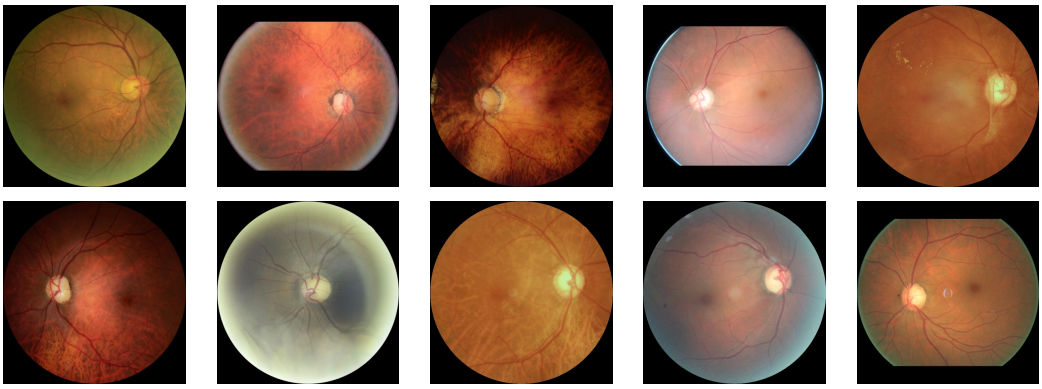


Tabla 4.1: Ejemplos de retinografías del dataset rotterdam [9]

imágenes que incluye el *dataset rotterdam* se presentan a modo de ejemplo en la Tabla 4.1.

El conjunto de datos de Rotterdam está dividido en los grupos de *test*, *train* y *validation*. Cada uno de estos subconjuntos se divide a su vez entre retinografías de pacientes con glaucoma y otros sin él. En la Tabla 4.2 se indica el número de imágenes que contiene cada carpeta para cada clase.

Carpeta	Clase	Número de imágenes
test	Glaucoma	385
	Normal	385
train	Glaucoma	4000
	Normal	4000
validation	Glaucoma	385
	Normal	385

Tabla 4.2: Resumen de las imágenes disponibles de cada clase en el dataset Rotterdam.

Distribución de los datos

Los datos siguen la distribución que se muestra en la Figura 4.1; es decir, cada una de las dos clases está representada por un total de 4770 imágenes. Además, como el número de retinografías que se tienen de cada clase es igual, los datos no presentan ningún tipo de problema respecto al balanceo de los mismos.

4.2.2. Dataset RIM-ONE

El *dataset RIM-ONE* [13] se utilizará como un conjunto de datos auxiliar a lo largo de este proyecto, puesto que se tienen *datasets* más interesantes para las tareas de clasificación

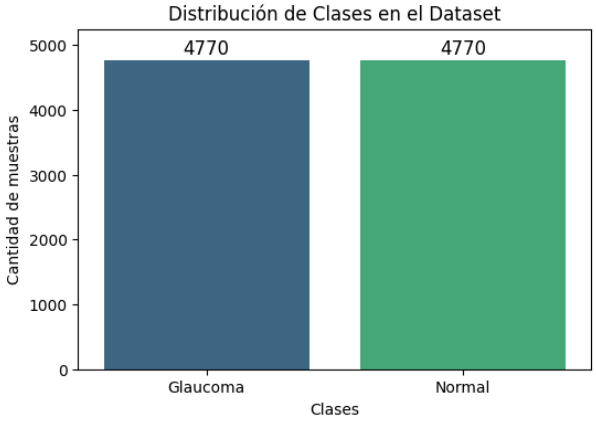


Figura 4.1: Número de datos de cada clase en el dataset Rotterdam.

y segmentación. Este conjunto de datos será usado para preentrenar modelos sobre los que realizar un segundo entrenamiento a partir del *dataset rotterdam*.

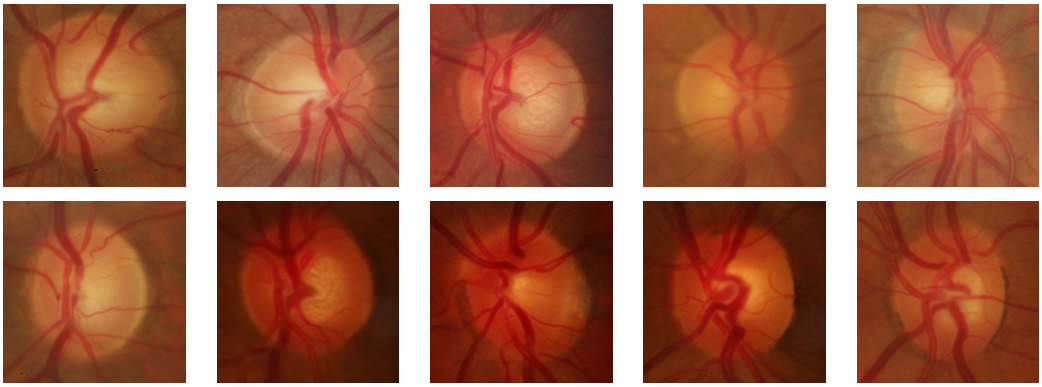


Tabla 4.3: Ejemplos de retinografías del dataset RIM-ONE [13]

Descripción de los datos

Este *dataset* contiene solo la región del nervio óptico extraído de retinografías como se muestra en los ejemplos de la Tabla 4.3. En consecuencia, como se busca clasificar retinografías completas, estas imágenes no pueden usarse directamente. Se verá en la propuesta de solución, que este *dataset* servirá para preentrenar ciertos modelos que empleen tan solo la región ONH.

Como se indica en la documentación de este dataset [12], originalmente fue concebido como un conjunto de imágenes de referencia para la segmentación del disco óptico a partir de imágenes tomadas en distintos hospitales de España. De hecho, aunque no se vaya a nombrar pues no se ha utilizado, existe un segundo repositorio con las segmentaciones correspondientes a las retinografías [13]. No obstante, su uso ha degenerado hacia el entrenamiento y evaluación de modelos de *Deep Learning* como es el caso.

Carpeta	Subcarpeta	Clase	Número de imágenes
partitioned_by_hospital	*test_set*	Glaucoma	56
		Normal	118
	training_set	Glaucoma	116
		Normal	195
partitioned_randomly	*test_set*	Glaucoma	52
		Normal	94
	training_set	Glaucoma	120
		Normal	219

Tabla 4.4: Resumen de la disposición original de los datos de RIM-ONE.

A colación de lo anterior, en un origen existían tres versiones distintas, pero se combinaron en una nueva versión pública, denominada *RIM-ONE DL* (*RIM-ONE for Deep Learning*). Esta nueva versión es la que se utiliza a lo largo de este proyecto. La versión utilizada de este conjunto de datos incluye 313 retinografías de pacientes sanos y 172 retinografías de pacientes con glaucoma. Además, todas ellas se acompañan de las correspondientes segmentaciones del disco y la copa ópticos. Las retinografías vienen organizadas siguiendo el esquema de la Tabla 4.4.

Por su parte, existe otro directorio con las segmentaciones. Estas vienen almacenadas siguiendo la organización de la Tabla 4.5. En el recuento de archivos del directorio que representa a cada clase, se tienen 4 archivos para cada imagen. Estos archivos se corresponden con dos imágenes con la segmentación de la copa y el disco, y dos archivos de texto con los puntos del contorno del disco y la copa.

Clase	Número de archivos
Glaucoma	688
Normal	1252

Tabla 4.5: Resumen de la disposición original de los datos de RIM-ONE.

Distribución de los datos

Los datos siguen la distribución que se muestra en la Figura 4.2; es decir, la clase que representa a las imágenes de la región ONH con glaucoma alberga un total de 172 ejemplos. Por su parte, se tienen 313 imágenes de la región ONH de pacientes sanos. A pesar de que el número de retinografías que se tienen de cada clase es bastante disparejo, pues tan solo en torno a un tercio son retinografías de ojos glaucomatosos, los datos están suficientemente balanceados. Además, el balanceo no es tan importante en este caso,

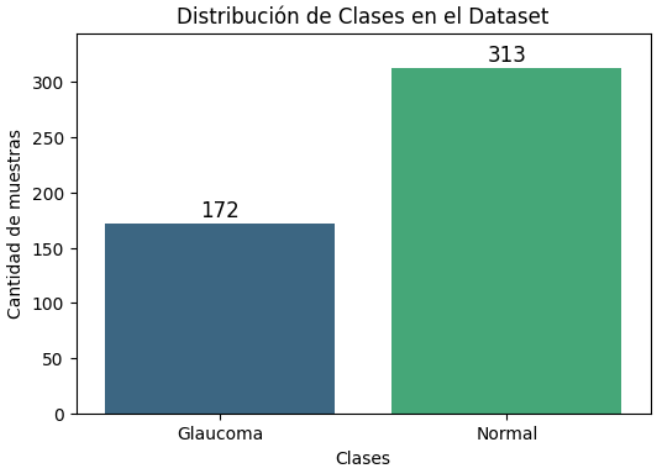


Figura 4.2: Número de datos de cada clase en el dataset RIM-ONE.

siempre que no supere unos límites, pues los datos tan solo sirven como un preentramiento, y el conjunto donde el balanceo es más importante es para el *dataset rotterdam* que es con el que se hace el entrenamiento principal.

4.2.3. Dataset DRISHTI-GS

El dataset *DRISHTI-GS* [11] es un conjunto de datos que alberga retinografías completas junto con otros archivos que identifican partes destacadas del nervio óptico. Este dataset será el conjunto de datos sobre el que se desarrollarán algoritmos de segmentación.

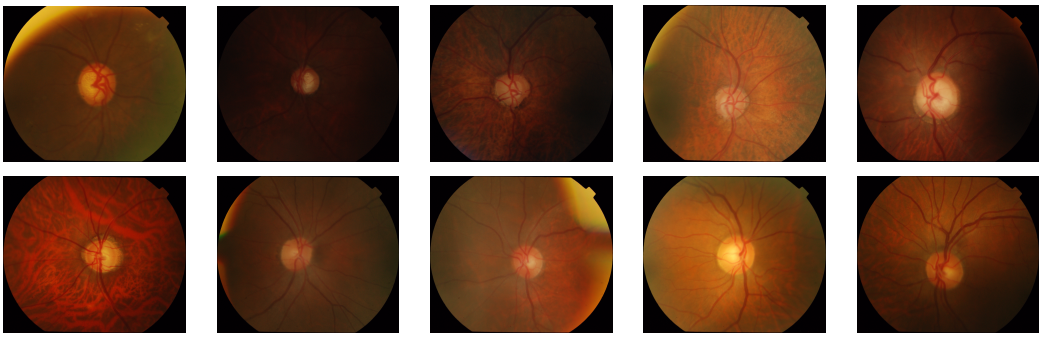


Tabla 4.6: Ejemplos de retinografías del dataset DRISHTI-GS [11]

Descripción de los datos

Como ya se ha mencionado, *DRISHTI-GS* es un conjunto de datos que alberga retinografías completas como las de la Tabla 4.6. Además, para cada una de estas imágenes se tiene una serie de archivos adicionales con píxeles destacados y otras imágenes. Por una parte, los archivos que identifican ciertos píxeles son tres. Estos señalan el contorno

del disco, de la copa y dónde se encuentra el centro del nervio óptico. Por otra parte, las imágenes son de carácter binario, e indican dónde se encuentra la copa o el disco, según corresponda. En las tablas 4.7 y 4.8 se muestran ejemplos de cada tipo de imágenes, respectivamente.

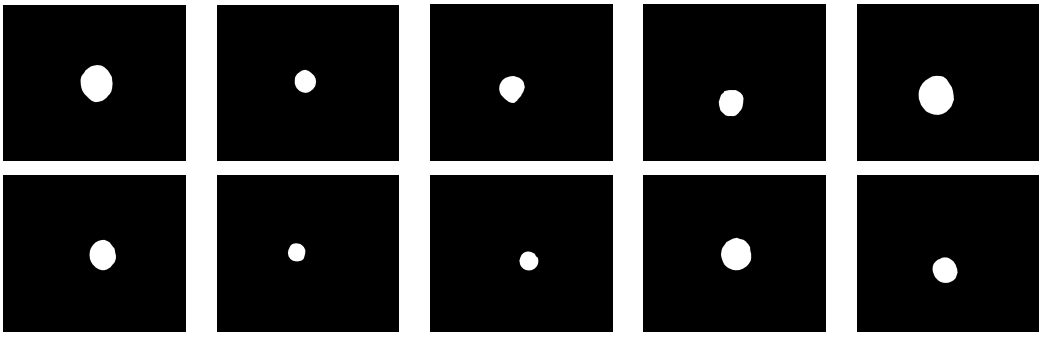


Tabla 4.7: Ejemplos de segmentaciones de la copa en el dataset DRISHTI-GS [11]

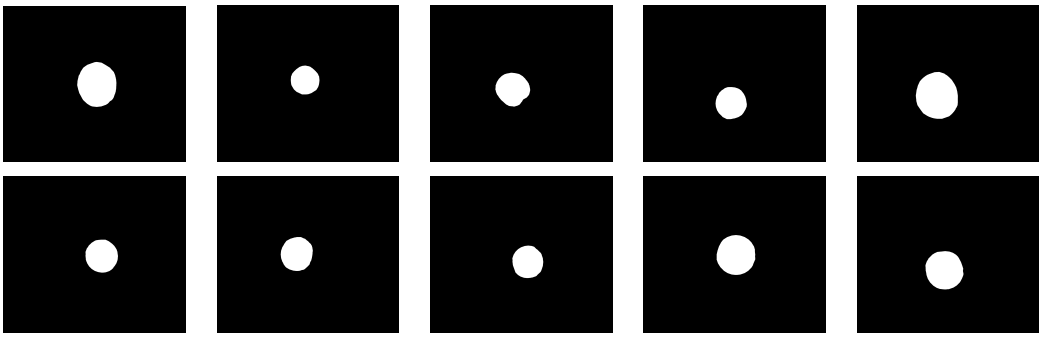


Tabla 4.8: Ejemplos de segmentaciones del disco en el dataset DRISHTI-GS [11]

De esta forma, podemos construir modelos de segmentación a partir de las imágenes binarias o de los archivos que indican el contorno correspondiente. Para el desarrollo de este trabajo se ha optado por la opción de las imágenes binarias por la facilidad que ofrecen en el entrenamiento de los modelos que se presentarán más adelante.

Este conjunto de datos contiene un total de 101 imágenes de retinografías completas con sus respectivos archivos asociados. A continuación, se analiza la distribución de los datos, aunque no será de especial relevancia, pues, independientemente de si la retinografía corresponde a una persona que tenga o no glaucoma, la segmentación debe producir el mismo resultado identificando las estructuras correspondientes del nervio óptico, que son las mismas en ambos casos; aunque, si bien es cierto que, en función de si se presenta o no la patología, las estructuras pueden variar ligeramente su tamaño, como se explicó en la Sección 3.1.3 para la copa óptica.

Queda por mencionar la colocación que sigue el conjunto de datos de *DRISHTI-GS*. Este está dividido en dos grupos: *train* y *test*. Cada uno de estos subconjuntos se divide

Carpeta	Clase	Número de imágenes
test	Glaucoma	38
	Normal	13
train	Glaucoma	32
	Normal	18

Tabla 4.9: Resumen de las imágenes disponibles de cada clase en el dataset DRISHTI-GS.

a su vez entre retinografías de pacientes con glaucoma y otros sin él. En la Tabla 4.9 se indica el número de imágenes que alberga cada carpeta para cada clase.

Distribución de los datos

En el *dataset DRISHTI-GS*, los datos siguen la distribución que se muestra en la Figura 4.3; es decir, la clase del glaucoma está representada por un total de 70 imágenes, mientras que para la clase normal se tienen 31 imágenes.

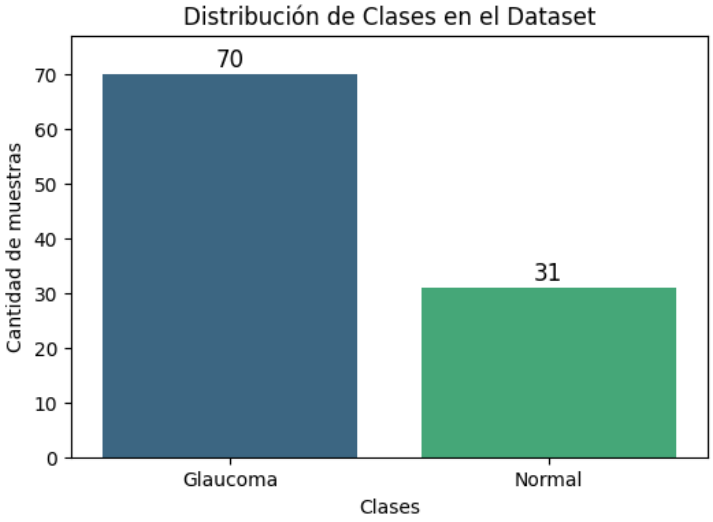


Figura 4.3: Número de datos de cada clase en el dataset DRISHTI-GS.

Aunque la diferencia entre imágenes pueda denotar que los datos están desbalanceados, esto no sucede. Como se ha indicado previamente, este dataset está orientado a segmentación. A la detección y localización de las estructuras correspondientes del nervio óptico, que son las mismas en ambos casos y tan solo pueden variar ligeramente su tamaño. Luego, la división en las clases de glaucoma y normal no es relevante, pues no se tiene en cuenta para el desarrollo del modelo que se construya a partir de estos datos.

4.3. Iteración 1. Clasificación de la retinografía completa

La aproximación más básica a la solución del problema planteado de detectar el glaucoma pasa por la construcción de un algoritmo de clasificación a partir de su entrenamiento con las imágenes de las retinografías. Para ello, se parte del *dataset* orientado a este propósito de entre los que se tiene; es decir, el de Rotterdam. Con él, se entrena el modelo y se evalúan los resultados. Este proceso se detalla en cada uno de los apartados siguientes.

4.3.1. Preparación de los datos

Puesto que el problema que se pretende solucionar en esta primera iteración es el de clasificación, se parte del conjunto de datos de Rotterdam descrito en la Sección 4.2.1. Esto se debe a que este *dataset* es el que alberga una mayor cantidad de datos dispuestos para clasificación. Los datos de esta fuente han sido reordenados de la siguiente manera:

- Conjunto de entrenamiento [*train*]. Este segmento de datos ha de ser el que mayor cantidad de los mismos albergue. Típicamente se reserva en torno a un 70 % del total de datos disponibles. En particular, para el entrenamiento de los modelos de esta propuesta se ha separado un 64 % de cada una de las clases.

- Conjunto de validación [*val*]. Este conjunto de datos se empleará durante la fase de entrenamiento. Durante este proceso, al finalizar cada época, se valida el resultado obtenido mediante este subconjunto de datos, lo que permite ajustar los parámetros de la red en función de los resultados obtenidos para mejorar el modelo en construcción. Con este fin, se ha reservado un 16 % de las retinografías con glaucoma y un 16 % de las retinografías de pacientes sanos.

  Es una práctica de uso habitual tomar un porcentaje de los datos de entrenamiento para la validación; y así es como se ha hecho en este caso. Para el proceso de entrenamiento en total se toma un 80 %, y de este se divide un 20 % para validación, lo que resulta en un 16 % del total, y el resto, un 64 % del total, para lo que es propiamente el entrenamiento.

- Conjunto de test [*test*]. Este conjunto de datos se empleará para probar el modelo construido. Se utilizará para obtener las métricas que miden la capacidad del modelo. En concreto, está compuesto por un 20 % de las retinografías con glaucoma y un 20 % de las retinografías de pacientes sanos.

Esta reordenación que se ha tomado sobre los datos tiene su motivo. En vez de reservar un 8 % para validación durante el entrenamiento y otro 8 % para prueba del modelo como se menciona en 4.2.1, se coge un 16 % y un 20 % respectivamente. Esta nueva disposición de los datos presenta dos ventajas diferenciadoras:

- La evaluación final del rendimiento del modelo es más robusta. Al contar con un mayor número de datos de prueba, se mejora la estimación del rendimiento real del modelo.
- El ajuste de los parámetros internos del modelo mejora. Con más datos para validar, las métricas de validación son más representativas.

Bien es cierto que no todo son ventajas, pues el aumento en estos subconjuntos de datos se produce a costa de reducir el número de datos de entrenamiento. Esto puede afectar a la capacidad del modelo para extraer patrones, especialmente si los datos son limitados. Sin embargo, como tenemos un total de 9540 imágenes, estas son suficientes y se puede permitir esta disminución de imágenes en el conjunto de entrenamiento.

Además de procesar las retinografías originales, que están a color, también se realiza un entrenamiento para la predicción del glaucoma a partir de las mismas imágenes pero en escala de grises durante esta iteración. La disposición de los datos es la misma en ambos casos, y solo cambia el número de canales que tiene la imagen. A esta modificación del dataset original se le ha denominado *Rotterdam_grises*.

4.3.2. Entrenamiento

En la etapa de entrenamiento se procede a construir los modelos que se pretenden que clasificaquen retinografías entre las de pacientes sanos y los que presenten glaucoma. Como se ha indicado previamente, en esta primera iteración se consideran dos modelos: uno orientado a clasificación con retinografías a color, y otro análogo con retinografías en escala de grises.

Durante esta fase se ajustan los parámetros internos de los modelos de *Deep Learning* a partir de los datos disponibles siguiendo la teoría explicada en la Sección 3.2.3. Para ello, se indican una serie de hiperparámetros que determinan algunos procesos que se llevan a cabo. En concreto, se indica el número de épocas que se debe entrenar el modelo; la paciencia, para evitar el sobreajuste; o el tamaño de entrada de las imágenes. Además, se indica el modelo base sobre el que se entrena para no tener que construir la red desde cero. En la Tabla 4.10, se detallan los hiperparámetros empleados durante el entrenamiento de cada uno de los dos modelos. Para seleccionarlos, se han tenido en cuenta aspectos como el coste computacional del proceso.

Como se observa en la Tabla 4.10, se ha empleado para realizar el entrenamiento el sistema YOLO. Este se ha importado desde la biblioteca *ultralytics*. Durante las iteraciones se verá que se usa tanto FastAI como YOLO, teniendo más presencia este último. Esto se debe a la rapidez del algoritmo, pues se basa en el paradigma " *You Only Look Once*"[60].

4.3.3. Evaluación

Una vez entrenados los modelos, han sido evaluados utilizando el subconjunto de los datos que se reservó para prueba: el conjunto de test. Para cuantificar la capacidad de clasificación de cada modelo adecuadamente, se ha empleado la métrica *accuracy* definida

Modelo	Descripción	Hiperparámetros	
rotterdam	rotterdam a color	Algoritmo	YOLO
		Modelo base	YOLO 11
		Dataset	Rotterdam
		Épocas	100
		Paciencia	15
		Tamaño imagen	256
rotterdam grises	Modelo escala de grises	Algoritmo	YOLO
		Modelo base	YOLO 11
		Dataset	Rotterdam_grises
		Épocas	100
		Paciencia	15
		Tamaño imagen	256

Tabla 4.10: Parámetros empleados en el entrenamiento de los modelos de la iteración 1

en la Sección 3.2.4. En la Tabla 4.11 se resumen los resultados obtenidos tanto para el conjunto de datos *val* como para el de *test*.

Modelo	Épocas	*Accuracy* validación	*Accuracy* test
rotterdam	52	0.938	0.933
rotterdam grises	41	0.921	0.912

Tabla 4.11: Resultados obtenidos en la iteración 1 tras el entrenamiento

A partir de los resultados obtenidos en la Tabla 4.11 se puede comparar de manera cuantitativa el rendimiento de los modelos considerados, identificando el que mejor comportamiento presenta en términos de la métrica *accuracy*. Atendiendo a los mismos, se observan mejores resultados para el modelo que recibe como datos de entrada imágenes a color en vez del modelo para escala de grises, tanto en el conjunto de validación como en el de test. Lo más representativo siempre será el de test, pues los datos de validación han sido empleados para ajustar los parámetros durante la etapa de entrenamiento y no representan fielmente el rendimiento que pueda tener el modelo en la realidad.

De esta forma, el resultado más destacado en términos de *accuracy* durante esta iteración, orientada a clasificación, ha sido de un 93.3 %. Esto implica que menos 7 de cada 100 imágenes son clasificadas de manera errónea, lo que significa un gran avance para esta primera iteración del proyecto. Además, razonando el motivo por el que se puede haber obtenido un mejor resultado para imágenes a color que para imágenes en escala de grises, puede deberse a la propia construcción de las imágenes en cada caso. Frente a los tres

canales RGB que componen la imagen a color, para la imagen en escala de grises tan solo se tiene uno, lo que supone una disminución en la cantidad de información que aporta cada retinografía en términos computacionales.

4.4. Iteración 2. Segmentación con Deep Learning y clasificación con Machine Learning

Una vez tratado el problema de clasificar las retinografías entre las de pacientes sanos y las de aquellos que presentan indicios de glaucoma, surge la necesidad de abordar el otro objetivo principal recogido en la Sección 1.2. Este se trata de la identificación de las estructuras propias de las retinografías empleando técnicas de segmentación.

A lo largo de esta sección, se explicará el dataset del que se parte, el entrenamiento del modelo, y el correspondiente análisis de los resultados que se han llevado a cabo para la consecución de este objetivo. A continuación, se detalla cada una de las partes de este proceso.

4.4.1. Preparación de los datos

Puesto que el problema que se pretende solucionar en esta segunda iteración es el de segmentación a partir de una retinografía, se parte del conjunto de datos DRISHTI-GS descrito en la Sección 4.2.3. Esto se debe a que este dataset es el único del que se dispone que alberga la información necesaria para segmentar las estructuras del disco y la copa ópticos a partir de retinografías completas.

Por otra parte, puesto que la segmentación se ha entrenado mediante YOLO, o los modelos que facilita FastAI, la disposición del conjunto de datos que se ha empleado para entrenar los modelos varía en función de los mismos:

- **FastAI**. Para los modelos que se entrenarán usando las arquitecturas que facilita FastAI, se dispone de una carpeta con las imágenes y otra con las segmentaciones que se pretenden obtener. Puesto que se van a probar distintos formatos de imágenes, como RGB, escala de grises o aplicando filtros de Gabor, tan solo será necesario cambiar la carpeta de donde se toman las imágenes. Puesto que para los filtros de Gabor, las pruebas iniciales determinaron un bajo rendimiento, en lo que sigue solo se tendrá en cuenta las imágenes a color y en escala de grises, sin ningún filtro.

  Así, se tienen las carpetas: *cup_segmentations*, con las segmentaciones de la copa; *disc_segmentations*, que almacena las segmentaciones del disco; *fundus*, que alberga las retinografías a color; y *fundus_filtered*, con las imágenes de la retinografía en escala de grises.

  Para los modelos entrenados a partir de FastAI no es necesario dividir los directorios para determinar los conjuntos de entrenamiento y validación. Esto se hace con el *dataloader* que es una estructura la cual se construye previo al entrenamiento y que determina los datos que se usarán. En este se dividen los datos en un 80 %

para entrenamiento y un 20 % para validación. Además, también especifica que las imágenes de entrada a la red neuronal serán de tamaño 224×224 píxeles.

- **YOLO**. Estos modelos requieren los datos de entrada organizados como se describió en la iteración 1. De esta manera, para el uso de YOLO se han considerado dos *datasets* diferenciados: uno para la segmentación del disco óptico y otro para la de la copa óptica, denominados *dataset_yolo_disc* y *dataset_yolo_cup* respectivamente. En ambos casos, con las imágenes a color. En cada uno de estos *datasets* se ha dispuesto una carpeta [*train*] para el entrenamiento, y otro directorio [*val*] para el proceso de validación.

  Además, tanto para el modelo del disco como para el de la copa se han separado los datos en 80 imágenes para *train* y 21 para *val*. Esto representa un 80 % y un 20 % del total de datos disponibles, respectivamente. En cada uno de los directorios, la carpeta de imágenes se acompaña con otra denominada *labels* que contiene un archivo de texto referido a cada imagen e incluye la información necesaria para poder determinar la segmentación esperada.

