UNIVERSIDAD PRIVADA ANTENOR ORREGO

FACULTAD DE INGENIERÍA

PROGRAMA DE ESTUDIO DE INGENIERÍA ELECTRÓNICA

TESIS PARA OPTAR EL TÍTULO PROFESIONAL DE

INGENIERO ELECTRÓNICA

Desarrollo de un sistema de diagnóstico en la detección temprana de cataratas utilizando redes neuronales convolucionales

---

Línea de investigación: Sistemas cognitivos

Autor/Autora/Autores:

Infante Cueva Wilder Pablo

Cruzado Benites Marco Salvador

Jurado evaluador:

Presidente : Linares Vertiz, Saul Noe

Secretario : Vargas Diaz, Luis Alberto

Vocal : Llanos Leon, Lenin Humberto

Asesor: Alvarado Rodríguez Luis Enrique

Código Orcid: https://orcid.org/0000-0001-6444-2922

Trujillo–Perú

2024

Fecha de Sustentación: año/mes/día

Informe Turnitin (PRIMERA Y ULTIMA HOJA DONDE INDICA EL PORCENTAJE, AMBAS HOJAS FIRMADAS POR EL ASESOR)

Desarrollo de un sistema de diagnóstico en la detección temprana de cataratas utilizando redes neuronales convolucionales

por PABLO MARCO CRUZADO BENITES

Fecha de entrega: 09-jul-2024 04:43p.m. (UTC-0500)
Identificador de la entrega: 2412826450
Nombre del archivo: na_de_cataratas_utilizando_redes_neuronales_convolucionales.docx (422.33K)
Total de palabras: 8343
Total de caracteres: 48479

Ing. Luis Alvarado Rodriguez
DNI: 43344790

Desarrollo de un sistema de diagnóstico en la detección temprana de cataratas utilizando redes neuronales convolucionales

INFORME DE ORIGINALIDAD

6%	6%	%	%
INDICE DE SIMILITUD	FUENTES DE INTERNET	PUBLICACIONES	TRABAJOS DEL ESTUDIANTE

FUENTES PRIMARIAS

1	www.medmultilingua.com Fuente de Internet	4%
2	hdl.handle.net Fuente de Internet	2%

Ing. Luis Alvarado Rodriguez
DNI: 43344790

Excluir citas	Apagado	Excluir coincidencias	< 2%
Excluir bibliografía	Activo		

Jurado de sustentación Oral

Linares Vertiz, Saul Noe

N° CIP 142213

Presidente

Vargas Diaz, Luis Alberto

N° CIP 104675

Secretario

Llanos Leon, Lenin Humberto

N° CIP 139213

Vocal

Entregado el:

Aprobado por:

Infante Cueva Wilder Pablo

DNI 44990809

Cruzado Benites Marco Salvador

DNI 42376849

Alvarado Rodríguez Luis Enrique

Asesor de Tesis

UNIVERSIDAD PRIVADA ANTENOR ORREGO

FACULTAD DE INGENIERÍA

PROGRAMA DE ESTUDIO DE INGENIERÍA ELECTRÓNICA

TESIS PARA OPTAR EL TÍTULO PROFESIONAL DE

INGENIERO ELECTRÓNICA

Desarrollo de un sistema de diagnóstico en la detección temprana de cataratas utilizando redes neuronales convolucionales

---

Línea de investigación: Sistemas cognitivos

Autor/Autora/Autores:

Infante Cueva Wilder Pablo

Cruzado Benites Marco Salvador

Jurado evaluador:

Presidente : Linares Vertiz, Saul Noe

Secretario : Vargas Diaz, Luis Alberto

Vocal : Llanos Leon, Lenin Humberto

Asesor:Alvarado Rodríguez Luis Enrique

Código Orcid: 0000-0001-6444-2922

Trujillo–Perú

2024

Fecha de Sustentación: año/mes/día

DECLARACION DE ORIGINALIDAD

Yo, Alvarado Rodríguez Luis Enrique, docente del Programa de Estudio de Pregrado de la Universidad Privada Antenor Orrego, asesor de la tesis titulada “Desarrollo de un sistema de diagnóstico en la detección temprana de cataratas utilizando redes neuronales convolucionales”, de los autores Infante Cueva Wilder Pablo y Cruzado Benites Marco Salvador

• El mencionado documento tiene un índice de puntuación de similitud del 6%. Así lo consigna el reporte de similitud emitido por el software Turnitin el día 09de julio del 2024

• He revisado con detalle dicho reporte de la tesis “Desarrollo de un sistema de diagnóstico en la detección temprana de cataratas utilizando redes neuronales convolucionales.” y no se advierte indicios de plagio.

• Las citas a otros autores y sus respectivas referencias cumplen con las normas establecidas por la Universidad.

Ciudad y fecha: 17 de julio 2024

Alvarado Rodríguez Luis Enrique
DNI: 43344790
ORCID: 0000-0001-6444-2922

Infante Cueva Wilder Pablo
DNI: 44990809
FIRMA:

GRADUADO

Cruzado Benites Marco Salvador

DNI: 42376849

FIRMA:

Dedicatoria

A mis padres, quienes con su amor,
apoyo incondicional y constantes
sacrificios han sido la base de mi
formación y logros.
Gracias por enseñarme el valor del
esfuerzo y la perseverancia.
Este logro es tanto mío como suyo.

Infante Cueva Wilder Pablo

A mi esposa, cuya paciencia, comprensión y
constante motivación me han impulsado a seguir
adelante en cada etapa de este proyecto. Gracias
por ser mi roca y mi fuente inagotable de
inspiración.
Este trabajo es para ti.

Cruzado Benites Marco Salvador

Agradecimientos

Agradezco profundamente a mis asesor,
el Ing. Luis Alvarado, por su guía experta,
sus valiosas sugerencias y su apoyo
constante durante todo el proceso
de elaboración de esta tesis.
Sin su orientación y dedicación,
este proyecto no hubiera sido posible.

Infante Cueva Wilder Pablo

Expreso mi gratitud a mi compañero
de investigación , quienes con su colaboración,
ideas y esfuerzo compartido, contribuyeron
significativamente al desarrollo de este
sistema de diagnóstico.
Cruzado Benites Marco Salvador

Resumen

En esta tesis se presenta el desarrollo de un sistema de diagnóstico para la detección temprana de cataratas utilizando redes neuronales convolucionales (CNNs). Las cataratas son la principal causa de discapacidad visual a nivel mundial, y su detección temprana es crucial para prevenir la progresión de la enfermedad y mejorar los resultados visuales de los pacientes. El sistema propuesto tiene como objetivo proporcionar una herramienta automatizada y precisa para identificar cataratas en imágenes oftalmológicas, facilitando así el acceso a la atención oftalmológica y mejorando la eficiencia del diagnóstico.

La metodología empleada incluye la recolección de un conjunto de datos de imágenes oftalmológicas, el diseño de una arquitectura de CNN optimizada para la detección de cataratas, y el entrenamiento del modelo utilizando técnicas de procesamiento de imágenes y aprendizaje profundo. El sistema se evaluó mediante métricas de rendimiento como precisión, sensibilidad y especificidad, demostrando una alta capacidad para distinguir entre imágenes con y sin cataratas.

Los resultados obtenidos muestran que el sistema desarrollado es capaz de detectar cataratas con un alto grado de precisión, lo que sugiere su viabilidad como herramienta de apoyo en entornos clínicos. Además, el uso de este sistema puede reducir la carga de trabajo de los oftalmólogos, permitir diagnósticos más rápidos y precisos, y aumentar la accesibilidad a exámenes oftalmológicos en áreas con recursos limitados.

En conclusión, el desarrollo de este sistema de diagnóstico basado en CNNs representa un avance significativo en la aplicación de la inteligencia artificial en la medicina oftalmológica, ofreciendo una solución prometedora para mejorar la detección temprana de cataratas y, en última instancia, contribuir a la reducción de la ceguera evitable a nivel mundial.

Palabras clave: Sistema de diagnóstico, detección temprana de cataratas, redes neuronales convolucionales, imágenes oftalmológicas, inteligencia artificial, sistemas cognitivos, diagnóstico automatizado, aprendizaje profundo, sensibilidad y especificidad, medicina oftalmológica.

Abstract

This thesis presents the development of a diagnostic system for the early detection of cataracts using convolutional neural networks (CNNs). Cataracts are the leading cause of visual impairment worldwide, and early detection is crucial to prevent disease progression and improve patient visual outcomes. The proposed system aims to provide an automated and accurate tool to identify cataracts in ophthalmic images, thereby facilitating access to eye care and improving diagnostic efficiency.

The methodology employed includes the collection of a dataset of ophthalmic images, the design of a CNN architecture optimized for cataract detection, and the training of the model using image processing and deep learning techniques. The system was evaluated using performance metrics such as accuracy, sensitivity, and specificity, demonstrating a high ability to distinguish between images with and without cataracts.

The results obtained show that the developed system can detect cataracts with a high degree of accuracy, suggesting its viability as a support tool in clinical settings. Additionally, the use of this system can reduce the workload of ophthalmologists, enable faster and more accurate diagnoses, and increase accessibility to eye examinations in resource-limited areas.

In conclusion, the development of this CNN-based diagnostic system represents a significant advancement in the application of artificial intelligence in ophthalmic medicine, offering a promising solution to improve the early detection of cataracts and, ultimately, contribute to the reduction of preventable blindness worldwide.

Key: Diagnostic system, early cataract detection, convolutional neural networks, ophthalmic images, artificial intelligence, cognitive systems, automated diagnosis, deep learning, sensitivity and specificity, ophthalmic medicine.

Presentación

Señores miembros del Jurado:

De conformidad con lo estipulado en el Reglamento de Grados y Títulos de la Universidad Privada Antenor Orrego, ponemos a su disposición el informe de tesis titulado “Desarrollo de un sistema de diagnóstico en la detección temprana de cataratas utilizando redes neuronales convolucionales” para que sea revisado y evaluado y de ser aprobado pueda ser defendido oralmente para optar el título profesional de Ingeniero Electrónico

De antemano, nos excusamos de los errores involuntarios en que se hubiera incurrido en el desarrollo y redacción del misma, esperando del honorable jurado un justo dictamen.

Infante Cueva Wilder Pablo

Cruzado Benites Marco Salvador

Tabla de contenidos

Dedicatoria.....

Agradecimiento .....

Resumen.....

Abstract.....

Presentación.....

Tabla de contenidos.....

Índice de tablas .....

Índice de figuras.....

I. INTRODUCCIÓN .....

1.1. Problema de Investigación .....

1.2. Objetivos.....

1.3. Justificación del estudio .....

II. MARCO DE REFERENCIA.....

2.1. Antecedentes del estudio.....

2.2. Marco teórico .....

2.3. Marco Conceptual.....

2.4. Sistema de Hipótesis .....

2.5. Variables e indicadores.....

III. METODOLOGÍA EMPLEADA.....

3.1. Tipo y nivel de investigación.....

3.2. Población y muestra de estudio .....

3.3. Diseño de investigación .....

3.4. Técnicas e instrumentos de investigación .....

3.5. Procesamiento y análisis de datos.....

IV. PRESENTACIÓN DE RESULTADOS .....

4.1. Propuesta de investigación (solo si la hubiere) .....

4.2. Análisis e interpretación de resultados .....

4.3. Docimasia de hipótesis .....

V. DISCUSIÓN DE RESULTADOS .....

CONCLUSIONES.....

RECOMENDACIONES .....

REFERENCIAS BIBLIOGRÁFICAS .....

ANEXOS .....

CAPÍTULO I
INTRODUCCIÓN

1.1 Problema de la investigación

a) Descripción de la realidad problemática

Según Lai et al. (2022) una evaluación de la Organización Mundial de la Salud, en 2021 habrá al menos 2.200 millones de personas con discapacidad visual en el mundo. Entre ellas, más de mil millones de personas no han resuelto o han podido evitar su deficiencia visual. Cataratas: Con 94 millones de casos en todo el mundo, son la causa más común de discapacidad visual o ceguera. La probabilidad de ceguera puede reducirse, su progresión puede detenerse y la vista de los pacientes puede mejorar con la identificación y el tratamiento precoz de las cataratas.

El cribado regular de cataratas sigue siendo crucial para identificar a los pacientes que requieren cirugía y prevenir la ceguera. La cirugía de cataratas es la solución más segura y efectiva para restaurar la visión cuando las cataratas han afectado significativamente la vista. Sin embargo, los métodos manuales de evaluación, como el uso de lámparas de hendidura y criterios de diagnóstico específicos, son costosos, engorrosos y sujetos a errores subjetivos. Esto se debe a la acumulación de experiencia personal en el diagnóstico manual, lo que puede afectar la precisión de las evaluaciones.

Lai et al. (2022) también menciona que la falta de oftalmólogos calificados, recursos limitados para la atención ocular y restricciones económicas han resultado en la ceguera de muchos pacientes que no recibieron tratamiento oportuno. Para abordar esta situación, se han desarrollado tecnologías de detección de cataratas altamente portátiles y automatizadas. Estas innovaciones tienen como objetivo detectar cataratas tempranas y mejorar la precisión del diagnóstico, lo que podría llevar a una atención oftalmológica más accesible y efectiva para prevenir la ceguera.

b) Descripción del problema

En muchas regiones del mundo, el acceso a la atención oftalmológica especializada es limitado, lo que dificulta la detección temprana y el tratamiento oportuno de enfermedades oculares comunes, como las cataratas. Esta falta de acceso a la atención oftalmológica puede tener consecuencias graves para la salud visual de la población, especialmente en comunidades marginadas o con recursos limitados.

Las cataratas, una de las principales causas de ceguera evitable a nivel mundial, son una condición ocular progresiva que afecta el cristalino del ojo, causando opacidad y deterioro de la visión. La detección temprana de las cataratas es fundamental para prevenir la progresión de la enfermedad y mejorar los resultados visuales de los pacientes. Sin embargo, la falta de acceso a exámenes oftalmológicos regulares y la

escasez de oftalmólogos capacitados dificultan la detección temprana de las cataratas, lo que resulta en diagnósticos tardíos y tratamientos menos efectivos.

Además, los métodos de diagnóstico convencionales, como el examen clínico y la evaluación visual, pueden ser subjetivos y estar sujetos a errores humanos, lo que puede llevar a diagnósticos inexactos y a retrasos en el tratamiento. Esto puede tener un impacto significativo en la calidad de vida de los pacientes y aumentar la carga económica asociada con el tratamiento de complicaciones relacionadas con las cataratas.

En este contexto, el desarrollo de un sistema de diagnóstico de cataratas basado en Redes Neuronales Convolucionales (CNNs) emerge como una solución potencial para mejorar el acceso a la atención oftalmológica y facilitar la detección temprana de esta enfermedad ocular. Sin embargo, se requiere investigación adicional para desarrollar y validar un sistema de diagnóstico preciso y confiable que pueda integrarse efectivamente en entornos clínicos y comunitarios, especialmente en regiones con recursos limitados.

c. Formulación del problema

¿Cómo detectar de manera temprana las cataratas?

1.2 Objetivos de la investigación

a) Objetivo general

Aplicar redes neuronales convolucionales para detectar cataratas.

b) Objetivos específicos

- Diseñar la arquitectura de la CNN para la detección de cataratas.
- Entrenar la CNN ajustando parámetros clave para mejorar la detección.
- Analizar resultados para identificar patrones y mejorar la CNN.

1.3 Justificación del estudio

Justificación Académica:

El desarrollo de un sistema de diagnóstico de cataratas utilizando Redes Neuronales Convolucionales (CNNs) presenta una oportunidad académica significativa para avanzar en el campo de la inteligencia artificial aplicada a la salud visual. Este proyecto proporciona una plataforma para la investigación y el desarrollo de algoritmos de aprendizaje automático sofisticados, así como para la exploración de nuevas técnicas de procesamiento de imágenes médicas. Además, el estudio de la aplicación de las CNNs en el diagnóstico de cataratas puede contribuir al avance del conocimiento en el campo de la oftalmología y la atención médica preventiva.

Justificación Técnica:

La utilización de Redes Neuronales Convolucionales para el diagnóstico de cataratas ofrece varias ventajas técnicas significativas. Estas redes son capaces de aprender automáticamente características complejas y abstractas de las imágenes oculares, lo que permite una detección precisa y eficiente de

la opacidad del cristalino. Además, las CNNs son altamente escalables y pueden procesar grandes volúmenes de datos de manera rápida y automatizada, lo que las hace adecuadas para aplicaciones clínicas en entornos de atención médica. Al aprovechar el potencial de las CNNs, este sistema de diagnóstico puede mejorar la eficacia y la precisión del diagnóstico de cataratas, lo que lleva a una atención oftalmológica más efectiva y a una detección temprana de esta enfermedad ocular.

Justificación Social:

El desarrollo de un sistema de diagnóstico de cataratas basado en CNNs tiene importantes implicaciones sociales en términos de salud pública y bienestar de la comunidad. La detección temprana de cataratas es fundamental para prevenir la progresión de la enfermedad y reducir la carga de discapacidad visual en la población. Al proporcionar un método preciso y eficiente para el diagnóstico de cataratas, este sistema puede mejorar el acceso a la atención oftalmológica, especialmente en comunidades con recursos limitados o áreas rurales donde el acceso a servicios médicos especializados puede ser escaso. Además, al facilitar la detección temprana y el tratamiento oportuno de las cataratas, este sistema puede contribuir a mejorar la calidad de vida de los pacientes y reducir los costos asociados con el tratamiento de complicaciones relacionadas con la enfermedad. En última instancia, el desarrollo y la implementación de esta tecnología pueden tener un impacto positivo en la

salud visual de la población y en la reducción de la carga de enfermedades oculares en la sociedad.

1.4 Aportes

El desarrollo de un sistema de diagnóstico de cataratas utilizando Redes Neuronales Convolucionales (CNNs) puede generar diversos aportes significativos a la investigación en varios ámbitos:

Avances en Inteligencia Artificial en Medicina: La aplicación de las CNNs en el diagnóstico médico, específicamente en oftalmología, contribuye al avance de la inteligencia artificial en medicina. Este proyecto ofrece la oportunidad de investigar y desarrollar algoritmos innovadores que puedan automatizar y mejorar la precisión de los diagnósticos oftalmológicos.

Mejora de la Precisión Diagnóstica: Al entrenar las CNNs con conjuntos de datos grandes y diversificados, se puede lograr una mayor precisión en el diagnóstico de cataratas. Este enfoque de aprendizaje automático puede identificar patrones sutiles y características distintivas en las imágenes oculares, lo que lleva a una detección temprana más efectiva de la opacidad del cristalino.

Exploración de Nuevas Técnicas de Procesamiento de Imágenes: El desarrollo de un sistema de diagnóstico de cataratas con CNNs implica la exploración de nuevas técnicas de procesamiento de imágenes médicas. Este proyecto puede conducir a la identificación de características específicas del

cristalino asociadas con las cataratas, así como al desarrollo de métodos avanzados para la segmentación y el análisis de imágenes oftalmológicas.

Investigación Clínica y Validación: La validación clínica del sistema en cohortes de pacientes reales proporciona una oportunidad para llevar a cabo investigaciones clínicas significativas. Estos estudios pueden evaluar la eficacia y la precisión del sistema en entornos clínicos reales, así como comparar su rendimiento con métodos de diagnóstico convencionales. Los resultados de estos estudios pueden contribuir al cuerpo de evidencia científica sobre la aplicación de la inteligencia artificial en el diagnóstico de cataratas y su impacto en la práctica clínica.

En resumen, el desarrollo de un sistema de diagnóstico de cataratas con CNNs puede generar importantes aportes a la investigación en inteligencia artificial en medicina, mejorar la precisión diagnóstica, explorar nuevas técnicas de procesamiento de imágenes y conducir a investigaciones clínicas significativas en el campo de la oftalmología. Estos avances tienen el potencial de mejorar la atención oftalmológica y los resultados de los pacientes, así como contribuir al avance del conocimiento en el campo de la salud visual.

CAPÍTULO II

MARCO DE REFERENCIA

2.1 Antecedentes del estudio:

Lai et al. (2022) en su investigación denominada **“The Use of Convolutional Neural Networks and Digital Camera Images in Cataract Detection”.**

El objetivo principal de esta investigación crear un sistema de detección de cataratas y emplea CNN con imágenes de cámaras digitales para identificar cataratas. La figura 1 ilustra la arquitectura CNNDCI propuesta. El estudio recopiló dos conjuntos de datos de GitHub.com, publicados por krishnabojha y piygot5. En este estudio, los conjuntos de datos de krishnabojha y piygot5 se denominan conjunto de datos I y conjunto de datos II, respectivamente. Todas las imágenes fueron fotografiadas con una cámara digital. Ambos conjuntos de datos contienen dos clases: cataratas y no cataratas. El conjunto de datos I contiene 9.668 imágenes, de las cuales 4.514 son de cataratas y 5.154 de no cataratas. El conjunto de datos II contiene 89 imágenes, 43 de cataratas y 46 de no cataratas.

Se empleó ImageDataGenerator, una de las funciones de Keras, para preprocesar los datos de las imágenes. Este estudio utilizó el conjunto de datos I para entrenar la CNN con procedimientos de validación cruzada quíntuple y, a continuación, proporcionó precisiones de clasificación. El conjunto de datos II se empleó para examinar el rendimiento de clasificación de los modelos CNN entrenados con diferentes datos. La Tabla 1 indica el número de datos en cada partición del conjunto de datos

I con validación cruzada quíntuple. La tabla 2 muestra los datos de entrenamiento y los datos de prueba del conjunto de datos I con validación cruzada quíntuple. La figura 2 ilustra las redes neuronales convolucionales que hacen frente a las imágenes oculares de cámaras digitales utilizadas en esta investigación.