  Queda destacar un elemento fundamental que se debe tener previo al entrenamiento de los modelos. Se trata de un archivo de extensión *yaml* que describe cómo debe tomar YOLO los datos. Estos se denominan *cup_config.yaml* y *disc_config.yaml* para la copa y el disco, respectivamente. Su contenido es el siguiente:

cup_config.yaml

```
path: /content/drive/My Drive/Colab Notebooks/dataset_modified/dataset
_yolo_cup
train: train/images
val: test/images
nc: 1
names: ['copa']
```

disc_config.yaml

```
path: /content/drive/My Drive/Colab Notebooks/dataset_modified/dataset
_yolo_disc
train: train/images
val: test/images
nc: 1
names: ['disco']
```

  Como se puede comprobar, en estos archivos se indica la ruta donde se encuentra el *dataset* en cuestión, junto con las rutas relativas de las imágenes dentro del directorio del *dataset*. Además, *nc* informa del número de clases de objetos que se pretenden segmentar; mientras que *names* muestra el nombre que recibirá la clase.

En esta iteración, además de entrenar los modelos de segmentación con los datos preparados como se ha descrito, también se han entrenado modelos de clasificación, los cuales están construidos mediante algoritmos de *Machine Learning*. Para estos, se parte de la hipótesis mencionada en la Sección 3.1.3 de que la copa aumenta su tamaño en presencia de glaucoma.

Partiendo de esta idea y mediante los modelos construidos para segmentación, se pasan a construir una serie de archivos CSV (Comma-separated values) que recogen un conjunto de medidas y/o proporciones oportunas tanto del disco como de la copa. A continuación, se describe cada uno de los archivos CSV que se utilizarán en el entrenamiento:

- ***datos_DRISHTI.csv***. Este fichero se ha construido a partir de las máscaras que se tienen para las retinografías del *dataset DRISHTI-GS*. De esta forma, se puede comprobar si puede ser útil para la clasificación la hipótesis de la que se parte. En caso de poder ser útil, se hace lo mismo para las segmentaciones. Así, se prueba si es usable en condiciones ideales para luego comprobarlo con las segmentaciones, que inevitablemente introducirán errores en las medidas.

  En este archivo, se presentan 4 columnas de datos. Las tres primeras, *diam_disco*, *diam_copa* y *prop_diam*, representan el diámetro del disco y de la copa, así como la proporción entre los mismos, respectivamente. Formalmente,

  $$prop\_diam = \frac{diam\_disco}{diam\_copa}.$$

  Cabe recordar que la definición de diámetro se incluyó en la Sección 3.3. Por último, la cuarta columna indica de manera binaria si la retinografía corresponde a un globo ocular con glaucoma, representado por 0, o sin él, representado por 1.

- ***datos_fastai.csv***. Este fichero coincide en lo referido a los datos que se almacenan con el archivo *datos_DRISHTI.csv*. La diferencia radica en que para su construcción no se utilizan las máscaras propias del *dataset*, sino que se emplean las predicciones hechas por el modelo FastAI que mejor rendimiento tenga.

- ***datos_YOLO.csv***. Este fichero es análogo al archivo *datos_fastai.csv*. El único cambio es que las medidas se toman a partir del modelo construido con YOLO en vez de con el de FastAI.

- ***datos_completo.csv***. Este archivo extiende la información que se tiene en los ficheros *datos_fastai.csv* y *datos_YOLO.csv*. En concreto, las primeras columnas se refieren a medidas tomadas de la predicción YOLO; las siguientes, a la predicción FastAI; y la última es la clasificación esperada. Respecto a las columnas que recogen atributos de las segmentaciones, tenemos los siguientes datos:

  - **min_radio_disco_yolo**: denota la distancia mínima entre el centroide del disco detectado con el modelo YOLO y el contorno del mismo.

- **max_radio_disco_yolo**: es la distancia máxima entre el centroide del disco detectado y el contorno del mismo con el modelo YOLO.
- **min_radio_copa_yolo**: mínimo de las distancias entre el centroide de la copa detectada con el modelo YOLO y el contorno de la misma.
- **max_radio_copa_yolo**: es la distancia máxima entre el centroide de la copa segmentada y el contorno de la misma con el modelo YOLO.
- **prop_radios_max_yolo**: cociente entre *max_radio_disco_yolo* y *max_radio_copa_yolo*.
- **prop_radios_min_yolo**: división de *min_radio_disco_yolo* entre *min_radio_copa_yolo*.
- **prop_radios_disco_yolo**: proporción entre los radios mínimo y máximo de la segmentación del disco con el modelo YOLO; es decir, entre *min_radio_disco_yolo* y *max_radio_disco_yolo*, respectivamente.
- **prop_radios_copa_yolo**: proporción para la copa detectada con el modelo YOLO entre el mínimo (*min_radio_copa_yolo*)y máximo (*max_radio_copa_yolo*) de sus radios.
- **min_radio_disco**: denota la distancia mínima entre el centroide del disco detectado con el modelo construido con FastAI y el contorno del mismo.
- **max_radio_disco**: es la distancia máxima entre el centroide del disco detectado y el contorno del mismo con el modelo construido con FastAI.
- **min_radio_copa**: mínimo de las distancias entre el centroide de la copa detectada con el modelo construido con FastAI y el contorno de la misma.
- **max_radio_copa**: es la distancia máxima entre el centroide de la copa segmentada y el contorno de la misma con el modelo construido con FastAI.
- **prop_radios_max**: cociente entre *max_radio_disco* y *max_radio_copa*.
- **prop_radios_min**: división de *min_radio_disco* entre *min_radio_copa*.
- **prop_radios_disco**: proporción entre los radios mínimo y máximo de la segmentación del disco con el modelo construido con FastAI; es decir, entre *min_radio_disco* y *max_radio_disco*, respectivamente.
- **prop_radios_copa**: proporción para la copa detectada con el modelo construido con FastAI entre el mínimo (*min_radio_copa*)y máximo (*max_radio_copa*) de sus radios.

4.4.2. Entrenamiento

Dentro de esta fase de la segunda iteración, se pueden distinguir distintas clases de entrenamiento en función del objetivo o la biblioteca de modelos usada; es decir, si se trata de un problema de segmentación o clasificación, y si se emplea YOLO o FastAI. Luego, a continuación se da una explicación para cada uno de los casos de manera análoga para todos ellos.

FastAI para el disco óptico

Usando la biblioteca FastAI para segmentar el disco óptico, se han contemplado una serie de arquitecturas que esta misma biblioteca recoge. En particular, se ha probado a construir modelos a partir de *Transfer Learning* [61] con las arquitectura *Resnet*, *Alexnet*, *Densenet* y *VGG*, donde para las que ofrecen la oportunidad se han utilizado variantes con más o menos parámetros, como sucede con *Resnet34* y *Resnet152* o con *VGG-16* y *VGG-19*.

Además de seleccionar la arquitectura con la que se construirán los modelos, también se ha de escoger el número de épocas que se van a entrenar los mismos. En este caso, se han ido probando diferentes ajustes y se ha tomado la opción que mejores resultados aportaba. Este trabajo se ha realizado de manera manual examinando el ajuste a lo largo de las épocas para que el modelo no sobreentrene y haya un sobreajuste, pero también para que ajuste lo suficiente y no quede sin entrenar. Otra forma podría haber sido introducir un hiperparámetro de *patience* o paciencia.

Finalmente, para terminar de definir los hiperparámetros del modelo, se debe escoger el conjunto de métricas en las que se basará FastAI para realizar los correspondientes ajustes en los parámetros para adaptar la red neuronal a la solución. Para esta fase de entrenamiento se ha empleado la métrica *accuracy_camvid* definida en la Sección 3.2.4 para todos los modelos. En la fase de evaluación se estudiará una mayor cantidad métricas para examinar en detalle los modelos escogidos.

De esta forma, en la Tabla 4.12 se recoge la arquitectura, el número de épocas y la métrica utilizada para ajustar el modelo tanto para la imagen en escala de grises como a color en la base RGB.

Arquitectura modelos	Resultado disco escala de grises		Resultado disco a color	
	Número etapas	Métrica	Número etapas	Métrica
Resnet34	13	*accuracy_camvid*	13	*accuracy_camvid*
Resnet152	20	*accuracy_camvid*	20	*accuracy_camvid*
Alexnet	25	*accuracy_camvid*	25	*accuracy_camvid*
Densenet	12	*accuracy_camvid*	12	*accuracy_camvid*
VGG-16	12	*accuracy_camvid*	12	*accuracy_camvid*
VGG-19	17	*accuracy_camvid*	17	*accuracy_camvid*

Tabla 4.12: Parámetros modelos FastAI de segmentación del disco en la iteración 2.

FastAI para la copa

De manera análoga a como se ha tratado el entrenamiento de los modelos para la segmentación del disco óptico, se construyen los modelos para la segmentación de la

copa. Así, se debe definir la arquitectura sobre la que se entrenará el modelo. Se han considerado las mismas que en el caso del disco. Además, también se define el número de épocas que se entrenará el modelo y la métrica para hacerlo. De igual modo que para el entrenamiento de los modelos de segmentación del disco, se ha ajustado el número de épocas para obtener los mejores resultados. Además, la métrica considerada también es la misma: *accuracy_camvid*. En la Tabla 4.13 se detallan los hiperparámetros a partir de los cuales se construyen los distintos modelos.

Arquitectura modelos	Resultado copa escala de grises		Resultado copa a color	
	Número etapas	Métrica	Número etapas	Métrica
Resnet34	12	*accuracy_camvid*	12	*accuracy_camvid*
Resnet152	18	*accuracy_camvid*	18	*accuracy_camvid*
Alexnet	25	*accuracy_camvid*	25	*accuracy_camvid*
Densenet	12	*accuracy_camvid*	12	*accuracy_camvid*
VGG-16	11	*accuracy_camvid*	11	*accuracy_camvid*
VGG-19	16	*accuracy_camvid*	16	*accuracy_camvid*

Tabla 4.13: Parámetros modelos FastAI de segmentación de la copa en la iteración 2.

YOLO para el disco

En la etapa de entrenamiento para el modelo YOLO se empleará YOLO11. Para la construcción del modelo que segmenta el disco se parte del *dataset DRISHTI-GS* con las retinografías a color. Además, se debe especificar durante esta fase una serie de características que acoten el funcionamiento del entrenamiento.

Modelo	Hiperparámetros	
yolo_disc	Algoritmo	YOLO
	Modelo base	YOLO 11
	Dataset máscaras	disc_segmentations
	Épocas	100
	Paciencia	-
	Tamaño imagen	256

Tabla 4.14: Parámetros modelo YOLO de segmentación del disco en la iteración 2.

Para ello, se indican una serie de hiperparámetros que determinan algunos procesos que se llevan a cabo. En concreto, se indica el número de épocas que se debe entrenar el

modelo; la paciencia, para evitar el sobreajuste; o el tamaño de entrada de las imágenes. Además, se indica el modelo base sobre el que se entrena para no tener que construir la red desde cero.

A colación de lo anterior, en la Tabla 4.14 se detallan los hiperparámetros empleados durante el entrenamiento del modelo que segmenta el disco óptico a partir de retinografías a color. Para seleccionar dichos hiperparámetros se han tenido en cuenta aspectos como el coste computacional del proceso.

YOLO para la copa

De manera análoga a como se ha tratado el entrenamiento del modelo construido a partir de YOLO para la segmentación del disco óptico, se construye el modelo para la segmentación de la copa. Así, se debe definir la arquitectura sobre la que se entrenará el modelo, el número de épocas que se entrenará el mismo, la dirección donde se encuentran las máscaras a partir de las que se entrenará y el tamaño de la imagen de entrada a la red neuronal.

Modelo	Hiperparámetros	
yolo_cup	Algoritmo	YOLO
	Modelo base	YOLO 11
	Dataset máscaras	cup_segmentations
	Épocas	100
	Paciencia	-
	Tamaño imagen	256

Tabla 4.15: Parámetros modelo YOLO de segmentación de la copa en la iteración 2

En la Tabla 4.15 se detallan los hiperparámetros a partir de los cuales se construye el modelo capaz de identificar la región de la copa óptica.

Entrenamiento de los algoritmos de *Machine Learning* para clasificación

Para el entrenamiento de los modelos de clasificación mediante técnicas de *Machine Learning* se usarán los algoritmos propuestos en la Sección 3.2.1. Además, para los que se pueda establecer una serie de hiperparámetros, se elegirán los óptimos; es decir, los que mejores resultados arroje su correspondiente entrenamiento. Esto es, para los modelos de *KNN*, elegir el número óptimo de vecinos a partir de los que se hace la predicción, o para el *clustering*, el conjunto de atributos que se manejan así como la cantidad de *clústers* en los que se divide el conjunto de datos.

Además, para todos los algoritmos que se contemplan, se medirán sus resultados en base al *accuracy* que presenten. De hecho, esta métrica será la que se use por los algoritmos de entrenamiento para ajustar las soluciones de los modelos. Pero no solo eso, sino que

también se considerará para cada clase, glaucoma y normal, la *precision* obtenida, junto con el *recall*, y a partir de las cuales, el *F1-score*. Para entender su significado, todas estas métricas aparecen definidas en la Sección 3.2.4.

4.4.3. Postprocesado

Tras el entrenamiento de los modelos de esta segunda iteración, se distinguen dos tipos de modelos. Por una parte, los construidos siguiendo técnicas propias del *Machine Learning*, y por otra, los modelos de segmentación sobre imágenes entrenados con redes neuronales profundas que se enmarcan dentro del campo *Deep Learning*.

Respecto a los segundos, las segmentaciones de las imágenes pueden contener ciertas imperfecciones solucionables de manera automática con las debidas técnicas de tratamiento de imagen. En esta sección se van a revisar las problemáticas que se han apreciado respecto a las mismas así como la solución abordada. Es aquí donde se aplicará todo el conocimiento matemático de teoría de conjuntos relatado en la Sección 3.3 de los antecedentes. En una primera instancia se revisarán las soluciones construidas usando modelos YOLO, para continuar con las proporcionadas a través de la biblioteca FastAI.

Problemas segmentación YOLO

Con YOLO se han construido dos modelos para segmentación: uno para el disco y otro para la copa óptica, que reciben el nombre de *yolo_disc* y *yolo_cup*, respectivamente. Para probar su funcionamiento, además de comprobar su rendimiento con el conjunto de datos *DRISHTI-GS* que se ha usado para entrenar los modelos, también se hacen las respectivas pruebas con el *dataset rotterdam*. Aunque no se tengan las segmentaciones de las clases identificadas en una retinografía para este último conjunto de datos, se puede comprobar visualmente si el modelo funciona adecuadamente.

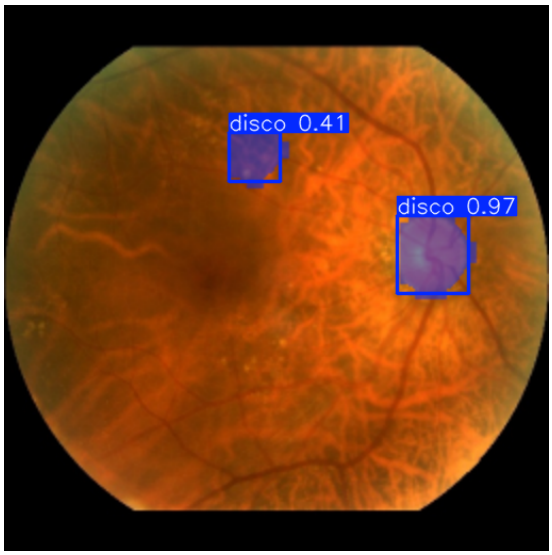


Figura 4.4: Ejemplo de error *yolo_disc*

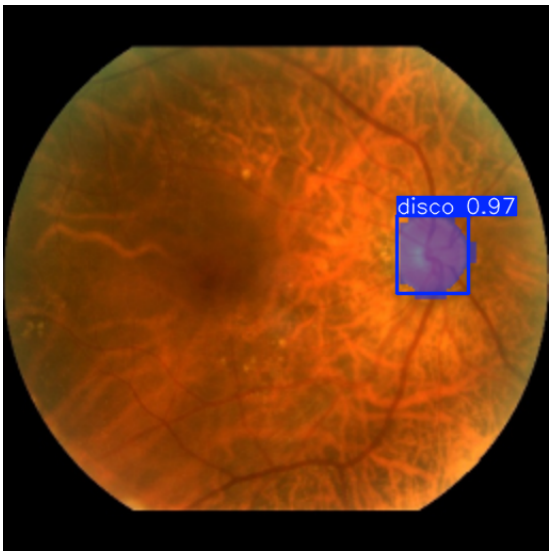


Figura 4.5: Solución de errores *yolo_disc*

En primer lugar, se estudia el modelo construido para el disco, denominado *yolo_disc*. De entre todas las retinografías que alberga el *dataset rotterdam*, solo se ha apreciado un error. Este consiste en la identificación de más de un disco. Este error se muestra en la Figura 4.4.

Como se puede comprobar en la Figura 4.4, YOLO aporta junto con la segmentación, un recuadro de localización, así como la probabilidad de que la región señalada se haya identificado correctamente. El error se soluciona sin más que tomando la región con mayor probabilidad. Se ha comprobado que siempre que ocurre este error, la máscara que identifica el disco real presenta una certeza mucho mayor como ocurre en el ejemplo de la Figura 4.4.

Cabe destacar además el buen funcionamiento del modelo, pues al probarlo en otro *dataset* completamente distinto al del entrenamiento sigue identificando correctamente el disco óptico. No obstante, en la posterior Sección 4.4.4 se evaluará de manera numérica el rendimiento de este modelo.

Una vez analizados y solucionados los errores que pueden cometerse al tratar de segmentar el disco óptico, se pasa a estudiar si para la copa óptica también se produce alguna desviación respecto del resultado esperado. Análogamente a como se hizo para el modelo del disco, además de comprobar su rendimiento con el *dataset DRISHTI-GS* que se ha usado para entrenar, también se harán pruebas con el conjunto de datos *rotterdam*. Así es como se ha observado no un error, pero sí una posible mejora a la solución facilitada por el modelo. Dicha mejora se ejemplifica en las figuras 4.6, 4.7 y 4.8.

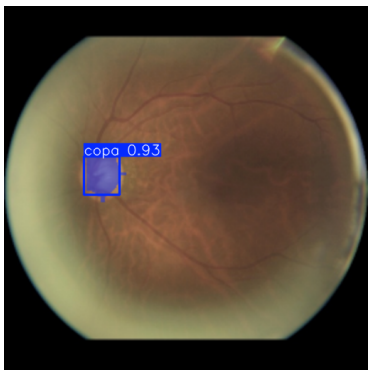


Figura 4.6: Resultado inicial

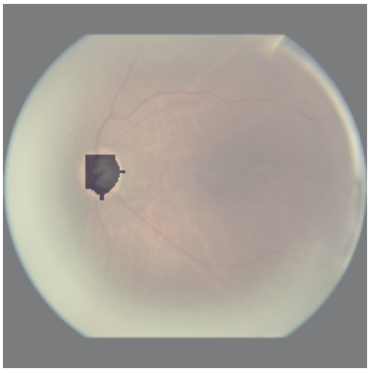


Figura 4.7: Segmentación

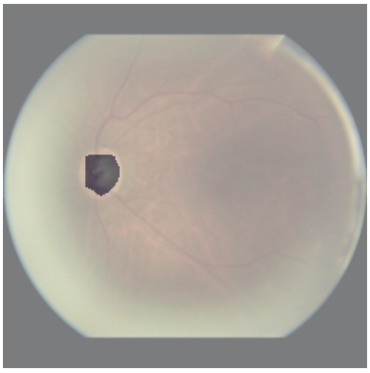


Figura 4.8: Predicción final

En la primera de las figuras, la 4.6 se muestra la predicción hecha con el modelo construido para la copa. Para poder ver mejor el resultado, en la Figura 4.7 se destaca la segmentación hecha por dicho modelo. Aquí, como se puede comprobar, aparecen ciertas *vesículas*. Para eliminarlas, se aplica un *kernel* para obtener el resultado de la Figura 4.8.

Finalmente, cabe destacar que, puesto que el disco óptico y la copa óptica se supone que deben ser conjuntos convexos, para mejorar el resultado se ha decidido calcular la envolvente convexa de las máscaras que identifican estas estructuras. Así, para la aplicación final se mostrarán tanto la solución inicial con las mejoras descritas, como el resultado que se obtiene al calcular la envolvente convexa.

Problemas segmentación FastAI

Análogamente a los modelos construidos con YOLO, se han escogido dos modelos para segmentación con FastAI: uno para el disco y otro para la copa óptica, que reciben el nombre de *fastai_disc* y *fastai_cup*, respectivamente, ambos construidos con el modelo *Resnet152*. Para probar su funcionamiento, además de comprobar su rendimiento con el conjunto de datos *DRISHTI-GS*, usado para entrenar los modelos, también se hacen las respectivas pruebas con el *dataset rotterdam*. Aunque no se tenga las segmentaciones de las clases identificadas en una retinografía para este último conjunto de datos, se comprobará visualmente si el modelo funciona adecuadamente.

Igual que para los modelos YOLO, en primer lugar, se estudia el modelo construido para el disco, *fastai_disc*. De entre todas las retinografías que alberga el *dataset rotterdam*, se han detectado dos tipos de desviaciones respecto a la segmentación esperada. Estas fallas en las segmentaciones, así como las soluciones planteadas, son las siguientes:

- **Más de una componente conexa**. En ocasiones, el modelo devuelve una segmentación formada por varias regiones disjuntas como se muestra en la Figura 4.9. En la mayoría de casos en los que esto sucede, suele deberse a la existencia de regiones con intensidad similar a la del disco óptico; es decir, a la presencia de brillos en la retinografía. Como consecuencia, el modelo interpreta erróneamente cada uno de los brillos como un disco.

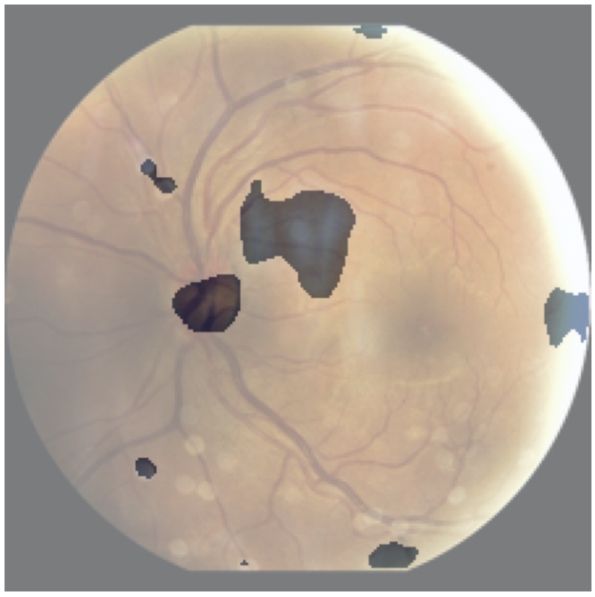


Figura 4.9: Ejemplo de segmentación con más de una componente conexa.

Solución. La respuesta esperada para el modelo es una única identificación de disco, pues para cada retinografía se sabe que solo puede existir uno. Luego, la solución pasa por seleccionar únicamente la componente conexa que identifica al disco. Sin embargo, esto es una tarea muy complicada. El motivo es que habría que

establecer condiciones para determinar cuál de todas las componentes conexas es la que identifica el disco óptico.

Otra solución pasa por aplicar filtros a partir de la transformada de Fourier [62]. Con esta técnica, se busca reducir los brillos que causan problemas en el proceso de segmentación. Así, reduciendo el efecto de estos brillos, se pretende obtener una única componente conexa que coincida con el disco óptico.

Sin embargo, aunque se hayan contemplado estas opciones e incluso implementado la primera de ellas, no es necesario hacer uso de las mismas para solventar esta cuestión. En su lugar, la solución que se ha tomado para el otro obstáculo que se presenta a continuación, también sirve para controlar el este problema.

- **Se identifica un área mayor como disco**. Otro problema que surge al segmentar la región del disco óptico es detectar un área mayor a la que le corresponde, como ocurre en el ejemplo de la Figura 4.10. El motivo de esta desviación respecto del resultado esperado es un menor contraste entre la zona del disco óptico respecto a la región inmediatamente exterior a él.

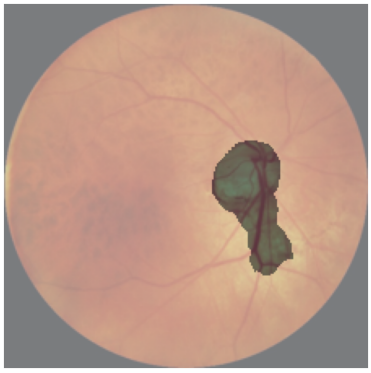


Figura 4.10: Desviación en la segmentación del disco

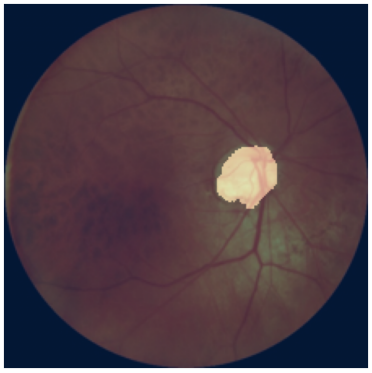


Figura 4.11: Detección del fondo de la retinografía

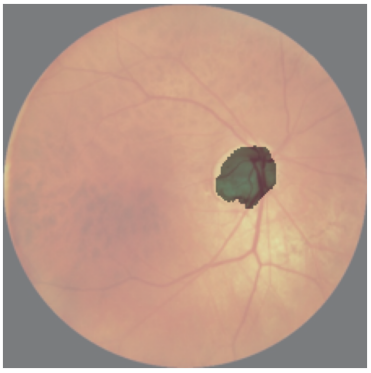


Figura 4.12: Solución aplicada al disco en FastAI

Solución. Una primera solución pasa por la aplicación de un kernel. Igual que para eliminar las vesículas, como se hizo con las predicciones de YOLO, se trata de aplicar un kernel que reduzca la región que no corresponde con el disco óptico. Esto funciona cuando las regiones mal detectadas son pequeñas. Sin embargo, para un caso como el de la Figura 4.10, donde las regiones mal detectadas son grandes, esto no representa una solución real.

Otra solución planteada es tratar de elaborar otro modelo para predicción de las segmentaciones a partir de un conjunto de datos diferente y combinar el resultado. Con este propósito, se ha entrenado un modelo para detectar el fondo de la retinografía; es decir, el área complementaria al disco en la imagen.

Para llevar a cabo esto, el conjunto de datos de entrenamiento consiste en las retinografías del *dataset DRISHTI-GS* a color, frente a las imágenes en escala de grises

de este mismo conjunto de datos con las que se ha entrenado el modelo *yolo_disc*. Así, se alteran los datos de entrada, obteniendo resultados distintos para combinarlos posteriormente. El modelo que identifica el fondo, el área complementaria a la región del disco, recibe el nombre de *fastai_disc_background*.

Así, aplicando este modelo construido como solución al problema por el que se identifica un área mayor como disco óptico para la retinografía de la Figura 4.10, se consigue un resultado como el de la Figura 4.11. Para conseguir la solución esperada a partir de los modelos que detectan el disco y su fondo, se calcula la región complementaria al fondo y se obtiene la solución final a partir de la intersección entre esta última zona y el área del disco calculada por el primer modelo. De esta forma, se alcanza la solución representada en la Figura 4.12.

De igual forma que se hizo para el disco y la copa con los modelos a base de YOLO, para mejorar la solución y reducir ciertas *vesículas* que pudieran aparecer, se aplica un *kernel*. Además, puesto que el disco óptico y la copa óptica deben ser conjuntos convexos, también se ha decidido calcular la envolvente convexa de las máscaras que identifican estas estructuras.

A partir del proceso descrito anteriormente, se alcanza un resultado mucho más fiel a la realidad. En las imágenes 4.13, 4.14, 4.15 y 4.16 se muestra cada etapa principal en ese orden. Como se identifica en la Figura 4.16, el resultado final tras el postprocesado se adecúa en mayor medida a la región original del disco óptico. A todo este proceso se le ha denotado con el nombre de *fastai_process*, el cual se mencionará para valorar los resultados obtenidos.