El modelo propuesto constaba de siete capas, incluidas dos capas convolucionales, dos capas Max-pooling, una capa plana y dos capas densas. Mientras tanto, las imágenes de entrada contenían tres colores básicos: rojo, verde y azul, con tamaños de 64x64. La CNN propuesta incluía dos capas convolucionales, 32 filtros con tamaños de kernel de 3x3, un stride de 1. Las capas convolucionales se activaron mediante unidad lineal rectificada (ReLU). Se empleó un tamaño de pool de 2x2 en dos capas de max-pooling. En la primera capa densa se utilizó la función de activación ReLU y 128 neuronas. Como clasificador binario, se emplearon una neurona y la función de activación Sigmoid en la última capa densa. En este estudio se obtuvo un rendimiento muy satisfactorio cuando la tasa de abandono se aproximaba a cero, por lo que la tasa de abandono se fijó en cero.

El algoritmo de aprendizaje utilizó la entropía cruzada binaria como función de pérdida destinada a minimizar el error de entrenamiento entre los valores predichos y los reales. En este estudio, se fijó una época de

1000 para el procedimiento de entrenamiento, y se utilizó el optimizador Adam con una tasa de aprendizaje de 0,001. Los componentes del modelo CNN propuesto se presentan en la Tabla 3. Tras el entrenamiento, se construyó el modelo CNN y se ejecutó en un servidor. Se pidió a los usuarios que utilizaran dispositivos móviles para fotografiar la apariencia de los globos oculares. A continuación, las imágenes de las cámaras digitales se cargaron en el servidor a través de un sitio web. Por último, los resultados detectados se generaron inmediatamente en los dispositivos móviles para los usuarios.

Agarwal et al. (2019) en su investigación denominada **"Mobile Application Based Cataract Detection System".** El objetivo de esta investigación es desarrollar un sistema robusto de detección de cataratas basado en algoritmos de aprendizaje automático y procesamiento de imágenes. Este módulo representa la culminación de los esfuerzos para combinar estos algoritmos en una aplicación cohesiva. Utilizando Android Studio y OpenCV, se ha aprovechado el poder de estas tecnologías para crear una sofisticada herramienta para la detección de cataratas. Android Studio, como principal entorno de desarrollo, ofrece una integración perfecta de las funcionalidades de aprendizaje automático y procesamiento de imágenes, mientras que las bibliotecas mejoran estas capacidades. El enfoque en la experiencia del usuario llevó a la creación de una interfaz fácil de usar dentro de Android Studio, garantizando tanto la facilidad de uso como la satisfacción del usuario.

Al recibir la información de los usuarios, la aplicación inicia procesos de backend basados en OpenCV. Este versátil software combina a la perfección técnicas de aprendizaje automático y de procesamiento de imágenes. Para entrenar el sistema, se incorporó un conjunto de datos con imágenes de cataratas, lo que permitió a OpenCV identificar cataratas mediante la detección de ojos, el cálculo de puntos característicos y el análisis de conjuntos de datos predefinidos. La metodología se desarrolla en cuatro pasos distintos, comenzando por la Recogida de Imágenes, seguida del Preprocesamiento de Datos, la Clasificación por KNN y la Validación de la Aplicación Desarrollada.

La etapa de Recogida de Imágenes implica la recopilación de datos de la base de datos de imágenes de Google, utilizando scripts y APIs para adquirir imágenes tanto de cataratas como de ojos normales. A continuación, se realiza el preprocesamiento de los datos, en el que se recortan las imágenes para aislar la región del ojo y garantizar la precisión. Estos conjuntos de datos se importan a Orange Tool para la extracción de características y la clasificación.

Así también, la clasificación mediante KNN, el siguiente paso, aprovecha las características de imagen extraídas y emplea KNN para la categorización basada en la similitud de características, eliminando la

necesidad de un entrenamiento explícito. Por último, la eficacia del sistema se valida mediante pruebas reales en los ojos de los usuarios, bajo el escrutinio de oftalmólogos cualificados. Gracias a un desarrollo meticuloso y una validación rigurosa, el sistema de detección de cataratas representa el potencial del aprendizaje automático y el procesamiento de imágenes en aplicaciones médicas.

Ganokratanaa et al., (2023)en su investigación denominada **“Advancements in Cataract Detection: The Systematic Development of LeNet-Convolutional Neural Network Models”.** En este trabajo se presenta una investigación y desarrollo de un sistema para detectar y clasificar anomalías de la córnea utilizando tecnología avanzada de procesamiento de imágenes. Los objetivos principales de esta investigación son: comprender el proceso de discriminación de anomalías corneales mediante técnicas de procesamiento de imágenes. Así como diseñar, construir y desarrollar un sofisticado programa informático para detectar anomalías de la córnea. Además de evaluar el rendimiento y la eficacia del programa desarrollado en la detección de anomalías corneales.

Para ello, se realizó un análisis exhaustivo de teorías pertinentes y trabajos de investigación anteriores para asimilar y adaptar los conocimientos a los procesos de diseño y desarrollo. Se analizó meticulosamente un amplio conjunto de datos compuesto por 3.500 imágenes de la córnea, tanto normales como con anomalías, para fines de entrenamiento y prueba. Antes de entrenar el conjunto de datos, se aplicaron metodologías de preprocesamiento para optimizar los datos de imagen y garantizar su idoneidad para los procedimientos de aprendizaje automático.

El modelo LeNet, un modelo principal para entrenar el conjunto de datos preparado, demostró un rendimiento y precisión superiores en comparación con otras técnicas propuestas en la literatura. Además, se utilizó en el diseño e implementación de una interfaz gráfica de usuario (GUI) intuitiva, que interactúa a la perfección con la base de datos construida expresamente para esta investigación. Este sistema integrado facilita la interacción con el usuario y la recuperación eficiente de los resultados corneales.

La culminación de esta investigación subraya el inmenso potencial y la eficacia de la metodología propuesta para identificar y detectar con precisión las anomalías de la córnea. Al aprovechar la potencia del modelo LeNet-CNN y el programa de software a medida, esta investigación supone un avance significativo en la detección de anomalías de córnea, lo que resulta muy prometedor para aplicaciones clínicas y mejora la precisión diagnóstica en el campo de la oftalmología.

Además, el modelo LeNet-CNN presenta varias ventajas destacadas que lo hacen muy adecuado para el discernimiento de anomalías de la enfermedad corneal, como su arquitectura fundacional eficaz y su capacidad para fundamentar subcaracterísticas destacadas en el análisis de imágenes. Este paradigma computacional capitaliza la extracción e integración de subcaracterísticas, contribuyendo a un análisis holístico de los datos de imagen. En consecuencia, la selección del modelo LeNet-CNN para diseñar una metodología de detección de anomalías en la córnea utilizando tecnología de procesamiento de imágenes es una elección acertada, como lo demuestran los resultados obtenidos.

Finalmente, el modelo LeNet-CNN exhibe una precisión excepcional en la detección de anomalías corneales, respaldado por un proceso analítico sofisticado que identifica atributos críticos con precisión. Por lo tanto, su aplicación en la clasificación y diagnóstico de enfermedades corneales en entornos clínicos es altamente prometedora para alcanzar los objetivos de la investigación. Investigaciones futuras podrían explorar conjuntos de datos más completos, abordando niveles de gravedad detallados y ubicaciones específicas de deposición de proteínas para desarrollar un sistema de clasificación multiclasificación. Asimismo, se pueden enfocar esfuerzos en mejorar el proceso de entrenamiento y las técnicas de mejora de imágenes mediante la obtención de imágenes de alta calidad y libres de ruido. Los factores ambientales, como las condiciones de iluminación y la posición de las imágenes, afectaron el rendimiento del sistema, lo que sugiere la necesidad de un mecanismo de notificación para guiar a los usuarios en la captura óptima de imágenes.

Hu et al. (2020) en su investigación denominada **"Unified Diagnosis Framework for Automated Nuclear Cataract Grading Based on Smartphone Slit-Lamp Images".**

El objetivo de esta investigación es evaluar la capacidad de clasificación de un algoritmo automatizado de detección de cataratas nucleares utilizando imágenes oculares captadas con una lámpara de hendidura basada en smartphone. Aunque los métodos de detección de cataratas a través del aprendizaje profundo están surgiendo, mejorar el mecanismo de clasificación sigue siendo una prioridad en el campo de la investigación. Por lo tanto, este estudio se centra en la capacidad del

algoritmo para detectar automáticamente la gravedad de la catarata en función del aspecto fotométrico de la región nuclear del cristalino.

Se utilizó una combinación de una red de aprendizaje profundo, ShufeNet, y un clasificador de máquina de vectores de soporte (SVM) para clasificar la gravedad de la catarata, evaluando las características conjugadas grises de la región nuclear. Para localizar la región nuclear del cristalino, se empleó el modelo YOLOv3. Posteriormente, se evaluó el rendimiento del algoritmo utilizando 819 imágenes oculares capturadas con lámpara de hendidura basada en smartphone. Los resultados mostraron una precisión del 93,5%, un Kappa del 95,4% y un F1 del 92,3%. Además, el área bajo la curva (AUC) fue de 0,9198. El método de validación propuesto permitió evaluar la gravedad de una catarata en 29 ms y todo el proceso de clasificación en menos de 1 segundo.

Este estudio tiene el potencial de mejorar la precisión del examen de cataratas, reducir la tasa de diagnósticos erróneos y la dificultad del examen médico. Además, el sistema de puntuación propuesto puede mejorar la calidad de las imágenes obtenidas por no oftalmólogos, especialmente en zonas subdesarrolladas o con escasez de recursos oftalmológicos. Por lo tanto, este método puede aumentar la accesibilidad al tratamiento médico oftalmológico. En conclusión, el algoritmo desarrollado en este estudio muestra un rendimiento prometedor en la

detección y clasificación de cataratas nucleares, lo que puede tener un impacto significativo en la práctica clínica y mejorar el acceso a la atención oftalmológica en áreas con recursos limitados.

Pratap y Kokil, (2021) en su investigación denominada **“Efficient Network selection for computer-aided cataract diagnosis Under noisy environment.** El objetivo de este estudio es proponer un método CACD robusto y eficiente basado en la selección de redes, capaz de diagnosticar con precisión las cataratas en condiciones de ruido gaussiano blanco aditivo (AWGN). Los métodos existentes para el diagnóstico de cataratas asistido por ordenador (CACD) enfrentan desafíos debido al ruido en las imágenes digitales de fondo de retina, lo que conduce a una disminución en el rendimiento.

El método propuesto implica un conjunto de redes de vectores de soporte independientes entrenadas local y globalmente con características extraídas a varios niveles de ruido. Se adopta una técnica de extracción automática de características mediante una red neuronal convolucional (CNN) preentrenada para extraer características de las imágenes de fondo de retina de entrada.

Se obtiene un conjunto de datos de imágenes de fondo de retina de buena calidad a partir del conjunto de datos EyePACS, y se generan imágenes de fondo de retina sintéticas ruidosas utilizando un modelo AWGN para un análisis eficaz. El rendimiento del método CACD propuesto se compara con los métodos CACD basados en CNN existentes a diferentes niveles de ruido. Los resultados demuestran la superioridad del método propuesto al mostrar un rendimiento robusto frente a AWGN.

Los resultados experimentales indican que el método propuesto supera a los métodos existentes en la literatura, particularmente en el manejo del ruido. Por tanto, el método propuesto constituye un prometedor punto de partida para futuras investigaciones sobre métodos CACD robustos basados en CNN.

.

2.2 Marco teórico

2.2.1 Cataratas

Las cataratas se definen como una opacidad progresiva del cristalino, la lente transparente dentro del ojo, que puede resultar en una disminución gradual de la agudeza visual y la capacidad de enfoque. Esta opacidad puede variar en densidad y extensión, desde áreas pequeñas y localizadas hasta nubosidad difusa que cubre todo el cristalino.

2.2.2 Características

Las cataratas pueden afectar uno o ambos ojos y pueden desarrollarse lentamente con el tiempo o progresar rápidamente en algunos casos.

Entre las características comunes de las cataratas se incluyen la visión borrosa o nublada, sensibilidad a la luz, dificultad para ver en la oscuridad, deslumbramiento, cambios en la percepción del color y halos alrededor de las luces. Estos síntomas pueden interferir significativamente con las actividades diarias, como conducir, leer, ver televisión y realizar tareas domésticas.

2.2.3 Factores de riesgo asociados:

Numerosos factores pueden aumentar el riesgo de desarrollar cataratas, algunos de los cuales son modificables y otros no. Los principales factores de riesgo incluyen:

1. **Edad:** El envejecimiento es el principal factor de riesgo para el desarrollo de cataratas. A medida que las personas envejecen, el cristalino tiende a volverse menos transparente y más rígido, lo que aumenta la probabilidad de desarrollar opacidad.
2. **Genética:** La predisposición genética también puede desempeñar un papel importante en la susceptibilidad a las cataratas. Se ha observado que ciertas mutaciones genéticas pueden aumentar el riesgo de desarrollar cataratas a edades más tempranas.
3. **Exposición a la Radiación Ultravioleta:** La exposición prolongada a la radiación ultravioleta (UV) del sol, especialmente sin protección ocular

adecuada, puede dañar las células del cristalino y aumentar el riesgo de cataratas.

4. **Tabaquismo:** Fumar tabaco se ha asociado de manera consistente con un mayor riesgo de desarrollar cataratas. Los productos químicos tóxicos presentes en el humo del tabaco pueden causar estrés oxidativo y daño celular en el ojo, contribuyendo al desarrollo de opacidad del cristalino.
5. **Enfermedades Sistémicas:** Condiciones médicas como la diabetes, la hipertensión arterial, la obesidad y enfermedades inflamatorias sistémicas pueden aumentar el riesgo de desarrollar cataratas. Estas enfermedades pueden afectar el metabolismo y la salud de las células del cristalino, promoviendo la formación de opacidad.

Figura 1

Cataratas

2.2.4 Impacto Epidemiológico de las Cataratas:

Las cataratas representan una carga significativa para la salud pública a nivel mundial, regional y poblacional. Algunos aspectos del impacto epidemiológico de las cataratas incluyen:

• Prevalencia: Las cataratas son una de las principales causas de discapacidad visual en todo el mundo. Se estima que más de 100 millones de personas están afectadas por cataratas en todo el mundo, y esta cifra sigue aumentando con el envejecimiento de la población.

• Incidencia: La incidencia de cataratas varía según la región geográfica y los factores de riesgo prevalentes en cada población. Las tasas de incidencia tienden a aumentar con la edad y están influenciadas por factores genéticos, ambientales y de estilo de vida.

• Impacto Socioeconómico: Las cataratas pueden tener un impacto significativo en la calidad de vida, la independencia funcional y la participación en la vida laboral y social. Además, representan una carga económica considerable en términos de costos directos e indirectos asociados con el diagnóstico, tratamiento y rehabilitación de la enfermedad.

• Desigualdades en el Acceso a la Atención: Existen disparidades significativas en el acceso a la atención oftalmológica y el tratamiento de las cataratas en diferentes regiones del mundo y dentro de las poblaciones. Factores como la disponibilidad de servicios de salud, infraestructura médica, recursos financieros y educación sanitaria pueden influir en el acceso equitativo a la atención oftalmológica.

2.2.5 Importancia de la Detección Temprana de Cataratas

La detección temprana de cataratas desempeña un papel crucial en la prevención de la progresión de la enfermedad y en la mejora de los resultados visuales de los pacientes afectados. A continuación, se exploran los beneficios de la detección temprana, así como los desafíos asociados con la detección tardía de cataratas y sus consecuencias para la calidad de vida y la carga económica.

2.2.6 Beneficios de la Detección Temprana:

Prevención de la Progresión de la Enfermedad: La detección temprana permite identificar las cataratas en sus etapas iniciales, cuando la opacidad del cristalino es mínima y la visión aún no se ve afectada significativamente. Esto brinda la oportunidad de intervenir antes de que la condición progrese y se vuelva más difícil de tratar.

Tratamiento Oportuno: Con la detección temprana, los pacientes pueden recibir tratamiento oportuno para sus cataratas, lo que puede incluir medidas conservadoras, como cambios en el estilo de vida y corrección visual con lentes, o procedimientos quirúrgicos, como la extracción del cristalino opaco y su reemplazo por una lente intraocular.

Mejora de los Resultados Visuales: Al abordar las cataratas en sus etapas iniciales, se pueden lograr mejores resultados visuales después del tratamiento. Los pacientes pueden experimentar una restauración significativa de la agudeza visual y la calidad

de la visión, lo que les permite realizar actividades diarias con mayor facilidad y comodidad.

Prevención de Complicaciones: La detección temprana y el tratamiento oportuno de las cataratas pueden ayudar a prevenir complicaciones graves asociadas, como el glaucoma secundario, la inflamación ocular y la pérdida permanente de la visión, que pueden ocurrir si las cataratas no se tratan adecuadamente.

2.2.7 Desafíos Asociados con la Detección Tardía:

Progresión de la Opacidad: Cuando las cataratas no se detectan ni se tratan en sus etapas iniciales, tienden a progresar con el tiempo, lo que resulta en una mayor opacidad del cristalino y una mayor afectación de la visión. Esto puede dificultar el tratamiento y disminuir las posibilidades de una recuperación completa de la visión.

Impacto en la Calidad de Vida: Las cataratas no tratadas pueden tener un impacto significativo en la calidad de vida de los pacientes, afectando su capacidad para realizar tareas cotidianas, como leer, conducir, trabajar y participar en actividades sociales. La pérdida de independencia funcional y la disminución de la autonomía pueden llevar a una disminución de la calidad de vida y el bienestar emocional.

2.2.8 Tecnologías Avanzadas en el Diagnóstico de Cataratas

El diagnóstico preciso de las cataratas es fundamental para proporcionar un tratamiento adecuado y mejorar los resultados visuales de los pacientes. En este apartado, se revisan tanto los métodos tradicionales como las tecnologías avanzadas utilizadas en el diagnóstico de cataratas, destacando el potencial de las técnicas de imagenología moderna y el papel emergente de la inteligencia artificial,

especialmente las redes neuronales convolucionales, en el diagnóstico asistido por computadora de esta enfermedad ocular.

2.2.9 Revisión de Métodos Tradicionales de Diagnóstico:

Examen Clínico: El examen clínico es la piedra angular del diagnóstico de cataratas y generalmente implica una evaluación detallada de la historia clínica del paciente, así como una exploración física del ojo utilizando instrumentos como el oftalmoscopio y la lámpara de hendidura. Durante el examen, el oftalmólogo busca signos clásicos de cataratas, como opacidad del cristalino, cambios en el color y la textura, y defectos en la refracción ocular.

Evaluación Visual: La evaluación visual incluye pruebas de agudeza visual, refracción ocular y evaluación de la visión con y sin corrección. Los pacientes con cataratas pueden experimentar una disminución en la agudeza visual y la calidad de la visión, así como otros síntomas como deslumbramiento, halos alrededor de las luces y cambios en la percepción del color.

2.2.10 Introducción a Técnicas de Imagenología Moderna:

Tomografía de Coherencia Óptica (OCT): La OCT es una técnica de imagenología no invasiva que utiliza luz infrarroja para obtener imágenes de alta resolución de las estructuras oculares, incluyendo el cristalino. La OCT proporciona una visualización detallada de la morfología del cristalino y puede ayudar en la detección temprana y

el seguimiento de cambios en la opacidad y la arquitectura del cristalino asociados con las cataratas.

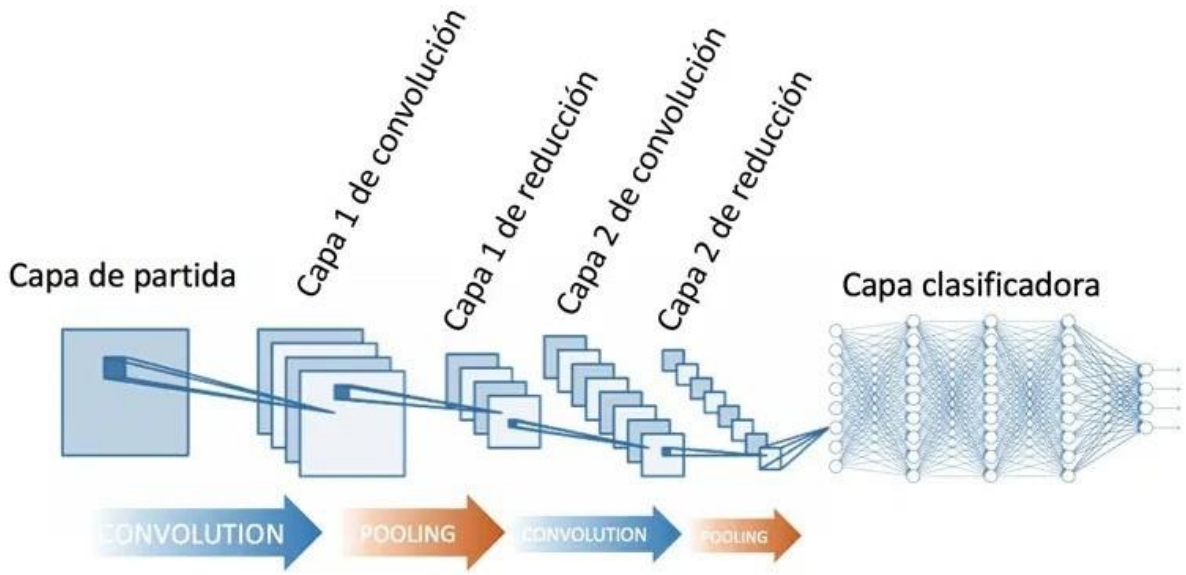
Fotografía de Segmento Anterior: La fotografía de segmento anterior es otra técnica de imagenología que se utiliza para capturar imágenes de alta resolución del segmento anterior del ojo, incluyendo el cristalino, la córnea y el iris. Estas imágenes pueden proporcionar información valiosa sobre la morfología y la opacidad del cristalino, así como la presencia de cualquier otra anormalidad ocular.