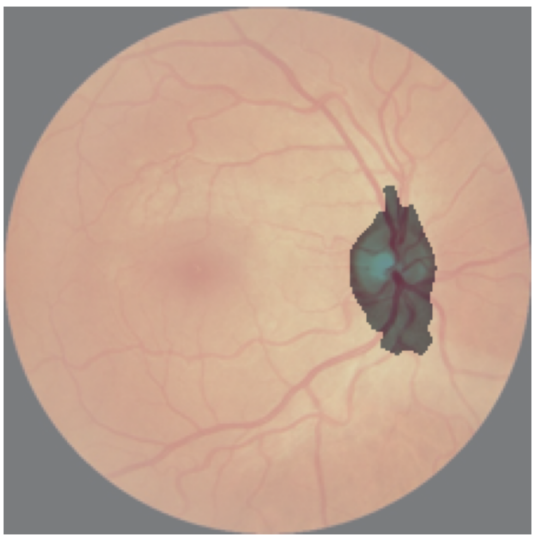


Figura 4.13: Predicción inicial del disco

Figura 4.14: Predicción del fondo

Respecto a la copa, apenas se ha mencionado su postprocesado puesto que es muy simple y ya ha sido relatado con anterioridad. Simplemente se hace la intersección de la segmentación predicha con la región del disco, dado que la copa se encuentra

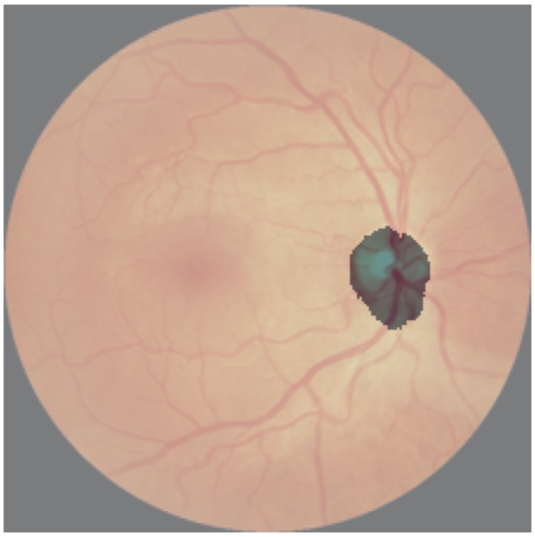


Figura 4.15: Resultado de la combinación de las dos segmentaciones realizadas

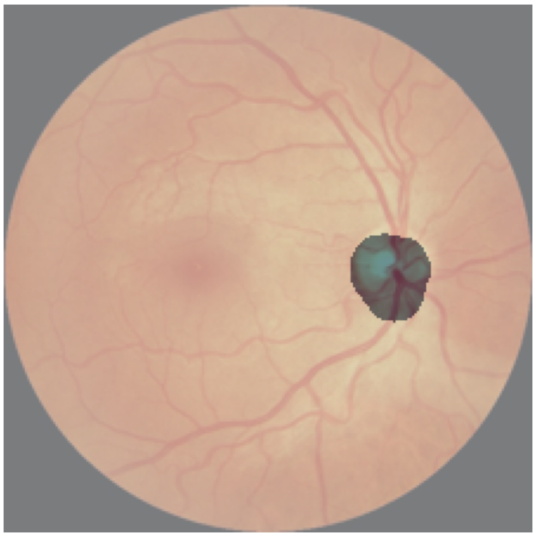


Figura 4.16: Predicción final tras aplicar un kernel y calcular la envolvente convexa

dentro del disco. Tras esto, se aplica un kernel para reducir las vesículas que pueda presentar la máscara y se calcula la envolvente convexa para obtener el resultado final que identifica la copa. Realmente es un proceso muy similar al que se ha explicado para la segmentación de la copa en los modelos YOLO.

4.4.4. Evaluación

Métricas para los modelos elegidos para el disco

El mejor modelo producido para la segmentación del disco utilizando la biblioteca FastAI es el entrenado con *Resnet152* para imágenes en escala de grises. A este modelo se le ha denominado *fastai_disc*. Por su parte, se tiene un modelo a partir de YOLO que se denomina *yolo_disc*. Junto con estos dos, también se ha considerado relevante calcular las métricas para el proceso descrito en la Sección 4.4.3 para la segmentación del disco óptico a partir de FastAI, al que se le ha nombrado como *fastai_process*.

La Tabla 4.16 presenta las métricas clave para evaluar la calidad de las segmentaciones producidas por el modelo de YOLO, junto con el mejor modelo de los de FastAI; además del proceso completo denominado *fastai_process*. En la Tabla 4.16 se evalúan los modelos escogidos, sin calcular ninguna mejora como lo de la envolvente convexa. Como se puede apreciar, el mejor modelo es *fastai_disc*, pues supera al resto en todas las métricas, a excepción de *accuracy_camvid*, donde es un 0.5 % inferior al modelo *yolo_disc*, lo que resulta despreciable en comparación con la superioridad que muestra en el resto de métricas frente a este último modelo.

Además, se observa que el proceso completo, al que se hace referencia como *fastai_process* y que se describió en la Sección 4.4.3 anterior, obtiene métricas un tanto inferiores para las imágenes del *dataset DRISHTI-DS*. Sin embargo, esto supone una solu-

Modelo	Coeficiente de Jaccard (IoU)	Coeficiente Dice	Pixel accuracy	Accuracy_camvid
fastai_disc	0.9471	0.9726	0.9983	0.9726
fastai_process	0.9243	0.9603	0.9976	0.9313
yolo_disc	0.8822	0.9371	0.9959	0.9778

Tabla 4.16: Métricas de los modelos que segmentan el disco

ción para contemplar una mayor variedad de imágenes, pues también resuelve el problema para las segmentaciones del *dataset rotterdam*. De esta forma, se sacrifica ligeramente el rendimiento para el conjunto de las 101 imágenes sobre las que se calculan las métricas (las del *dataset DRISHTI-DS*), en favor de una mayor generalización para resolver el problema propuesto.

A continuación se relata el resultado obtenido para cada métrica:

- **Coeficiente de Jaccard (IoU)**. Los valores para esta métrica indican que el modelo *fastai_disc* presenta la mayor superposición entre predicción y etiqueta real, lo que indica una segmentación muy precisa. Por su parte, *yolo_disc* tiene el peor desempeño en esta métrica, aunque sigue siendo aceptable.
- **Coeficiente Dice**. Para esta métrica, nuevamente *fastai_disc* lidera con la mejor segmentación general; mientras que *yolo_disc* es la que peor coeficiente Dice muestra, aunque mantiene un desempeño razonable.
- ***Pixel Accuracy***. Todos los modelos tienen una precisión de píxeles extremadamente alta, indicando que la mayoría de los píxeles se clasifican correctamente. Sin embargo, esta métrica puede ser engañosa si hay clases dominantes, como es el caso. Así, esta métrica no resulta representativa para la selección de un modelo u otro, puesto que en cada retinografía la mayor parte de la misma representa el fondo. Esto incrementa el valor del *pixel Accuracy* en todos los casos, consiguiendo que esta métrica no resulte representativa.
- ***Accuracy_camvid***. A pesar de que el modelo *yolo_disc* tiene peor IoU y Dice, logra el mejor *accuracy_camvid*. Esto sugiere que su clasificación global por clase es buena. Aunque la segmentación precisa, distinguiendo adecuadamente la frontera entre el disco óptico y el fondo, sea inferior.

Otros entrenamientos para la segmentación del disco

También se ha entrenado un modelo con ayuda de FastAI sobre el conjunto de datos *DRISHTI-DS* sobre el que se ha aplicado un filtro de Gabor como los descritos en la Sección 3.2.5. Para ello, se utilizó la arquitectura U-net con *Resnet34*. Desde un inicio se ha descartado la vía que proporciona la aplicación de filtros de este tipo, pues se obtienen métricas peores que para los modelos escogidos que se acaban de comentar.

En concreto, para el modelo a base de *Resnet34* con filtros de Gabor, se ha obtenido para la métrica *accuracy_camvid*, la que se utiliza para el entrenamiento del modelo, un valor de 0.92. De esta forma, usar el filtro de Gabor empeora en un 5 % las segmentaciones de los modelos sin filtro; es decir, de *yolo_disc* y *fastai_disc*, que presentan un 0.9778 y 0.9726 para *accuracy_camvid*, respectivamente.

Métricas para los modelos elegidos para la copa

El mejor modelo producido para la segmentación de la copa óptica utilizando la biblioteca FastAI es el entrenado con *Resnet152* para imágenes en escala de grises. A este modelo se le ha denominado *fastai_cup*. Por su parte, se tiene un modelo a partir de YOLO que se denomina *yolo_cup*.

La Tabla 4.17 presenta las métricas clave para evaluar la calidad de las segmentaciones producidas por el modelo de YOLO, junto con el mejor modelo de los de FastAI.

Modelo	Coeficiente de Jaccard (IoU)	Coeficiente Dice	Pixel accuracy	Accuracy_camvid
fastai_cup	0.8913	0.9413	0.9982	0.9231
yolo_cup	0.8034	0.8887	0.9964	0.9573

Tabla 4.17: Métricas de los modelos que segmentan la copa

Como se puede apreciar en la Tabla 4.17, *fastai_cup* supera en 3 de 4 métricas al modelo *yolo_cup*. Entre esas tres métricas, se incluyen las más sensibles a la segmentación precisa, como son IoU y Dice. Esto indica que realiza una mejor segmentación a nivel de detalle y contornos, siendo especialmente útil si se requiere precisión en los bordes de la copa. La otra métrica en la que es mejor es *pixel_accuracy*, que ya se explicó para la segmentación del disco óptico, que no resulta relevante para este problema.

Por su parte, *yolo_cup* aunque tiene métricas de segmentación más bajas, logra un mejor *accuracy_camvid*, de en torno a un 2.5 % superior. Esto indica que indica correctamente la copa y el fondo, aunque no delinea con tanta precisión.

Observación sobre las métricas para segmentación

De la evaluación de las métricas realizada se desprenden dos ideas:

1. La métrica que se ha utilizado para ajustar los modelos ha sido en todos los casos *accuracy_camvid*. Además, resulta interesante que los modelos entrenados con YOLO solo superan al resto en esta misma métrica. De aquí, se desprende la idea de que la arquitectura YOLO optimiza mejor para la métrica indicada en el entrenamiento.

2. Para el resto de métricas que no son *accuracy_camvid*, FastAI presenta mejores resultados. En concreto, muestra un rendimiento superior respecto a los coeficientes IoU y Dice. Esto implica que FastAI identifica de una forma más fina los bordes;

mientras que YOLO localiza muy bien la región, pero no afina tanto en los bordes del objeto buscado.

Todo esto se puede comprobar sin más que observar ejemplos como los de las figuras 4.4 y 4.7, para ver como YOLO identifica perfectamente la región pero con más vértices, y la Figura 4.10 para los modelos de FastAI, en la que se aprecia como, quizá no detecta la región tan bien, pero la segmentación no tiene tantas aristas.

Métricas de los modelos de Machine Learning

De igual manera que se establecen métricas para los modelos de *Deep Learning* entrenados con imágenes como datos de entrada, se tiene una serie de métricas para los algoritmos de *Machine Learning* empleados para la clasificación a partir de los datos extraídos de las segmentaciones. Esto permite comparar los modelos de manera objetiva para escoger el que mejor rendimiento tenga y obtener conclusiones valiosas sobre los datos a los que se aplican.

A colación de lo anterior, se van a medir los modelos construidos según su *accuracy*; es decir, los casos que aciertan. Además, en caso de que se esté interesado en un modelo en específico, se pueden analizar otras métricas como el *recall*, la precisión o el *F1-score* para cada una de las clases. Para comenzar, se presenta la métrica *accuracy* para cada uno de los conjuntos de datos tratados con cada uno de los algoritmos de *Machine Learning* considerados. Esto se presenta en la Tabla 4.18.

Algoritmo Machine Learning	*Accuracy* original	*Accuracy* completo	*Accuracy* YOLO	*Accuracy* FastAI
KNN	87.10 %	76.39 %	77.70 %	67.35 %
KNN con K-Fold	87.09 %	72.10 %	77.70 %	63.26 %
Naive Bayes	87.10 %	72.53 %	78.38 %	46.26 %
Bernoulli-Naive Bayes	67.74 %	51.07 %	50.68 %	50.34 %
Árboles de decisión	74.19 %	66.09 %	67.57 %	61.90 %
Random Forest	90.32 %	75.54 %	66.22 %	62.59 %
Regresión logística	83.87 %	73.82 %	79.05 %	57.14 %
SVM lineal	87.10 %	74.68 %	78.38 %	53.74 %
SVM radial	80.65 %	73.82 %	75.68 %	60.54 %
MLP	87.10 %	75.54 %	78.38 %	57.82 %
Clustering	84.28 %	70.24 %	69.09 %	52.18 %

Tabla 4.18: Parámetros empleados en el entrenamiento de los modelos de la iteración 2

Antes de comenzar con el análisis de los resultados, caben destacar una serie de consideraciones sobre los mismos:

- Para K-NN se calculan los parámetros óptimos previamente, tanto para el algoritmo K-NN propiamente, como para el mismo con la introducción de *K-Fold*.

- Para el clustering se toman los dos atributos que mayor información aportan y se utilizan dos *clústers*. La decisión del número de *clústers* que se usa se determina a partir del método del codo [63]. En todas las situaciones, las opciones han sido escoger entre 2, 3 y 4 *clústers*. La decisión final se ha tomado para que haya el mismo número de *clústers* que de clases en las que clasificar los datos.

- En todos los algoritmos que hacen uso de árboles de decisión, el resultado de train es 100 % de *accuracy* siempre. El motivo es que se va dividiendo el espacio hasta que todos los datos están clasificados en su grupo correctamente. En la Tabla 4.18 solo se muestran los datos de test, por esto no aparece el valor de 100 % en el *accuracy*.

Con esto se pretende remarcar que los resultados obtenidos para cada uno de los algoritmos recogidos en la Tabla 4.18 son óptimos; es decir, estos resultados son los que se han alcanzado utilizando los parámetros que maximizan el rendimiento. Si bien es cierto que también se pueden contemplar otras opciones para tratar de mejorarlo, como escalar los datos. Para el caso contemplado, se hicieron pruebas y no mejoraron los resultados; luego no había motivo para escoger utilizar alguna de estas técnicas.

Ahora sí se está en disposición de comenzar con el análisis de los resultados. Como es de esperar, los mejores resultados obtenidos para todos los algoritmos contemplados son los que se corresponden al conjunto de datos que extrae las medidas a partir de las máscaras originales. El resto de *datasets* parte de características fruto de la segmentación. En consecuencia, se introduce un error que afecta al rendimiento de los algoritmos. Otro detalle a destacar es que, como ocurre con el algoritmo de Bernoulli-Naive Bayes, los clasificadores construidos no superan el 50 %. Esto implica que este algoritmo no supera el rendimiento que tendría un clasificador aleatorio.

Además, como las métricas de clasificación son inferiores para esta iteración que para la primera, no se analizarán en mayor profundidad los modelos, pues el interés reside en obtener el modelo que mejor resultados aporte. Luego, no corresponde realizar un análisis exhaustivo en esta ocasión.

Para finalizar con esta segunda iteración, cabe mencionar que los resultados obtenidos para la clasificación entre sistema ocular sano y con glaucoma son mejores si se emplean técnicas de *Deep Learning* y no de *Machine Learning*. El motivo de esta afirmación es que se han obtenido mejores resultados para la primera iteración, en la que se usó *Deep Learning*, que en la segunda iteración con *Machine Learning*. Por consiguiente, se abandona la vía de clasificar el estado del ojo en lo referido al glaucoma mediante técnicas de *Machine Learning*. En cada iteración que se haga a continuación, todos los modelos se desarrollarán con técnicas de *Deep Learning*. Además, como estas últimas contemplan una mayor cantidad de datos, es de esperar que arrojen mejores resultados que los modelos de *Machine Learning* a los que se proporciona una cantidad baja de atributos.

4.5. Iteración 3. Clasificación Deep Learning sin preentrenar a partir de segmentación

En las dos primeras iteraciones, descritas en las secciones 4.3 y 4.4, se han resuelto los dos objetivos principales de este proyecto: el de clasificación y el de segmentación del disco y la copa ópticos. Sin embargo, en base a los antecedentes médicos recopilados en la Sección 3.1 surge una cuestión. Si la enfermedad afecta al nervio óptico deteriorando la visión, la información que permita decidir si el paciente sufre de glaucoma debe localizarse en esa región.

De esta forma, parece natural emplear la segmentación realizada para extraer la parte del nervio óptico de la retinografía completa y atacar el problema de clasificación desde el mismo conjunto de datos, pero solo teniendo en cuenta la parte segmentada. Así, se puede tratar de probar que la región del nervio óptico es la que posee una mayor cantidad de información sobre la patología del glaucoma.

En esta sección se explicará el proceso para la construcción de un *dataset* con imágenes que solo contengan la región del nervio óptico, y las diferentes variantes que se pueden tomar en la forma de los datos. Tras esto, se especificarán los hiperparámetros utilizados para la construcción de los modelos, para terminar con un análisis de los resultados obtenidos en esta iteración.

4.5.1. Preparación de los datos

Con el objetivo descrito para esta iteración de obtener una clasificación entre ojos sanos y con glaucoma a partir de la región del nervio óptico en las retinografías, se debe preparar un conjunto de datos adecuado. Para ello, a partir del dataset denominado como *rotterdam* en la Sección 4.3, se extrae la región de interés.

A la vista de los resultados descritos en la Sección 4.4 durante la segunda iteración, ambos modelos son útiles para localizar la región del disco óptico. Sin embargo, el modelo desarrollado a partir de YOLO presenta una clara ventaja, pues no solo segmenta la región buscada, sino que también la localiza dentro de una caja. De esta forma, se ha procesado el *dataset rotterdam* para quedarse con las cajas en torno al nervio óptico, y construir el nuevo conjunto de datos siguiendo el siguiente proceso:

1. Se abre una de las carpetas del *dataset rotterdam*.

2. Se procesan sus archivos recortando la parte predicha por el modelo de segmentación *yolo_disc*. En caso de que se localicen dos zonas, se toma la que mayor porcentaje de confianza aporta, pues solo puede existir una única zona para el nervio óptico.

3. Se almacenan los recortes obtenidos a partir de la carpeta correspondiente del *dataset Rotterdam* en una carpeta igual dentro del nuevo *dataset rotterdam_RIM-ONE*.

4. Se repite el proceso con cada uno de los subdirectorios que contienen retinografías del *dataset rotterdam*.

Queda añadir un pequeño detalle. Para asegurar que la imagen recortada de la retinografía abarca la región del nervio óptico al completo, se amplia un 5 % los márgenes del cuadro obtenido a partir del modelo *yolo_disc* utilizado. Así, partiendo del proceso descrito, se construye el dataset denominado como *rotterdam_RIM-ONE*.

Por otra parte, se pueden considerar una serie de variaciones de este dataset para estudiar un abanico más amplio de posibilidades. En esta iteración, se ha tomado el mismo conjunto de datos considerando una sola capa de escala de grises. De igual manera que se ha procesado el *dataset rotterdam_RIM-ONE* construido para extraer cada capa, se opera para obtener el nuevo conjunto de datos con las mismas regiones del nervio óptico en escala de grises. Este último *dataset* se ha denominado *rotterdam_RIM-ONE_grises*.

4.5.2. Entrenamiento

En la etapa de entrenamiento se procede a construir los modelos que se pretenden que clasifiquen retinografías entre las de pacientes sanos y los que presenten glaucoma analizando únicamente la región del nervio óptico. Como se ha indicado previamente, en esta tercera iteración se consideran dos modelos: uno orientado a clasificación con la parte del nervio óptico de las retinografías a color, y otro análogo con retinografías en escala de grises.

Durante esta fase se ajustan los parámetros internos de los modelos de *Deep Learning* a partir de los datos disponibles siguiendo la teoría explicada en la Sección 3.2.3. Para ello, se indican una serie de hiperparámetros que determinan algunos procesos que se llevan a cabo en el proceso de entrenamiento. En concreto, se indica el número de épocas que se debe entrenar el modelo; la paciencia, para evitar el sobreajuste; o el tamaño de entrada de las imágenes. Además, se indica el modelo base sobre el que se entrena para no tener que construir la red desde cero. En la Tabla 4.19, se detallan los hiperparámetros empleados durante el entrenamiento de cada uno de los dos modelos. Para seleccionarlos, se han tenido en cuenta aspectos como el coste computacional del proceso.

Como se observa en la Tabla 4.19, se ha empleado para realizar el entrenamiento el sistema YOLO de igual forma que se hizo en la primera iteración, importándolo desde la biblioteca *ultralytics*. Como también se destacó en la primera iteración, la clasificación de imágenes siempre se hace a lo largo de este proyecto a partir de modelos YOLO; puesto que, dado el gran número de imágenes a procesar, se utiliza un algoritmo que opera con rapidez.

4.5.3. Evaluación

Una vez entrenados los modelos, han sido evaluados utilizando el subconjunto de los datos que se reservó con este propósito. Estos datos residen en el directorio de test de cada *dataset*, que es el mismo en ambos casos con la distinción entre color y escala de grises en función del conjunto de datos. Para cuantificar la capacidad de clasificación de cada modelo adecuadamente, se ha empleado la métrica *accuracy* definida en la Sección 3.2.4.

Modelo	Descripción	Hiperparámetros	
rotterdam RIM-ONE	Nervio óptico de rotterdam a color	Algoritmo	YOLO
		Modelo base	YOLO 11
		Dataset	rotterdam_RIM-ONE
		Épocas	100
		Paciencia	15
		Tamaño imagen	256
rotterdam RIM-ONE grises	Nervio óptico de rotterdam en escala de grises	Algoritmo	YOLO
		Modelo base	YOLO 11
		Dataset	rotterdam_RIM-ONE_grises
		Épocas	100
		Paciencia	15
		Tamaño imagen	256

Tabla 4.19: Parámetros empleados en el entrenamiento de los modelos de la iteración 3

En la Tabla 4.20 se resumen los resultados obtenidos tanto para el conjunto de datos *val* como para el de *test*.

Modelo	Épocas	*Accuracy* validación	*Accuracy* test
rotterdam_RIM-ONE	68	0.948	0.947
rotterdam_RIM-ONE_grises	34	0.928	0.921

Tabla 4.20: Resultados obtenidos en la iteración 3 tras el entrenamiento

A partir de los resultados obtenidos en la Tabla 4.20 se puede comparar de manera cuantitativa el rendimiento de los modelos considerados, identificando el que mejor comportamiento presenta en términos de la métrica *accuracy*. Atendiendo a los mismos, se observan mejores resultados para el modelo que recibe como datos de entrada la región del nervio óptico a color en vez del modelo en escala de grises, tanto en el conjunto de validación como en el de test. Una vez más, se remarca que el conjunto de test es el de mayor relevancia, pues los datos de validación han sido empleados para ajustar los parámetros durante la etapa de entrenamiento y no representan, con la misma fidelidad que lo hace el conjunto de datos de test, el rendimiento que pueda tener el modelo en la realidad.

De esta forma, para la métrica *accuracy*, el modelo *rotterdam_RIM-ONE* que utiliza la región del nervio óptico a color alcanza un mejor rendimiento que el modelo *rotterdam_RIM-ONE_grises* con la región del nervio óptico en escala de grises. En concreto,

para el primero de ellos se clasifica un 94.7 % de las imágenes de manera correcta; mientras que para el segundo, se clasifica un 92.1 % de las retinografías correctamente.

Aterrizando estos porcentajes, para el modelo *rotterdam_RIM-ONE* se clasifican 53 de cada 1000 retinografías de manera incorrecta. Por su parte, para el modelo *rotterdam_RIM-ONE_grises*, se clasifican erróneamente 79 de cada 1000 retinografías. En consecuencia, el modelo en escala de grises clasifica en torno a 25 retinografías más de manera errónea que el modelo a color. Esto también sucedió para la iteración 1, donde se obtenían mejores resultados para unas imágenes a color que para las mismas en escala de grises. Además, igual que se razonó en ese momento, esto puede deberse a la propia construcción de las imágenes en cada caso. Frente a los tres canales RGB que componen la imagen a color, para la imagen en escala de grises tan solo se tiene uno, lo que supone una disminución en la cantidad de información que aporta cada retinografía en términos computacionales.

Queda una comparación de gran relevancia por hacer. La motivación de esta tercera iteración es poder comparar si los resultados mejoran al contemplar únicamente la región del nervio óptico. En otras palabras, el objetivo de esta iteración es comprobar la hipótesis de que la región del nervio óptico es la que aporta mayor información en una retinografía para la detección del glaucoma. Por este motivo, en la Tabla 4.21 se recogen los resultados para la métrica *accuracy* para el conjunto de test obtenidos con cada uno de los modelos construidos para clasificación mediante *Deep Learning* durante las iteraciones primera y tercera.

Iteración	Modelo	*Accuracy* test
Iteración 1	rotterdam	0.933
	rotterdam_grises	0.912
Iteración 3	rotterdam_RIM-ONE	0.947
	rotterdam_RIM-ONE_grises	0.921

Tabla 4.21: Resumen de la métrica *accuracy* para los modelos de las iteraciones 1 y 3

Para poder realizar una comparación objetiva, se divide la Tabla 4.21 en dos partes. Una primera donde se recogen los modelos a color, y una segunda para los modelos en escala de grises. Esto se considera así, dado que no es comparable para aprobar o rechazar la hipótesis de que el análisis del nervio óptico aporta mayor cantidad de información, tomar un modelo con datos a color con uno en escala de grises. El motivo es que si se detecta una pérdida o aumento de información, puede deberse a otras causas como la diferencia de capas de la imagen. En consecuencia, se construyen las tablas 4.22 y 4.23 para comparar los resultados de manera adecuada.

En la Tabla 4.22 se presentan los valores de la métrica *accuracy* para los modelos a color durante las iteraciones 1 y 3. Como se puede comprobar, para el modelo que contempla el área circundante al nervio óptico, se obtiene un valor superior al del modelo que emplea la retinografía completa para la detección del glaucoma. Concretamente, la métrica *accuracy* es un 1.4 % superior cuando el modelo se enfoca de manera exclusiva en

Iteración	Modelo	*Accuracy* test
Iteración 1	rotterdam	0.933
Iteración 3	rotterdam_RIM-ONE	0.947

Tabla 4.22: Comparación *accuracy* para los modelos a color de las iteraciones 1 y 3

la región del nervio óptico. En términos de clasificaciones correctas, esta mejora implica que de cada 1000 imágenes hay 14 más que se clasifican correctamente para el modelo *rotterdam_RIM-ONE*, el de la iteración 3.

Iteración	Modelo	*Accuracy* test
Iteración 1	rotterdam_grises	0.912
Iteración 3	rotterdam_RIM-ONE_grises	0.921

Tabla 4.23: Comparación *accuracy* de los modelos en escala de grises de las iteraciones 1 y 3

Por su parte, en la Tabla 4.23 se presentan los valores de la métrica *accuracy* para los modelos en escala de grises durante las iteraciones 1 y 3. Como se puede comprobar, para el modelo que contempla el área circundante al nervio óptico, se obtiene un valor superior al del modelo que emplea la retinografía completa para la detección del glaucoma. Concretamente, la métrica *accuracy* es un 0.9 % superior cuando el modelo se enfoca de manera exclusiva en la región del nervio óptico. En términos de clasificaciones correctas, esta mejora implica que de cada 1000 imágenes hay 9 más que se clasifican correctamente para el modelo *rotterdam_RIM-ONE_grises*, el de la iteración 3.

Tanto con los modelos a color como con los modelos en escala de grises, se obtiene un mejor resultado si se toma solo la región del nervio óptico en lugar de la retinografía completa. De esta forma, se puede concluir que la hipótesis que se pretendía demostrar en esta iteración es correcta. En consecuencia, para determinar la presencia de glaucoma será suficiente - de hecho mejor - tomar el área del nervio óptico en las retinografías.