2.2.11 Potencial de Técnicas de Inteligencia Artificial y Aprendizaje Automático:

Redes Neuronales Convolucionales (CNNs): Las CNNs han demostrado un gran potencial en el diagnóstico asistido por computadora de cataratas. Estas redes pueden entrenarse utilizando grandes conjuntos de datos de imágenes de ojos con y sin cataratas, para identificar patrones y características distintivas asociadas con la opacidad del cristalino. Una vez entrenadas, las CNNs pueden analizar rápidamente nuevas imágenes y proporcionar diagnósticos precisos y objetivos de la presencia y gravedad de las cataratas.

Figura 2

Red neuronal

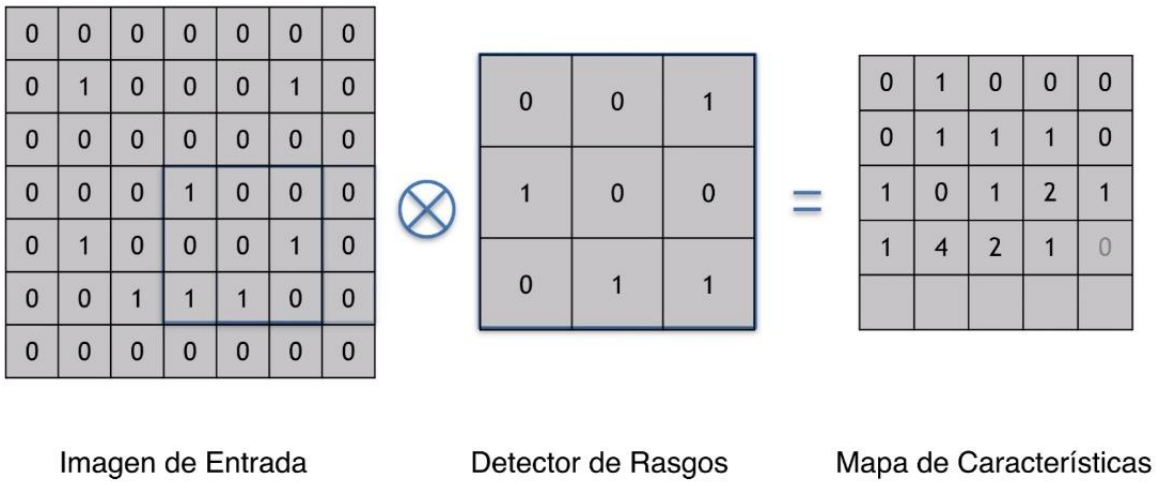


El principal uso de las CNN, un tipo de red neuronal artificial, es el procesamiento de imágenes. Lo que distingue a estos modelos es su capacidad para manejar datos con una topología cuadriculada, como una imagen formada por píxeles dispuestos en un formato ancho x alto. Las CNN son especialmente útiles para los problemas de visión por ordenador por su naturaleza jerárquica, que les permite detectar patrones temporales y espaciales en los datos de entrada.

La arquitectura de una CNN consta de varias capas, cada una de las cuales tiene una finalidad distinta. Las capas convolucionales, de agrupamiento y totalmente enlazadas son las más frecuentes en una CNN.

Figura 3

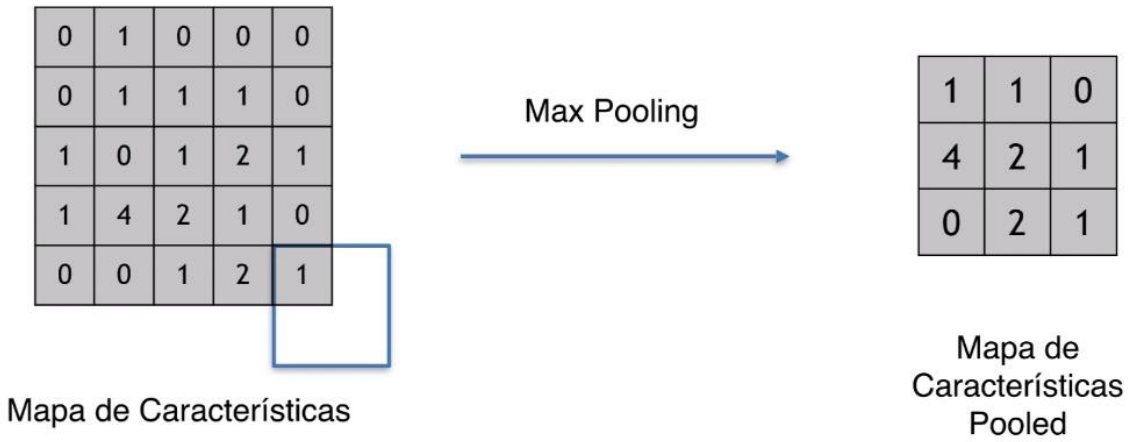
Capas de la red



Las capas convolucionales procesan los datos entrantes aplicando una serie de filtros. Cada filtro tiene la capacidad de identificar un determinado elemento dentro de la imagen, como una esquina, un borde o un color. La CNN puede crear un mapa de características que muestre la presencia de estas características en varias zonas de la imagen aplicando estos filtros a la imagen.

Figura 4

Mapas de pooling



Al reducir la dimensionalidad de los datos, las capas de agrupamiento disminuyen la necesidad de cálculo en capas posteriores y ayudan a minimizar el sobreajuste. Aunque existen otros tipos de agrupación, la agrupación máxima -que toma el valor máximo de un grupo de píxeles- es la más utilizada.

Las capas totalmente conectadas emplean todas las características recuperadas por las capas anteriores para completar la tarea en cuestión, que en este caso es la categorización de imágenes.

2.3 Marco Conceptual

Accesibilidad: La capacidad de los pacientes y proveedores de atención médica de acceder y utilizar el sistema de diagnóstico.

Algoritmo de segmentación: Un algoritmo que identifica y aísla el cristalino del ojo en una imagen.

Características: Atributos específicos del cristalino del ojo que se utilizan para entrenar el sistema de diagnóstico.

Cataratas: Opacidad del cristalino del ojo, que puede causar visión borrosa, distorsionada o incluso ceguera.

Conjunto de datos: Una colección de imágenes de ojos, tanto con cataratas como sin ellas, que se utiliza para entrenar y evaluar el sistema de diagnóstico.

Desarrollo de un sistema de diagnóstico: El proceso de crear un sistema que pueda identificar y clasificar enfermedades con precisión y eficiencia.

Detección temprana: Identificación de una enfermedad en su etapa inicial, cuando el tratamiento es más eficaz.

Especificidad: La capacidad del sistema de diagnóstico para identificar correctamente la ausencia de cataratas.

Ética: Las consideraciones morales y sociales relacionadas con el desarrollo y uso de sistemas de diagnóstico médico.

Evaluación del rendimiento: El proceso de determinar la precisión y confiabilidad del sistema de diagnóstico.

Función de clasificación: Un algoritmo que utiliza las características extraídas del cristalino del ojo para determinar si está presente o no una catarata.

Impacto clínico: El potencial del sistema de diagnóstico para mejorar la atención médica y los resultados de los pacientes.

Interpretabilidad: La capacidad de comprender cómo el sistema de diagnóstico llegó a su decisión.

Precisión: La proporción de casos correctamente diagnosticados por el sistema.

Redes neuronales convolucionales (CNNs): Un tipo de red neuronal artificial inspirada en la estructura y función del cerebro humano. Las CNNs son particularmente efectivas para el reconocimiento de imágenes y el análisis de datos visuales.

Sensibilidad: La capacidad del sistema de diagnóstico para identificar correctamente las cataratas.

Sistema de diagnóstico: Un conjunto de herramientas y métodos utilizados para identificar y clasificar enfermedades.

2.4 Hipótesis

El desarrollo de un sistema de diagnóstico utilizando redes neuronales convolucionales sirve para la detección temprana de cataratas

2.5 Variables e indicadores

Variable 1: Aplicación de Redes Neuronales Convolucionales

1. **Definición:**

   Se define un algoritmo de clasificación basado en Redes Neuronales Convolucionales (CNNs) que toma como entrada las imágenes oftalmológicas y las procesa para determinar la probabilidad de que la imagen contenga cataratas. El algoritmo clasifica las imágenes en dos categorías: "cataratas" o "sin cataratas", en función de esta probabilidad.

   Indicadores:

   a. Precisión del Modelo (Accuracy):

   Definición: La precisión del modelo (accuracy) es la proporción de predicciones correctas realizadas por el modelo sobre el total de predicciones realizadas. En el contexto de la aplicación de redes neuronales convolucionales (CNNs) para el diagnóstico de cataratas, la precisión indica la capacidad general del modelo para clasificar correctamente las imágenes oftalmológicas como positivas (con cataratas) o negativas (sin cataratas).

   Fórmula: Accuracy = (Verdaderos Positivos + Verdaderos Negativos) / (Total de Predicciones)

   Interpretación: Una alta precisión indica que el modelo clasifica correctamente la mayoría de las imágenes, minimizando tanto los falsos positivos como los falsos negativos.

   b. Sensibilidad (Recall):

Definición: La sensibilidad, también conocida como recall, es la proporción de casos positivos que fueron correctamente identificados por el modelo. En el contexto de la aplicación de CNNs para el diagnóstico de cataratas, la sensibilidad indica la capacidad del modelo para detectar correctamente las imágenes que realmente contienen cataratas.

Fórmula: Sensibilidad = Verdaderos Positivos / (Verdaderos Positivos + Falsos Negativos)

Interpretación: Una alta sensibilidad indica que el modelo es capaz de detectar la mayoría de los casos positivos de cataratas, minimizando los falsos negativos.

c. Especificidad:

Definición: La especificidad es la proporción de casos negativos que fueron correctamente identificados por el modelo. En el contexto de la aplicación de CNNs para el diagnóstico de cataratas, la especificidad indica la capacidad del modelo para descartar correctamente las imágenes que no contienen cataratas.

Fórmula: Especificidad = Verdaderos Negativos / (Verdaderos Negativos + Falsos Positivos)

Interpretación: Una alta especificidad indica que el modelo es capaz de evitar la clasificación errónea de imágenes sin cataratas como positivas.

Variable 2: Detección de cataratas

Definición: La definición operacional de la detección de cataratas en el contexto del desarrollo de un sistema de diagnóstico utilizando Redes Neuronales Convolucionales (CNNs) implica establecer claramente los criterios y procedimientos que se utilizarán para identificar la presencia o ausencia de cataratas en las imágenes oftalmológicas.

Indicadores:

a. Sensibilidad (Recall):

Definición: La sensibilidad, también conocida como recall, es la proporción de casos positivos que fueron correctamente identificados por el modelo. En el contexto de la detección de cataratas, la sensibilidad indica la capacidad del sistema de diagnóstico para detectar correctamente las imágenes que realmente contienen cataratas.

Fórmula: Sensibilidad = (Verdaderos Positivos) / (Verdaderos Positivos + Falsos Negativos)

Interpretación: Una sensibilidad alta indica que el modelo es capaz de detectar la mayoría de los casos positivos de cataratas, minimizando los falsos negativos.

b. Especificidad:

Definición: La especificidad es la proporción de casos negativos que fueron correctamente identificados por el modelo. En el contexto de la detección de

cataratas, la especificidad indica la capacidad del sistema de diagnóstico para descartar correctamente las imágenes que no contienen cataratas.

Fórmula: Especificidad = (Verdaderos Negativos) / (Verdaderos Negativos + Falsos Positivos)

Interpretación: Una alta especificidad indica que el modelo es capaz de evitar la clasificación errónea de imágenes sin cataratas como positivas.

c. Valor Predictivo Positivo (PPV):

Definición: El valor predictivo positivo (PPV) es la proporción de casos positivos identificados por el modelo que son realmente positivos. En otras palabras, es la probabilidad de que una imagen clasificada como positiva realmente tenga cataratas.

Fórmula: PPV = (Verdaderos Positivos) / (Verdaderos Positivos + Falsos Positivos)

Interpretación: Un alto valor predictivo positivo indica que la mayoría de las imágenes clasificadas como positivas por el modelo realmente tienen cataratas, lo que reduce los falsos positivos.

d. Valor Predictivo Negativo (NPV):

Definición: El valor predictivo negativo (NPV) es la proporción de casos negativos identificados por el modelo que son realmente negativos. Es la probabilidad de que una imagen clasificada como negativa realmente no tenga cataratas.

Fórmula: NPV = (Verdaderos Negativos) / (Verdaderos Negativos + Falsos Negativos)

Interpretación: Un alto valor predictivo negativo indica que la mayoría de las imágenes clasificadas como negativas por el modelo realmente no tienen cataratas, lo que reduce los falsos negativos.

Estos indicadores son fundamentales para evaluar el rendimiento y la precisión del sistema de diagnóstico de cataratas basado en Redes Neuronales Convolucionales (CNNs).

Tabla 1 Operacionalización de la variable independiente

Variable 1	**Definición operacional**	**Indicadores**	**Instrumento**	**Unidad de Medida**
Redes neuronales	Se define un algoritmo de clasificación basado en Redes Neuronales Convolucionales (CNNs) que toma como entrada las imágenes oftalmológicas y las procesa para determinar la probabilidad de que la imagen	Precisión del Modelo (Accuracy):	Reporte de simulación	%
		Sensibilidad (Recall)	Reporte de simulación	%
		Especificidad	Reporte de simulación	%

	contenga cataratas. El algoritmo clasifica las imágenes en dos categorías: "cataratas" o "sin cataratas", en función de esta probabilidad.			

Fuente: elaboración propia

Tabla 2 Operacionalización de la variable dependiente

Variable 2	Definición operacional	Indicadores	Instrumento	Unidad de Medida
Detección de cataratas	La definición operacional de la detección de cataratas en el contexto del desarrollo de un sistema de diagnóstico utilizando Redes Neuronales Convolucionales (CNNs) implica establecer claramente los criterios y procedimientos que se utilizarán para identificar la presencia o ausencia de cataratas en las imágenes oftalmológicas.	Sensibilidad (Recall):	Ficha de observación	%
		Valor Predictivo Positivo (PPV):	Ficha de observación	%
		Valor Predictivo Negativo (NPV):	Ficha de observación	%

Fuente: elaboración propia

CAPÍTULO III
METODOLOGÍA EMPLEADA

3.1 Tipo y nivel de investigación

A. Tipo de investigación

- De acuerdo a la orientación

La investigación es aplicada, ya que está enfocada a generar soluciones a problemas generales, utilizando conocimientos científicos anticipadamente confirmados.

- De acuerdo a la técnica de contrastación

La investigación es explicativa porque intenta entender las causas del fenómeno de estudio.

B. Nivel de investigación

Se considera en la presente investigación, el nivel de investigación es aplicado. En la investigación se utilizan los principios de la inteligencia artificial y el aprendizaje automático, junto con datos clínicos y de imágenes oftalmológicas, para desarrollar un sistema de diagnóstico preciso y confiable para las cataratas.

3.2 Población y muestra de estudio

A. Población

La población se refiere al conjunto completo de imágenes oftalmológicas disponibles que pueden ser utilizadas para entrenar, validar y probar el modelo de CNN. Esta población puede consistir en imágenes de pacientes de diferentes edades, géneros y condiciones oculares, que representan la diversidad de casos clínicos encontrados en la práctica médica.

B. Muestra

La muestra, por otro lado, es un subconjunto representativo de la población total de imágenes oftalmológicas que se selecciona para ser utilizado en una etapa específica del proceso de desarrollo del sistema de diagnóstico, las cuales serán de 4000 fotos, las cuales se van a dividir en:

Muestra de Entrenamiento: Es un subconjunto de imágenes seleccionadas aleatoriamente de la población total, que se utiliza para entrenar el modelo de CNN. Estas imágenes son utilizadas por el algoritmo de aprendizaje para ajustar los pesos y parámetros de la red neuronal, de modo que el modelo pueda aprender a reconocer patrones y características asociadas con la presencia o ausencia de cataratas.

Muestra de Validación: Es otro subconjunto de imágenes seleccionadas de la población total, que se utiliza para ajustar los hiperparámetros del modelo y evitar el sobreajuste durante el entrenamiento. Estas imágenes se utilizan para evaluar el rendimiento del modelo en un conjunto independiente de datos, lo que ayuda a optimizar la arquitectura y los parámetros de la CNN.

Muestra de Prueba: Es un tercer subconjunto de imágenes seleccionadas de la población total, que se utiliza para evaluar el rendimiento final del modelo entrenado. Estas imágenes no se utilizan durante el entrenamiento ni la validación, y se utilizan para

calcular métricas de evaluación como precisión, sensibilidad, especificidad y valores predictivos positivos y negativos, que proporcionan una medida de la capacidad del modelo para detectar cataratas.

C. Unidad de análisis

Imágenes de cataratas

3.3 Diseño de investigación

A. Experimental

En la presente investigación se considera de diseño de investigación experimental que permite estudiar las relaciones de causa y efecto entre variables al manipular una o más variables independientes y observar los efectos en una variable dependiente. En el contexto del desarrollo de un sistema de diagnóstico de cataratas utilizando Redes Neuronales Convolucionales (CNNs), un diseño de investigación experimental puede ser adecuado para evaluar la efectividad y precisión del sistema en la detección de la enfermedad ocular.

B. Muestreo

No probabilístico por conveniencia

3.4 Técnicas e instrumentos de investigación

A. Matriz de técnicas de investigación

Tabla 3 Técnicas de recolección de datos

Técnica	Forma de aplicación	Forma de obtención
Observación directa	Se realiza la observación directa de imágenes oftalmológicas para identificar características específicas asociadas con las cataratas, como la opacidad del cristalino.	Se observan y registran las características de las imágenes oftalmológicas utilizando software especializado de análisis de imágenes médicas.
Revisión de Registros	Se examinan registros médicos y bases de datos oftalmológicas para recopilar datos clínicos y de imágenes de pacientes con y sin cataratas.	Se accede a registros médicos electrónicos, historias clínicas y bases de datos oftalmológicas para obtener información relevante sobre pacientes diagnosticados con cataratas.
Muestras Biológicas	Se recolectan muestras de imágenes oculares, como imágenes de retina, para su análisis y procesamiento en el desarrollo del sistema de diagnóstico de cataratas.	Se obtienen imágenes oftalmológicas directamente de los pacientes durante exámenes oftalmológicos regulares o sesiones de imagenología específicas.

Fuente: elaboración propia

Instrumentos:

La matriz de confusión servirá para calcular las métricas de evaluación, como precisión, sensibilidad, especificidad y valores predictivos positivos y negativos, que proporcionan una comprensión completa del rendimiento del modelo en la detección de cataratas. En resumen, la matriz de confusión es una herramienta valiosa que permite una evaluación detallada y cuantitativa del rendimiento de un sistema de diagnóstico de cataratas basado en CNNs.

3.5 Procesamiento y análisis de los datos

A. **Técnicas de procesamiento**

División de los Datos: El conjunto de datos se divide en dos partes: un conjunto de entrenamiento, que se utiliza para ajustar los parámetros del modelo, y un conjunto de prueba, que se reserva exclusivamente para evaluar el rendimiento del modelo después del entrenamiento.

Entrenamiento del Modelo: Se utilizan los datos de entrenamiento para ajustar los pesos y parámetros de la CNN durante el proceso de entrenamiento. El objetivo es que el modelo aprenda a reconocer patrones y características asociadas con la presencia o ausencia de cataratas en las imágenes oftalmológicas.

Predicción en el Conjunto de Prueba: Una vez que el modelo ha sido entrenado, se utilizan las imágenes del conjunto de prueba para realizar predicciones sobre la presencia o ausencia de cataratas en cada

imagen. Estas predicciones se comparan luego con las etiquetas reales de las imágenes (es decir, si tienen o no cataratas).

Construcción de la Matriz de Confusión: Utilizando las predicciones del modelo y las etiquetas reales del conjunto de prueba, se construye la matriz de confusión. Esta matriz muestra el número de predicciones correctas e incorrectas del modelo para cada clase (cataratas y no cataratas).

Técnicas de procesamiento

En el contexto del desarrollo de un sistema de diagnóstico de cataratas utilizando Redes Neuronales Convolucionales (CNNs), varias técnicas de procesamiento de imágenes pueden ser utilizadas para preparar y mejorar la calidad de los datos antes de alimentarlos al modelo de CNN. Estas técnicas son fundamentales para garantizar que el modelo pueda aprender de manera efectiva patrones y características relevantes de las imágenes oftalmológicas. Aquí hay algunas técnicas de procesamiento de imágenes que podrían aplicarse:

Redimensionamiento de Imágenes: Las imágenes oftalmológicas pueden tener diferentes tamaños y resoluciones. El redimensionamiento consiste en ajustar todas las imágenes a un tamaño uniforme, lo que facilita el procesamiento y la comparación entre ellas.

Normalización de Píxeles: La normalización de píxeles implica ajustar el rango de valores de los píxeles de las imágenes para que estén

dentro de un rango específico, como [0, 1] o [-1, 1]. Esto ayuda a estandarizar los datos y facilita el entrenamiento del modelo.

Corrección de Contraste y Brillo: Algunas imágenes oftalmológicas pueden tener problemas de contraste o brillo, lo que puede dificultar la identificación de características relevantes. La corrección de contraste y brillo ajusta estos parámetros para mejorar la calidad visual de las imágenes.

Filtrado de Ruido: Las imágenes oftalmológicas pueden verse afectadas por diferentes tipos de ruido, como el ruido gaussiano o el ruido impulsivo. Se pueden aplicar filtros de suavizado, como el filtro de mediana o el filtro gaussiano, para reducir el impacto del ruido en las imágenes.

Segmentación de Regiones de Interés: La segmentación implica identificar y aislar regiones específicas de interés en las imágenes oftalmológicas, como el área del cristalino. Esto permite al modelo centrarse en las áreas relevantes para el diagnóstico de cataratas.

Aumento de Datos: El aumento de datos consiste en generar nuevas imágenes de entrenamiento mediante la aplicación de transformaciones como rotaciones, escalados, traslaciones y cambios de luminosidad a las imágenes originales. Esto aumenta la variabilidad del conjunto de datos y ayuda al modelo a generalizar mejor.

Eliminación de Artefactos: Se pueden eliminar artefactos no deseados de las imágenes, como reflejos, manchas o bordes de

imagen, para mejorar la calidad visual y facilitar la detección de características relevantes por parte del modelo.

Estas son solo algunas de las técnicas de procesamiento de imágenes que pueden ser útiles en el desarrollo del sistema de diagnóstico de cataratas con CNNs. La selección y combinación de estas técnicas dependerá de las características específicas de los datos y los requisitos del modelo.

B. **Técnicas de análisis**

Cálculo de Métricas de Rendimiento: A partir de la matriz de confusión, se calculan diversas métricas de evaluación del rendimiento del modelo, como la precisión, sensibilidad, especificidad y el área bajo la curva ROC (Receiver Operating Characteristic). Estas métricas proporcionan una medida cuantitativa del rendimiento del modelo en la detección de cataratas.

Análisis de Resultados: Finalmente, se analizan las métricas de rendimiento obtenidas para evaluar la capacidad del modelo para identificar correctamente la presencia o ausencia de cataratas. Esto proporciona información sobre la eficacia y fiabilidad del sistema de diagnóstico desarrollado.

CAPÍTULO IV
PRESENTACIÓN DE LOS RESULTADOS

4.1 Análisis e interpretación de resultados.

Se hizo uso de Ocular disease intelligent recognition (ODIR) como base de datos oftalmológica, la misma que contiene imágenes y datos de más de 5000 pacientes, esta base contiene fotos de los ojos a color tanto del derecho como izquierdo junto a los diagnósticos médicos.

Con este conjunto de datos se pretende representar el conjunto real de pacientes con estas afecciones.

La base clasifica a los pacientes en ocho categorías:

Normal (N),

Diabetes (D),

Glaucoma (G),

Catarata (C),

Degeneración Macular Asociada a la Edad (A),

Hipertensión (H),

Miopía patológica (M),

Otras enfermedades/anomalías (O)

Para el desarrollo del modelo de predicción se harán uso de las categorías Normal o Cataratas.

1. Predicción usando vgg19
    a. Creación y compilación de modelo

    Iniciamos cargando el dataset especificado junto con las librerías necesarias

Figura 5

Modelo vgg19

```
import numpy as np
import pandas as pd
import cv2
import random
from tqdm import tqdm
import matplotlib.pyplot as plt
from tensorflow.keras.preprocessing.image import ImageDataGenerator

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

```

Obtenemos las siguientes imágenes

```
/kaggle/input/ocular-disease-recognition-odir5k/full_df.csv
/kaggle/input/ocular-disease-recognition-odir5k/preprocessed_
images/3450_right.jpg
/kaggle/input/ocular-disease-recognition-odir5k/preprocessed_
images/1548_left.jpg
/kaggle/input/ocular-disease-recognition-odir5k/preprocessed_
images/4186_left.jpg
```

Exportamos los datos que relacionan cada una de las imágenes con los diagnósticos dados

Figura 6

Exportación de datos

```
df = pd.read_csv("/kaggle/input/ocular-disease-recognition-odir5k/full_df.csv")
df.head(3)
```

Obtenemos en resumen el siguiente cuadro de clasificaciones, identificando las inagene para cada ojo, tanto izquierdo como derecho, con los diagnósticos dados:

Figura 7

Cuadro de clasificaciones

	ID	Patient Age	Patient Sex	Left-Fundus	Right-Fundus	Left-Diagnostic Keywords	Right-Diagnostic Keywords	N	D	G	C	A	H	M	O	filepath	labels	target	filename
0	0	69	Female	0_left.jpg	0_right.jpg	cataract	normal fundus	0	0	0	1	0	0	0	0	../input/ocular-disease-recognition-odir5k/ODI...	['N']	[1, 0, 0, 0, 0, 0, 0, 0]	0_right.jpg
1	1	57	Male	1_left.jpg	1_right.jpg	normal fundus	normal fundus	1	0	0	0	0	0	0	0	../input/ocular-disease-recognition-odir5k/ODI...	['N']	[1, 0, 0, 0, 0, 0, 0, 0]	1_right.jpg
2	2	42	Male	2_left.jpg	2_right.jpg	laser spot, moderate non proliferative retinopathy	moderate non proliferative retinopathy	0	1	0	0	0	0	0	1	../input/ocular-disease-recognition-odir5k/ODI...	['D']	[0, 1, 0, 0, 0, 0, 0, 0]	2_right.jpg

Con la identificación realizada se procede a extraer las imágenes necesarias para el entrenamiento de la red.

Figura 8

Extracción de imágenes

```
23
24  def has_cataract(text):
25      if "cataract" in text:
26          return 1
27      else:
28          return 0
29  df["left_cataract"] = df["Left-Diagnostic Keywords"].apply(lambda x: has_cataract(x))
30  df["right_cataract"] = df["Right-Diagnostic Keywords"].apply(lambda x: has_cataract(x))
31  left_cataract = df.loc[(df.C ==1) & (df.left_cataract == 1)]["Left-Fundus"].values
32  left_cataract[:15]
33
34  right_cataract = df.loc[(df.C ==1) & (df.right_cataract == 1)]["Right-Fundus"].values
35  right_cataract[:15]
36
37  print("Number of images in left cataract: {}".format(len(left_cataract)))
38  print("Number of images in right cataract: {}".format(len(right_cataract)))
39
40
```

Realizamos a la vez una distinción entre ojo derecho e izquierdo

Figura 9

pruebas para ambos ojos

```
array(['0_left.jpg', '81_left.jpg', '103_left.jpg', '119_left.jpg',
       '254_left.jpg', '294_left.jpg', '330_left.jpg', '448_left.jpg',
       '465_left.jpg', '477_left.jpg', '553_left.jpg', '560_left.jpg',
       '594_left.jpg', '611_left.jpg', '625_left.jpg'], dtype=object)

array(['24_right.jpg', '81_right.jpg', '112_right.jpg', '188_right.jpg',
       '218_right.jpg', '345_right.jpg', '354_right.jpg', '477_right.jpg',
       '553_right.jpg', '560_right.jpg', '625_right.jpg', '726_right.jpg',
       '769_right.jpg', '949_right.jpg', '955_right.jpg'], dtype=object)
```

De igual manera realizamos la extracción de imágenes para las etiquetas normales y cataratas

Figura 10

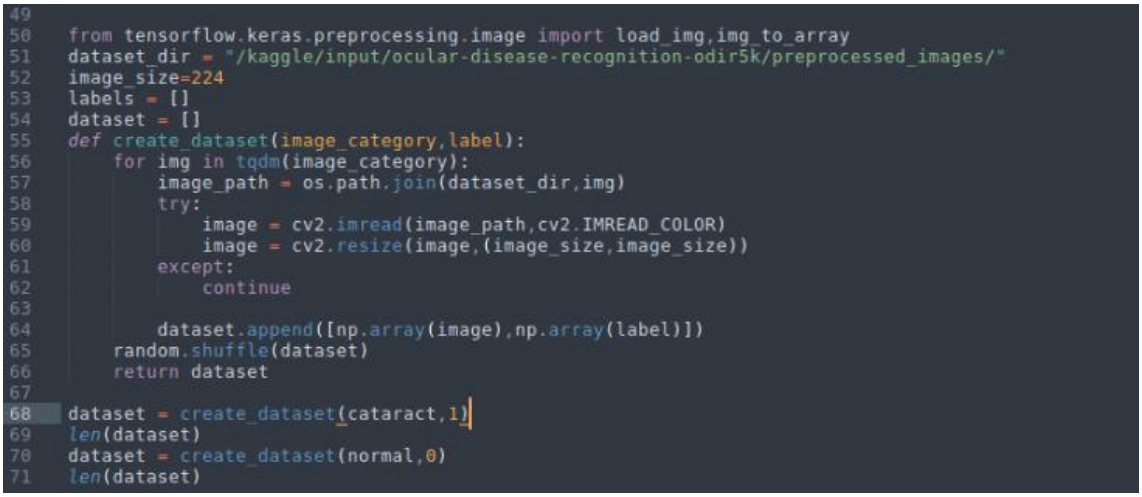
pruebas extracción normal

```
left_normal = df.loc[(df.C ==0) & (df["Left-Diagnostic Keywords"] == "normal fundus")]["Left-Fundus"].sample(250,random_state=42).values
right_normal = df.loc[(df.C ==0) & (df["Right-Diagnostic Keywords"] == "normal fundus")]["Right-Fundus"].sample(250,random_state=42).values
right_normal[:15]

cataract = np.concatenate((left_cataract,right_cataract),axis=0)
normal = np.concatenate((left_normal,right_normal),axis=0)
print(len(cataract),len(normal))
```

Procedemos con la creación de datasets con las imágenes extraídas

Figura 11

Creación de datasets



```
from tensorflow.keras.preprocessing.image import load_img,img_to_array
dataset_dir = "/kaggle/input/ocular-disease-recognition-odir5k/preprocessed_images/"
image_size=224
labels = []
dataset = []
def create_dataset(image_category,label):
    for img in tqdm(image_category):
        image_path = os.path.join(dataset_dir,img)
        try:
            image = cv2.imread(image_path,cv2.IMREAD_COLOR)
            image = cv2.resize(image,(image_size,image_size))
        except:
            continue

        dataset.append([np.array(image),np.array(label)])
    random.shuffle(dataset)
    return dataset

dataset = create_dataset(cataract,1)
len(dataset)
dataset = create_dataset(normal,0)
len(dataset)
```

Realizamos un muestreo rápido verificando la generación del dataset

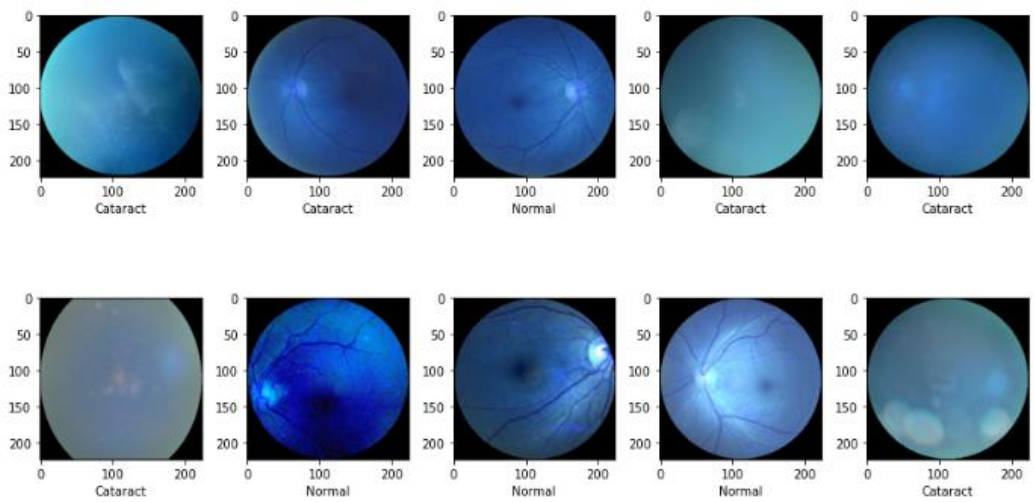
Figura 12

Muestreo y generación de dataset

```
73
74  plt.figure(figsize=(12,7))
75  for i in range(10):
76      sample = random.choice(range(len(dataset)))
77      image = dataset[sample][0]
78      category = dataset[sample][1]
79      if category== 0:
80          label = "Normal"
81      else:
82          label = "Cataract"
83      plt.subplot(2,5,i+1)
84      plt.imshow(image)
85      plt.xlabel(label)
86  plt.tight_layout()
87
```

Figura 13

Pruebas iniciales catarata y ojos normal



Realizamos una división entre elementos de entrada y salida

Figura 14

Pruebas para entrada y salida

```
88
89  x = np.array([i[0] for i in dataset]).reshape(-1,image_size,image_size,3)
90  y = np.array([i[1] for i in dataset])
91  from sklearn.model_selection import train_test_split
92  x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2)
93
```

Procedemos con la creación del modelo y obtenemos el sumario del modelo.

Figura 15

creación de modelo

```
x = np.array([i[0] for i in dataset]).reshape(-1,image_size,image_size,3)
y = np.array([i[1] for i in dataset])
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size=0.2)


from tensorflow.keras.applications.vgg19 import VGG19
vgg = VGG19(weights="imagenet",include_top = False,input_shape=(image_size,image_size,3))
for layer in vgg.layers:
    layer.trainable = False
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Flatten,Dense
model = Sequential()
model.add(vgg)
model.add(Flatten())
model.add(Dense(1,activation="sigmoid"))
model.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
vgg19 (Functional)           (None, 7, 7, 512)         20024384
_________________________________________________________________
flatten (Flatten)            (None, 25088)             0
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 25089
=================================================================
Total params: 20,049,473
Trainable params: 25,089
Non-trainable params: 20,024,384
_________________________________________________________________
```

Compilamos el proyecto

Figura 16

Compilación red

```
model.compile(optimizer="adam",loss="binary_crossentropy",metrics=["accuracy"])
from tensorflow.keras.callbacks import ModelCheckpoint,EarlyStopping
checkpoint = ModelCheckpoint("vgg19.h5",monitor="val_acc",verbose=1,save_best_only=True,
                             save_weights_only=False,period=1)
earlystop = EarlyStopping(monitor="val_acc",patience=5,verbose=1)
history = model.fit(x_train,y_train,batch_size=32,epochs=15,validation_data=(x_test,y_test),
                    verbose=1,callbacks=[checkpoint,earlystop])
```

Resumen de la compilación del modelo realizado

```
Epoch 13/15
28/28 [==============================] - 2s 86ms/step - loss: 4.7724e-05 - accuracy: 1.0000 - val_loss: 0.1677 - val_accur
acy: 0.9771
Epoch 14/15
28/28 [==============================] - 2s 87ms/step - loss: 3.0803e-05 - accuracy: 1.0000 - val_loss: 0.1641 - val_accur
acy: 0.9771
Epoch 15/15
28/28 [==============================] - 2s 86ms/step - loss: 2.4925e-05 - accuracy: 1.0000 - val_loss: 0.1604 - val_accur
acy: 0.9771
```

Realizamos las verificaciones y comprobaciones del modelo obtenido

Figura 17

Pruebas de verificación

```
122
123  loss,accuracy = model.evaluate(x_test,y_test)
124  print("loss:",loss)
125  print("Accuracy:",accuracy)
126
127
```

```
129
130  from sklearn.metrics import confusion_matrix,classification_report,accuracy_score
131  y_pred = model.predict_classes(x_test)
132  accuracy_score(y_test,y_pred)
133
134
135  print(classification_report(y_test,y_pred))
136
```

	precision	recall	f1-score	support
0	1.00	0.95	0.98	111
1	0.96	1.00	0.98	107
accuracy			0.98	218
macro avg	0.98	0.98	0.98	218
weighted avg	0.98	0.98	0.98	218

Realizamos el diagrama de confusión para comprobar la precisión de nuestro modelo.

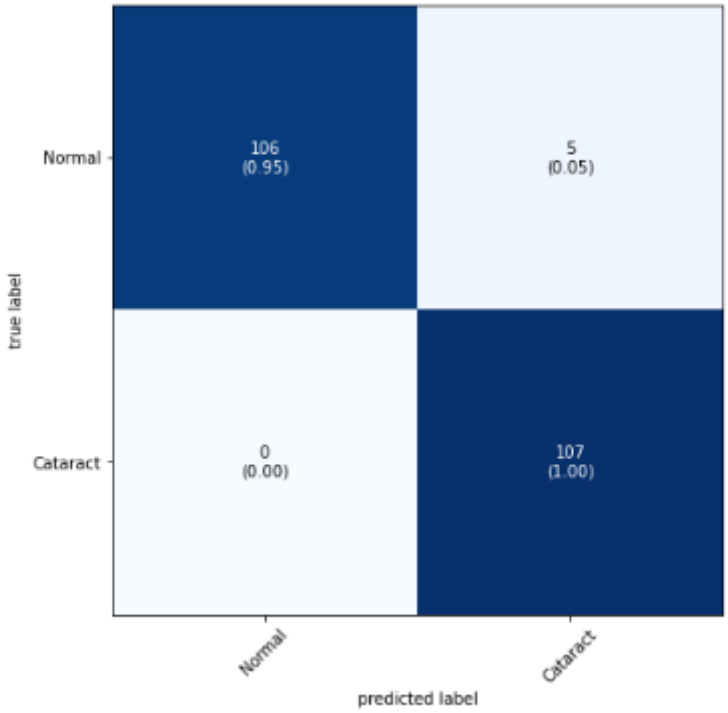
Figura 18

Creación de matriz de confusión

```
139
140  from mlxtend.plotting import plot_confusion_matrix
141  cm = confusion_matrix(y_test,y_pred)
142  plot_confusion_matrix(conf_mat = cm,figsize=(8,7),class_names = ["Normal","Cataract"],
143                        show_normed = True);
144
```

Figura 19

matriz de confusión



Obtenemos las curvas de aprendizaje para nuestro modelo.

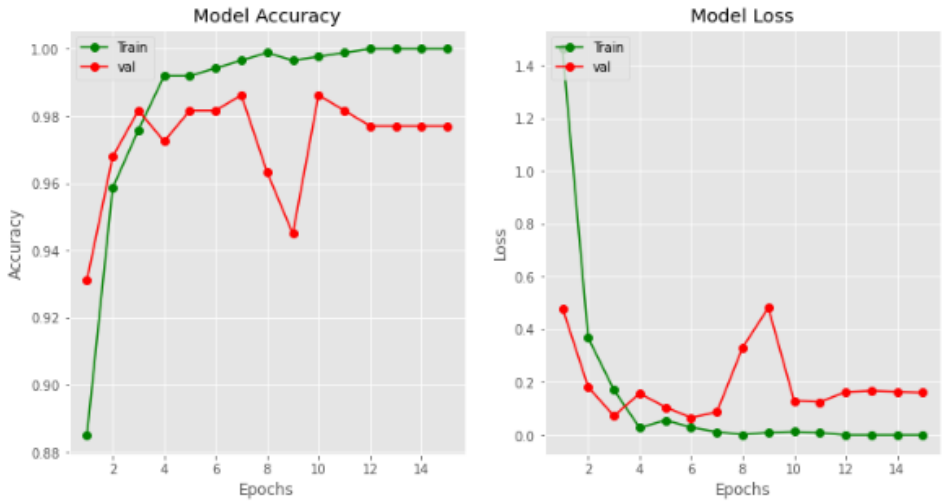
Figura 20

curvas de aprendizaje

```
146
147   plt.style.use("ggplot")
148   fig = plt.figure(figsize=(12,6))
149   epochs = range(1,16)
150   plt.subplot(1,2,1)
151   plt.plot(epochs,history.history["accuracy"],"go-")
152   plt.plot(epochs,history.history["val_accuracy"],"ro-")
153   plt.title("Model Accuracy")
154   plt.xlabel("Epochs")
155   plt.ylabel("Accuracy")
156   plt.legend(["Train","val"],loc = "upper left")
157
158   plt.subplot(1,2,2)
159   plt.plot(epochs,history.history["loss"],"go-")
160   plt.plot(epochs,history.history["val_loss"],"ro-")
161   plt.title("Model Loss")
162   plt.xlabel("Epochs")
163   plt.ylabel("Loss")
164   plt.legend(["Train","val"],loc = "upper left")
165   plt.show()
166
167
```