4.6. Iteración 4. Clasificación Deep Learning preentrenando a partir de segmentación

En las iteraciones realizadas hasta el momento, se han resuelto los dos objetivos principales de este proyecto y se ha probado la hipótesis de que la región del nervio óptico es la que mayor información aporta para la detección del glaucoma. En esta iteración se plantea una nueva cuestión. Se pretende comprobar si se mejora el resultado cuando se preentrena el modelo con otro conjunto de datos para realizar un ajuste previo de los parámetros de la red neuronal sobre la que se fundamenta.

En esta sección se explicará el proceso que se ha llevado a cabo para manipular el tercero de los *datasets* contemplados, el de RIM-ONE. Además, los modelos se entrenarán

teniendo en cuenta que se obtienen mejores resultados recortando una parte de las retinografías. A colación de lo anterior, se contemplará una mayor variedad de los modelos para poder determinar cuál es la mejor solución al problema abordado.

4.6.1. Preparación de los datos

En primer lugar, se ha de llevar a cabo las manipulaciones oportunas en el *dataset RIM-ONE* para usarlo durante el preentrenamiento de los modelos. Análogamente a lo que se describió en la Sección 4.3 para la iteración 1 con el *dataset rotterdam*, se dividen los datos en los mismos subconjuntos y con la misma fracción de imágenes en cada uno de ellos. A continuación se describe esta situación:

- Conjunto de entrenamiento [*train*]. Para este conjunto se han reservado un 64 % de cada una de las clases de imágenes que se tienen.
- Conjunto de validación [*val*]. De igual forma que con el *dataset rotterdam*, se ha tomado un 16 % del total de imágenes con el nervio óptico propio del glaucoma y un 16 % de las que muestran la región ONH de un ojo sano.
- Conjunto de test [*test*]. En total se tendrá un 20 % de cada clase de imágenes para probar el modelo construido.

Además, para la primera iteración también se consideró el mismo conjunto de datos pero con las imágenes en escala de grises. En esta iteración se ha ido más allá. Como se han validado las hipótesis que se querían probar previamente, se crea una mayor variedad en la forma de los datos para generar más modelos y tener un abanico amplio para elegir el que mejor rendimiento aporte. En consecuencia, se ha considerado la misma distribución para los datos de *RIM-ONE* con distintas variaciones: un conjunto de datos en escala de grises y otro para cada uno de los canales de la base RGB. Así, para esta iteración se tienen los siguientes *datasets* de preentrenamiento: *RIM-ONE*, *RIM-ONE_grises*, *RIM-ONE_rojo*, *RIM-ONE_verde* y *RIM-ONE_azul*.

Por otra parte, los *datasets* de los que se parte para la segunda fase del entrenamiento son los mismos que los descritos en la Sección 4.5 para la iteración 3. Además, se consideran una serie de variaciones de estos *datasets* para estudiar un abanico más amplio de posibilidades, que son los mismos que para el conjunto de datos *RIM-ONE*. Aquí también se toman *datasets* en función de las capas RGB y escala de grises.

Las imágenes recortadas a partir de las retinografías completas siguen teniendo tres capas de color en la base RGB. Se ha considerado construir un *dataset* con cada una de las capas de manera independiente. Esto permite estudiar qué capas reportan una mayor cantidad de información. Así, se han construido los siguientes *datasets* a partir del *dataset rotterdam_RIM-ONE*: *rotterdam_RIM-ONE_rojo* para la capa roja; *rotterdam_RIM-ONE_verde* para la capa verde; y *rotterdam_RIM-ONE_azul* para la capa azul. Estos conjuntos de datos se añaden a los que se tenían previamente, que eran *rotterdam* y *rotterdam_grises*.

4.6.2. Entrenamiento

Como ya se ha mencionado, en esta iteración el entrenamiento tendrá dos fases bien diferenciadas. Por una parte, se preentrenarán los modelos con cada uno de los *datasets* construidos a partir del conjunto de datos de *RIM-ONE*. Una vez terminado este proceso, se llevará a cabo un segundo entrenamiento con el conjunto de datos de *rotterdam* correspondiente; es decir, el que presente el mismo estilo que con el que se ha preentrenado para *RIM-ONE*.

En las tablas 4.24 y 4.25 se muestran los hiperparámetros que se han usado para entrenar en la primera y segunda etapa, respectivamente. Como se observa en las tablas mencionadas, los modelos de *RIM-ONE* se entrenan a partir de los modelos base de YOLO, y los de *rotterdam* se construyen sobre los propios de *RIM-ONE*. Además, el tamaño de imagen de la entrada es igual en ambos casos, pues si no, el preentrenamiento no tendría la misma utilidad.

Modelo	Descripción	Hiperparámetros	
RIM-ONE	RIM-ONE a color	Algoritmo	YOLO
		Modelo base	YOLO 11
		Dataset	RIM-ONE
		Épocas	100
		Paciencia	15
		Tamaño imagen	256
RIM-ONE grises	RIM-ONE en escala de grises	Algoritmo	YOLO
		Modelo base	YOLO 11
		Dataset	RIM-ONE_grises
		Épocas	100
		Paciencia	15
		Tamaño imagen	256
RIM-ONE rojo	RIM-ONE capa roja de RGB	Algoritmo	YOLO
		Modelo base	YOLO 11
		Dataset	RIM-ONE_rojo
		Épocas	100
		Paciencia	15
		Tamaño imagen	256
RIM-ONE verde	RIM-ONE capa verde de RGB	Algoritmo	YOLO
		Modelo base	YOLO 11
		Dataset	RIM-ONE_verde
		Épocas	100
		Paciencia	15
		Tamaño imagen	256
RIM-ONE azul	RIM-ONE capa azul de RGB	Algoritmo	YOLO
		Modelo base	YOLO 11
		Dataset	RIM-ONE_azul
		Épocas	100
		Paciencia	15
		Tamaño imagen	256

Tabla 4.24: Parámetros empleados en el preentrenamiento de los modelos de la iteración 4

Modelo	Descripción	Hiperparámetros	
rotterdam RIM-ONE preentrenado	Nervio óptico de rotterdam a color	Algoritmo	YOLO
		Modelo base	RIM-ONE
		Dataset	rotterdam_RIM-ONE _preentrenado
		Épocas	100
		Paciencia	15
		Tamaño imagen	256
rotterdam RIM-ONE grises preentrenado	Nervio óptico de rotterdam en escala de grises	Algoritmo	YOLO
		Modelo base	RIM-ONE_grises
		Dataset	rotterdam_RIM-ONE _grises_preentrenado
		Épocas	120
		Paciencia	20
		Tamaño imagen	256
rotterdam RIM-ONE rojo	Nervio óptico de rotterdam capa roja	Algoritmo	YOLO
		Modelo base	RIM-ONE_rojo
		Dataset	rotterdam_RIM-ONE_rojo
		Épocas	120
		Paciencia	20
		Tamaño imagen	256
rotterdam RIM-ONE verde	Nervio óptico de rotterdam capa verde	Algoritmo	YOLO
		Modelo base	RIM-ONE_verde
		Dataset	rotterdam_RIM-ONE_verde
		Épocas	100
		Paciencia	15
		Tamaño imagen	256
rotterdam RIM-ONE azul	Nervio óptico de rotterdam capa azul	Algoritmo	YOLO
		Modelo base	RIM-ONE_azul
		Dataset	rotterdam_RIM-ONE_azul
		Épocas	100
		Paciencia	15
		Tamaño imagen	256

Tabla 4.25: Parámetros empleados en el entrenamiento de los modelos de la iteración 4

4.6.3. Evaluación

En las tablas 4.26 y 4.27 se muestran los resultados de cada una de las etapas de los entrenamientos.

Modelo	Épocas	*Accuracy* validación	*Accuracy* test
RIM-ONE	39	0.922	0.888
RIM-ONE_grises	29	0.922	1
RIM-ONE_rojo	46	0.857	0.816
RIM-ONE_verde	42	0.909	0.867
RIM-ONE_azul	32	0.935	0.897

Tabla 4.26: Resultados obtenidos en la iteración 4 tras el entrenamiento

Para los *datasets* construidos a partir de *RIM-ONE*, vemos que no se obtienen valores tan altos como para los de *rotterdam* en anteriores iteraciones respecto a la clasificación de los datos de prueba, con la excepción del caso de escala de grises. La principal causa que puede tener esto es la baja cantidad de datos con los que se entrena. No obstante, esto no supone un problema, pues el propósito aquí es establecer unos parámetros que se aproximen a los de la solución buscada.

Por otra parte, se tienen los resultados para la segunda etapa del entrenamiento en la Tabla 4.27. En esta fase, como era de esperar por la cantidad de datos con la que se entrena, se mejoran significativamente los datos respecto a los de los modelos de la primera etapa de entrenamiento. Además, la parte interesante de esta iteración es comparar los modelos resultado de la misma con los de la iteración 3.

Modelo	Épocas	*Accuracy* validación	*Accuracy* test
rotterdam RIM-ONE color preentrenado	52	0.94	0.943
rotterdam RIM-ONE grises preentrenado	105	0.949	0.937
rotterdam RIM-ONE rojo	101	0.92	0.912
rotterdam RIM-ONE verde	57	0.934	0.939
rotterdam RIM-ONE azul	32	0.898	0.897

Tabla 4.27: Resultados obtenidos en la iteración 4 tras el entrenamiento

Ya se ha visto que los modelos de la tercera iteración mejoran los de las iteraciones anteriores al fijar la atención en la región del nervio óptico y utilizar *Deep Learning*. Por

tanto, queda comparar si los modelos preentrenados aportan mejores resultados que en el caso de que no se haga este proceso al inicio. En la Tabla 4.28 se comparan los resultados en los conjuntos de prueba para los modelos a color con la métrica *accuracy*.

Modelo	*Accuracy* test
rotterdam RIM-ONE color	0.947
rotterdam RIM-ONE color preentrenado	0.943

Tabla 4.28: Resultados obtenidos en la iteración 4 tras el entrenamiento

En la Tabla 4.28, se puede apreciar que el preentrenamiento no tiene siempre un resultado positivo. En este caso, preentrenar el modelo implica una reducción de un 0.4 % de *accuracy*. Que en términos absolutos, de las 1908 imágenes que se tienen para test, implica errar en el diagnóstico de 8 casos más. Queda ver qué ocurre con los modelos que trabajan con imágenes en escala de grises. Los resultados relativos a los mismos se presentan en la Tabla 4.28 en base a la métrica *accuracy*.

Modelo	*Accuracy* test
rotterdam RIM-ONE_grises	0.921
rotterdam RIM-ONE grises preentrenado	0.937

Tabla 4.29: Resultados obtenidos en la iteración 4 tras el entrenamiento

En la Tabla 4.29, se observa que en este caso el preentrenamiento ha tenido un resultado favorable. En este caso, preentrenar el modelo aumenta significativamente el *accuracy* del modelo, incrementando el mismo hasta en un 1.6 %. Analizando igual que con los modelos a color en términos absolutos, de las 1908 imágenes que se tienen para test, implica fallar en el diagnóstico de 30 casos más.

Como conclusión para esta cuarta iteración, se han construido modelos con un rendimiento muy alto y se han mejorado algunos de los que ya se tenían. Este es el caso del modelo en escala de grises, o del modelo del canal de color verde, que ha resultado incluso en un incremento del rendimiento del de escala de grises, aunque la diferencia se hace menor con el debido preentrenamiento. Sin embargo, aunque es habitual que con un preentrenamiento de la red neuronal se obtengan mejores resultados puesto que los parámetros internos son más cercanos a los óptimos, esto no siempre ocurre como se ha podido comprobar con los *datasets* a color.

4.7. Iteración 5. Construcción de ensembles

Como se ha explicado en la Sección 3 de antecedentes, la construcción de ensembles es una técnica que combina múltiples modelos individuales para crear un modelo más robusto y preciso. La ventaja que presenta la adición de esta estrategia es que aprovecha los puntos fuertes de cada modelo individual y compensa las debilidades tomando decisiones en común. La estructura de los ensembles que se contemplan en esta propuesta queda representada por el diagrama de la Figura 4.17.

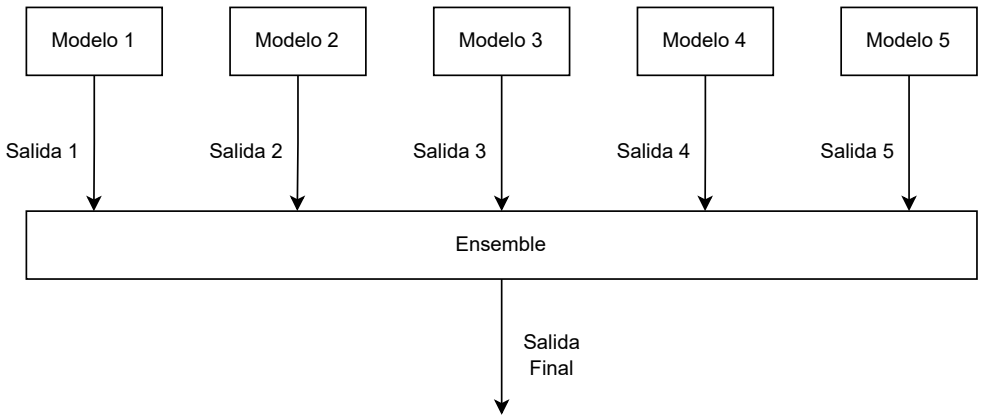


Figura 4.17: Diagrama ilustrativo entradas y salidas del ensemble construido.

4.7.1. Ensemble por votación

En una primera aproximación a la solución mediante la construcción de ensembles, se ha analizado la elaboración de un ensemble por medio de votación mayoritaria. Este ensemble es de los más sencillos que se pueden construir. Consiste en que cada modelo que lo compone predice una clase, y la clase más votada es la que se elige como predicción final.

De esta forma, se han escogido tres modelos para comprobar el funcionamiento de este tipo de ensembles. En particular, los modelos con los que se han hecho la prueba son *rotterdam_RIM-ONE_preentrenado*, *rotterdam_RIM-ONE_verde* y *rotterdam_RIM-ONE_azul*. A partir de los mismos, se han obtenido los resultados recogidos en la Tabla 4.30, donde se han utilizado los datos de *test* del *dataset rotterdam*.

A partir de los resultados mostrados en la Tabla 4.30 se pueden calcular las métricas que cuantifican el rendimiento del modelo para el propósito que se persigue. Estas se muestran en la Tabla 4.31.

Para una segunda versión del ensemble por votación, se podrían elegir como modelos base los 5 modelos que mejor resultados de clasificación han aportado para el conjunto de datos de test, que representan a los modelos del 1 al 5 en el diagrama de la Figura 4.17, y se elige como predicción final la que más modelos base voten.

	Número de fallos	Número total de imágenes	Porcentaje fallos	Porcentaje aciertos
Glaucoma	41	948	4.32 %	95.68 %
Normal	80	955	8.37 %	91.63 %

Tabla 4.30: Resultados obtenidos para la prueba del ensemble por votación construido

	Accuracy	fpr	Precisión	*Recall*	F1
Glaucoma	0.9364	0.0429	0.9189	0.9567	0.9374
Normal	0.9389	0.0810	0.9227	0.9588	0.9404

Tabla 4.31: Métricas ensemble de prueba por votación

Esta es una primera propuesta para la construcción de ensembles por votación que ayuden a resolver el problema. Versiones más sofisticadas pueden incluir sistemas como, por ejemplo, votación ponderada. El análisis de esta variante queda fuera del alcance del proyecto, pero puede ser interesante, pues algún modelo puede aportar una mayor fiabilidad, favoreciendo así más a los modelos con mayor tasa de aciertos.

Dada la facilidad de la elaboración de los ensembles con votación, estos son muy útiles para ejemplificar cómo funciona la combinación de modelos para construir un ensemble que pueda mejorar los resultados. Esta posibilidad es muy básica y existen otras formas de construcción de ensembles mejores como la que se verá posteriormente. Esto se debe a que la votación, ya sea ponderada o no, puede aportar robustez dado que se consideran predicciones de distintos modelos; sin embargo, por la propia naturaleza del método, los resultados serán una media o media ponderada, según corresponda, de los obtenidos para cada modelo. A continuación se explica cómo se ha abordado esta situación para este proyecto: mediante técnicas de *Machine Learning*.

4.7.2. Ensemble mediante algoritmos de Machine Learning

Frente al enfoque mediante votación, cabe la posibilidad del uso de técnicas más avanzadas. En particular, es posible el desarrollo de algoritmos de Machine Learning para la construcción de un ensemble. Esta es la otra posibilidad que se ha contemplado en la elaboración del ensemble que combina los datos de clasificación para producir uno solo.

Preparación de los datos

Para construir el conjunto de datos con los que se construirá el ensemble, se parte de los 5 modelos que mejores resultados de clasificación han aportado para el conjunto de datos de test durante las 4 iteraciones descritas anteriormente, de igual forma que se hace para el ensemble por votación de la Sección 4.7.1. Estos modelos representan a los nombrados del 1 al 5 en el diagrama de la Figura 4.17, y son *rot-*

terdam, *rotterdam_RIM-ONE*, *rotterdam_RIM-ONE_grises_preentrenado*, *rotterdam_RIM-ONE_rojo* y *rotterdam_RIM-ONE_verde*. Nótese que *rotterdam_RIM-ONE_preentrenado* aporta un mejor rendimiento en la clasificación de imágenes que algunos de los escogidos, pero se ha descartado puesto que se considera el mismo modelo sin preentrenar, y se pretende tomar opciones distintas para mejorar la generalización del modelo.

La construcción del conjunto completo de datos consta de dos partes análogas la una a la otra. En primer lugar, se crea un archivo tipo csv donde se almacena para cada retinografía del conjunto de datos de validación del *dataset rotterdam*, la clasificación obtenida por cada uno de los modelos y su clasificación real, ambos en forma binaria; es decir, con 0 para representar el glaucoma y 1 para el normal. Para cada modelo, el dato de entrada será la retinografía completa o la región del nervio óptico con la modificación en cuanto a las capas de la imagen que corresponda. En un segundo paso, se opera de igual forma para procesar las clasificaciones de las imágenes de la parte del *dataset rotterdam* orientada a test. Así, se crea un nuevo archivo CSV con las mismas características que el creado con la parte del *dataset rotterdam* de validación.

En resumen, los datos de entrada para esta etapa, en la que se busca construir un ensemble utilizando técnicas de Machine Learning, constan de dos archivos csv con las clasificaciones de las retinografías obtenidas con los mejores modelos construidos durante las 4 primeras iteraciones, junto con la clasificación real de las mismas imágenes. Estos archivos csv tienen la forma que se muestra en la Tabla 4.32.

completa_color	onh_color	onh_gris	onh_verde	onh_rojo	clasif_real
0	0	0	0	0	0
0	1	0	0	0	0
⋮	⋮	⋮	⋮	⋮	⋮
0	0	1	1	1	1
1	1	1	0	1	1
⋮	⋮	⋮	⋮	⋮	⋮

Tabla 4.32: Formato csv empleado para construir el ensemble

Por último, se analiza el número de datos referidos a retinografías con glaucoma y normales que se tienen para cada uno de los dos csv, tanto el de validación como el de test. En la Figura 4.18 se recogen los datos que se tienen para el csv construido con el subconjunto *val*; y, por su parte, en la Figura 4.19 los del subconjunto test.

En la Figura 4.18 se observa que los datos están perfectamente balanceados y se tienen más de 750 ejemplos de cada una de las clases, lo que será suficiente para poder entrenar el modelo. Por su parte, en la Figura 4.19 se observa que los datos también están balanceados y se tienen del orden de 950 ejemplos de cada una de las clases. Estos datos serán usados para tratar de aproximar el rendimiento que presente el modelo. A colación de lo anterior, dado el gran número de datos de prueba que se tienen, los resultados obtenidos serán

suficientemente representativos. pues los datos están balanceados y se tienen un total de más de 1900 ejemplos para test.

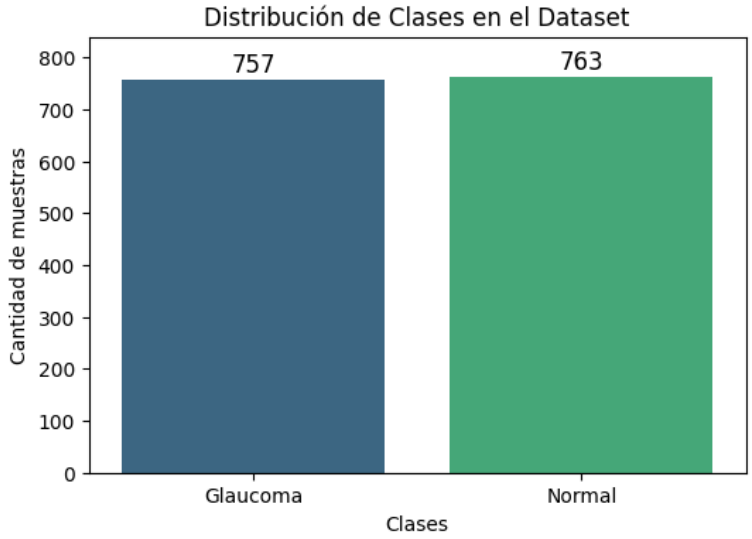


Figura 4.18: Número de datos que se tiene para cada clase en el csv de validación.

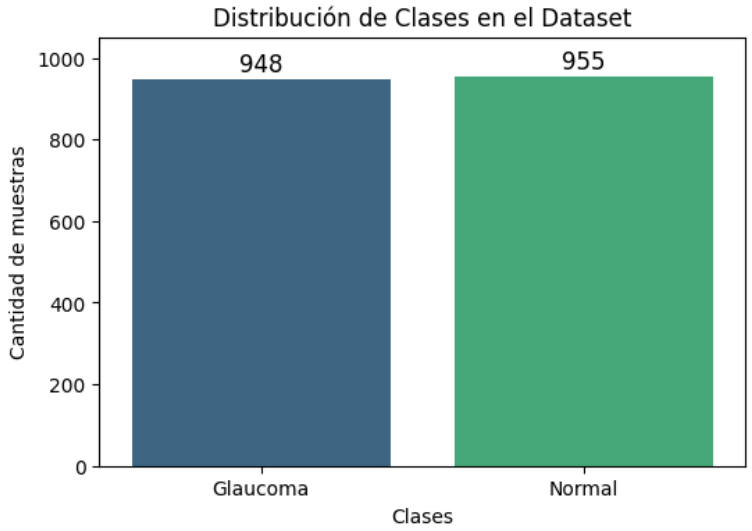


Figura 4.19: Número de datos que se tiene para cada clase en el csv de test.

Entrenamiento

Como se acaba de introducir el describir los datos utilizados, se utilizará el CSV construido a partir de los datos de validación para entrenar el modelo de Machine Learning desarrollado; mientras que, para el CSV elaborado con los datos orientados a *test*, se mantendrá su propósito.

El motivo de introducir los datos de validación para entrenar, y no los dispuestos en el subconjunto de *train* para este objetivo, se debe a que los datos ya han sido utilizados para entrenar los modelos previos. El uso de los mismos datos para entrenar tanto el ensemble como los modelos sobre los que se sustenta, tiene el inconveniente del sobreajuste y la falta de generalización. Si los modelos y el ensemble se entrenan con los mismos datos, el ensemble puede aprender a replicar el comportamiento de los modelos sobre ese conjunto, incluyendo sus errores, sin realmente aprender a generalizar.

En la construcción del ensemble no se ha contemplado un único algoritmo de *Machine Learning*, sino un conjunto de ellos para poder escoger el que mejor resultados arroje. En particular, se ha utilizado SVM lineal, radial, polinomial y sigmoidea; además de MLP. Para la construcción de los modelos a partir de estos algoritmos, se dividen los datos de validación entre los atributos y el resultado esperado. Todos estos datos se usarán para entrenar. Por otra parte, para los datos que se usarán para probar el modelo construido, se opera de igual forma estableciendo la misma división de los datos de test.

Evaluación de los resultados

En primer lugar, se van a comparar los modelos obtenidos a partir de la métrica *accuracy*. Esta mostrará el porcentaje de clasificaciones correctas tanto para las detecciones de glaucoma como para las de pacientes sanos. La recopilación de esta métrica para cada uno de los modelos entrenados se muestra en la Tabla 4.33. Como ocurre con todos los modelos construidos mediante algoritmos propios del *Machine Learning*, los datos con los que se entrena tendrán un *accuracy* superior al de los datos de test.

Modelo	*Accuracy train*	*Accuracy* test
SVM lineal	95.46 %	94.85 %
SVM radial	95.59 %	94.01 %
SVM polinomial	95.13 %	94.96 %
SVM sigmoideo	92.11 %	91.28 %
MLP	95.72 %	94.75 %

Tabla 4.33: Resultados accuracy de los ensembles entrenados mediante Machine Learning

En la Tabla 4.33 se observa que todos los modelos superan el 90 % de *accuracy*. Esto es de esperar, pues todos los modelos previos en los que se sustenta el ensemble superan el 90 %. En caso contrario, el ensemble estaría empeorando el rendimiento de las predicciones. Por otra parte, para comparar fielmente los resultados, debe fijarse la vista en los

resultados para el conjunto de prueba. Además, cabe mencionar que, como el *accuracy* en entrenamiento y test con muy similares en todos los casos, los ensembles aquí construidos no presentan sobreajuste. Por tanto, estos ensembles generalizan bien y mantienen el rendimiento para nuevos datos de prueba.

En la columna *accuracy test*, se observa que para el modelo SVM sigmoidal no se alcanza ni siquiera un 92 % de acierto, siendo el peor de los ensembles construidos. Siguiendo esta tónica, otro ensemble que no corresponde usar es el SVM radial, pues es el otro que peor rendimiento tiene en cuanto a nivel de aciertos totales. Continuando con el análisis, se comprueba que el SVM polinomial se adapta mejor a los datos que el SVM lineal, como es de esperar. Dado que una recta es un caso particular de polinomio, el algoritmo SVM lineal tiene menor alcance que el SVM polinomial. Además, estos dos son los que mejor *accuracy* tienen, con un 94.85 % y un 94.96 %, respectivamente; es decir, son los que clasifican más casos correctamente. Otra opción a contemplar como ensemble es el perceptrón multicapa (MLP), pues les sigue muy de cerca con un 94.75 % de casos totales acertados.

Por otra parte, se van a ir analizando por separado cada una de las métricas, explicadas en la Sección 3.2.4, que se tienen de todos los ensembles construidos. Finalmente, se dará una conclusión de cuál es el ensemble a utilizar.

- **SVM lineal**. Vamos a analizar las métricas obtenidas con este ensemble recogidas en la Tabla 4.34. Estas métricas se obtienen a partir de los datos de la matriz de confusión de la Figura 4.20.