Figura 21

Gráficas curvas de aprendizaje



b. Predicción de nuestro modelo

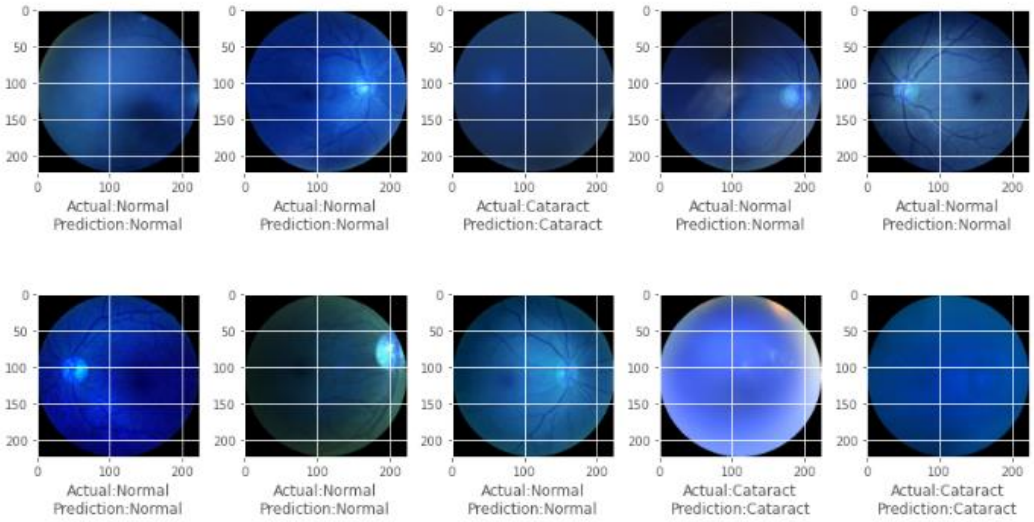
Figura 22

Predicción de modelo creado

```
168
169   plt.figure(figsize=(12,7))
170   for i in range(10):
171       sample = random.choice(range(len(x_test)))
172       image = x_test[sample]
173       category = y_test[sample]
174       pred_category = y_pred[sample]
175
176       if category== 0:
177           label = "Normal"
178       else:
179           label = "Cataract"
180
181       if pred_category== 0:
182           pred_label = "Normal"
183       else:
184           pred_label = "Cataract"
185
186       plt.subplot(2,5,i+1)
187       plt.imshow(image)
188       plt.xlabel("Actual:{}\nPrediction:{}".format(label,pred_label))
189   plt.tight_layout()
190
191
```

Figura 23

Grafica de predicción



2. Predicción usando CNN

    a. Creación y compilación del modelo

    Iniciamos cargando las imágenes y librerías necesarias

Figura 24

Creación CNN

```
import os
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
from matplotlib.image import imread
pwd
'/kaggle/working'
my_data_dir='/kaggle/input/cataract-image-dataset/processed_images/'
os.listdir(my_data_dir)

train_path = my_data_dir+'train'
test_path = my_data_dir+'test'
os.listdir(train_path)

os.listdir(test_path)
```

Procedemos a preparar los archivos para entrenamiento y validación

i. Entrenamiento

Figura 25

Entrenamiento red CNN

```
plant_types = ["normal", "cataract"]

cataract = []
normal = []

total_images = 0
for plant_type in plant_types:
    file_list = os.listdir(os.path.join(train_path, plant_type))

    if plant_type == "cataract":
        cataract.extend(file_list)
    elif plant_type == "normal":
        normal.extend(file_list)

    print(f"Number of {plant_type} images:", len(file_list))

    total_images += len(file_list)

print("Total images:", total_images)
```

```
Number of normal images: 246
Number of cataract images: 245
Total images: 491
```

ii. Validación

Figura 26

Validación red CNN

```
plant_types = ["normal", "cataract"]

cataract = []
normal = []

total_images = 0
for plant_type in plant_types:
    file_list = os.listdir(os.path.join(test_path, plant_type))

    if plant_type == "cataract":
        cataract.extend(file_list)
    elif plant_type == "normal":
        normal.extend(file_list)

    print(f"Number of {plant_type} images:", len(file_list))

    total_images += len(file_list)

print("Total images:", total_images)
```

```
Number of normal images: 60
Number of cataract images: 61
Total images: 121
```

Obtenemos el promedio de la matriz de imágenes

iii. Normal

Figura 27

Matriz normal de imágenes

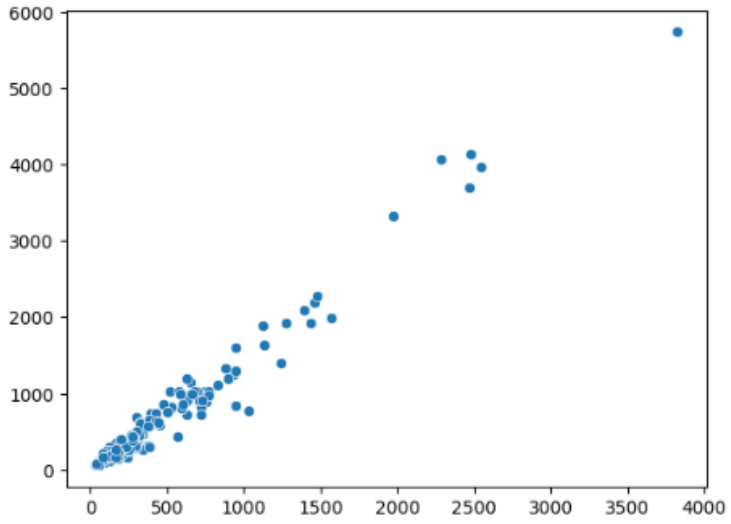
```
x= []
y = []
for image in os.listdir(train_path+'/normal'):

    img = imread(train_path+'/normal/'+image)
    d1,d2,colors = img.shape
    x.append(d1)
    y.append(d2)
sns.scatterplot(x=x,y=y)

np.mean(x)
np.mean(y)

```

```
406.5081300813008
```

```
594.1422764227642
```

Figura 28

Datos dispersados



El gráfico muestra una serie de puntos de datos dispersos. El eje horizontal (x) va desde 0 hasta 4000, y el eje vertical (y) va desde 0 hasta 6000.

Los puntos de datos están concentrados principalmente en la esquina inferior izquierda del gráfico, cerca del origen, y se dispersan hacia arriba y hacia la derecha. Esto sugiere una correlación positiva entre las variables representadas en ambos ejes.

iv. Catarata

Figura 29

Importación de modulos para visualizar datos

```
x= []
y = []
for image in os.listdir(train_path+'/cataract'):

    img = imread(train_path+'/cataract/'+image)
    d1,d2,colors = img.shape
    x.append(d1)
    y.append(d2)
sns.scatterplot(x=x,y=y)

np.mean(x)
np.mean(y)
```

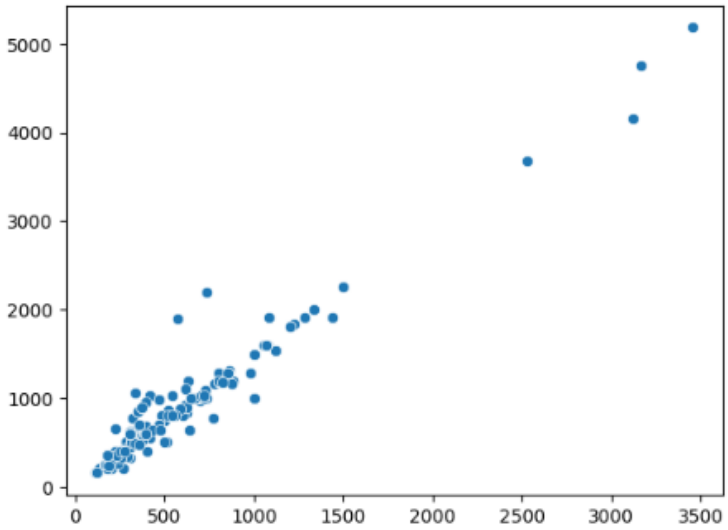
importamos los módulos necesarios para manipular el sistema de archivos y visualizar datos. Definimos la ruta al directorio que contiene nuestras imágenes de entrenamiento y creamos listas vacías para almacenar datos. Iteramos sobre cada imagen en el subdirectorio 'cataract' para leer los colores (aunque la función read_colors no está definida), obteniendo las dimensiones de cada imagen. Estas dimensiones se almacenan en las listas correspondientes, y calculamos la media de ambas

listas. Finalmente, utilizamos sns.scatterplot para crear un gráfico de dispersión que muestra las dimensiones de las imágenes.

```
606.1183673469387

925.3591836734694
```

Figura 30

Datos dispersados luego de dimensiones de imágenes



v. Preparación de ImageDataGenerator

Figura 31

Preparación *de* ImageDataGenerator

```
90
91   from tensorflow.keras.preprocessing.image import ImageDataGenerator
92   image_gen = ImageDataGenerator(rotation_range=20,
93                                  width_shift_range=0.10,
94                                  height_shift_range=0.10,
95                                  #rescale=1/255,
96                                  shear_range=0.1,
97                                  zoom_range=0.1,
98                                  horizontal_flip=True,
99                                  fill_mode='nearest'
100                                )
101  plt.imshow(image_gen.random_transform(normal_img))
102
103
```

Figura 32

Preparación del modelo

```
103
104  from tensorflow.keras.callbacks import EarlyStopping
105  early_stop = EarlyStopping(monitor='val_loss',patience=10)
106  batch_size = 32
107
108  train_image_gen = image_gen.flow_from_directory(train_path,
109                                                  target_size=(500, 800),
110                                                   color_mode='rgb',
111                                                  batch_size=batch_size,
112                                                  class_mode='binary')
113
114  test_image_gen = image_gen.flow_from_directory(test_path,
115                                                 target_size=(500, 800),
116                                                 color_mode='rgb',
117                                                 batch_size=batch_size,
118                                                 class_mode='binary')
119
120
```

Figura 33

Creación del modelo

```
121
122
123  from keras.models import Sequential
124  from keras.layers import Conv2D, MaxPooling2D, Flatten, Dense, BatchNormalization, Dropout
125  from tensorflow.keras.models import Sequential
126  from tensorflow.keras.layers import Activation, Dropout, Flatten, Dense, Conv2D, MaxPooling2D, BatchNormalization
127  model = Sequential()
128
129  model.add(Conv2D(filters=16, kernel_size=(3,3),input_shape=image_shape, activation='relu'))
130  model.add(MaxPooling2D(pool_size=(2, 2)))
131  model.add(BatchNormalization())
132  model.add(Dropout(0.2))
133
134
135  model.add(Conv2D(filters=32, kernel_size=(3,3), activation='relu'))
136  model.add(MaxPooling2D(pool_size=(2, 2)))
137  model.add(BatchNormalization())
138  model.add(Dropout(0.2))
139
140
141  model.add(Conv2D(filters=32, kernel_size=(3,3), activation='relu'))
142  model.add(MaxPooling2D(pool_size=(2, 2)))
143  model.add(BatchNormalization())
144  model.add(Dropout(0.2))
145
146  model.add(Flatten())
147
148  model.add(Dense(32, activation='relu'))
149  model.add(BatchNormalization())
150
151  model.add(Dense(1, activation='sigmoid'))
152
153  model.compile(loss='binary_crossentropy',
154                optimizer='adam',
155                metrics=['accuracy'])
156  model.summary()
157
158
```

Figura 34

Valores del modelo

Layer (type)	Output Shape	Param #
conv2d_33 (Conv2D)	(None, 498, 798, 16)	448
max_pooling2d_33 (MaxPooling2D)	(None, 249, 399, 16)	0
batch_normalization_42 (BatchNormalization)	(None, 249, 399, 16)	64
dropout_15 (Dropout)	(None, 249, 399, 16)	0
conv2d_34 (Conv2D)	(None, 247, 397, 32)	4,640
max_pooling2d_34 (MaxPooling2D)	(None, 123, 198, 32)	0
batch_normalization_43 (BatchNormalization)	(None, 123, 198, 32)	128
dropout_16 (Dropout)	(None, 123, 198, 32)	0
conv2d_35 (Conv2D)	(None, 121, 196, 32)	9,248
max_pooling2d_35 (MaxPooling2D)	(None, 60, 98, 32)	0
batch_normalization_44 (BatchNormalization)	(None, 60, 98, 32)	128
dropout_17 (Dropout)	(None, 60, 98, 32)	0
flatten_9 (Flatten)	(None, 188160)	0
dense_18 (Dense)	(None, 32)	6,021,152
batch_normalization_45 (BatchNormalization)	(None, 32)	128
dense_19 (Dense)	(None, 1)	33

```
158
159    results = model.fit(train_image_gen,epochs=30,
160                        validation_data=test_image_gen,
161                        callbacks=[early_stop])
162
```

```
Epoch 9/30
16/16 ━━━━━━━━━━━━━━━━━━━━ 67s 3s/step - accuracy: 0.8798 - loss: 0.2945 - val_accuracy: 0.8099
- val_loss: 0.3990
Epoch 10/30
16/16 ━━━━━━━━━━━━━━━━━━━━ 68s 3s/step - accuracy: 0.9140 - loss: 0.2238 - val_accuracy: 0.7851
- val_loss: 0.4590
```

```
163
164    summary = pd.DataFrame(model.history.history)
165    summary.tail()
166
```

	accuracy	loss	val_accuracy	val_loss
5	0.881874	0.299929	0.719008	0.542652
6	0.883910	0.302457	0.801653	0.443780
7	0.896130	0.278413	0.760331	0.454978
8	0.871690	0.289570	0.809917	0.398959
9	0.920570	0.232096	0.785124	0.459022

Figura 35

Curvas de aprendizaje

```
167
168   plt.figure(figsize=(10,6))
169   plt.plot(summary.loss, label="loss")
170   plt.plot(summary.val_loss, label="val_loss")
171   plt.legend(loc="upper right")
172   plt.ylabel("Loss")
173   plt.xlabel("Epoch")
174   plt.show()
175
176
```

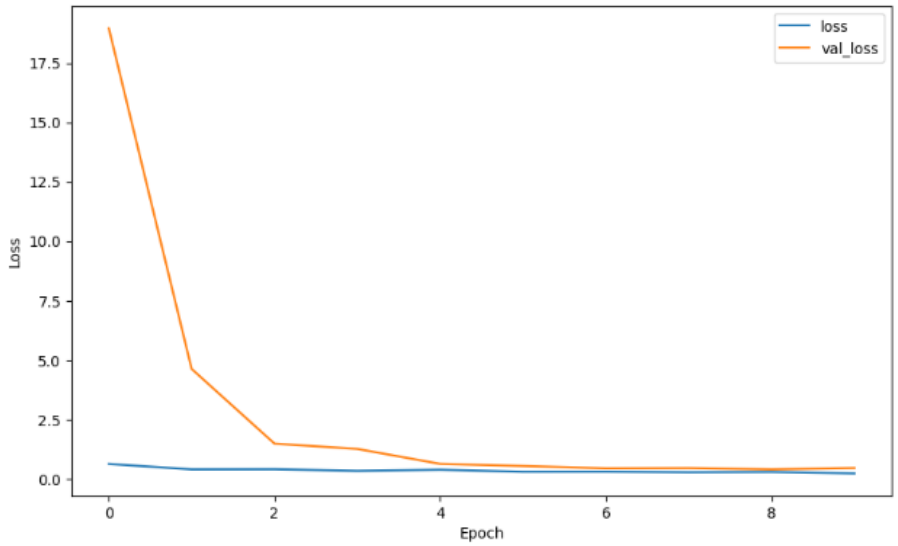


Figura 36

Gráficas de curvas de aprendizaje

```
175
176   plt.figure(figsize=(10,6))
177   plt.plot(summary.accuracy, label="accuracy")
178   plt.plot(summary.val_accuracy, label="val_accuracy")
179   plt.legend(loc="upper left")
180   plt.ylabel("Accuracy")
181   plt.xlabel("Epoch")
182   plt.show()
183
184
```

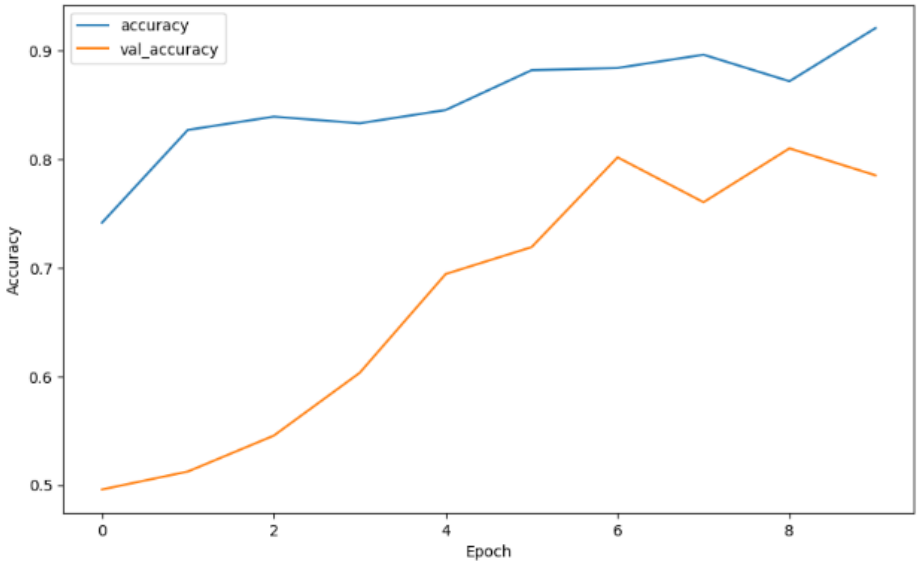


Figura 37

Métricas y evaluación

```
184
185   model.metrics_names
186   model.evaluate(test_image_gen)
187   pred_probabilities = model.predict(test_image_gen)
188   pred_probabilities
189
190   predictions = pred_probabilities > 0.5
191   from sklearn.metrics import classification_report,confusion_matrix
192   print(classification_report(test_image_gen.classes,predictions))
193
```

	precision	recall	f1-score	support
0	0.55	0.39	0.46	61
1	0.52	0.67	0.58	60
accuracy			0.53	121
macro avg	0.53	0.53	0.52	121
weighted avg	0.53	0.53	0.52	121

Figura 38

Predicción

```
train_image_gen.class_indices

from tensorflow.keras.preprocessing import image
import numpy as np
import tensorflow as tf

model = tf.keras.models.load_model('eyes_model.keras')

image_path = "/kaggle/input/cataract-image-dataset/processed_images/test/cataract/image_259.png"
img = image.load_img(image_path, target_size=(500, 800))
img_array = image.img_to_array(img)
img_array = np.expand_dims(img_array, axis=0)
img_array /= 255.

prediction = model.predict(img_array)

predicted_class = np.argmax(prediction)
print('Prediction:', predicted_class)
```

```
{'cataract': 0, 'normal': 1}
```

3. Predicción usando RNN

   a. Creación y compilación del modelo

      Cargamos librerías

Figura 39

Compilación RNN

```
import os
import cv2
import random
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import tensorflow as tf
from PIL import Image
from tensorflow import keras
from tensorflow.keras import layers
from tensorflow.keras.models import Sequential
from tensorflow.keras.preprocessing.image import ImageDataGenerator # Corrected import statement
from tensorflow.keras.layers import Conv2D, MaxPooling2D, Flatten, Dense
from tensorflow.keras.callbacks import TensorBoard
from sklearn.metrics import classification_report
import warnings
warnings.filterwarnings('ignore')

```

Cargamos directorios y redimensionamos las imágenes

Figura 40

Redimensionamiento de imágenes

```
21
22  rescale = tf.keras.layers.Rescaling(1./255)
23  train_ds = tf.keras.utils.image_dataset_from_directory(
24      directory='/kaggle/input/eye-diseases-classification/dataset',
25      batch_size=32,
26      image_size=(256, 256),
27      validation_split=0.2,
28      subset="training",
29      seed=123,
30      label_mode='categorical',
31  )
32  train_ds = train_ds.map(lambda x, y: (rescale(x), y))
33  validation_ds = tf.keras.utils.image_dataset_from_directory(
34      directory='/kaggle/input/eye-diseases-classification/dataset',
35      batch_size=32,
36      image_size=(256, 256),
37      validation_split=0.2,
38      subset="validation",
39      seed=123,
40      label_mode='categorical',
41  )
42  validation_ds = validation_ds.map(lambda x, y: (rescale(x), y))
43  test_ds = tf.keras.utils.image_dataset_from_directory(
44      directory='/kaggle/input/eye-diseases-classification/dataset',
45      batch_size=32,
46      image_size=(256, 256),
47      label_mode='categorical',
48      shuffle=False,
49  )
50  test_ds = test_ds.map(lambda x, y: (rescale(x), y))
51
```

```
Found 4217 files belonging to 4 classes.
Using 3374 files for training.
Found 4217 files belonging to 4 classes.
Using 843 files for validation.
Found 4217 files belonging to 4 classes.
```

i. Revisamos las imágenes antes del escalamiento

Figura 41

Revisión de imágenes

```
54
55  print("Shape of the first image in the training dataset:", next(iter(train_ds))[0][0].shape)
56  print("Shape of the first image in the validation dataset:", next(iter(validation_ds))[0][0].shape)
57  print("Shape of the first image in the test dataset:", next(iter(test_ds))[0][0].shape)
58
59
```

```
Shape of the first image in the training dataset: (256, 256, 3)
Shape of the first image in the validation dataset: (256, 256, 3)
Shape of the first image in the test dataset: (256, 256, 3)
```

ii. Revisamos el valor de pixel luego del procesamiento

Figura 42

Revisión de pixelaje

```
min_pixel_value = float('inf')
max_pixel_value = float('-inf')

for images, _ in train_ds:
    batch_min = tf.reduce_min(images)
    batch_max = tf.reduce_max(images)

    min_pixel_value = tf.minimum(min_pixel_value, batch_min)
    max_pixel_value = tf.maximum(max_pixel_value, batch_max)

print("Minimum pixel value:", min_pixel_value.numpy())
print("Maximum pixel value:", max_pixel_value.numpy())
```