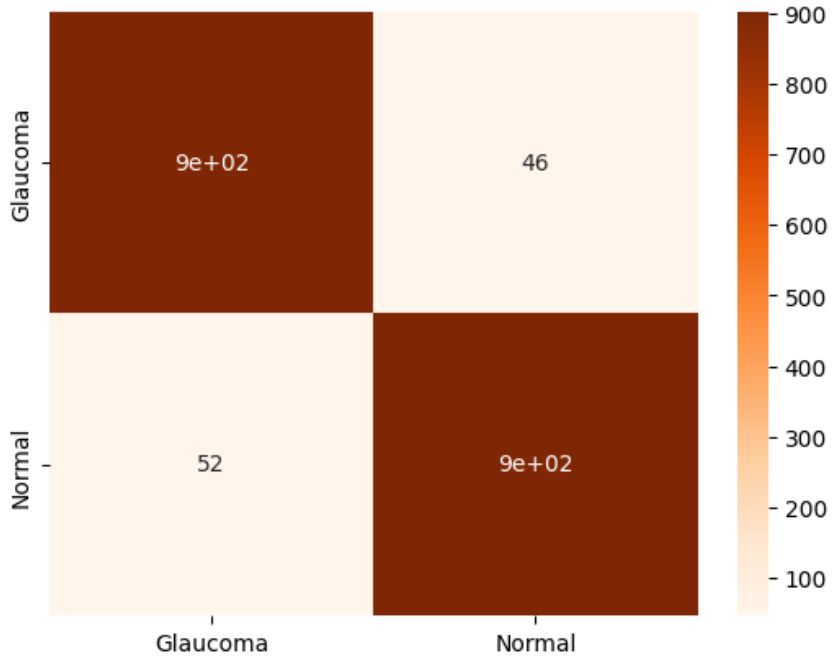


Figura 4.20: Matriz de confusión del ensemble construido con el algoritmo SVM lineal

	Precision	Recall	F1
Glaucoma	0.95	0.95	0.95
Normal	0.95	0.95	0.95

Tabla 4.34: Métricas utilizadas para el ensemble construido con el algoritmo SVM lineal

Comenzando con el análisis de las métricas, se observa que para ambas clases el modelo alcanza un 95 % de precisión. Luego, hay solo un 5 % de falsos positivos para cada clase. Además, como el *recall* de ambas clases es de un 0.95, para cada clase hay un 5 % de falsos negativos. Por último, cabe destacar que estas métricas son fiables puesto que como se vio en la imagen 4.18, las clases están bien balanceadas y existen en torno a 950 datos de prueba de cada clase.

Queda por interpretar estas métricas en el ámbito médico del problema que se está tratando. Por una parte, el *recall* del glaucoma indica que no se omiten diagnósticos positivos; es decir, hay un 5 % de probabilidad de que el modelo no detecte un paciente realmente enfermo. Por otra parte, la precisión significa que el 95 % de los diagnosticados con glaucoma realmente lo presentan.

- **SVM radial**. Las métricas para este ensemble, obtenidas a partir de los datos de la matriz de confusión de la Figura 4.20, se muestran en la Tabla 4.35.

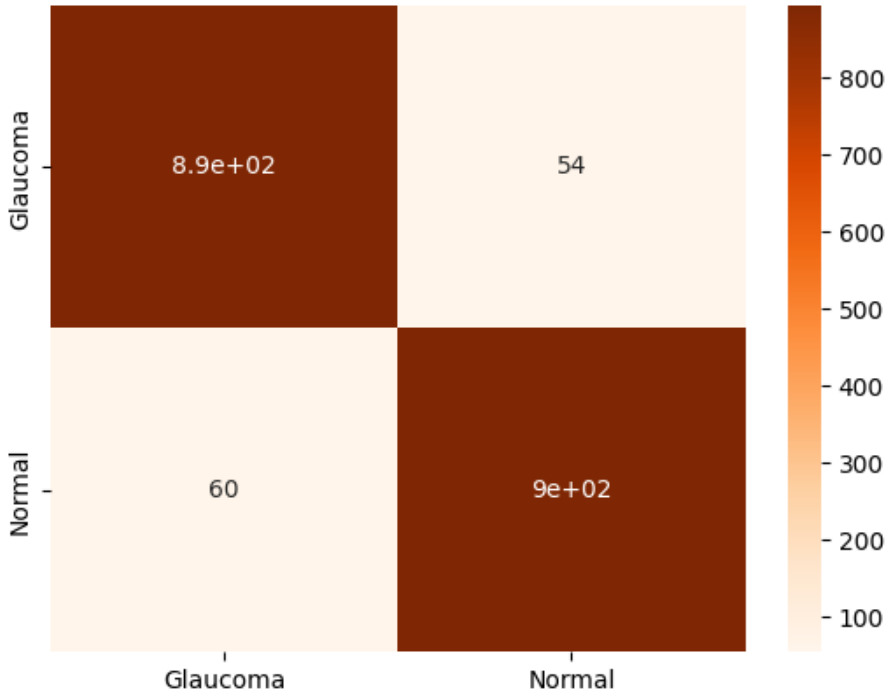


Figura 4.21: Matriz de confusión del ensemble construido con el algoritmo SVM radial.

	Precision	Recall	F1
Glaucoma	0.94	0.94	0.94
Normal	0.94	0.94	0.94

Tabla 4.35: Métricas utilizadas para el ensemble construido con el algoritmo SVM radial

En primer lugar, se tiene que para ambas clases el modelo alcanza un 94 % de precisión. Luego, hay un 6 % de falsos positivos para cada clase. Además, el *recall* de ambas clases es de 0.94, y para cada clase hay un 6 % de falsos negativos. Por último, cabe destacar que estas métricas son fiables puesto que como se vio en la imagen 4.18, las clases están balanceadas y existe un total de 1900 datos de prueba de cada clase.

Queda por interpretar estas métricas referidas al problema de detección del glaucoma que se está abordando. Por una parte, el *recall* del glaucoma indica que existe un 6 % de probabilidad de que el modelo no detecte un paciente realmente enfermo. Por otra parte, la precisión implica que el 94 % de los diagnosticados con glaucoma realmente padecen la enfermedad.

- **SVM polinomial**. El ensemble construido con este algoritmo presenta las métricas recogidas en la Tabla 4.36. Estas se calculan a partir de los datos de la matriz de confusión de la Figura 4.22.

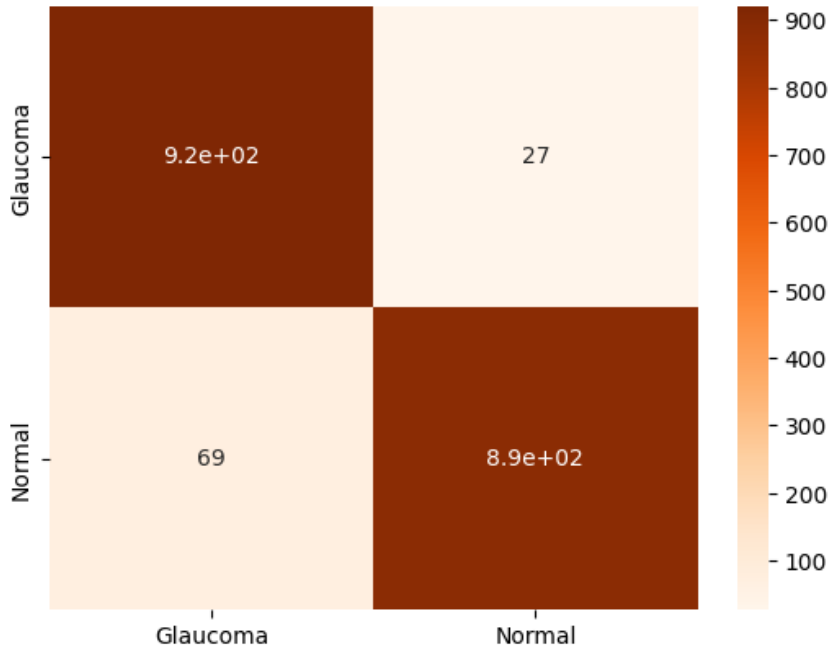


Figura 4.22: Matriz de confusión del ensemble construido con el algoritmo SVM polinomial.

	Precision	Recall	F1
Glaucoma	0.93	0.97	0.95
Normal	0.97	0.93	0.95

Tabla 4.36: Métricas utilizadas para el ensemble construido con el algoritmo SVM polinomial

En primer lugar, se tiene que para el glaucoma el modelo alcanza un 93 % de precisión; mientras que, para los casos donde no se presenta glaucoma, se tiene una precisión del 97 %. Luego, hay un 7 % de falsos positivos para la clase "Glaucoma", y un 3 % para la "Normal". Además, el *recall* de la clase "Glaucoma" es de 0.93, y para la clase "Normal" alcanza un 97 %. Así, hay un 7 % de falsos negativos para la predicción del ojo glaucomatoso, y un 3 % para la del ojo sano. Por último, cabe destacar que estas métricas son fiables puesto que, como se vio en la imagen 4.18, cada clase tiene aproximadamente el mismo número de ejemplos, y este es alto.

Queda por interpretar estas métricas referidas al problema de detección del glaucoma que se está abordando. A partir de las métricas comentadas, el ensemble está orientado a evitar falsos negativos de glaucoma. Además, el *recall* del glaucoma indica que existe un 3 % de probabilidad de que el modelo no detecte un paciente realmente enfermo. Por otra parte, la precisión implica que el 93 % de los diagnosticados con glaucoma realmente padecen la enfermedad.

- **SVM sigmoideo**. Aquí se van a analizar las métricas recogidas en la Tabla 4.37 para el ensemble construido a partir del algoritmo SVM con kernel sigmoideo. Para ello, se calculan en base a los datos de la matriz de confusión de la Figura 4.23

	Precision	Recall	F1
Glaucoma	0.97	0.85	0.91
Normal	0.87	0.98	0.92

Tabla 4.37: Métricas utilizadas para el ensemble SVM sigmoideo

Aquí se presenta el caso contrario al descrito para el ensemble con SVM polinomial. Para el glaucoma, el modelo alcanza un 97 % de precisión; mientras que, para los casos sin glaucoma, se tiene una precisión del 87 %. Luego, hay un 3 % de falsos positivos para la clase "Glaucoma", y un 13 % para la "Normal". Además, siguiendo la métrica *recall*, hay un 15 % de falsos negativos para la predicción del ojo glaucomatoso, y un 2 % para la del ojo sano. Queda mencionar que, como se observa en la imagen 4.18, los datos están balanceados y se tienen suficientes ejemplos de cada clase, luego las métricas obtenidas son fiables.

Queda por interpretar estas métricas referidas al problema de detección del glaucoma que se está abordando. A partir de las métricas comentadas, el ensemble está orientado a evitar falsos negativos de glaucoma. Además, el *recall* del glaucoma

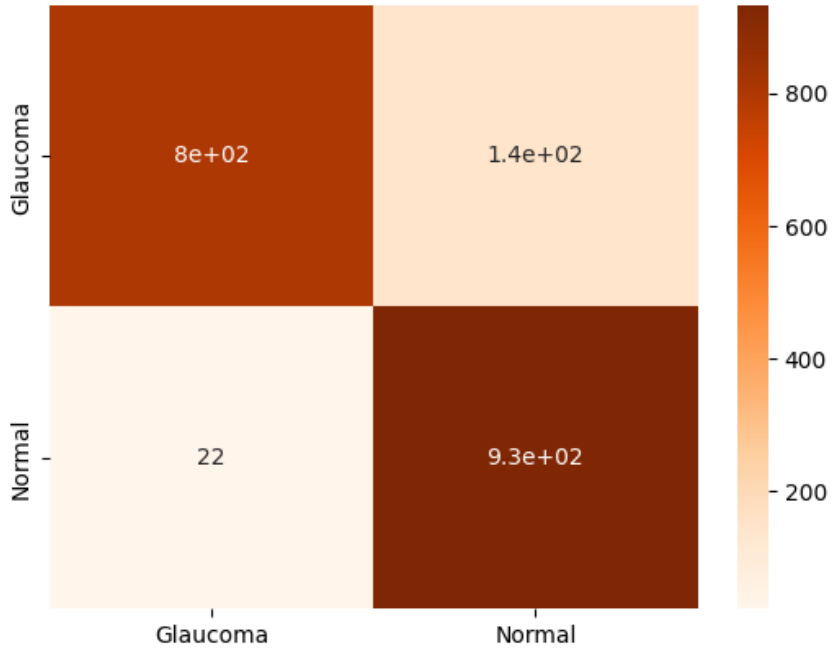


Figura 4.23: Matriz de confusión del ensemble con algoritmo SVM sigmoideo.

indica que existe un 3 % de probabilidad de que el modelo no detecte un paciente realmente enfermo. Por otra parte, la precisión implica que el 93 % de los diagnosticados con glaucoma realmente padecen la enfermedad.

- **MLP**. Para este ensemble se tienen los datos de la matriz de confusión de la Figura 4.24, a partir de los cuales se aportan las métricas recogidas en la Tabla 4.38.

	Precision	Recall	F1
Glaucoma	0.94	0.95	0.95
Normal	0.95	0.94	0.95

Tabla 4.38: Métricas utilizadas para el ensemble MLP

Por una parte, se tiene que para el glaucoma el modelo alcanza un 94 % de precisión; mientras que, para los casos donde no se presenta glaucoma, se tiene una precisión del 95 %. Esto representa que hay un 6 % de falsos positivos para la clase "Glaucoma", y un 5 % para la "Normal". Además, el *recall* de la clase "Glaucoma" es de 0.95, y para la clase "Normal", de 0.94. Así, hay un 5 % de falsos negativos para la predicción de casos con glaucoma, y un 6 % para los casos del ojo sano. Queda por mencionar que estas métricas son fiables dada la cantidad de ejemplos que se tienen y el balanceo en los datos como se ilustra en la imagen 4.18.

Queda por interpretar estas métricas referidas al problema de detección del glaucoma que se está abordando. A partir de las métricas comentadas, el ensemble está

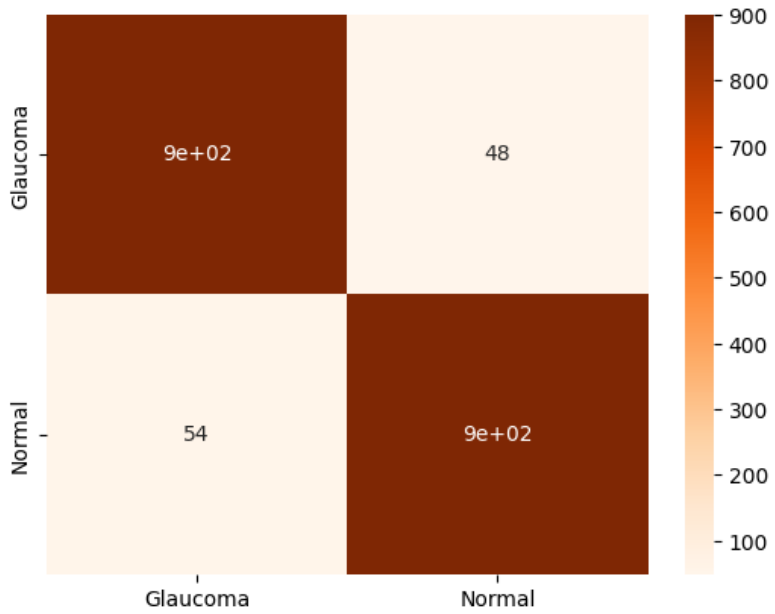


Figura 4.24: Matriz de confusión del ensemble construido con el algoritmo MLP.

orientado a evitar falsos negativos de glaucoma. Además, el *recall* del glaucoma indica que existe un 3 % de probabilidad de que el modelo no detecte un paciente realmente enfermo. Por otra parte, la precisión implica que el 93 % de los diagnosticados con glaucoma realmente padecen la enfermedad.

Balance y elección entre los ensembles construidos

El objetivo de esta sección es elegir el mejor de los ensembles construidos. Para ello, se realizará un balance entre las métricas de cada uno de ellos, es estudiará su significado y se terminará el análisis decidiendo cuál de todos los ensembles es el mejor para el entorno médico.

Puesto que el propósito es determinar el mejor de los ensembles, una métrica importante será el *accuracy*, que queda recogido en la Tabla 4.33. Esto se debe a que se busca un modelo capaz de clasificar la mayor parte de casos de manera correcta, independientemente de su tipo. Teniendo esto en cuenta, los ensembles que menor porcentaje de casos clasifican correctamente son el SVM sigmoideo y el SVM radial, con un 91.28 % y un 94.01 %, respectivamente. Una vez descartados los dos modelos que peor rendimiento tienen respecto a la métrica *accuracy*, quedan los ensembles construidos con los algoritmos SVM lineal, SVM polinómico y MLP, con un 94.85 %, 94.96 % y 94.75 %, respectivamente. En este caso, se siguen teniendo en cuenta estos tres últimos ensembles, puesto que la diferencia en el valor *accuracy* de ellos no es suficiente como para escoger uno en concreto y es preferible estudiarlos siguiendo otras métricas.

Continuando con el estudio, podemos observar que para la métrica F1 de los ensembles construidos con los algoritmos SVM lineal, SVM polinómico y MLP, se tiene para todos

los mismos valores de 0.95 tanto para la clase "normalçomo "glaucoma". Este valor mide la eficacia global del modelo a partir de las métricas *precision* y *recall*. Luego, los tres modelos son muy buenos, dado que detectan correctamente los casos positivos y evitan falsos positivos. No obstante, da la igualdad de valores, no se puede determinar cuál es mejor en el proceso de diagnóstico siguiendo la métrica F1.

Finalmente, queda estudiar la relación entre las precisiones y *recalls* de los distintos modelos para elegir el que mejores resultados aporte, aunque en cualquiera de los tres casos contemplados vayan a ser buenos dadas las métricas que ya se han tratado. En primer lugar, se puede apreciar que para todas las métricas del ensemble SVM lineal se tienen valores de 0.95 como se muestra en la Tabla 4.34, y para el ensemble MLP, de 0.94 y 0.95 como se recoge en la Tabla 4.38. Así, como MLP empeora alguno de los valores en *precision* y *recall* a los del SVM lineal, y el *accuracy* también es inferior, se descarta. Luego, quedan por elegir un ensemble entre el SVM lineal y el SVM polinomial.

Para el SVM polinomial, se recuerda que el *accuracy* era ligeramente superior al del SVM lineal. La diferencia principal entre los ensembles radica en que el SVM lineal es un modelo equilibrado, dado que clasifica ambas clases por igual, sin sesgo hacia ninguna, al contrario que el SVM polinomial. En concreto, para el ensemble construido con SVM polinomial, de todos los pacientes que realmente tienen glaucoma, se detecta el 97 %, dado que este es el *recall* para el glaucoma. Luego, se tienen pocos falsos negativos; es decir, pocos enfermos se quedan sin ser diagnosticados. Además, la precisión en la clasificación de pacientes sanos es de un 97 %. Esto indica que para esta clase hay un 3 % de falsos positivos; lo que radica en pocos sanos mal diagnosticados como enfermos.

Dependiendo del contexto, puede ser que uno de los ensembles sea más favorable que otro. En algunos casos, es interesante que las métricas estén más equilibradas. Sin embargo, para el caso médico lo primordial es un alto *recall* para la clase de la enfermedad y una alta precisión en la clase de pacientes sanos. El motivo es que, como se ha explicado, se pretende que queden el menor número de enfermos sin detectar, así como el menor número posible de pacientes sanos mal diagnosticados como enfermos. Esto se debe a que dejar sin diagnosticar a un enfermo puede tener graves consecuencias, como la aparición de la ceguera; mientras que diagnosticar como enfermo a un paciente sano también tiene sus implicaciones, como una serie de costos en el tratamiento, además de tener que estar expuesto al mismo de manera innecesaria. Bajo estas condiciones, el mejor ensemble es el construido mediante el algoritmo SVM polinomial.

Comparando con los resultados que han obtenido los trabajos mencionados en el estado del arte en la Sección 3.4, el único que se aproxima a los resultados obtenidos aquí es el segundo de los modelos de [80]. Para este modelo se han obtenido unos resultados casi idénticos que para el ensemble lineal construido en este trabajo, siendo superado por este último en casi un 1 % de *accuracy*. Luego, si el método mencionado en el estado del arte proporciona un rendimiento ligeramente inferior al SVM lineal, con los mismos razonamientos que se han dado anteriormente, se tiene que el ensemble a partir de SVM polinomial que aquí se ha elaborado, aporta mejores resultados. Luego, la solución construida en este trabajo supera todas las existentes para la detección del glaucoma a partir del *dataset rotterdam*.

Otra propuesta para mejorar los resultados

Como se ha explicado, las métricas más importantes, las que se pretenden maximizar sus resultados, son el *recall* para el glaucoma y la *precision* para el normal, pues no se quiere dejar ningún caso de glaucoma sin diagnosticar. No obstante, también se debe seguir prestando atención al resto de métricas.

Para llevar esto a cabo, se introduce aquí el uso de la métrica F-β explicada en la Sección 3.2.4. Así, se han comparado los modelos entrenados a través del valor F2 para el glaucoma, F0.5 para el normal y F1 para ambos casos. A esto se añade la métrica *accuracy* ya estudiada. Toda esta información se recopila en la Tabla 4.39.

Modelo	*accuracy*	F2 Glaucoma	F1 Glaucoma	F0.5 Normal	F1 Normal
rotterdam	0.933	0.935	0.933	0.935	0.933
rotterdam grises	0.912	0.927	0.914	0.924	0.910
rotterdam RIM-ONE	0.947	0.950	0.947	0.950	0.947
rotterdam RIM-ONE preentrenado	0.943	0.938	0.943	0.940	0.944
rotterdam RIM-ONE grises	0.921	0.943	0.924	0.939	0.919
rotterdam RIM-ONE grises preentrenado	0.937	0.938	0.937	0.938	0.938
rotterdam RIM-ONE rojo	0.912	0.914	0.912	0.914	0.912
rotterdam RIM-ONE verde	0.939	0.947	0.940	0.946	0.938
rotterdam RIM-ONE azul	0.897	0.898	0.897	0.898	0.897

Tabla 4.39: Comparación de las métricas más relevantes de los modelos de clasificación

Los resultados han sido ordenados en función del F2 para el glaucoma, y en caso de empate, teniendo en cuenta el resto de métricas. Así, los mejores siete modelos siguen la clasificación:

1. *rotterdam_RIM-ONE*
2. *rotterdam_RIM-ONE_verde*
3. *rotterdam_RIM-ONE_grises*
4. *rotterdam_RIM-ONE_preentrenado*. Este se descarta por tener el mismo modelo pero sin preentrenar en una mejor posición.
5. *rotterdam_RIM-ONE_grises_preentrenado*. Este se descarta por tener el mismo modelo pero sin preentrenar en una mejor posición.
6. *rotterdam*
7. *rotterdam_grises*

La primera observación que surge es que, cuando se tienen en cuenta todas estas métricas, se observa que los modelos sin preentrenar ofrecen mejores resultados. Sin embargo, esto puede deberse a que el preentrenamiento se hace con imágenes de otro *dataset*; mientras que el entrenamiento final y el test se hacen con el mismo *dataset*. Aunque las imágenes de entrenamiento y test sean diferentes, pueden tener similitudes por pertenecer a un mismo *dataset*, por lo que habría que contar con más datos para comprobar si realmente preentrenando con otro conjunto de datos el modelo generaliza mejor.

Otra observación es que, ahora que se consideran más métricas, se observa como todos los mejores modelos de los que se muestran en la lista son los que se fijan en la parte del nervio óptico. Esto corrobora una vez más que esta región es la que mayor información aporta para la detección del glaucoma.

Por otra parte, los resultados coinciden en su mayor parte con la clasificación hecha para el *accuracy*. Esto sugiere que, en este caso, el criterio de selección basado en el *accuracy* es suficiente y consistente con otras métricas más sensibles al balance entre *precision* y *recall*. No obstante, se han entrenado los mismos ensembles con estos modelos para tratar de mejorar los resultados.

El entrenamiento de los ensembles a partir de los modelos que mejor *recall* han mostrado por separado no ha mostrado resultados relevantes. En ninguno de ellos el *recall* para el glaucoma supera el 94 %. Además, tampoco se identifica ninguna mejora en el *accuracy* del ensemble final, pues tampoco supera el 94 % para esta métrica.

Capítulo 5

Integración de los modelos. Construcción de una aplicación.

Una vez desarrollados los modelos para el diagnóstico del glaucoma, así como los de segmentación de las estructuras de la región del nervio óptico, el siguiente paso es su integración en una aplicación funcional. Esta etapa corresponde con la fase de despliegue de la metodología CRISP-DM. Esto permite trasladar la funcionalidad desarrollada para la detección del glaucoma a un entorno más accesible para que pueda ser usada por oftalmólogos, sin necesidad de tener conocimiento en programación.

Este capítulo presenta la construcción de dicha aplicación. Para ello, se realizan los procesos de análisis, diseño, implementación y pruebas en los que se basa el desarrollo de la misma. Se presentan los principales *workflows* de desarrollo de la aplicación de forma resumida, ya que el grueso del trabajo se lo ha llevado la construcción, entrenamiento y evaluación de los modelos, y esta aplicación es un prototipo para integrar todo el desarrollo previo. A continuación se va a ir detallando una por una cada fase implicada.

5.1. Análisis - Especificación de requisitos

La fase de análisis en el desarrollo de software establece qué debe hacer el sistema. Esta etapa precede a la de diseño, en la que se define cómo se va a hacer. Por tanto, el objetivo principal del análisis es comprender, documentar y especificar los requisitos funcionales y no funcionales del sistema que se va a desarrollar, así como los requisitos de usuario.

5.1.1. Requisitos de usuario

Los requisitos de usuario describen lo que el usuario espera que el sistema haga. Por otro lado, un caso de uso es una descripción detallada de cómo un usuario o actor interactúa con el sistema para lograr un objetivo específico. Para ello, la descripción de los requisitos de usuario contempla un diagrama de casos de uso, los especifica y produce de una manera ordenada los requisitos buscados.

Diagrama de casos de uso

En la Figura 5.1 se presenta el diagrama de casos de uso que resume de manera visual lo que el usuario espera del sistema.

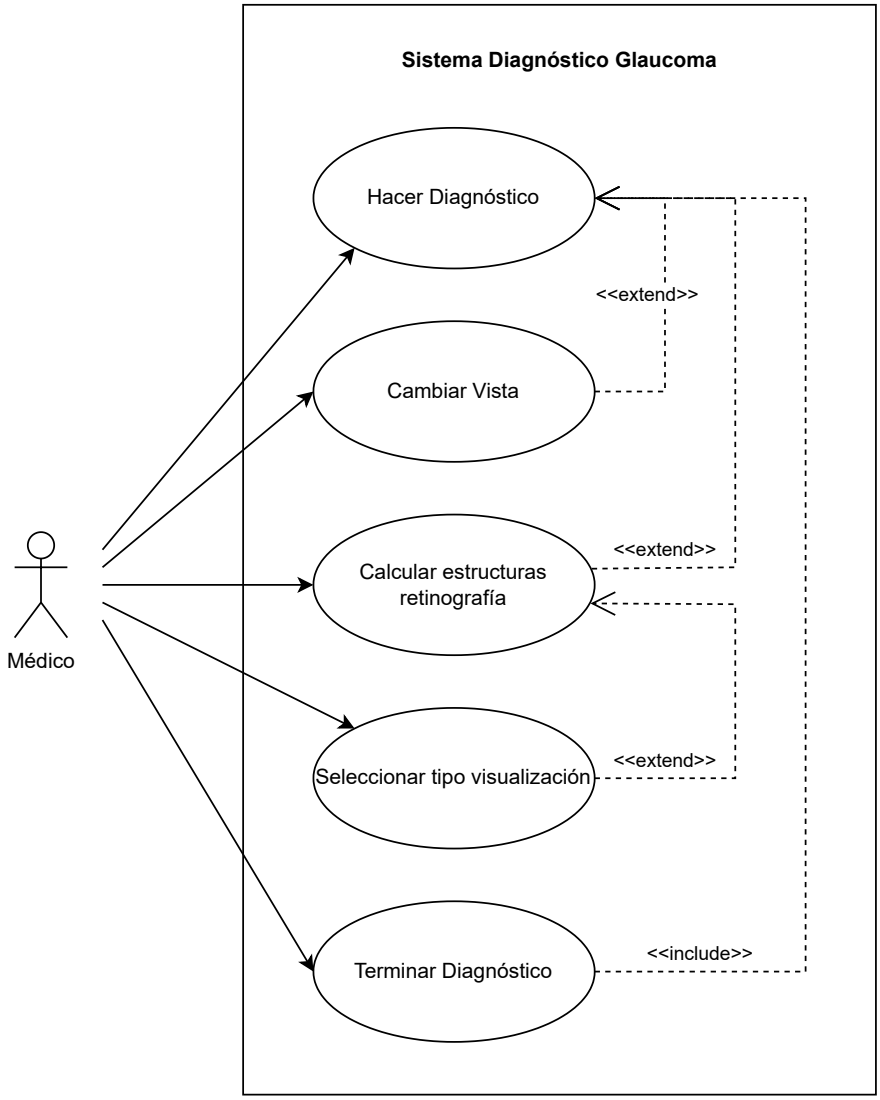


Figura 5.1: Diagrama de casos de uso de la aplicación construida.