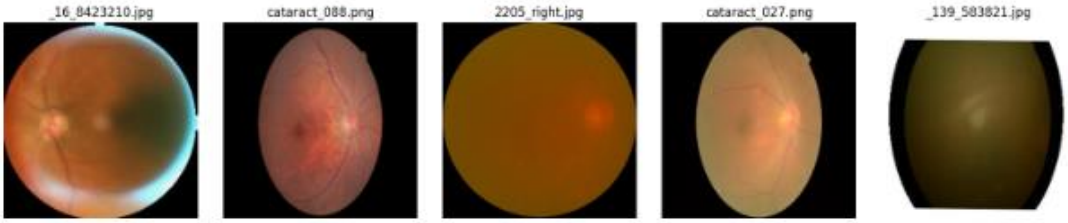
```
Minimum pixel value: 0.0
Maximum pixel value: 1.0
```

iii. Visualización de las imágenes

Figura 43

Visualización de imágenes

```
def visualize_images(path, target_size=(256, 256), num_images=5):

    image_filenames = [f for f in os.listdir(path) if os.path.isfile(os.path.join(path, f))]

    if not image_filenames:
        raise ValueError("No images found in the specified path")

    selected_images = random.sample(image_filenames, min(num_images, len(image_filenames)))

    fig, axes = plt.subplots(1, num_images, figsize=(15, 3), facecolor='white')

    for i, image_filename in enumerate(selected_images):
        image_path = os.path.join(path, image_filename)
        image = Image.open(image_path)
        image = image.resize(target_size)

        axes[i].imshow(image)
        axes[i].axis('off')
        axes[i].set_title(image_filename)

    plt.tight_layout()
    plt.show()

path_to_visualize = "/kaggle/input/eye-diseases-classification/dataset/cataract"

visualize_images(path_to_visualize, num_images=5)
```




```
path_to_visualize = "/kaggle/input/eye-diseases-classification/dataset/normal"

visualize_images(path_to_visualize, num_images=5)
```

iv. Construcción del modelo y revisión del sumario

Figura 44
Modelamiento y revisión

```
# Define RNN model
model = tf.keras.Sequential([
    tf.keras.layers.Conv2D(32, (3, 3), activation='relu', input_shape=(256, 256, 3)),
    tf.keras.layers.MaxPooling2D((2, 2)),
    tf.keras.layers.Conv2D(64, (3, 3), activation='relu'),
    tf.keras.layers.MaxPooling2D((2, 2)),
    tf.keras.layers.Conv2D(64, (3, 3), activation='relu'),
    tf.keras.layers.MaxPooling2D((2, 2)),
    tf.keras.layers.Flatten(),
    tf.keras.layers.Dense(64, activation='relu'),
    tf.keras.layers.Reshape((64, 1)),  # Reshape for RNN input
    tf.keras.layers.SimpleRNN(32),  # Simple RNN layer
    tf.keras.layers.Dense(4, activation='softmax')  # Output layer
])

# Compile the model
model.compile(optimizer='adam',
              loss='categorical_crossentropy',
              metrics=['accuracy'])

# Print model summary
print(model.summary())
```

Figura 45
Valores obtenidos

Layer (type)	Output Shape	Param #
conv2d (Conv2D)	(None, 254, 254, 32)	896
max_pooling2d (MaxPooling2D)	(None, 127, 127, 32)	0
conv2d_1 (Conv2D)	(None, 125, 125, 64)	18,496
max_pooling2d_1 (MaxPooling2D)	(None, 62, 62, 64)	0
conv2d_2 (Conv2D)	(None, 60, 60, 64)	36,928
max_pooling2d_2 (MaxPooling2D)	(None, 30, 30, 64)	0
flatten (Flatten)	(None, 57600)	0
dense (Dense)	(None, 64)	3,686,464
reshape (Reshape)	(None, 64, 1)	0
simple_rnn (SimpleRNN)	(None, 32)	1,088
dense_1 (Dense)	(None, 4)	132

v. Modelo Fit

Figura 46
Modelo FIT

```
from tensorflow.keras.callbacks import EarlyStopping
early_stopping = EarlyStopping(patience=5, restore_best_weights=True)

history = model.fit(train_ds,
                    validation_data=validation_ds,
                    epochs=5,
                    callbacks=[early_stopping])

```

```
Epoch 1/5
106/106 ━━━━━━━━━━━━━━━━━━━━ 215s 2s/step - accuracy: 0.4840 - loss: 1.1158 - val_accuracy:
0.6607 - val_loss: 0.7685
Epoch 2/5
106/106 ━━━━━━━━━━━━━━━━━━━━ 260s 2s/step - accuracy: 0.7023 - loss: 0.7009 - val_accuracy:
0.6975 - val_loss: 0.6745
Epoch 3/5
106/106 ━━━━━━━━━━━━━━━━━━━━ 259s 2s/step - accuracy: 0.7272 - loss: 0.6383 - val_accuracy:
0.7331 - val_loss: 0.6005
Epoch 4/5
106/106 ━━━━━━━━━━━━━━━━━━━━ 265s 2s/step - accuracy: 0.7570 - loss: 0.5683 - val_accuracy:
0.7461 - val_loss: 0.6202
Epoch 5/5
106/106 ━━━━━━━━━━━━━━━━━━━━ 261s 2s/step - accuracy: 0.7787 - loss: 0.5330 - val_accuracy:
0.7711 - val_loss: 0.5730
```

Figura 47
Evaluación modelo FIT

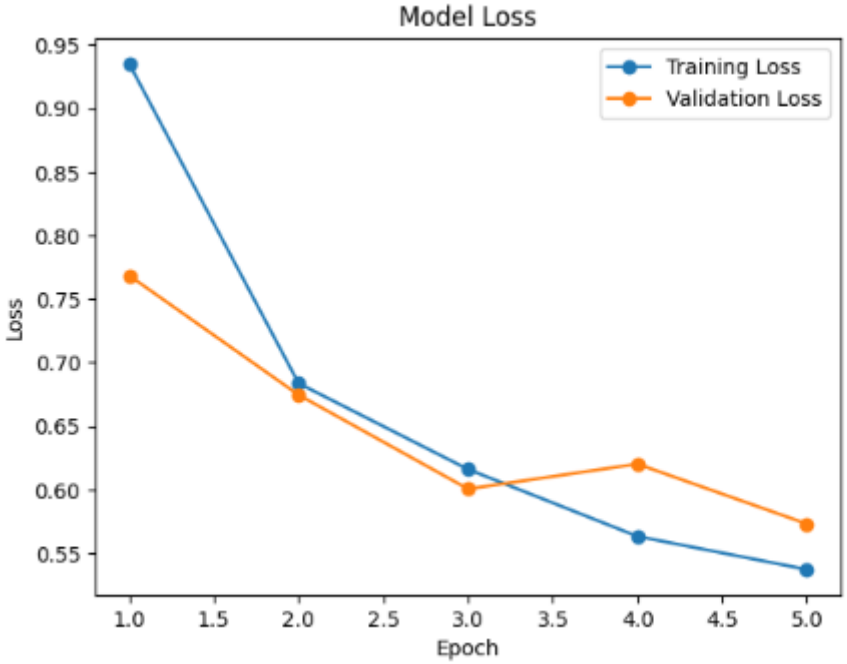
```
test_loss, test_accuracy = model.evaluate(test_ds)
print("Test accuracy:", test_accuracy)

```

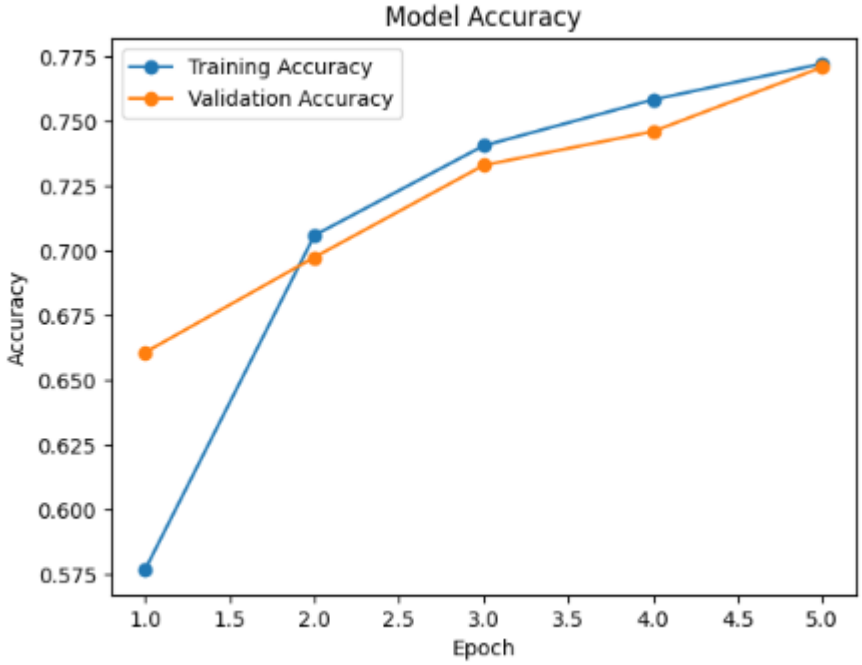
```
132/132 ━━━━━━━━━━━━━━━━━━━━ 77s 579ms/step - accuracy: 0.6361 - loss: 0.7641
Test accuracy: 0.7811239957809448
```

```

epochs = range(1, len(history.history['loss']) + 1)

plt.plot(epochs, history.history['loss'], label='Training Loss', marker='o')
plt.plot(epochs, history.history['val_loss'], label='Validation Loss', marker='o')
plt.title('Model Loss')
plt.xlabel('Epoch')
plt.ylabel('Loss')
plt.legend()
plt.show()

plt.plot(epochs, history.history['accuracy'], label='Training Accuracy', marker='o')
plt.plot(epochs, history.history['val_accuracy'], label='Validation Accuracy', marker='o')
plt.title('Model Accuracy')
plt.xlabel('Epoch')
plt.ylabel('Accuracy')
plt.legend()
plt.show()


```

Figura 48

Curvas de aprendizaje





4. Predicción usando aprendizaje automático

    a. Cargamos librerías y dataset

Figura 49

Importación de librerías y dataset

```
import os
import numpy as np
import pandas as pd
from PIL import Image, ImageChops, ImageEnhance, ImageOps
import matplotlib.pyplot as plt
import scipy.stats as stats
import seaborn as sns
from keras.preprocessing import image
import tensorflow as tf
from tensorflow import keras
import skimage.color as color
sns.set_theme()
import skimage.io as io
import cv2, PIL, glob, pathlib
from sklearn.model_selection import KFold, train_test_split
from sklearn.metrics import *
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import classification_report
from sklearn.model_selection import cross_val_score
from sklearn.metrics import make_scorer
from sklearn.svm import SVC
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score
from sklearn.metrics import accuracy_score
from sklearn.tree import export_graphviz
from IPython.display import Image
import keras.utils as image
import cv2
import cv2 as cv
from PIL import Image
import time

data_dir = '/kaggle/input/eye-diseases-classification/dataset'
entradas_dir = os.listdir( data_dir )
for carpeta in entradas_dir:
  print(carpeta)
```

```
glaucoma
normal
diabetic_retinopathy
cataract
```

```
entradas_diabetic = os.listdir('/kaggle/input/eye-diseases-classification/dataset/diabetic_retinopathy')
entradas_cataract = os.listdir('/kaggle/input/eye-diseases-classification/dataset/cataract')
entradas_glaucoma = os.listdir('/kaggle/input/eye-diseases-classification/dataset/glaucoma')
entradas_normal = os.listdir('/kaggle/input/eye-diseases-classification/dataset/normal')

t = (len(entradas_diabetic), len(entradas_cataract), len(entradas_glaucoma), len(entradas_normal))

print("Cantidad de imagenes:")
print("\nRetinopatía diabetica:",len(entradas_diabetic))
print("Cataratas:",len(entradas_cataract))
print("Glaucoma:",len(entradas_glaucoma))
print("Normal:",len(entradas_normal))
print("\nTotal: ", sum(t))
```

```
Cantidad de imagenes:

Retinopatia diabetica: 1098
Cataratas: 1038
Glaucoma: 1007
Normal: 1074

Total:  4217
```

b. Muestreo de imágenes

Figura 50

Muestreo de imágenes

```
58
59   fig = plt.subplots(figsize =(12, 8))
60   count_clases = [len(entradas_diabetic), len(entradas_cataract), len(entradas_glaucoma), len(entradas_normal)]
61   plt.bar(entradas_dir, count_clases, color ='b', edgecolor ='grey', label ='IT')
62
63
```

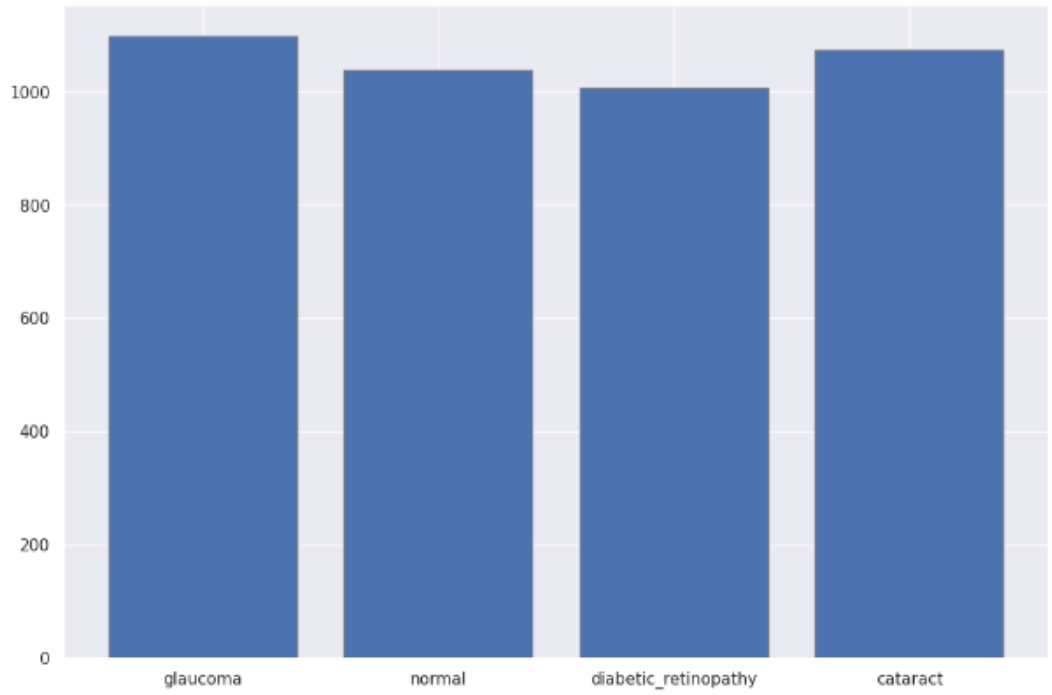
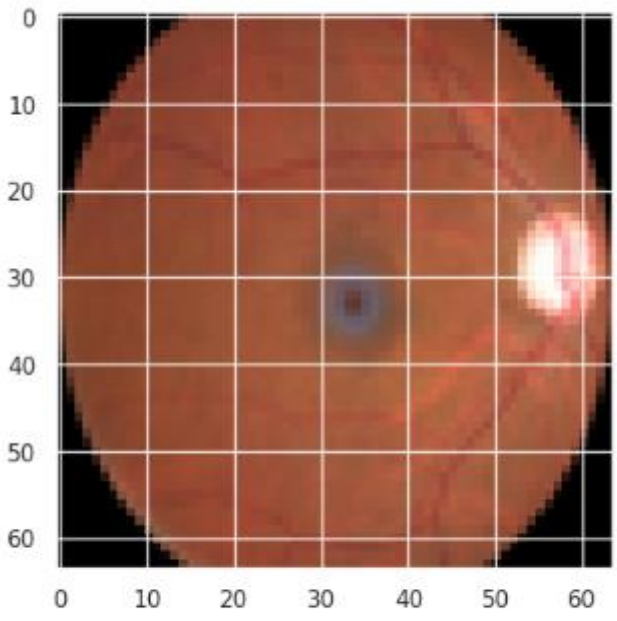


Figura 51

Iteración de imágenes

```
64
65  X_mostrar = []
66  y_mostrar = []
67  label_dict = {x:i for i, x in enumerate(entradas_dir)}
68  tamaño=64
69  for clase in entradas_dir:
70      ruta_clase_actual = os.path.join(data_dir, clase)
71      for i, img_nombre in enumerate(os.listdir(ruta_clase_actual)):
72          img = cv2.imread(os.path.join(ruta_clase_actual, img_nombre))
73          imgRGB = cv.cvtColor(img, cv.COLOR_BGR2RGB)
74          img_array = Image.fromarray(imgRGB)
75          img_resized = img_array.resize((tamaño, tamaño))
76          X_mostrar.append(np.array(img_resized))
77          y_mostrar.append(label_dict[clase])
78  print(label_dict['cataract'])
79  plt.imshow(X_mostrar[0])
80  plt.show()
81  print(X_mostrar[0].shape)
82
83
```

El fragmento de código incluye un bucle for que itera sobre archivos en un directorio, lee imágenes, las convierte de formato BGR a RGB utilizando funciones de OpenCV, las redimensiona a 64x64 píxeles y las agrega a una lista. También hay un diccionario de etiquetas asociado que se actualiza dentro del bucle.



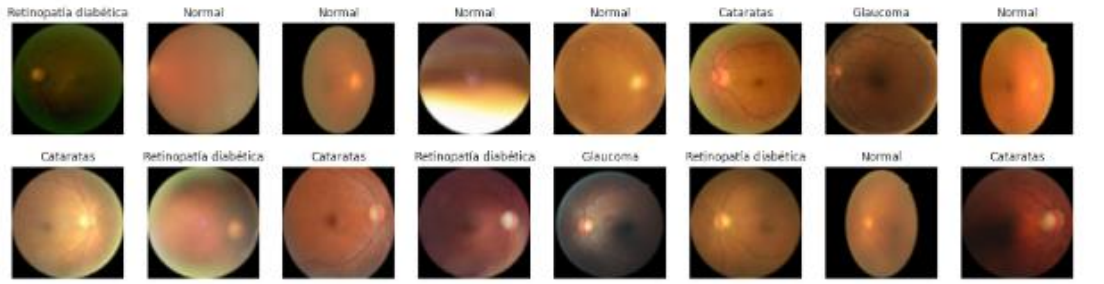

```
(64, 64, 3)
```

c. Dimensionamiento de imágenes

Figura 52

Dimensionamiento de imágenes

```
X_mostrar = np.array(X_mostrar)
y_mostrar = np.array(y_mostrar)

print('Imagenes: {} \nLabels : {}'.format(X_mostrar.shape , y_mostrar.shape))

plt.figure(1, figsize=(20,5))
n=0
for i in range(16):
  n += 1
  r = np.random.randint(0, X_mostrar.shape[0], 1)
  plt.subplot(2, 8, n)
  plt.imshow(X_mostrar[r[0]])

  if y_mostrar[r[0]] == 0:
    plt.title('{}'.format('Retinopatía diabética'))
  if y_mostrar[r[0]] == 1:
    plt.title('{}'.format('Cataratas'))
  if y_mostrar[r[0]] == 2:
    plt.title('{}'.format('Glaucoma'))
  if y_mostrar[r[0]] == 3:
    plt.title('{}'.format('Normal'))
  plt.xticks([]), plt.yticks([])

plt.show()
```



El código importa las bibliotecas os, cv2 (OpenCV) y numpy, las cuales son comunes para operaciones de directorio, procesamiento de imágenes y cálculos numéricos. A través de un bucle for, itera sobre archivos en un directorio, lee imágenes y las convierte de formato BGR a RGB utilizando OpenCV, redimensiona las imágenes a 64x64 píxeles y las agrega a una lista. Además, actualiza un diccionario de etiquetas asociado. Los comentarios en español en el código mencionan enfermedades oculares específicas como “Retinopatía diabética”, “Cataratas” y “Glaucoma”, indicando que las imágenes procesadas están relacionadas con estas condiciones médicas.

d. Comprobación de imágenes en blanco y negro

Figura 53

Comprobación de imágenes

```
X_test = []
test_img = cv2.imread(os.path.join("/kaggle/input/eye-diseases-classification/dataset/cataract", "_0_4015166.jpg"))
test_imgRGB = cv.cvtColor(test_img, cv.COLOR_BGR2RGB)
print(test_imgRGB.shape)
X_test.append(test_imgRGB.flatten())
X_test = np.array(X_test)
X_test.shape

X_test = []
test_img = cv2.imread(os.path.join("/kaggle/input/eye-diseases-classification/dataset/cataract", "_0_4015166.jpg"))
print(test_img.shape)
test_imgRGB = cv.cvtColor(test_img, cv.COLOR_BGR2GRAY)
print(test_imgRGB.shape)
X_test.append(test_imgRGB.flatten())
X_test = np.array(X_test)
X_test.shape
```

```
(256, 256, 3)
(256, 256)


(1, 65536)
```

e. Tratamiento de imágenes y etiquetas

Figura 54

Tratamiento de imágenes

```
X = []
y = []

label_dict = {x:i for i, x in enumerate(entradas_dir)}
print(label_dict)
tamaño=64

for clase in entradas_dir:
    ruta_clase_actual = os.path.join(data_dir, clase)
    for i, img_nombre in enumerate(os.listdir(ruta_clase_actual)):
        img = cv2.imread(os.path.join(ruta_clase_actual, img_nombre))
        imgGRAY = cv.cvtColor(img, cv2.COLOR_BGR2GRAY)
        img_resized = cv2.resize(imgGRAY, (tamaño, tamaño))
        X.append(img_resized.flatten())
        y.append(label_dict[clase])
print('Tamaño de X: {} \nTamaño de y: {}'.format(len(X),len(y)))
```

```
{'glaucoma': 0, 'normal': 1, 'diabetic_retinopathy': 2, 'cataract': 3}


Tamaño de X: 4217
Tamaño de y: 4217
```

f. Partición de datos para entrenamiento y testeo

Figura 55

Partición de datos

```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.2, shuffle = True, random_state = 68)

print('Cantidad y dimensión de los datos de: \nEntrenamiento: {} \nTest: {}'.format(X_train.shape, X_test.shape))
```

```
Cantidad y dimensión de los datos de:
Entrenamiento: (3373, 4096)
Test: (844, 4096)
```