Especificación de los casos de uso

A lo largo de las tablas 5.1, 5.2, 5.3, 5.4 y 5.5 se especifican cada uno de los casos de uso recogidos en el diagrama de la Figura 5.1.

CU_01	**Hacer Diagnóstico**
Precondición	-
Descripción	El sistema permite subir una retinografía y realiza el diagnóstico en base a la misma
Secuencia	Secuencia normal: 1. El usuario sube una retinografía. 2. Se realiza el diagnóstico. 3. Se muestra el diagnóstico y la retinografía por pantalla. 4. Si el usuario decide cambiar la vista <<Punto de extensión>> Cambiar Vista. 5. Si el usuario decide segmentar <<Punto de extensión>> Calcular estructuras retinografía.
Postcondición	-
Excepciones	-
Rendimiento	15-20 segundos
Importancia	Alta
Frecuencia	Una vez por diagnóstico

Tabla 5.1: Caso de Uso 1

CU_02	**Cambiar vista**
Precondición	Haber subido la imagen de una retinografía
Descripción	El sistema cambia la visualización de la retinografía completa a la región del nervio óptico y viceversa.
Secuencia	Secuencia normal: 1. Usuario hace click en el botón <<Cambiar vista>>. 2. Se toma la región buscada. 3. Se cambia la vista. Alternativa 1: 1. Usuario hace click en el botón. 2. No se detecta la región 3. No se cambia la vista.
Postcondición	La región de la retinografía que se muestra es distinta

Excepciones	- Excepción 1: El sistema no detecta el disco óptico - Flujo alternativo para la excepción 2: Secuencia Alternativa 1 - Excepción 2: Se introduce una imagen que no sea una retinografía - Flujo alternativo para la excepción 2: Secuencia Alternativa 1
Rendimiento	Inmediato
Importancia	Normal
Frecuencia	Las veces que el usuario lo utilice

Tabla 5.2: Caso de Uso 2

CU_03	**Calcular estructuras retinografía**
Precondición	Haber subido la imagen de una retinografía
Descripción	Cálculo de las segmentaciones del disco óptico y la copa.
Secuencia	Secuencia normal: 1. El usuario solicita las segmentaciones 2. El módulo correspondiente calcula las segmentaciones 3. Se muestra la opción de escoger la segmentación que se quiere contrastar. 4. Si el usuario decide seleccionar una segmentación <<Punto de extensión>> Seleccionar tipo visualización.
Postcondición	La opción para elegir la segmentación es visible
Excepciones	-
Rendimiento	30 segundos si solo aparece modelo YOLO 1 minuto y 30 segundos si también se muestra la solución con FastAI
Importancia	Alta
Frecuencia	Una vez por diagnóstico

Tabla 5.3: Caso de Uso 3

CU_04	**Seleccionar tipo visualización**
Precondición	Haber calculado las estructuras de la retinografía
Descripción	El usuario puede ver las segmentaciones calculadas para el disco y la copa
Secuencia	Secuencia normal: 1. El usuario selecciona el tipo de segmentación que quiere comprobar 2. El sistema muestra la segmentación escogida por pantalla.
Postcondición	La segmentación seleccionada es visible
Excepciones	-
Rendimiento	Inmediato
Importancia	Alta
Frecuencia	Las veces que el usuario lo requiera

Tabla 5.4: Caso de Uso 4

CU_05	**Terminar Diagnóstico**
Precondición	Haber subido la imagen de una retinografía
Descripción	Finalización del diagnóstico para poder realizar uno nuevo
Secuencia	Secuencia Normal: 1. El usuario selecciona la opción de terminar diagnóstico. 2. El sistema elimina los archivos generados para el diagnóstico. 3. El sistema vuelve al punto de inicio para realizar otro diagnóstico
Postcondición	Se eliminan los archivos generados
Excepciones	-
Rendimiento	10 segundos
Importancia	Baja
Frecuencia	Una vez por diagnóstico

Tabla 5.5: Caso de Uso 5

Requisitos de usuario

Una vez descritos los casos de uso que se han recogido para la implementación de la aplicación, se pasa a enumerar cada uno de las requisitos de usuario encontrados:

- **RU_01**. El usuario podrá subir una retinografía a la aplicación.
- **RU_02**. El usuario consultará el diagnóstico para la retinografía adjuntada.
- **RU_03**. El usuario podrá ver entre ver la retinografía completa o solo la parte del nervio óptico.
- **RU_04**. El usuario podrá ver la segmentación de las estructuras básicas de una retinografía.
- **RU_05**. El usuario puede visualizar las segmentaciones ordenadas.
- **RU_06**. El usuario cerrará el diagnóstico para poder realizar otro.

5.1.2. Requisitos funcionales y no funcionales

Los requisitos funcionales y no funcionales son dos tipos fundamentales de requisitos en el desarrollo de software que ayudan a definir qué debe hacer el sistema y cómo debe hacerlo, respectivamente.

Requisitos funcionales

- **RF_01**. El sistema almacenará la imagen subida por el usuario.
- **RF_02**. El sistema hará un diagnóstico de si la retinografía tiene glaucoma o no.
- **RF_03**. El sistema recortará y almacenará la parte del nervio óptico de la retinografía.
- **RF_04**. El sistema permitirá las distintas vistas de la retinografía; es decir, la retinografía completa y la parte del nervio óptico.
- **RF_05**. El sistema segmentará las estructuras de la retinografía.
- **RF_06**. El sistema permitirá consultar las segmentaciones construidas.
- **RF_07**. El sistema permitirá terminar el diagnóstico borrando todas las imágenes generadas.

Requisitos no funcionales

Requisitos de eficiencia:

- **RNF_01**. El sistema debe tardar un máximo de 2 minutos en realizar el diagnóstico.
- **RNF_02**. El sistema deberá emplear un tiempo máximo de 6 minutos en calcular todas las segmentaciones.
- **RNF_03**. Las operaciones que consisten en un cambio en la visualización de los datos deben tener un tiempo de ejecución máximo de 1 segundo.
- **RNF_04**. Las imágenes generadas por el sistema deben ser visibles en menos de 3 segundos.

Requisitos de fiabilidad:

- **RNF_05**. La localización del nervio óptico debe realizarse correctamente el 98 % de las veces.
- **RNF_06**. La tasa de fallo de las segmentaciones ejecutadas debe ser menor al 5 %.

Requisitos de usabilidad:

- **RNF_07**. El sistema será accesible desde ordenador y dispositivos móviles, siendo el primero de ellos donde se usará principalmente la aplicación.
- **RNF_08**. La interfaz de usuario debe ser clara, intuitiva y "amigable" para un uso sencillo.
- **RNF_09**. La aplicación tendrá un diseño *responsive* que se adecúe al tamaño del *viewport*.
- **RNF_10**. La curva de aprendizaje para familiarizarse con el sistema de detección del glaucoma debe ser inferior a 10 minutos.
- **RNF_11**. El sistema contará con manuales de usuario para facilitar el uso por parte de los mismos.

Otros requisitos no funcionales:

- **RNF_12**. La aplicación desarrollada deberá ser compatible con todo tipo de dispositivos; es decir, será multiplataforma.
- **RNF_13**. El tratamiento de los datos se hará respetando lo establecido en la Ley Orgánica de Protección de Datos (Ley Orgánica 3/2018, de 5 de diciembre, de Protección de Datos Personales y garantía de los derechos digitales).
- **RNF_14**. Se empleará *dash* para el desarrollo de la aplicación.
- **RNF_15**. El sistema no debe estar fuertemente acoplado para permitir la reutilización y portabilidad de parte del código.

5.2. Diseño

Como se ha mencionado con anterioridad, la fase de diseño en el desarrollo de software es la etapa en la que se planifica cómo se va a construir el sistema definido en la fase de análisis. Por tanto, representa un vínculo entre los requisitos - el qué debe hacer el sistema - y la codificación - el cómo se va a implementar.

Para la fase de diseño de la aplicación que se está describiendo, se ha considerado oportuno explicar la arquitectura lógica de la aplicación, así como los diagramas de secuencia que explican los proceso más complejos que llevará a cabo y una primera aproximación de su interfaz.

5.2.1. Arquitectura lógica

La arquitectura lógica del sistema representa de una manera organizada los componentes funcionales del sistema y la relación entre ellos, sin tener en cuenta su implementación física en materia de servidores y otros componentes hardware. En la Figura 5.2 queda descrita la arquitectura lógica para la aplicación construida, la cual sigue el patrón arquitectónico "Capas".

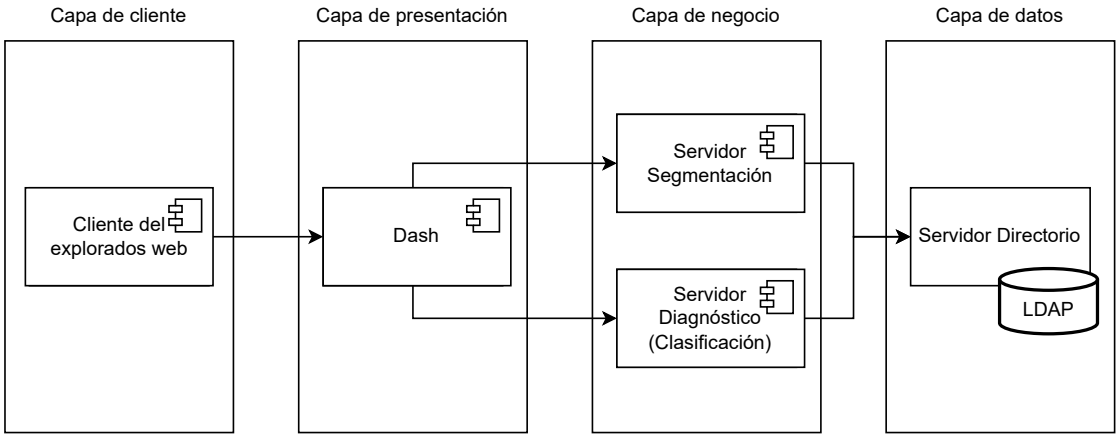


Figura 5.2: Diagrama descriptivo de la arquitectura lógica de la aplicación.

A continuación se detallan cada uno de los componentes que constituyen la arquitectura lógica según la capa a la que pertenecen:

- Capa de cliente. Representa la capa de acceso del usuario a la aplicación. Esta se dispone para poder ser usada como aplicación web.

- Capa de presentación. Se trata de la parte de la aplicación encargada de interactuar directamente con el usuario, mostrando la interfaz y gestionando las entradas. Con este propósito se ha hecho uso de la biblioteca *Dash*.

- Capa de negocio. Parte de la arquitectura que implementa la lógica y las reglas de negocio de una aplicación. Para este caso, la lógica fundamental la componen dos módulos: *Segmentation Server* y *Diagnosis (Clasification) Server*, que se encargan de segmentar y clasificar las imágenes (realizar el diagnóstico), respectivamente.
- Capa de datos. Se refiere a la organización y gestión de los datos. En concreto, define cómo se estructuran los datos, cómo se relacionan entre sí y cómo se accede a los mismos.

5.2.2. Diagramas de secuencia

En las figuras 5.3 y 5.4 se modela la interacción entre los componentes del sistema para realizar los procesos de diagnóstico y de segmentación, respectivamente.

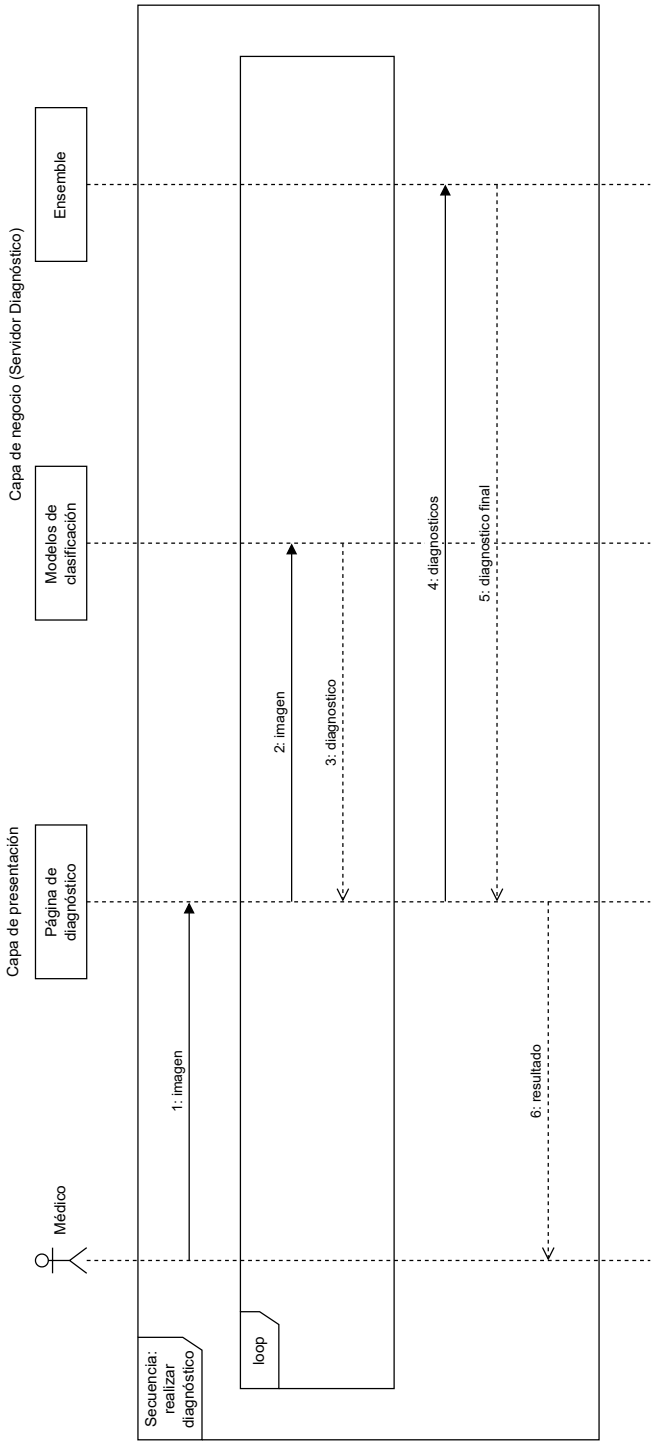


Figura 5.3: Diagrama de secuencia para el proceso de diagnóstico.

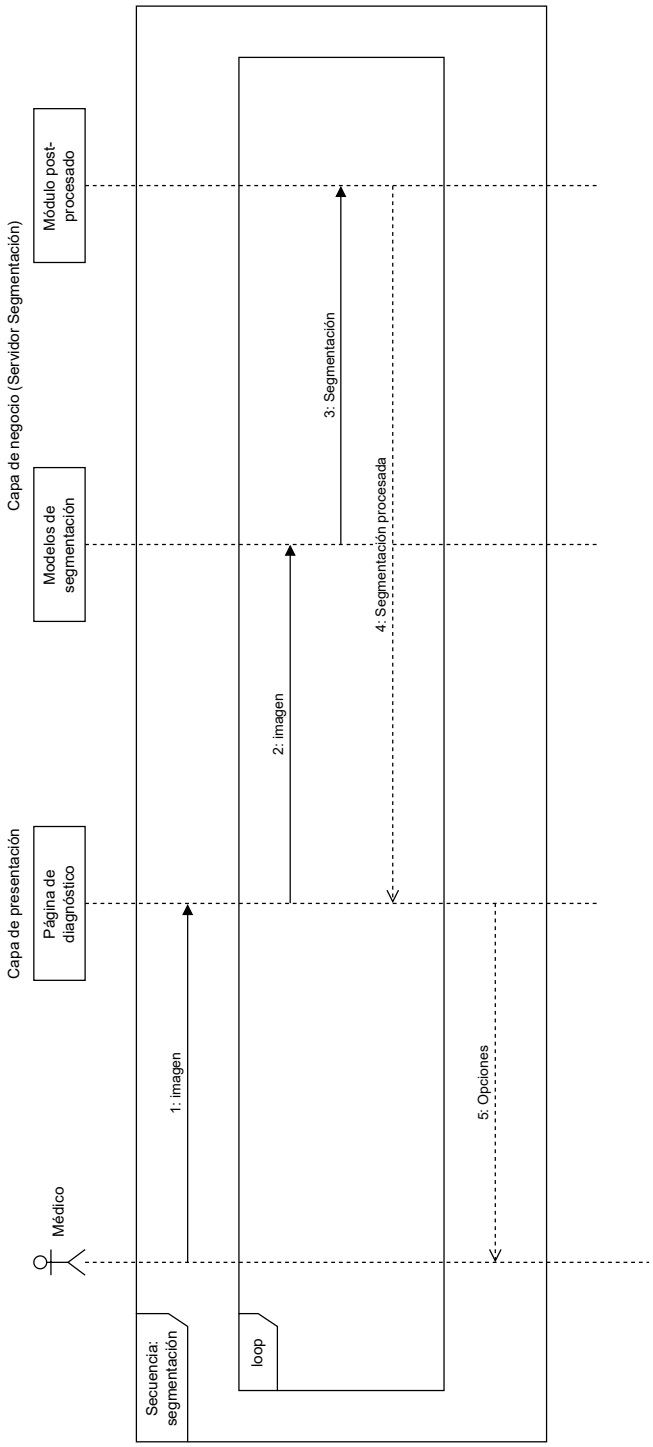


Figura 5.4: Diagrama de secuencia para el proceso de segmentación.

5.2.3. Diseño de interfaz

En las figuras 5.5, 5.6 y 5.7 se representan unos bocetos con el diseño propuesto para la aplicación construida.

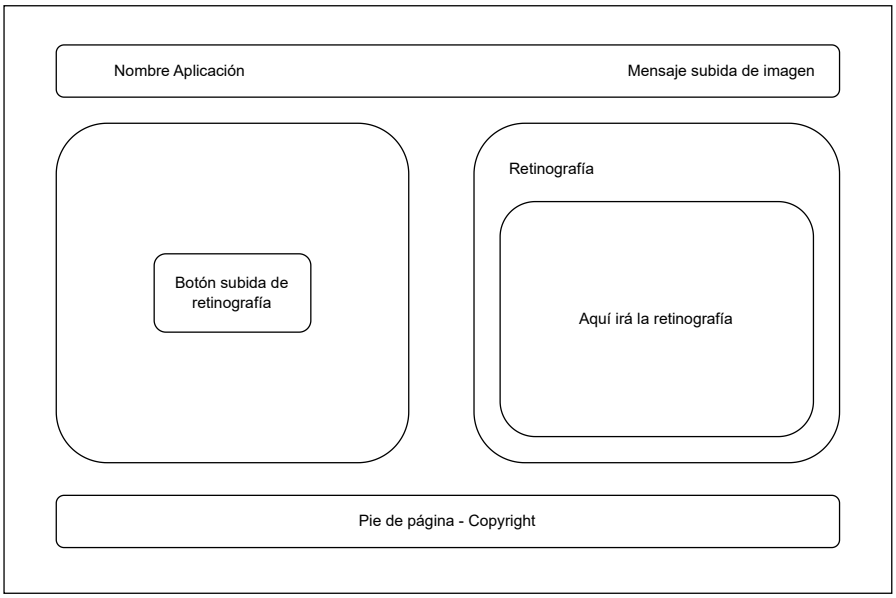


Figura 5.5: Interfaz de usuario propuesta inicio aplicación pre-diagnóstico.

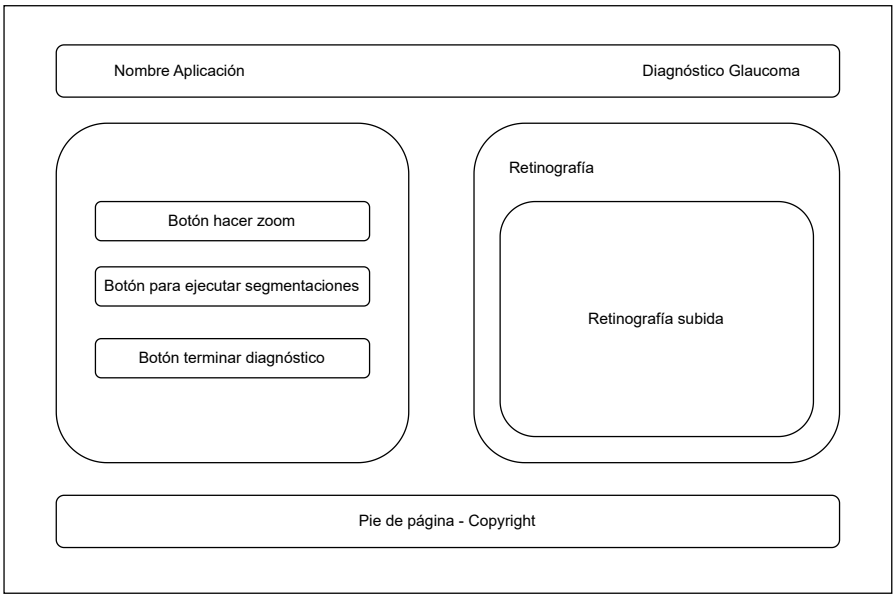


Figura 5.6: Interfaz de usuario propuesta tras diagnóstico antes de segmentar.

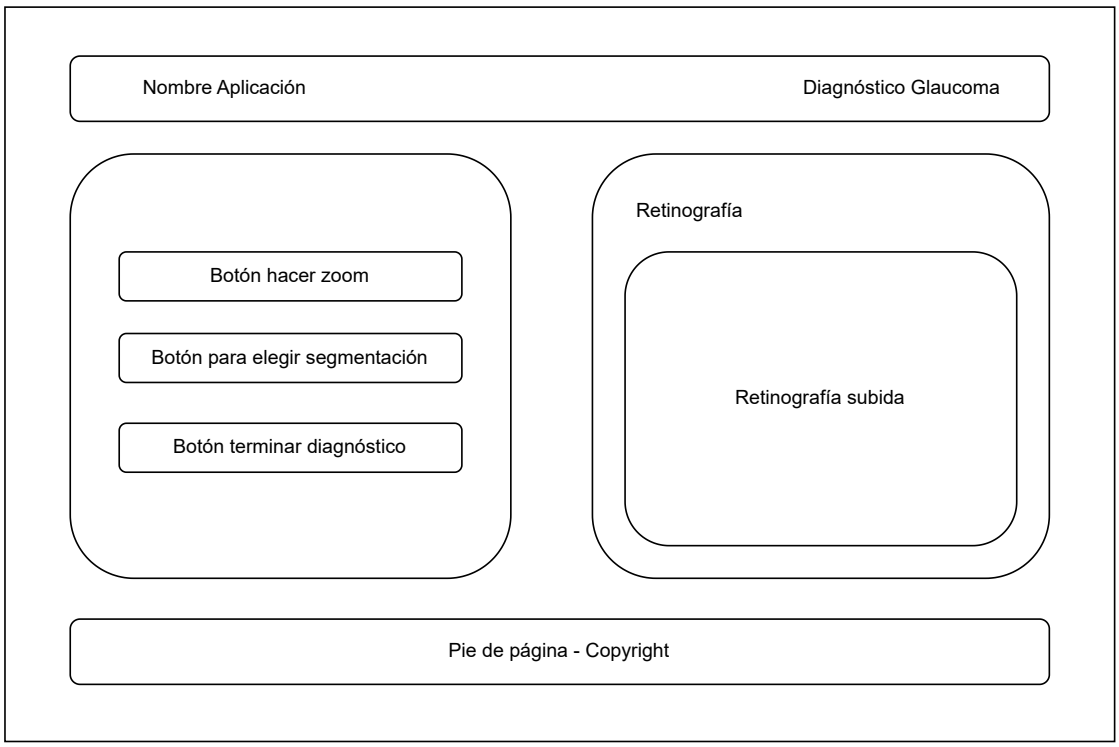


Figura 5.7: Interfaz de usuario propuesta tras segmentación.

5.3. Implementación

5.3.1. Tecnologías y herramientas utilizadas

Para el desarrollo de la aplicación se han empleado las siguientes tecnologías:

- **Dash**. Esta biblioteca ha sido empleada para construir la aplicación, tanto la parte de *frontend* como la de las llamadas a los módulos que se encargan de la predicción de diagnósticos y estructuras.
- **Ultralytics YOLO**. Implementación optimizada del modelo de detección de objetos YOLO, desarrollada por la empresa Ultralytics. Ofrece modelos rápidos y precisos sobre los que es fácil entrenar y desplegar otros modelos tanto para clasificación como para segmentación. Está escrito en Python y usa PyTorch como backend.
- **FastAI**. FastAI es una biblioteca de alto nivel construida sobre PyTorch, diseñada para facilitar el entrenamiento rápido y eficaz de modelos de Deep Learning usando menos código.

Por otra parte, se ha hecho uso de una serie de herramientas tanto en la parte de programación como para llevar a cabo el diseño y análisis de la aplicación:

- **Google Colab**. Entorno basado en *Jupyter Notebook* que permite ejecutar código Python desde el navegador con acceso a recursos GPU. Sobre esta herramienta se construye todo el código de la aplicación desarrollada.
- **Google Drive**. Herramienta utilizada para almacenar todos los archivos generados por su facilidad para acceder desde distintos dispositivos que permite conectarse con *Google Colab*.
- **Draw.io**. Herramienta que se ha empleado en la elaboración de todos los diagramas: diagrama de casos de uso, diagramas de secuencia y representación de las interfaces de usuario previstas.

5.3.2. Interfaz de usuario implementada

Con el uso de las tecnologías y herramientas descritas en la Sección 5.3.1 se implementa finalmente la aplicación que se ha venido describiendo a lo largo del Capítulo 5. Además, en las figuras 5.8, 5.9, 5.10, 5.11, 5.12, 5.13, 5.14 y 5.15 se incluye la interfaz de usuario que se ha desarrollado a partir de la propuesta en la Sección 5.2.3.

Figura 5.8: Interfaz de usuario del programa pre-diagnóstico.

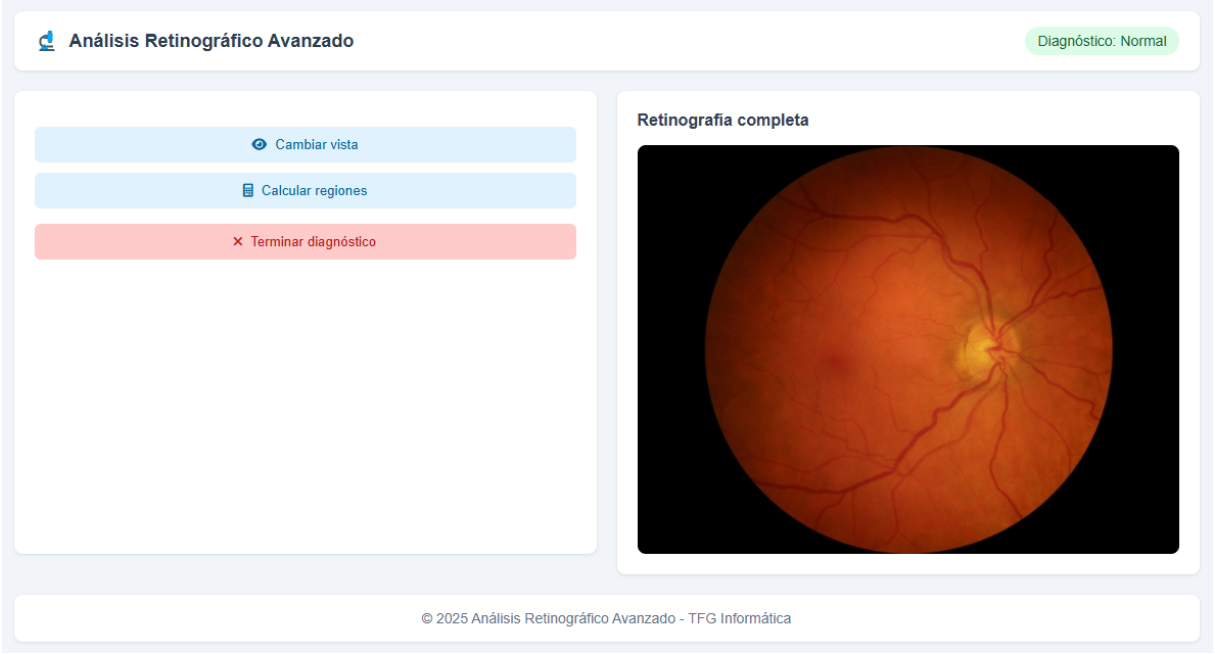


Figura 5.9: Interfaz de usuario del programa cuando no se diagnostica glaucoma.

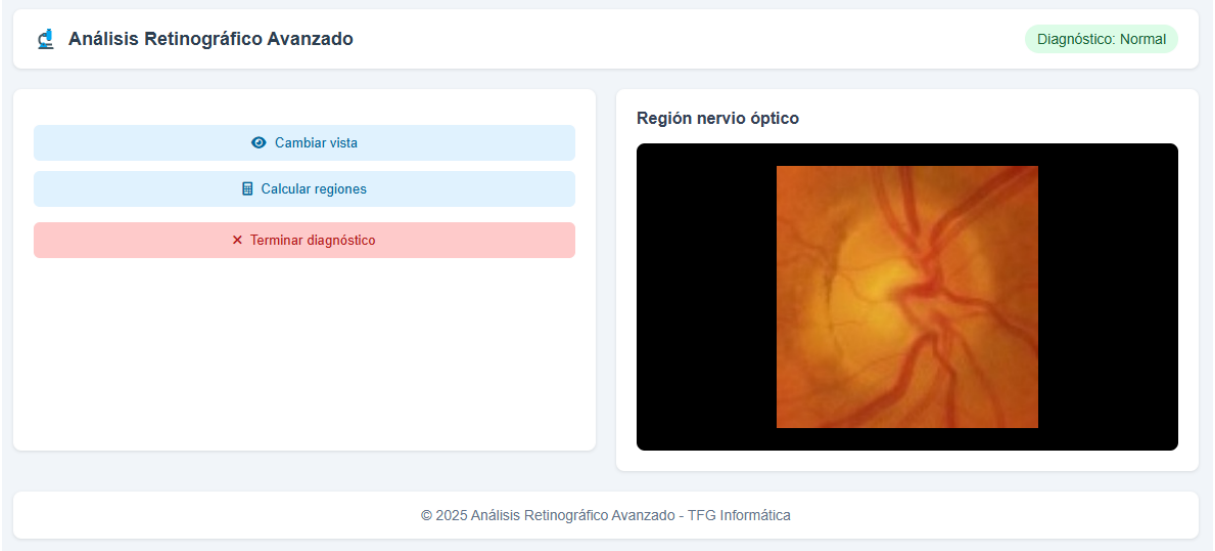


Figura 5.10: Interfaz de usuario del programa haciendo zoom en el nervio óptico.

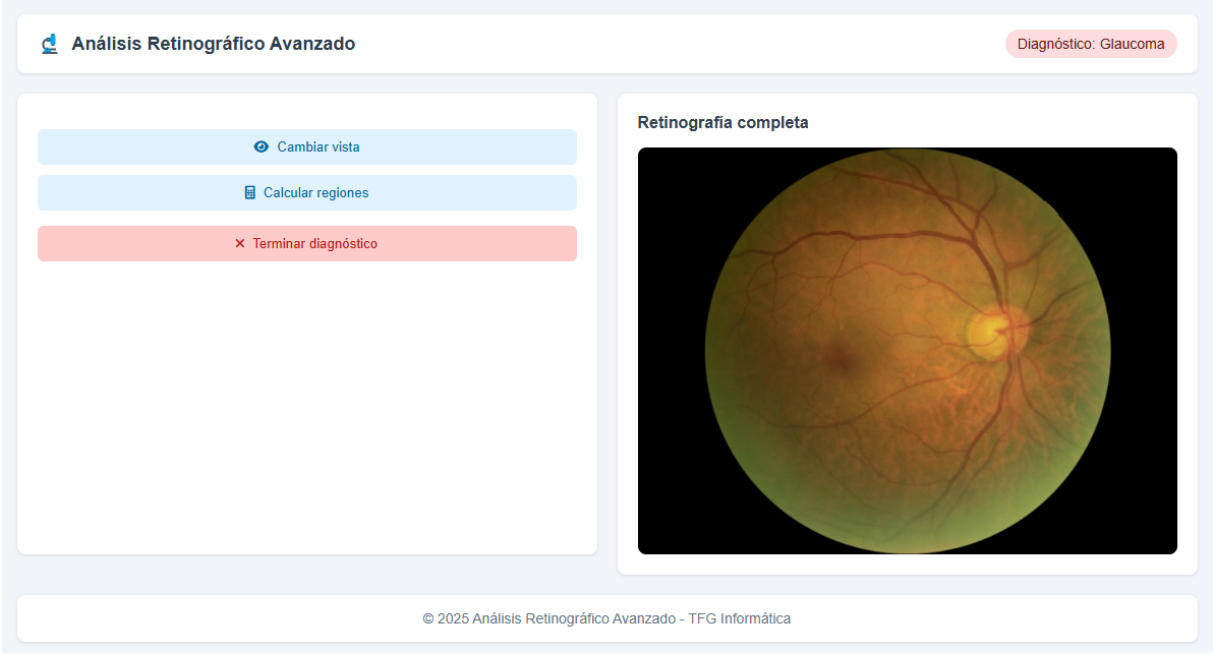


Figura 5.11: Interfaz de usuario del programa cuando se diagnostica el glaucoma.

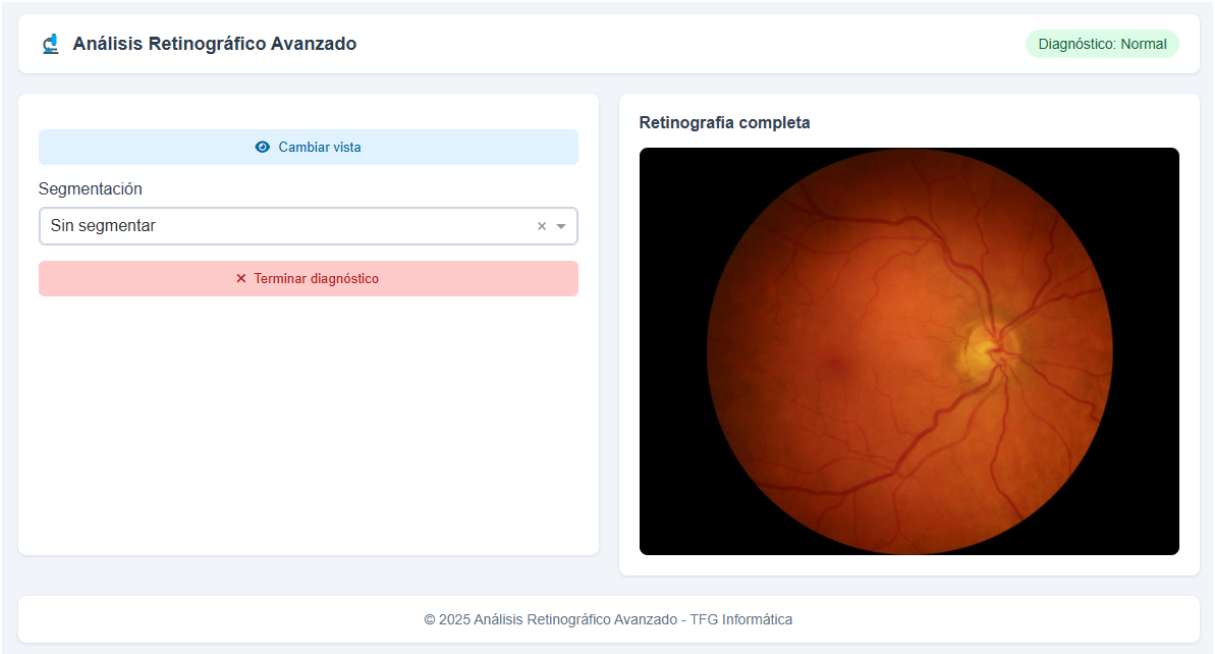


Figura 5.12: Interfaz de usuario del programa tras segmentación.

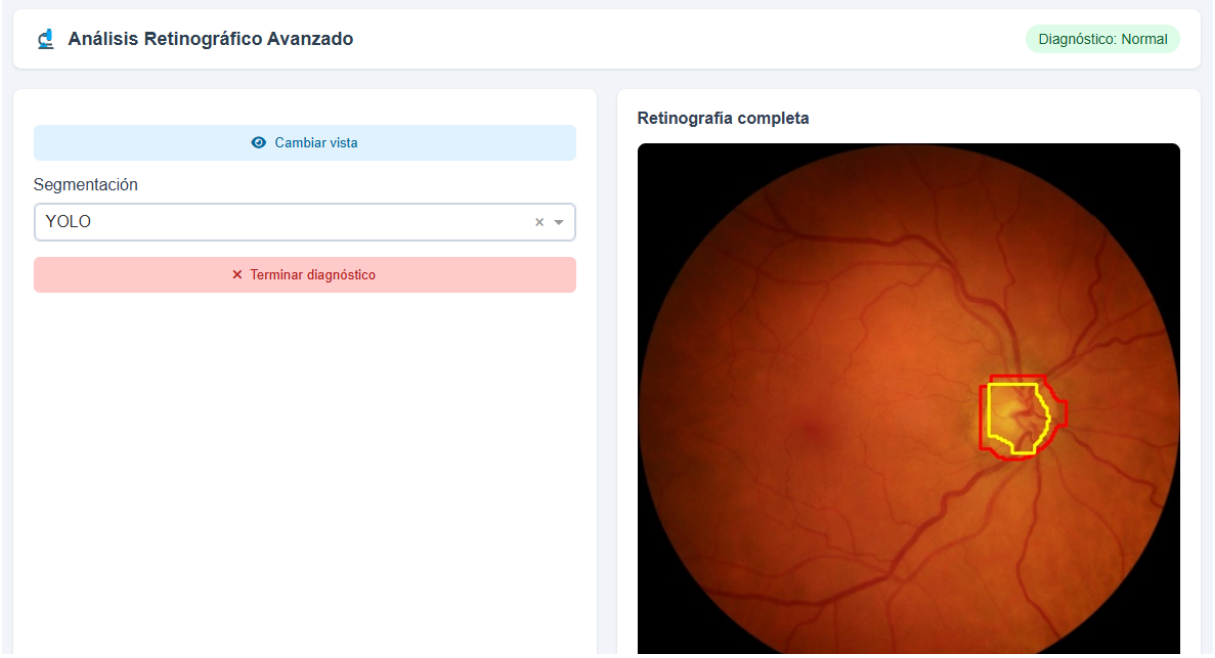


Figura 5.13: Interfaz de usuario del programa con la segmentación de YOLO.

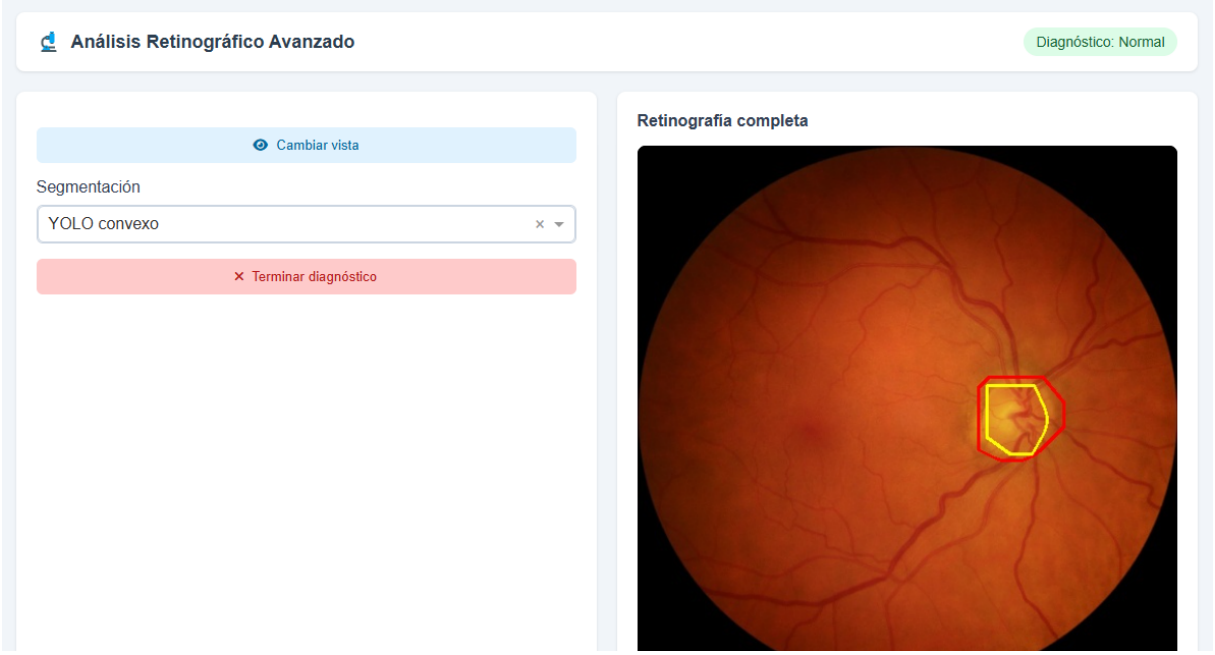


Figura 5.14: Interfaz de usuario del programa con la segmentación de YOLO convexo.

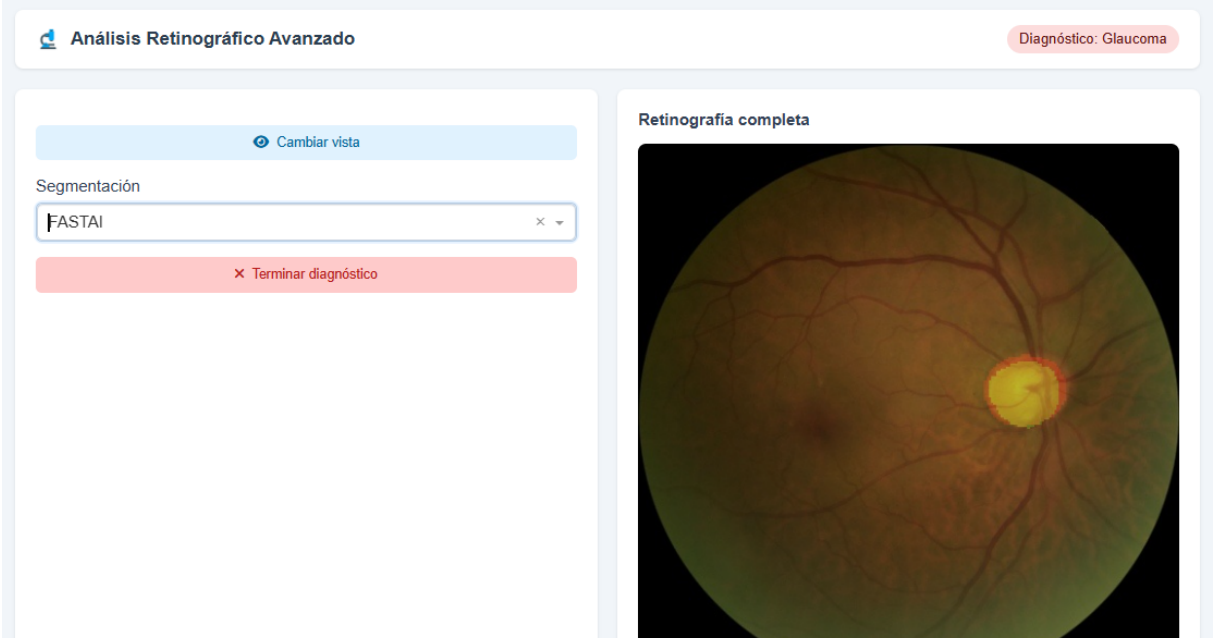


Figura 5.15: Interfaz de usuario del programa con la segmentación de FastAI.

5.3.3. Pruebas

Una vez implementada la aplicación, queda realizar las pruebas oportunas para comprobar su buen funcionamiento. Con este propósito, se ha decidido hacer pruebas de caja negra, en concreto pruebas de sistema, ya que se efectúan sobre el sistema completo. Se describen a continuación.

Caja negra - Test E2E (*End to end*)

En las tablas 5.6, 5.7, 5.8, 5.9 y 5.10 se proponen una serie de casos de prueba de caja negra para confirmar que cada caso de uso propuesto se ha implementado de acuerdo a los objetivos esperados:

PR-E2E_1	**Procesar diagnóstico**
Propósito	Comprobar si al subir una retinografía se aporta un diagnóstico
Datos de entrada	Imagen de una retinografía
Resultado esperado	Diagnóstico y muestra de la imagen
Resultado obtenido	Diagnóstico y muestra de la imagen

Tabla 5.6: Prueba de caja negra 1

PR-E2E_2	**Cambio entre la región ONH y la retinografía completa**
Propósito	Comprobar que el cambio entre la región ONH y la retinografía completa se realiza correctamente
Datos de entrada	-
Resultado esperado	Se cambia la imagen de la retinografía completa a la región ONH y viceversa
Resultado obtenido	Se cambia la imagen de la retinografía completa a la región ONH y viceversa

Tabla 5.7: Prueba de caja negra 2

PR-E2E_3	**Procesar diagnóstico**
Propósito	Comprobar que el botón de cálculo de segmentación cambia a una lista de selección
Datos de entrada	-
Resultado esperado	El botón de cálculo de segmentación cambia a una lista de selección
Resultado obtenido	El botón de cálculo de segmentación cambia a una lista de selección

Tabla 5.8: Prueba de caja negra 3

PR-E2E_4	**Seleccionar tipo de visualización**
Propósito	Comprobar si se muestra la segmentación elegida
Datos de entrada	Opción de los tipos de segmentación
Resultado esperado	Se muestra muestra la retinografía y la región ONH con la segmentación esperada
Resultado obtenido	Se muestra muestra la retinografía y la región ONH con la segmentación esperada

Tabla 5.9: Prueba de caja negra 4

PR-E2E_5	**Terminar diagnóstico**
Propósito	Comprobar que al terminar el diagnóstico se eliminan los datos generados y se vuelve a la página de inicio
Datos de entrada	-
Resultado esperado	Se eliminan los archivos de las retinografías y se regresa a la página inicial
Resultado obtenido	Se eliminan los archivos de las retinografías y se regresa a la página inicial

Tabla 5.10: Prueba de caja negra 5

Capítulo 6

Conclusiones y trabajo futuro

Como uno de los últimos puntos a tratar, se presentan las conclusiones que se han alcanzado a lo largo de este proyecto. En esta sección se aportará una reflexión crítica y global sobre el desarrollo del Trabajo de Fin de Grado. Para ello, se analiza el grado de cumplimiento de los objetivos propuestos inicialmente, así como la idoneidad de la metodología empleada en cada una de las fases del proyecto. Del mismo modo, se expone una valoración personal sobre la experiencia adquirida a lo largo del proceso, destacando tanto los aspectos positivos como los desafíos encontrados. Para finalizar, se incluye una perspectiva propia sobre la proyección de este proyecto, así como las vías futuras para mejorarlo.

6.1. Conclusiones

En esta parte del capítulo se analiza de manera objetiva el avance que ha tenido el proyecto en base a sus objetivos. Además, se aportará una valoración personal del mismo.

6.1.1. Perspectiva del proyecto

A continuación se va a justificar el cumplimiento de los objetivos propuestos en un primer momento, los cuales han sido satisfechos por completo:

- Se ha construido una aplicación que permite el diagnóstico del glaucoma a partir de retinografías mediante técnicas de *Deep Learning*, lo que se corresponde con el cumplimiento del objetivo **OBJ-01**.
- Se han entrenado modelos capaces de identificar el disco y la copa ópticos. Esto conlleva la consecución del objetivo **OBJ-02** al construir con ellos un módulo que emplea métodos de segmentación sobre las retinografías.
- Se ha estudiado el rendimiento de los modelos construidos para la clasificación y segmentación de retinografías respecto a las métricas más adecuadas para cada uno de ellos. Así, se cumple el objetivo **OBJ-03**.

Fruto de la realización de los objetivos anteriores, en este Trabajo Fin de Grado se ha conseguido elaborar una aplicación capaz de detectar el glaucoma con un rendimiento incluso superior a algunos trabajos científicos publicados recientemente. En concreto, del *dataset rotterdam* utilizado, los resultados obtenidos superan a todos los que se han consultado. Por otra parte, también mejora a otras soluciones mencionadas a lo largo de esta memoria como la de la Universidad de Tohoku [59]. Además, este desarrollo ha permitido identificar las estructuras propias de una retinografía. Con todo lo anterior, se han podido comprobar hipótesis como, por ejemplo, si la región del nervio óptico es la que mayor información reporta para la detección del glaucoma, o si la dimensión de la copa óptica aumenta en casos con esta misma patología.

Para llevar esto a cabo, siguiendo unas restricciones temporales y produciendo una solución de calidad, es necesario el empleo de una metodología que permita estructurar el trabajo para maximizar el rendimiento. En particular, las metodologías escogidas han sido SCORE y CRISP-DM. Ha sido un acierto contar con metodologías ágiles para la planificación temporal del proyecto, como ha sido SCORE, así como para estructurar el desarrollo de la solución, para lo que se ha utilizado CRISP-DM. Seguir estas metodologías ha permitido mantener un ritmo constante en la realización del proyecto, así como recibir una buena retroalimentación por parte de los tutores del TFG, permitiendo mejorar las soluciones construidas.

6.1.2. Perspectiva y valoración personal

La realización de este Trabajo de Fin de Grado ha sido una experiencia muy enriquecedora. El desarrollo del mismo me ha permitido profundizar en temas de mi interés en el campo del aprendizaje automático. En particular, he podido aprender técnicas de *Deep Learning* nuevas para mí, como las de segmentación, y profundizar en otras ya conocidas, como las de clasificación.

Por otra parte, el tema elegido para la realización del proyecto ha sido un completo acierto, pues me ha permitido aplicar mis conocimientos para resolver un problema real como es la detección del glaucoma, lo que ha despertado mi interés en la utilización de técnicas de aprendizaje profundo en el campo de la medicina. Además, resulta muy satisfactorio poder desarrollar una herramienta con un impacto social positivo como la que se ha construido para la consecución de los objetivos propuestos en el inicio del proyecto.

A colación de lo anterior, en un principio las expectativas se basaban en la consecución de los objetivos establecidos para resolver el problema de manera adecuada. Sin embargo, estas han ido cambiando durante el desarrollo. La obtención de unos resultados de alto valor que apuntaban a superar las soluciones existentes para la detección automática del glaucoma ha ido elevando las expectativas del producto final así como la motivación. Finalmente, estas expectativas se han cumplido, consiguiendo una herramienta que supera otras soluciones planteadas a este problema.

Como se ha explicado con anterioridad, todas las soluciones construidas para la detección del glaucoma mediante el *dataset rotterdam* presentan peores resultados que los obtenidos en el presente trabajo. No solo esto, sino que otros trabajos como del que se hace

eco el Huffington Post [57] de la Universidad de Tohoku [59], también obtienen métricas inferiores a las de este proyecto.

Además, como se ha mencionado anteriormente, se han adquirido y consolidado conocimientos técnicos. No solo esto, sino que los conocimientos adquiridos van más allá, pues he empleado metodologías con las que no había tenido la oportunidad de trabajar, aunque ya hubiese tratado con la metodología SCRUM de la que proceden. En general, puesto que este trabajo abarca áreas de conocimiento tan diversas como la ciencia de datos, la topología, el procesamiento de imágenes o la anatomía ocular, ha sido muy interesante poder aprender una gran cantidad de conceptos nuevos y tan diversos. Además de aprender conceptos teóricos y de uso de metodologías, he podido usar herramientas nuevas como la librería *dash* para poder mostrar los resultados de una manera más visual, así como la biblioteca *ultralytics* para entrenar modelos usando YOLO.

Queda destacar que durante la realización del presente TFG también se han presentado dificultades o problemas a solventar. Entre ellos, ha sido la mejora de las predicciones de las segmentaciones que se han mostrado en la Sección 4.4.3. En particular, los correspondientes a la segmentación del disco óptico con los modelos de FastAI son los que han supuesto un mayor reto. Otro problema que ha surgido, y quizá uno de los más relevantes pues ha obligado a retrasar el inicio del proyecto, ha sido la obtención del *dataset* de entrenamiento, pues en un principio se iba a disponer de un conjunto de datos al que finalmente se denegó su acceso. Por otra parte, cabe destacar la incertidumbre de realizar una planificación durante un curso académico en el que se tiene que atender a otras obligaciones como exámenes o prácticas, lo que dificulta las cosas. Para solventar este problema ha sido de vital importancia contar con metodologías ágiles que permiten adaptarse mejor a la situación.

Espero que este proyecto pueda tener un impacto positivo con su implementación en centros clínicos para facilitar la detección del glaucoma y conseguir una herramienta de apoyo fundamental para los especialistas de la salud. Resulta algo conmovedor el valor que puede tener el buen uso de la Inteligencia Artificial para facilitar y mejorar la vida de las personas, y en particular, de su salud.

6.2. Trabajo futuro

En el contexto de este Trabajo Fin de Grado, resulta especialmente relevante señalar que, al igual que sucede con otros proyectos de software, requiere de una mejora y revisión continuas. Esto se debe a que los sistemas basados en aprendizaje automático dependen en gran medida de los datos disponibles. También de otros factores como la evolución de las arquitecturas, pues sin ir más lejos, en este proyecto se ha utilizado YOLO 11 y recientemente se ha actualizado a una versión YOLO 12. Esto implica la necesidad constante de actualizar y validar los modelos construidos.

Por otra parte, este proyecto puede utilizarse como punto de partida para desarrollos más complejos. En consecuencia, este trabajo no debe considerarse un producto final cerrado, sino una base sobre la que seguir construyendo para alcanzar soluciones más

precisas, robustas y aplicables en el ámbito clínico. A continuación, se presentan algunas vías de investigación futuras y otras acciones oportunas, así como mejoras que se pueden implementar a partir de la investigación aquí realizada:

- Disponer de un *dataset* real de las retinografías del centro médico donde se va a usar la solución. En un inicio, esta era la fuente de donde se iban a tomar los datos, pero el Comité de Ética de Investigación Clínica del Área de Salud de Valladolid no concedió la solicitud. De esta forma, hubo que prescindir de esta vía por motivos temporales, ya que solicitarlo a la Consejería de Sanidad retrasaría mucho los tiempos previstos.

  Sería de gran utilidad tener acceso a estos datos para poder adaptar la solución a las imágenes tomadas con los dispositivos del Hospital Clínico. En general, aumentar y diversificar el dataset incorporando retinografías de distintas poblaciones, dispositivos y calidades de imagen tendrá implicaciones positivas sobre el rendimiento del modelo.

- Integrar la aplicación construida en el entorno médico-hospitalario; es decir, realizar un despliegue en clínicas o centros de atención primaria como herramienta de apoyo para el diagnóstico del glaucoma.

- Llevar a cabo una validación clínica. Para comprobar la eficacia del modelo construido, sería de gran utilidad estudiar su rendimiento con pacientes reales y adaptarlo para mejorar su capacidad de predicción. Además, también sería de interés comparar su rendimiento frente al de oftalmólogos.

- Predicción de la evolución del glaucoma. Esta sería una nueva vía de investigación. El objetivo de esta mejora sería avanzar desde la detección temprana del glaucoma hacia la predicción de su evolución en el tiempo. De este modo, se trataría de desarrollar un modelo capaz de estimar la velocidad o el patrón de progresión del glaucoma, permitiendo mejorar la toma de decisiones clínicas, como ajustar la frecuencia de las revisiones o personalizar los tratamientos.