4.1. modelo sin clasificación – GaussianNB

Figura 56

Modelo GaussianNB

```
157
158   from sklearn.naive_bayes import GaussianNB
159   from sklearn.model_selection import cross_val_score
160   from sklearn.model_selection import KFold
161   from sklearn.metrics import *
162   inicio = time.time()
163   estimadorGNB = GaussianNB()
164   estimadorGNB.fit(X_train, y_train)
165   fin = time.time()
166   print("Tiempo de ejecución:", fin-inicio)
167   predicciones = estimadorGNB.predict(X_test)
168   print("accuracy test: %.8f"%accuracy_score(estimadorGNB.predict(X_test), y_test))
169   print("accuracy train: %.8f"%accuracy_score(estimadorGNB.predict(X_train), y_train))
170
```

```
Tiempo de ejecución: 0.22707223892211914
accuracy test: 0.43246445
accuracy train: 0.43996442
```

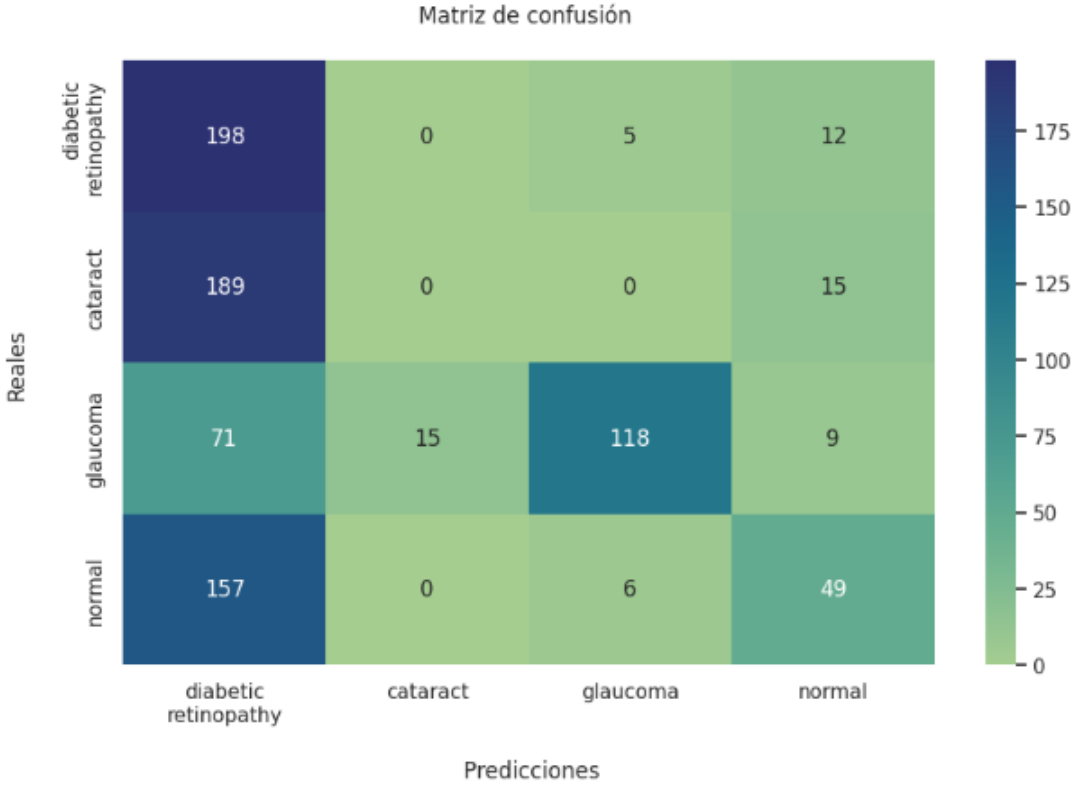
```
172
173   score = cross_val_score(estimadorGNB, X, y, cv=KFold(5, shuffle=True), scoring=make_scorer(accuracy_score))
174   print("accuracy score: %.8f (+/- %.5f)"%(np.mean(score), np.std(score)))
175
```

```
accuracy score: 0.43276776 (+/- 0.01009)
```

```
177
178   plt.figure(figsize=(10,6))
179   fx=sns.heatmap(confusion_matrix(y_test, predicciones), annot=True, fmt=".0f", cmap="crest")
180   fx.set_title('Matriz de confusión \n');
181   fx.set_xlabel('\n Predicciones\n')
182   fx.set_ylabel('Reales\n');
183   fx.xaxis.set_ticklabels(['diabetic\nretinopathy','cataract','glaucoma', 'normal'])
184   fx.yaxis.set_ticklabels(['diabetic\nretinopathy','cataract','glaucoma', 'normal'])
185   plt.show()
186
```

Figura 57

Matriz de confusión modelo GaussianNB




```
188  print(classification_report(y_test, predicciones))
```

	precision	recall	f1-score	support
0	0.32	0.92	0.48	215
1	0.00	0.00	0.00	204
2	0.91	0.55	0.69	213
3	0.58	0.23	0.33	212
accuracy			0.43	844
macro avg	0.45	0.43	0.37	844
weighted avg	0.46	0.43	0.38	844

```
194  estimadorGNB.get_params()
```

```
{'priors': None, 'var_smoothing': 1e-09}
```

4.2. DecisionTree Classifier

Figura 58

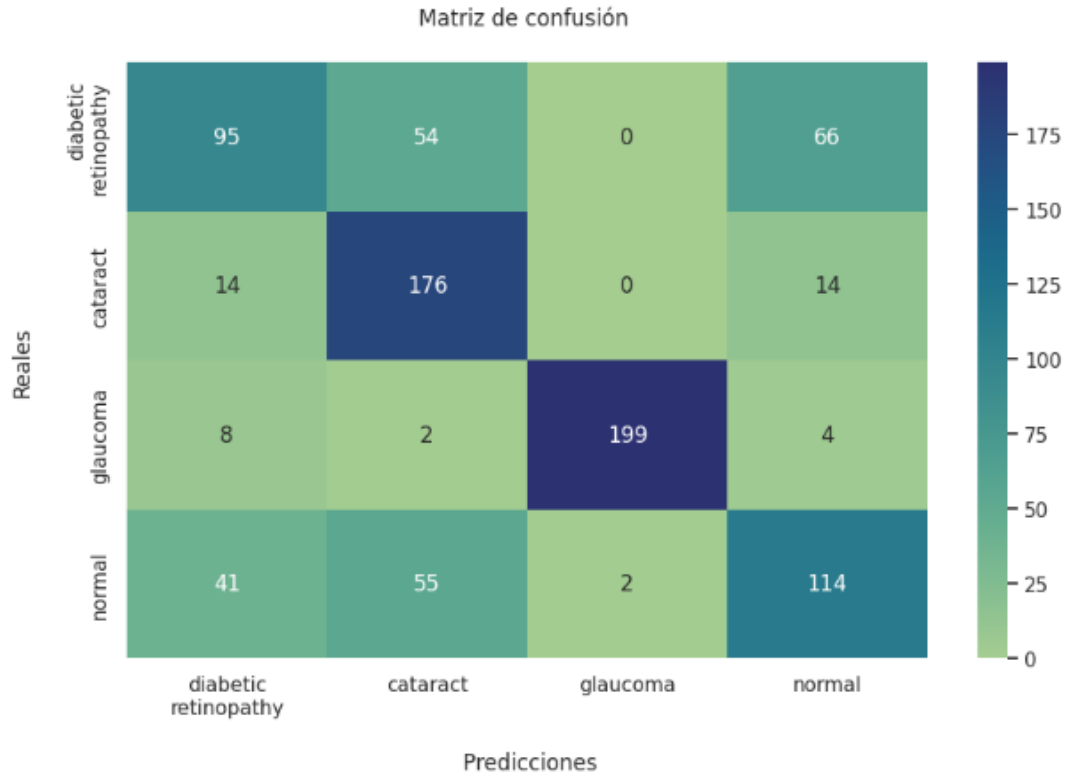
DecisionTree Classifier modelo GaussianNB

```
197
198   from sklearn.tree import DecisionTreeClassifier
199
200   inicio = time.time()
201
202   est = DecisionTreeClassifier(max_depth=6)
203   est.fit(X_train,y_train)
204
205   fin = time.time()
206   print("Tiempo de ejecución:", fin-inicio)
207
208   #print("%.3f"%accuracy_score(est.predict(X_test), y_test))
209   predicciones = est.predict(X_test)
210   print("accuracy test: %.3f"%accuracy_score(est.predict(X_test), y_test))
211   print("accuracy train: %.3f"%accuracy_score(est.predict(X_train), y_train))
212
```

```
Tiempo de ejecución: 5.084472894668579
accuracy test: 0.694
accuracy train: 0.758
```

```
214
215   from sklearn.tree import DecisionTreeClassifier
216   from sklearn.model_selection import KFold
217   from sklearn.model_selection import cross_val_score
218
219   est = DecisionTreeClassifier(max_depth=6)
220   est.fit(X_train,y_train)
221   predicciones = est.predict(X_test)
222   s = cross_val_score(est, X, y, cv=KFold(5, shuffle=True), scoring=make_scorer(accuracy_score))
223   print("accuracy %.3f (+/- %.5f)"%(np.mean(s), np.std(s)))
224
225   plt.figure(figsize=(10,6))
226   fx=sns.heatmap(confusion_matrix(y_test, predicciones), annot=True, fmt=".0f", cmap="crest")
227   fx.set_title('Matriz de confusión \n');
228   fx.set_xlabel('\n Predicciones\n')
229   fx.set_ylabel('Reales\n');
230   fx.xaxis.set_ticklabels(['diabetic\nretinopathy','cataract','glaucoma', 'normal'])
231   fx.yaxis.set_ticklabels(['diabetic\nretinopathy','cataract','glaucoma', 'normal'])
232   plt.show()
233
```

Figura 59
Matriz de confusión GaussianNB




```
235
236   print(classification_report(y_test, predicciones))
237
```

	precision	recall	f1-score	support
0	0.60	0.44	0.51	215
1	0.61	0.86	0.72	204
2	0.99	0.93	0.96	213
3	0.58	0.54	0.56	212
accuracy			0.69	844
macro avg	0.70	0.69	0.69	844
weighted avg	0.70	0.69	0.69	844

i. Curvas de aprendizaje

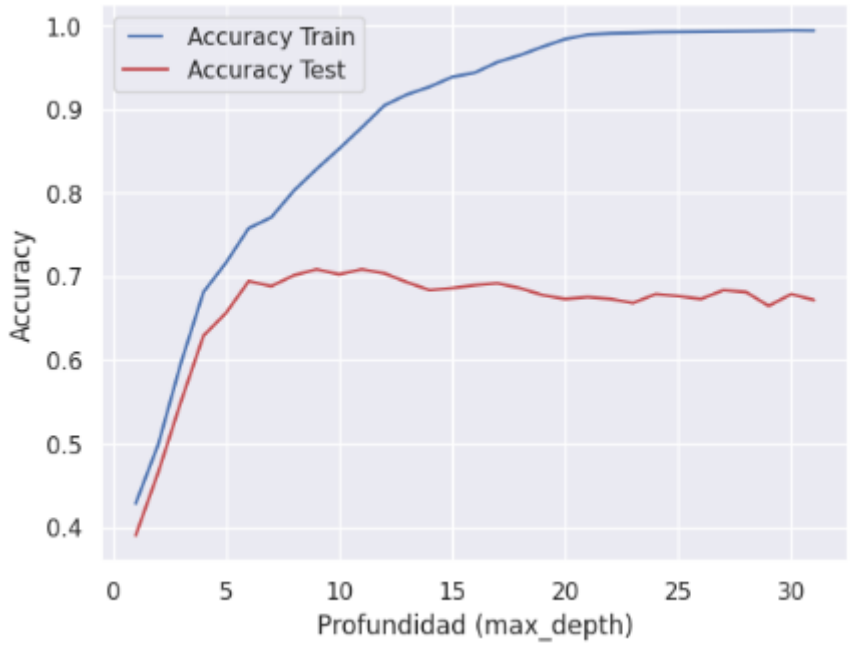
Figura 60

Código curvas de aprendizaje

```
239
240  Acu_test = []
241  Acu_train = []
242  for i in range(1,32):
243      dtree = DecisionTreeClassifier(max_depth=i)
244      dtree.fit(X_train,y_train)
245      y_pred = dtree.predict(X_test)
246      x_train_pred = dtree.predict(X_train)
247      Acu_test.append(round(accuracy_score(y_test,y_pred),4))
248      Acu_train.append(round(accuracy_score(y_train,x_train_pred),4))
249
250
251  from matplotlib.legend_handler import HandlerLine2D
252  max_depths = np.linspace(1, 31, 31, endpoint=True)
253  line1, = plt.plot(max_depths, Acu_train, "b", label="Accuracy Train")
254  line2, = plt.plot(max_depths, Acu_test, "r", label="Accuracy Test")
255  plt.legend(handler_map={line1: HandlerLine2D(numpoints=2)})
256  plt.ylabel("Accuracy")
257  plt.xlabel("Profundidad (max_depth)")
258  plt.show()
259
260
```

Figura 61

Curvas de aprendizaje



4.3. Random forest classifier

Figura 62

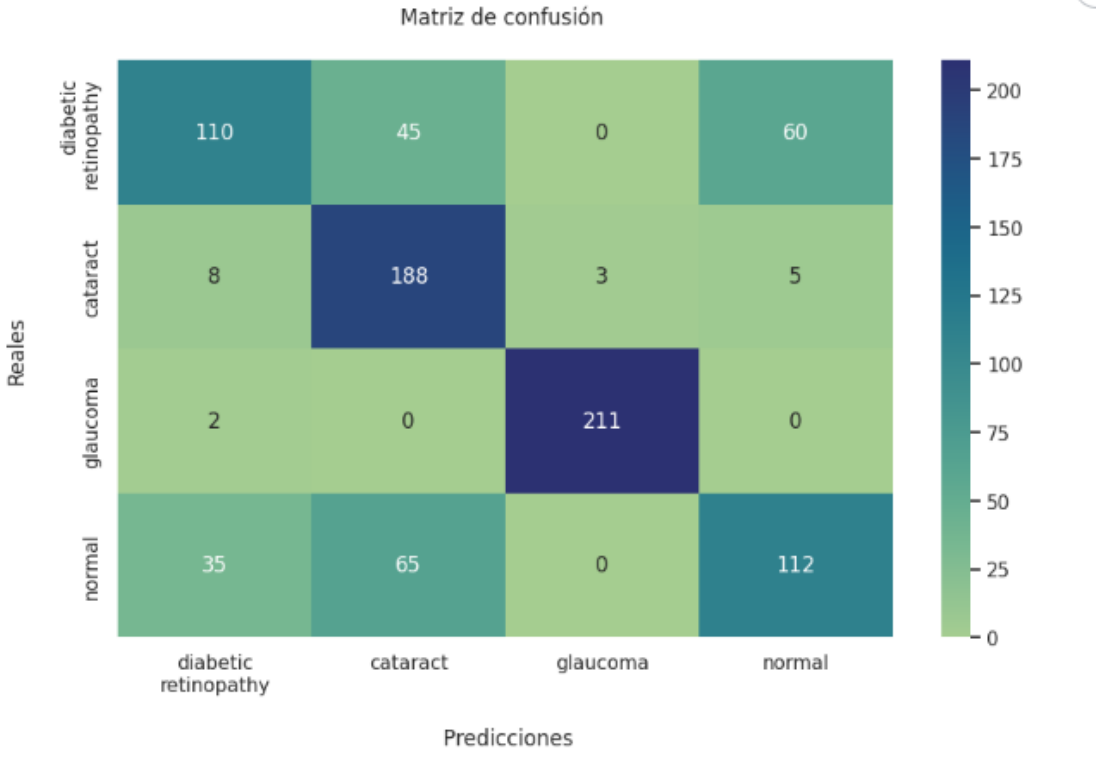
Random forest classifier

```
260
261   from sklearn.ensemble import RandomForestClassifier
262   from sklearn.model_selection import cross_val_score
263   from sklearn.model_selection import KFold
264   from sklearn.metrics import *
265
266   inicio = time.time()
267   estimador = RandomForestClassifier(max_depth=6, n_estimators=120)
268   estimador.fit(X_train, y_train)
269   fin = time.time()
270   print("Tiempo de ejecución:", fin-inicio)
271   predicciones = estimador.predict(X_test)
272   #print(accuracy_score(estimador.predict(X_test), y_test))
273   print("accuracy test: %.3f"%accuracy_score(estimador.predict(X_test), y_test))
274   print("accuracy train: %.3f"%accuracy_score(estimador.predict(X_train), y_train))
275
276
```

```
Tiempo de ejecución: 7.353305101394653
accuracy test: 0.743
accuracy train: 0.808
```

```
279
280   from sklearn.ensemble import RandomForestClassifier
281   from sklearn.model_selection import cross_val_score
282
283   estimador = RandomForestClassifier(max_depth=6, n_estimators=120)
284   estimador.fit(X_train, y_train)
285   predicciones = estimador.predict(X_test)
286   s = cross_val_score(estimador, X_train, y_train, cv=KFold(5, shuffle=True), scoring=make_scorer(accuracy_score))
287   print("accuracy %.3f (+/- %.5f)"%(np.mean(s), np.std(s)))
288
289   plt.figure(figsize=(10,6))
290   fx=sns.heatmap(confusion_matrix(y_test, predicciones), annot=True, fmt=".0f", cmap="crest")
291   fx.set_title('Matriz de confusión \n');
292   fx.set_xlabel('\n Predicciones\n')
293   fx.set_ylabel('Reales\n');
294   fx.xaxis.set_ticklabels(['diabetic\nretinopathy','cataract','glaucoma', 'normal'])
295   fx.yaxis.set_ticklabels(['diabetic\nretinopathy','cataract','glaucoma', 'normal'])
296   plt.show()
297
298
```

Figura 63
Matriz de confusión




```
298
299   print(classification_report(y_test, predicciones))
300
```

	precision	recall	f1-score	support
0	0.71	0.51	0.59	215
1	0.63	0.92	0.75	204
2	0.99	0.99	0.99	213
3	0.63	0.53	0.58	212
accuracy			0.74	844
macro avg	0.74	0.74	0.73	844
weighted avg	0.74	0.74	0.73	844

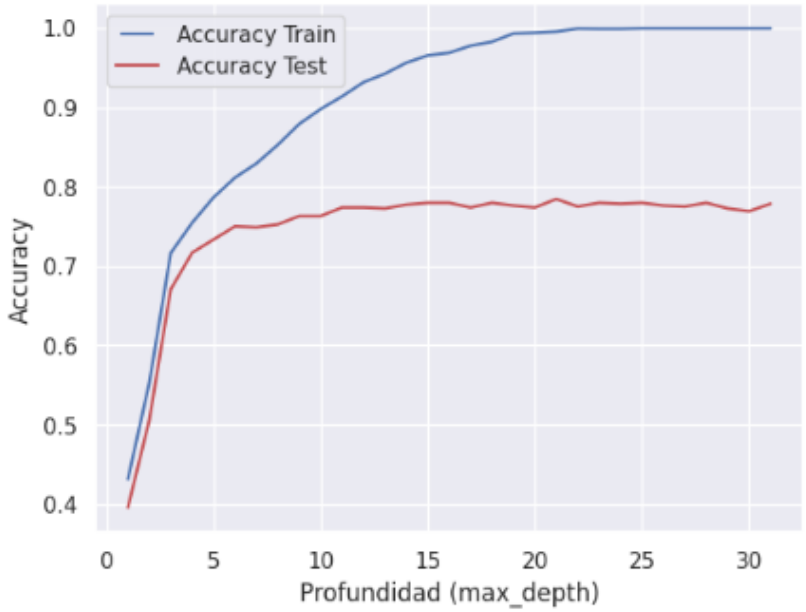
Figura 64

Código para curvas de aprendizaje

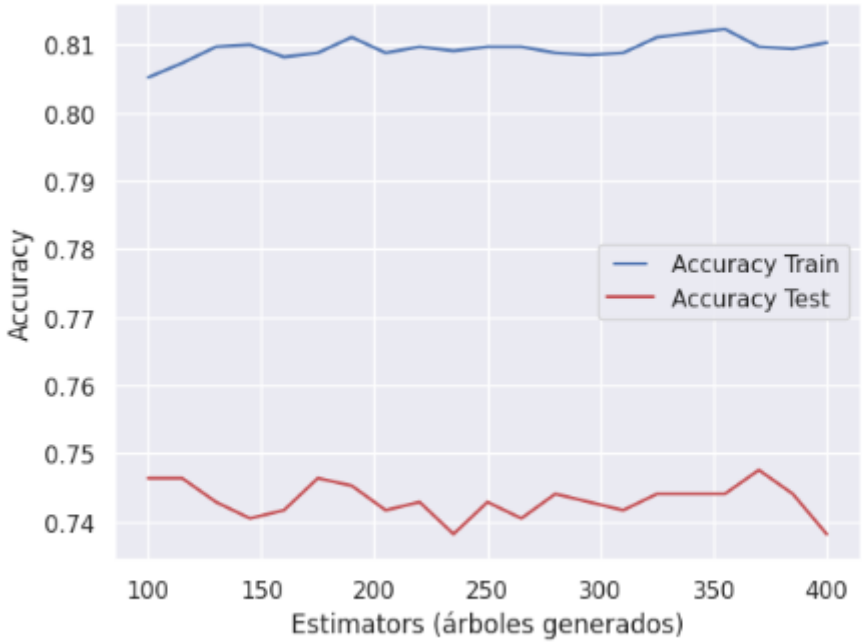
```
Acu_test = []
Acu_train = []
for i in range(1,32):
    rforest = RandomForestClassifier(max_depth=i)
    rforest.fit(X_train,y_train)
    y_pred = rforest.predict(X_test)
    x_train_pred = rforest.predict(X_train)
    Acu_test.append(round(accuracy_score(y_test,y_pred),4))
    Acu_train.append(round(accuracy_score(y_train,x_train_pred),4))

from matplotlib.legend_handler import HandlerLine2D
max_depths = np.linspace(1, 31, 31, endpoint=True)
line1, = plt.plot(max_depths, Acu_train, "b", label="Accuracy Train")
line2, = plt.plot(max_depths, Acu_test, "r", label="Accuracy Test")
plt.legend(handler_map={line1: HandlerLine2D(numpoints=2)})
plt.ylabel("Accuracy")
plt.xlabel("Profundidad (max_depth)")
plt.show()
```