- Clasificación de tipos de glaucoma. Existe más de un tipo de glaucoma, como el de ángulo abierto o el de ángulo cerrado [4]. En esta investigación se ha primado el hecho de distinguir pacientes sanos de los que presentan glaucoma. En una segunda versión se podría tratar de diferenciar qué tipo de glaucoma se presenta. Una opción de adaptarlo a los modelos construidos sería entrenar un modelo que distinga las clases de glaucoma. Con el modelo actual, si se predice la presencia de la patología, se pasaría la imagen a este segundo modelo para clasificar de qué tipo de glaucoma se trata.

- Desarrollar un plan para introducir a los oftalmólogos en el uso de la herramienta.

- Implementar técnicas de *Active Learning* o aprendizaje activo [61]. Con estos métodos, el modelo enviaría los casos más inciertos a un experto para su revisión. Una

vez fueran validados, esos casos se usarían para seguir entrenando al modelo. De esta forma, se promueve un aprendizaje continuo con la consiguiente mejora del modelo.

Otra opción que se podría implementar consistiría en un método de entrenamiento continuo en el que el sistema se pueda actualizar continuamente a medida que se confirma si las predicciones son correctas o no, para tratar de mejorar los resultados.

Parte III

Apéndices

Apéndice A

Manual de instalación

La aplicación se presenta en un cuaderno *Jupyter Notebook* de extensión `.ipynb` denominado `app.ipynb`. Para poder poner en marcha la misma, basta con ejecutar de manera secuencial los cuadros de código del cuaderno mencionado. En concreto, el proceso que se sigue durante esta ejecución secuencial es:

1. Instalar las librerías necesarias: *ultralytics*, *dash* y *dash-iconify*. Para ello se utilizan los comandos:

   ```
   pip install ultralytics
   ```

   ```
   pip install dash dash-iconify
   ```

   Además, se importan las bibliotecas imprescindibles con:

   ```
   from google.colab import drive
   ```

   ```
   from fastai.vision.all import *
   ```

   ```
   from ultralytics import YOLO
   ```

   ```
   import cv2
   ```

   ```
   import numpy as np
   ```

   ```
   import os
   ```

   ```
   from scipy.spatial import ConvexHull
   ```

   ```
   import joblib
   ```

   y se definen las rutas a las que se accederá para importar los modelos utilizados en la aplicación. Todo esto corresponde con el punto 1 del archivo `app.ipynb`.

2. Ejecutar el código imprescindible para importar los modelos. YOLO no requiere ningún código adicional. Por su parte para FastAI se deben definir las métricas y otros datos que usaron los modelos. Esto corresponde con los puntos 2 y 3.1 del archivo `app.ipynb`.

3. Se define la funcionalidad que utilizará la aplicación para realizar los diagnósticos y las segmentaciones. Lo que corresponde con el punto 3.2 del fichero `app.ipynb`.

4. Se inicia la aplicación y se está en disposición de usarla. Para ello, se ejecuta el código del punto 3.3 del archivo `app.ipynb` o del punto 3.4 del mismo fichero en función de si se quiere utilizar o no FastAI en las segmentaciones. La primera de las opciones es mucho más rápida.

Nota. En el archivo `app.ipynb` se muestran dos opciones de ejecución de la aplicación. Son iguales, pero en una se ha quitado la opción de segmentar con los modelos de FastAI para mejorar el rendimiento. Estos modelos tienen una gran cantidad de parámetros y pueden resultar lentos. En la Sección 5.3.2 se muestra un ejemplo de los resultados producidos con esta opción. Tras el apartado de ejecución de la aplicación sin FastAI, se puede ejecutar la aplicación con FastAI en la sección siguiente del archivo `app.ipynb`.

Apéndice B

Contenido adjunto

En esta sección se recopilan los archivos y recursos adicionales que complementan el desarrollo del proyecto. Estos elementos incluyen las implementaciones de código, gráficas obtenidas y otros materiales relevantes que han sido generados o utilizados durante la elaboración del trabajo, como es el caso de los *datasets* construidos para el entrenamiento de los modelos.

Comencemos por explicar los archivos que corresponden con el código implementado. El proyecto consta de dos cuadernos de *Jupyter Notebooks*, los cuáles se describen a continuación:

- `app.ipynb`. Este cuaderno de *Jupyter* contiene el código necesario para inicializar la aplicación. De esta forma, como se explicó en el apéndice A, ejecutando este archivo de manera secuencial se puede probar la aplicación desarrollada. Además, se puede observar el funcionamiento de los modelos construidos para la propuesta de la solución, ya que aquí se integran todos los seleccionados por tener un buen rendimiento.

  Por otra parte, en este cuaderno se incluyen dos versiones de la aplicación. En la primera, no se incluyen los modelos de segmentación con FastAI. Esta es mucho más rápida que la segunda, la cual también contiene los modelos de segmentación de FastAI.

- `development.ipynb`. En este cuaderno se incluye el proceso seguido para la construcción de la solución. Para organizarlo de una mejor manera, este archivo se divide en distintos puntos, en los que se explica:

  - **Solución mediante clasificación de retinografías con Deep Learning**. En este apartado se incluye el entrenamiento de todos los modelos destinados a clasificación mediante *Deep Learning*. Además, también se recogen las métricas para cada uno de ellos.
  - **Solución mediante segmentación de retinografías con Deep Learning y clasificación con Machine Learning**. En este apartado se incluye todo

lo referido al problema de segmentación. Costa de dos partes principales. En la primera, los modelos destinados a segmentación construidos tanto mediante YOLO como mediante FastAI. Por otra parte, también se incluye un apartado en el que se aborda el problema de clasificación mediante *Machine Learning*. Cabe mencionar que para todos los modelos de esta sección del cuaderno también se recogen las métricas para cada uno de ellos.

- **Propuesta de implementación**. En este apartado se construyen los ensembles descritos en la presente memoria a lo largo de la Sección 4.7. Además, también se recoge el proceso para desarrollar la aplicación final que se tiene en el archivo `app.ipynb` explicada en el Capítulo 5.

Dada la cantidad de memoria necesaria para almacenar todo el contenido generado a lo largo del TFG, en la carpeta habilitada para subir el contenido adjunto se suben los dos cuadernos de *Jupyter Notebooks* mencionados y los *datasets* originales. Estos siguen la siguiente estructura:

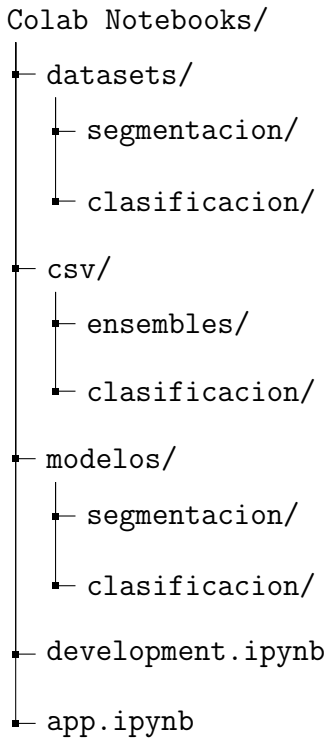
```
GRP-GestionINF5G - TFG/
├─ datasets/
│  ├─ drishti-gs/
│  ├─ rim-one/
│  └─ rotterdam/
├─ development.ipynb
└─ app.ipynb
```

Por otra parte, para acceder al contenido completo, se tiene una cuenta de Google Drive a la que se podrá acceder en caso de querer ejecutar la aplicación desarrollada; es decir, el archivo `app.ipynb`. En este mismo lugar, también se disponen todos los *datasets* con las modificaciones hechas para todos los tipos de entrenamiento tratados.

Para acceder a la cuenta de Google Drive se debe utilizar el siguiente correo y contraseña:

- Correo: **tfg.informatica.carlos@gmail.com**
- Contraseña: **TFG_informatica_jvc_24-25**

A continuación se describe la estructura de directorios que se tiene en Google Drive:



Aquí, se tienen los *datasets* utilizados para entrenar todos los modelos, los archivos csv construidos para entrenar los ensembles y la clasificación mediante *Machine Learning* a partir de segmentación, los modelos seleccionados a lo largo del proyecto y los dos cuadernos de Jupyter que se acaban de explicar.

Bibliografía

[1] *Los dispositivos electrónicos y la vista*, 15 de enero de 2025. Disponible en: https://www.aao.org/salud-ocular/consejos/los-dispositivos-electr%C3%B3nicos-y-la-vista

[2] *¿Qué es el glaucoma? Causas, síntomas, diagnóstico, tratamiento*, 16 de enero de 2025. Disponible en: https://www.aao.org/salud-ocular/enfermedades/que-es-la-glaucoma

[3] *¿Qué es el Glaucoma? — Sociedad Española de Glaucoma*. Disponible en: https://www.sociedadglaucoma.com/que-es-el-glaucoma/

[4] Institut Català de Retina, *Glaucoma, Causas, tipos, riesgos y tratamiento — ICR*, 17 de febrero de 2025. Disponible en: https://icrcat.com/enfermedades-oculares/glaucoma/

[5] Esteva A. et al., *Dermatologist-level classification of skin cancer with deep neural networks*, Nature, vol. 542, pp. 115–118, 25 de enero de 2017. DOI: https://doi.org/10.1038/nature21056

[6] Litjens G. et al., *A survey on deep learning in medical image analysis*, Medical Image Analysis, vol. 42, pp. 60–88, 26 de julio de 2017. DOI: https://doi.org/10.1016/j.media.2017.07.005

[7] Gulshan V. et al., *Development and Validation of a Deep Learning Algorithm for Detection of Diabetic Retinopathy in Retinal Fundus Photographs*, JAMA, vol. 316, no. 22, p. 2402, 29 de noviembre de 2016. DOI: https://doi.org/10.1001/jama.2016.17216

[8] Ting D.S.W. et al., *Artificial intelligence and deep learning in ophthalmology*, PubMed, 1 de febrero de 2019. DOI: https://doi.org/10.1136/bjophthalmol-2018-313173

[9] *Glaucoma Dataset: EyePACS-AIROGS-Light-V2*, 9 de marzo de 2024. Disponible en: https://www.kaggle.com/datasets/deathtrooper/glaucoma-dataset-eyepacs-airogs-light-v2

[10] *Glaucoma detection*, 13 de julio de 2022. Disponible en: https://www.kaggle.com/datasets/sshikamaru/glaucoma-detection

[11] *Drishti-GS - RETINA DATASET FOR ONH SEGMENTATION*, 18 de octubre de 2021. Disponible en: https://www.kaggle.com/datasets/lokeshsaipureddi/drishtigs-retina-dataset-for-onh-segmentation

[12] Fumero F.J. et al., *RIM-ONE DL: A Unified Retinal Image Database for Assessing Glaucoma Using Deep Learning*, Image Analysis & Stereology, vol. 39, no. 3, pp. 161–167, 25 de noviembre de 2020. DOI: https://www.ias-iss.org/ojs/IAS/article/view/2346

[13] Miag-Ull, *GitHub - miag-ull/rim-one-dl*. Disponible en: https://github.com/miag-ull/rim-one-dl?tab=readme-ov-file

[14] *SMDG, a standardized Fundus Glaucoma dataset*, 23 de abril de 2023. Disponible en: https://www.kaggle.com/datasets/deathtrooper/multichannel-glaucoma-benchmark-dataset

[15] Schwaber K., Sutherland J., *The scrum guide. The definitive guide to scrum: The rules of the game*. Disponible en: https://scrumguides.org/docs/scrumguide/v2020/2020-Scrum-Guide-US.pdf

[16] Marapaulaflores M., *Roles en metodología SCRUM ¿Cuáles son y por qué son tan importantes? – Manuales Datlas*, 31 de marzo de 2023. Disponible en: https://blogdatlas.wordpress.com/2023/04/02/roles-en-metodologia-scrum-cuales-son-y-por-que-son-tan-importantes-manuales-datlas/

[17] Wirth R., Hipp J., *CRISP-DM: Towards a Standard Process Model for Data Mining*

[18] Aoun L., *Do you need a data scientist? - AstroLabs - Medium*, 28 de enero de 2018. Disponible en: https://medium.com/astrolabs/do-you-need-a-data-scientist-ea4d79974f9c

[19] Hicks M., Foster J.S., *SCORE*, Communications of the ACM, vol. 53, no. 10, pp. 30–31, 1 de octubre de 2010. DOI: https://doi.org/10.1145/1831407.1831421

[20] Russell S.J., Norvig P., *Artificial Intelligence: A Modern Approach*, 1995.

[21] China C.R., *Tipos de machine learning*, 4 de abril de 2025. Disponible en: https://www.ibm.com/es-es/think/topics/machine-learning-types

[22] Gurney K., *An Introduction to Neural Networks*, CRC Press, 1997.

[23] Bregón Bregón A., *Apuntes tema 4 Sistemas Inteligentes*, 2023-2024.

[24] Bregón Bregón A., *Apuntes tema 5 Sistemas Inteligentes*, 2023-2024.

[25] Bregón Bregón A., *Apuntes tema 6 Sistemas Inteligentes*, 2023-2024.

[26] Bregón Bregón A., *Apuntes tema 7 Sistemas Inteligentes*, 2023-2024.

[27] Bregón Bregón A., *Apuntes tema 8 Sistemas Inteligentes*, 2023-2024.

[28] China C.R., *Tipos de machine learning*, 4 de abril de 2025. Disponible en: https://www.ibm.com/es-es/think/topics/machine-learning-types

[29] IBM, *Aprendizaje no supervisado*, 5 de junio de 2025. Disponible en: https://www.ibm.com/es-es/topics/unsupervised-learning

[30] Sun J., Du W., Shi N., *A Survey of kNN Algorithm*, Information Engineering and Applied Computing, vol. 1, no. 1, 10 de mayo de 2018. DOI: https://doi.org/10.18063/ieac.v1i1.770

[31] Cortes C., Vapnik V., *Support-vector networks*, Machine Learning, vol. 20, no. 3, pp. 273–297, 1 de septiembre de 1995. DOI: https://doi.org/10.1007/bf00994018

[32] Jin X., Han J., *K-Means clustering*, Encyclopedia of Machine Learning and Data Mining, pp. 695–697, 2017. DOI: https://doi.org/10.1007/978-1-4899-7687-1_431

[33] Maćkiewicz A., Ratajczak W., *Principal components analysis (PCA)*, Computers & Geosciences, vol. 19, no. 3, pp. 303–342, 1 de marzo de 1993. DOI: https://doi.org/10.1016/0098-3004(93)90090-r

[34] Bank D., Koenigstein N., Giryes R., *Autoencoders*, arXiv (Cornell University), 2020. DOI: https://arxiv.org/abs/2003.05991

[35] Stewart G.W., *On the Early History of the Singular Value Decomposition*, SIAM Review, vol. 35, no. 4, pp. 551–566, 1 de diciembre de 1993. DOI: https://doi.org/10.1137/1035134

[36] *Documento sin título*. Disponible en: https://sociedadoftalmologicademadrid.com/revistas/revista-2013/m2013-03.html

[37] Cadena Ser C., *Cadena SER*, 11 de marzo de 2025. Disponible en: https://cadenaser.com/comunitat-valenciana/2025/03/11/el-hospital-universitario-del-vinalopo-de-elche-diagnostica-mas-de-400-casos-de-glaucoma-cada-ano-radio-elche/

[38] Heydari A., *Glaucoma Facts and Stats - Glaucoma Research Foundation*, 16 de mayo de 2025. Disponible en: https://glaucoma.org/articles/glaucoma-facts-and-stats

[39] Central Ocular, *Retinografía: Qué es y cuando está indicada - Central Ocular*, 16 de junio de 2025. Disponible en: https://www.centralocular.com/que-es-una-retinografia-cuando-esta-indicada-diferencias-con-angiografia/

[40] IMO Grupo Miranza, *Pruebas diagnósticas - Retinografía — IMO*, 1 de julio de 2022. Disponible en: https://www.imo.es/pruebas-diagnosticas/retinografia/

[41] IBM, *Sobreajuste*, 23 de enero de 2025. Disponible en: https://www.ibm.com/es-es/think/topics/overfitting

[42] *Sobreajuste: regularización L2*. Disponible en: https://developers.google.com/machine-learning/crash-course/overfitting/regularization?hl=es-419

[43] Taunk D., *L1 vs L2 Regularization: The intuitive difference - Analytics Vidhya - Medium*, 22 de enero de 2024. Disponible en: https://medium.com/analytics-vidhya/l1-vs-l2-regularization-which-is-better-d01068e6658c

[44] Yadav H., *Dropout in Neural Networks*, 21 de enero de 2025. Disponible en: https://towardsdatascience.com/dropout-in-neural-networks-47a162d621d9/

[45] *Métricas de evaluación de modelos en el aprendizaje automático*, 25 de septiembre de 2023. Disponible en: https://www.datasource.ai/es/data-science-articles/metricas-de-evaluacion-de-modelos-en-el-aprendizaje-automatico

[46] *Classification: Accuracy, recall, precision, and related metrics*. Disponible en: https://developers.google.com/machine-learning/crash-course/classification/accuracy-precision-recall

[47] Hui J., *mAP (mean Average Precision) for Object Detection - Jonathan Hui - Medium*, 6 de febrero de 2020. Disponible en: https://jonathan-hui.medium.com/map-mean-average-precision-for-object-detection-45c121a31173

[48] GeeksforGeeks, *Mean Average Precision (mAP) in Computer Vision*, 3 de abril de 2025. Disponible en: https://www.geeksforgeeks.org/mean-average-precision-map-in-computer-vision/

[49] C. Cubiwan, *Convertir RGB a escala de grises*, 2018. [En línea]. Disponible en: https://construyendoachispas.blog/2016/08/02/convertir-rgb-a-escala-de-grises/.

[50] Colaboradores de Wikipedia, *Filtro de gabor*, 2021. [En línea]. Disponible en: https://es.wikipedia.org/wiki/Filtro_de_Gabor.

[51] A. Akter, N. Nosheen, S. Ahmed, M. Hossain, M.A. Yousuf, M.A.A. Almoyad, K.F. Hasan, M.A. Moni, *Robust clinical applicable CNN and U-Net based algorithm for MRI classification and segmentation for brain tumor*, Expert Systems with Applications, vol. 238, p. 122347, 2023. DOI: 10.1016/j.eswa.2023.122347. [En línea]. Disponible en: https://doi.org/10.1016/j.eswa.2023.122347.

[52] A. Esteva, B. Kuprel, R.A. Novoa, J. Ko, S.M. Swetter, H.M. Blau, S. Thrun, *Dermatologist-level classification of skin cancer with deep neural networks*, Nature, vol. 542, no. 7639, pp. 115–118, 2017. DOI: 10.1038/nature21056. [En línea]. Disponible en: https://pmc.ncbi.nlm.nih.gov/articles/PMC8382232//#F6.

[53] J.R.H. Lee, M. Pavlova, M. Famouri, A. Wong, *Cancer-Net SCa: tailored deep neural network designs for detection of skin cancer from dermoscopy images*, BMC Medical Imaging, vol. 22, no. 1, 2022. DOI: 10.1186/s12880-022-00871-w. [En línea]. Disponible en: https://doi.org/10.1186/s12880-022-00871-w.

[54] N. Nyecc, *Iris tumors*, 2016. [En línea]. Disponible en: https://eyecancer.com/eye-cancer/image-galleries/iris-tumors/.

[55] *Seguridad social: cotización / recaudación de trabajadores*, [En línea]. Disponible en: https://www.seg-social.es/wps/portal/wss/internet/Trabajadores/CotizacionRecaudacionTrabajadores/36537#36538.

[56] J.R. Munkres, *Topology*, 2018.

[57] Redacción HuffPost, *Revolución entre los oftalmólogos: científicos hallan la manera de detectar al "ladrón silencioso de la vista"*, 2025. [En línea]. Disponible en: https://www.huffingtonpost.es/life/salud/revolucion-oftalmologos-cientificos-hallan-forma-detectar-ladron-silencioso-vista-rp.html.

[58] *AI revolutionizes glaucoma care: Tohoku University develops Specialist-Level Screening System*, [En línea]. Disponible en: https://www.tohoku.ac.jp/en/press/ai_revolutionizes_glaucoma_care_tohoku_university_develops_specialist_level_screening_system.html.

[59] P. Sharma, N. Takahashi, T. Ninomiya, M. Sato, T. Miya, S. Tsuda, T. Nakazawa, *A hybrid multi model artificial intelligence approach for glaucoma screening using fundus images*, npj Digital Medicine, vol. 8, no. 1, 2025. DOI: 10.1038/s41746-025-01473-w. [En línea]. Disponible en: https://www.nature.com/articles/s41746-025-01473-w.

[60] Wikipedia contributors, *You only look once*, 2025. [En línea]. Disponible en: https://en.wikipedia.org/wiki/You_Only_Look_Once.

[61] J. Murel, E. Kavlakoglu, *Aprendizaje por transferencia*, 2025. [En línea]. Disponible en: https://www.ibm.com/es-es/think/topics/transfer-learning.

[62] GeeksforGeeks, *Fast Fourier Transform in Image Processing*, 2025. [En línea]. Disponible en: https://www.geeksforgeeks.org/fast-fourier-transform-in-image-processing/.

[63] Colaboradores de Wikipedia, *Método del codo (agrupamiento)*, 2024. [En línea]. Disponible en: https://es.wikipedia.org/wiki/M%C3%A9todo_del_codo_(agrupamiento).

[64] GeeksforGeeks, *UNET Architecture explained*, 2025. [En línea]. Disponible en: https://www.geeksforgeeks.org/u-net-architecture-explained/.

[65] A.I. Aramendia, *The U-Net : A Complete Guide — Medium*, 2025. [En línea]. Disponible en: https://medium.com/@alejandro.itoaramendia/decoding-the-u-net-a-complete-guide-810b1c6d56d8.

[66] J. Redmon, S. Divvala, R. Girshick, A. Farhadi, *You only look once: Unified, Real-Time Object Detection*, 2015. [En línea]. Disponible en: https://arxiv.org/abs/1506.02640.

[67] J. Pedro, *Detailed Explanation of YOLOv8 Architecture — Part 1*, 2023. [En línea]. Disponible en: https://medium.com/@juanpedro.bc22/detailed-explanation-of-yolov8-architecture-part-1-6da9296b954e.

[68] Ultralytics, *Architecture Summary*, 2025. [En línea]. Disponible en: https://docs.ultralytics.com/yolov5/tutorials/architecture_description/.

[69] GeeksforGeeks, *Darknet 53*, 2024. [En línea]. Disponible en: https://www.geeksforgeeks.org/darknet-53/.

[70] N. Neville, *Spatial pyramid pooling explained - Neville - medium*, 2024. [En línea]. Disponible en: https://cvinvolution.medium.com/spatial-pyramid-pooling-explained-3a1dd9ddf661.

[71] M. R., *PANET: Path Aggregation Network in YOLOV4 - Clique Community - Medium*, 2021. [En línea]. Disponible en: https://medium.com/clique-org/panet-path-aggregation-network-in-yolov4-b1a6dd09d158.

[72] Ultralytics, *Computer Vision Tasks supported by Ultralytics YOLO11*, 2025. [En línea]. Disponible en: https://docs.ultralytics.com/es/tasks/.

[73] *Sueldos para el puesto de Gestor de Proyectos TIC en España*, [En línea]. Disponible en: https://www.glassdoor.es/Sueldos/gestor-de-proyectos-tic-sueldo-SRCH_KO0,23.htm.

[74] *¿Cuánto cobra un analista programador? (Sueldo 2025) — Jobted.es*, [En línea]. Disponible en: https://www.jobted.es/salario/analista-programador.

[75] *¿Cuánto cobra un data scientist? (Sueldo 2025) — Jobted.es*, [En línea]. Disponible en: https://www.jobted.es/salario/data-scientist.

[76] *¿Cuánto cobra un programador? (Sueldo 2025) — Jobted.es*, [En línea]. Disponible en: https://www.jobted.es/salario/programador.

[77] M. Mucha, F. Derma, T. Pál, *Calculadora de salario por hora*, 2025. [En línea]. Disponible en: https://www.omnicalculator.com/es/finanzas/salario-por-hora.

[78] D. Deathtrooper, *Benchmark-94.16 % (Xception, noise and preprocess)*, 2023. [En línea]. Disponible en: https://www.kaggle.com/code/deathtrooper/benchmark-94-16-xception-noise-and-preprocess.

[79] S. Shirshaka, *BPEYE Glaucoma Test*, 2024. [En línea]. Disponible en: https://www.kaggle.com/code/shirshaka/bpeye-glaucoma-test.

[80] D. Deathtrooper, *Keras; Benchmark-94.94 % (ConvNEXTTiny)*, 2023. [En línea]. Disponible en: https://www.kaggle.com/code/deathtrooper/keras-benchmark-94-94-convnexttiny.

[81] D. Deathtrooper, *Glaucoma classification easy setup, 88.9 % baseline*, 2023. [En línea]. Disponible en: https://www.kaggle.com/code/deathtrooper/glaucoma-classification-easy-setup-88-9-baseline.

[82] Kibo. *Degeneración macular*. 18 de enero de 2021. Disponible en: https://tusdudasdesalud.com/vision/enfermedades-ojo/degeneracion-macular/

[83] Sociedad de Cirugía Ocular. *Retina y Vítreo - Sociedad de Cirugía Ocular*. 22 de diciembre de 2023. Disponible en: https://sociedadcirugiaocular.com/cirugias/retina-y-vitreo/

[84] *Anatomía de la Retina*. Disponible en: https://www.doctordiegoruizcasas.com/anatomia-globo-ocular/retina

[85] a2ECFcZPlw. *K Nearest Neighbour (KNN) – Distance Metrics with Python Sklearn*. 25 de noviembre de 2023. Disponible en: https://infoaryan.com/blog/understanding-k-nearest-neighbour-knn-python-sklearn/

[86] *Decision Trees*. Disponible en: https://www.researchgate.net/figure/Example-of-a-supervised-machine-learning-algorithm-a-decision-tree-Decision-trees-come_fig1_347070799

[87] *Regresión logística: Cómo calcular una probabilidad con la función sigmoidea*. Disponible en: https://developers.google.com/machine-learning/crash-course/logistic-regression/sigmoid-function?hl=es-419

[88] Birch. *Photo by Carl Warner - SlideServe*. 21 de marzo de 2019. Disponible en: https://www.slideserve.com/birch/photo-by-carl-warner-powerpoint-ppt-presentation

[89] Yosvani Orlando Lao-León, Ariam Rivas-Méndez, Milagros Caridad Pérez-Pravia, Fernando Marrero-Delgado. *Procedimiento para el pronóstico de la demanda mediante redes neuronales artificiales*. 2017. Disponible en: https://www.redalyc.org/journal/1815/181549596004/html/

[90] Colaboradores de Wikipedia. *Archivo:Perceptrón 5 unidades.svg - Wikipedia, la enciclopedia libre*. Disponible en: https://es.m.wikipedia.org/wiki/Archivo:Perceptr%C3%B3n_5_unidades.svg

[91] Imad Dabbura. *Imad Dabbura - Transformer Architecture explained*. Disponible en: https://imaddabbura.github.io/posts/nlp/Transformer-Architecture-Explained.html

[92] Jose Cuartas. *El concepto de la convolución en gráficos, para comprender las Convolutional Neural Networks (CNN) o redes convolucionadas*. 29 de diciembre de 2021. Disponible en: https://josecuartas.medium.com/el-concepto-de-la-convoluci%C3%B3n-en-gr%C3%A1ficos-para-comprender-las-convolutional-neural-networks-cnn-519d2eee009c

[93] Debasish Kalita. *Basics of CNN in Deep Learning*. 1 de mayo de 2025. Disponible en: https://www.analyticsvidhya.com/blog/2022/03/basics-of-cnn-in-deep-learning/

[94] Douglas Karr. *What Is Artificial Intelligence? A Comprehensive Guide For Business Professionals — Martech Zone*. 16 de diciembre de 2023. Disponible en: https://martech.zone/what-is-artificial-intelligence/

[95] Aswini Kumar Samantaray, Amol D. Rahulkar. *Design of Adaptive Gabor Wavelet Filter Banks*. En: *Design of Adaptive Gabor Wavelet Filter Banks*. 1 de enero de 2024, pp. 131–146. DOI: 10.1007/978-3-031-57279-1_7. Disponible en: https://doi.org/10.1007/978-3-031-57279-1_7