Figura 65

Curvas de aprendizaje




```
Acu_test_random = []
Acu_train_random = []
for i in range(100, 520, 20):
    rforest = RandomForestClassifier(max_depth=6, n_estimators=i)
    rforest.fit(X_train,y_train)
    y_pred = rforest.predict(X_test)
    x_train_pred = rforest.predict(X_train)
    Acu_test_random.append(round(accuracy_score(y_test,y_pred),4))
    Acu_train_random.append(round(accuracy_score(y_train,x_train_pred),4))
from matplotlib.legend_handler import HandlerLine2D
max_depths = np.linspace(100, 400, 21, endpoint=True)
line1, = plt.plot(max_depths, Acu_train_random, "b", label="Accuracy Train")
line2, = plt.plot(max_depths, Acu_test_random, "r", label="Accuracy Test")
plt.legend(handler_map={line1: HandlerLine2D(numpoints=2)})
plt.ylabel("Accuracy")
plt.xlabel("Estimators (árboles generados)")
plt.show()
```



4.4. Supper vector machine

Figura 66

Supper vector machine

```
342
343  inicio = time.time()
344  estimador = SVC(kernel="poly")
345  estimador.fit(X_train, y_train)
346  fin = time.time()
347  print("Tiempo de ejecución:", fin-inicio)
348  predictions=estimador.predict(X_test)
349  print("accuracy test: %.3f"%accuracy_score(estimador.predict(X_test), y_test))
350  print("accuracy train: %.3f"%accuracy_score(estimador.predict(X_train), y_train))
351  print(classification_report(y_test, predictions))
352
353
354
```

Figura 67

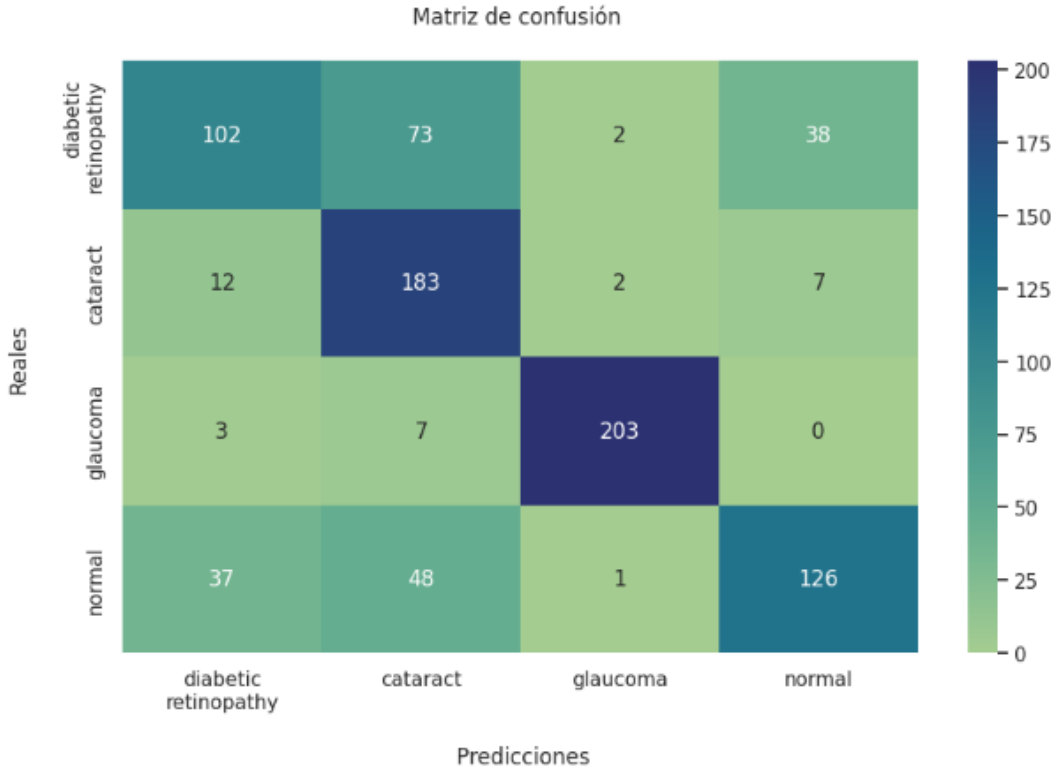
Tiempo de ejecución Supper vector machine

```
Tiempo de ejecución: 23.382349491119385
accuracy test: 0.727
accuracy train: 0.804
              precision    recall  f1-score   support

           0       0.66      0.47      0.55       215
           1       0.59      0.90      0.71       204
           2       0.98      0.95      0.96       213
           3       0.74      0.59      0.66       212

    accuracy                           0.73       844
   macro avg       0.74      0.73      0.72       844
weighted avg       0.74      0.73      0.72       844
```

```
354
355  s = cross_val_score(estimador, X_train, y_train, cv=KFold(5, shuffle=True),
356                      scoring=make_scorer(accuracy_score))
357  print("accuracy %.3f (+/- %.5f)"%(np.mean(s), np.std(s)))
358  plt.figure(figsize=(10,6))
359  fx=sns.heatmap(confusion_matrix(y_test, predictions), annot=True, fmt=".0f", cmap="crest")
360  fx.set_title('Matriz de confusión \n');
361  fx.set_xlabel('\n Predicciones\n')
362  fx.set_ylabel('Reales\n');
363  fx.xaxis.set_ticklabels(['diabetic\nretinopathy','cataract','glaucoma', 'normal'])
364  fx.yaxis.set_ticklabels(['diabetic\nretinopathy','cataract','glaucoma', 'normal'])
365  plt.show()
366
```

Figura 68

Matriz de confusión



4.5. linear

Figura 69

Linear

```
369
370   inicio = time.time()
371
372   estimador = SVC(kernel="linear")
373   estimador.fit(X_train, y_train)
374   fin = time.time()
375   print("Tiempo de ejecución:", fin-inicio)
376
377   predictions=estimador.predict(X_test)
378   print("accuracy test: %.3f"%accuracy_score(estimador.predict(X_test), y_test))
379   print("accuracy train: %.3f"%accuracy_score(estimador.predict(X_train), y_train))
380   print(classification_report(y_test, predictions))
381
```

Figura 70

Tiempo de ejecución

```
Tiempo de ejecución: 37.34801197052002
accuracy test: 0.737
accuracy train: 0.999
              precision    recall  f1-score   support

           0       0.62      0.56      0.59       215
           1       0.71      0.64      0.67       204
           2       0.99      1.00      0.99       213
           3       0.64      0.75      0.69       212

    accuracy                           0.74       844
   macro avg       0.74      0.74      0.73       844
weighted avg       0.74      0.74      0.74       844
```

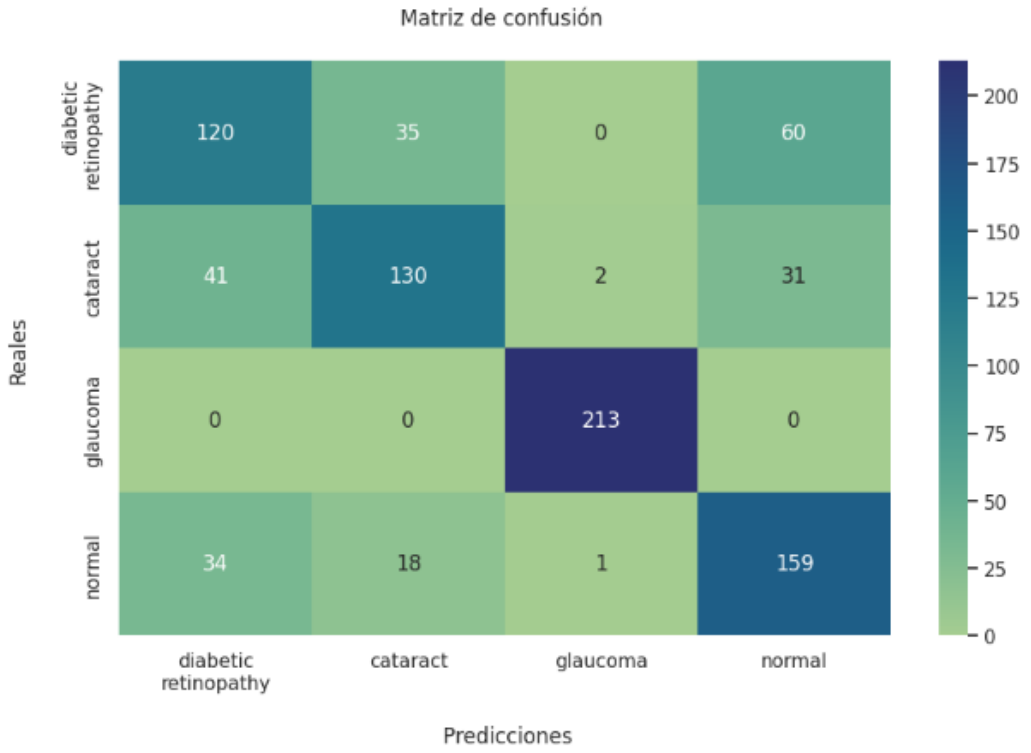
Figura 71

Código para etiquetas

```
383
384   s = cross_val_score(estimador, X_train, y_train, cv=KFold(5, shuffle=True),
385                       scoring=make_scorer(accuracy_score))
386   print("accuracy %.3f (+/- %.5f)"%(np.mean(s), np.std(s)))
387   plt.figure(figsize=(10,6))
388   fx=sns.heatmap(confusion_matrix(y_test, predictions), annot=True, fmt=".0f", cmap="crest")
389   fx.set_title('Matriz de confusión \n');
390   fx.set_xlabel('\n Predicciones\n')
391   fx.set_ylabel('Reales\n');
392   fx.xaxis.set_ticklabels(['diabetic\nretinopathy','cataract','glaucoma', 'normal'])
393   fx.yaxis.set_ticklabels(['diabetic\nretinopathy','cataract','glaucoma', 'normal'])
394   plt.show()
395
```

Figura 72

Matriz de confusión



4.6. sigmoid

Figura 73

Sigmoid

```
396
397  inicio = time.time()
398  estimador = SVC(kernel="sigmoid")
399  estimador.fit(X_train, y_train)
400  fin = time.time()
401  print("Tiempo de ejecución:", fin-inicio)
402  predictions=estimador.predict(X_test)
403  print("accuracy test: %.3f"%accuracy_score(estimador.predict(X_test), y_test))
404  print("accuracy train: %.3f"%accuracy_score(estimador.predict(X_train), y_train))
405  print(classification_report(y_test, predictions))
406
```

Figura 74

Tiempo de ejecución Sigmoid

```
Tiempo de ejecución: 29.102810621261597
accuracy test: 0.252
accuracy train: 0.270
              precision    recall  f1-score   support

           0       0.64      0.04      0.08       215
           1       0.40      0.13      0.19       204
           2       0.29      0.50      0.36       213
           3       0.18      0.34      0.24       212

    accuracy                           0.25       844
   macro avg       0.38      0.25      0.22       844
weighted avg       0.38      0.25      0.22       844
```

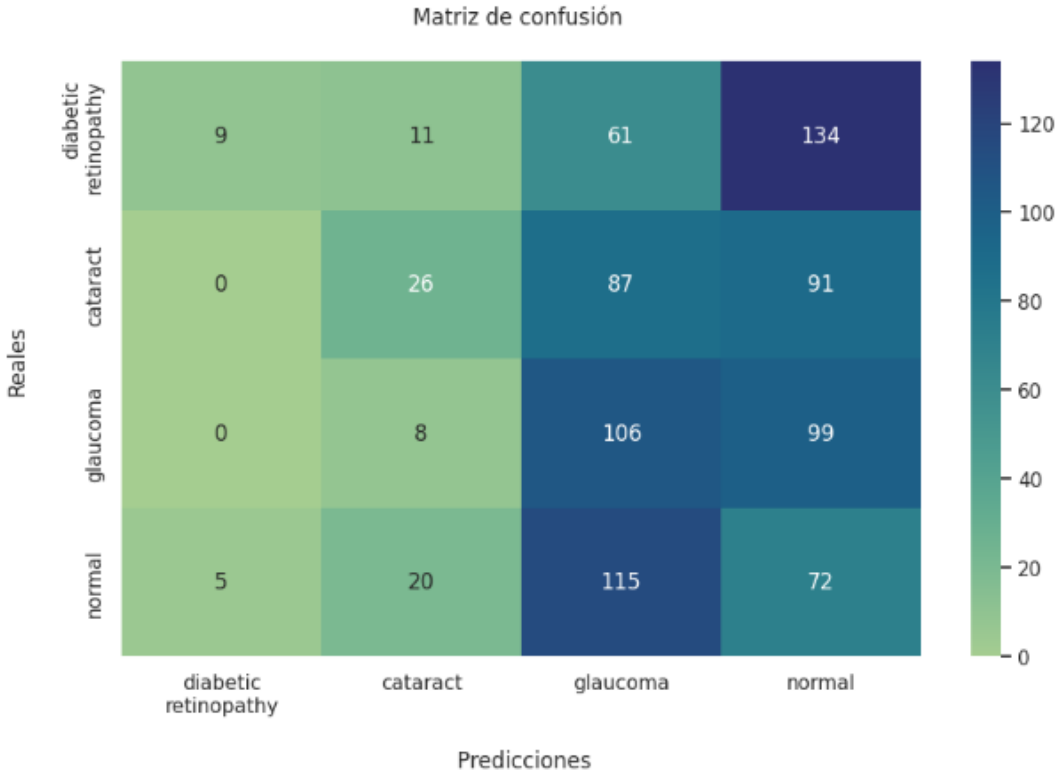
Figura 75

Etiquetas Sigmoid

```
408
409   s = cross_val_score(estimador, X_train, y_train, cv=KFold(5, shuffle=True),
410       scoring=make_scorer(accuracy_score))
411   print("accuracy %.3f (+/- %.5f)"%(np.mean(s), np.std(s)))
412   plt.figure(figsize=(10,6))
413   fx=sns.heatmap(confusion_matrix(y_test, predictions), annot=True, fmt=".0f", cmap="crest")
414   fx.set_title('Matriz de confusión \n');
415   fx.set_xlabel('\n Predicciones\n')
416   fx.set_ylabel('Reales\n');
417   fx.xaxis.set_ticklabels(['diabetic\nretinopathy','cataract','glaucoma', 'normal'])
418   fx.yaxis.set_ticklabels(['diabetic\nretinopathy','cataract','glaucoma', 'normal'])
419   plt.show()
420
```

Figura 76

Matriz de confusión Sigmoid



4.7. RBF

Figura 77

RBF

```
421
422   inicio = time.time()
423   estimador = SVC(kernel="rbf")
424   estimador.fit(X_train, y_train)
425
426   fin = time.time()
427   print("Tiempo de ejecución:", fin-inicio)
428
429   predictions=estimador.predict(X_test)
430   print("accuracy test: %.3f"%accuracy_score(estimador.predict(X_test), y_test))
431   print("accuracy train: %.3f"%accuracy_score(estimador.predict(X_train), y_train))
432   print(classification_report(y_test, predictions))
433
```

Figura 78

Tiempo de ejecución RBF

```
Tiempo de ejecución: 27.070806980133057
accuracy test: 0.739
accuracy train: 0.776
              precision    recall  f1-score   support

           0       0.70      0.53      0.60       215
           1       0.62      0.83      0.71       204
           2       0.96      0.99      0.97       213
           3       0.68      0.61      0.65       212

    accuracy                           0.74       844
   macro avg       0.74      0.74      0.73       844
weighted avg       0.74      0.74      0.73       844
```

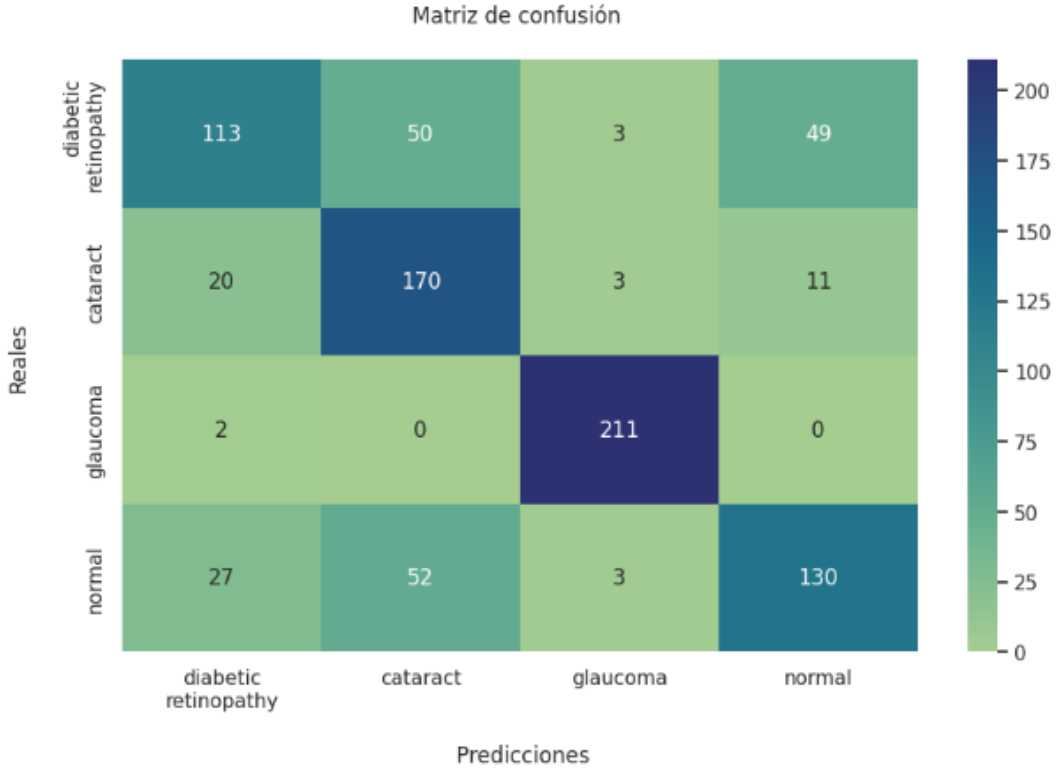
Figura 79

Etiquetas RBF

```
434
435   s = cross_val_score(estimador, X_train, y_train, cv=KFold(5, shuffle=True),
436                       scoring=make_scorer(accuracy_score))
437   print("accuracy %.3f (+/- %.5f)"%(np.mean(s), np.std(s)))
438   accuracy 0.730 (+/- 0.01866)
439   plt.figure(figsize=(10,6))
440   fx=sns.heatmap(confusion_matrix(y_test, predictions), annot=True, fmt=".0f", cmap="crest")
441   fx.set_title('Matriz de confusión \n');
442   fx.set_xlabel('\n Predicciones\n')
443   fx.set_ylabel('Reales\n');
444   fx.xaxis.set_ticklabels(['diabetic\nretinopathy','cataract','glaucoma', 'normal'])
445   fx.yaxis.set_ticklabels(['diabetic\nretinopathy','cataract','glaucoma', 'normal'])
446   plt.show()
447
```

Figura 80

Matriz de confusión RBF



CAPÍTULO V
DISCUSIÓN DE LOS RESULTADOS

5.1 Conclusiones

A. La arquitectura de la CNN diseñada para la detección de cataratas demostró ser efectiva y precisa en la clasificación de imágenes de ojos afectados por cataratas y normales. Utilizando la base de datos de Ocular Disease Intelligent Recognition (ODIR), que contiene imágenes y datos de más de 5000 pacientes, se logró crear una arquitectura robusta. El uso de la VGG19 permitió una extracción eficiente de características, facilitando la identificación de patrones específicos de cataratas. La red convolucional mostró un desempeño satisfactorio en términos de precisión y recall, lo que sugiere que la arquitectura es adecuada para el propósito de detección de cataratasSe diseñó una arquitectura de CNN que implementa una capa convolucional inicial, seguida de operaciones de convolución y funciones de activación no lineales. Esta arquitectura incluyó técnicas como Dropout para reducir el sobreajuste.

B. El proceso de entrenamiento de la CNN implicó la optimización de varios parámetros clave, incluyendo la tasa de aprendizaje, el tamaño del lote y el número de épocas. Durante el entrenamiento, se utilizó la técnica de validación cruzada para garantizar la robustez del modelo. Los ajustes realizados permitieron mejorar la precisión del modelo y reducir el overfitting. Las curvas de aprendizaje mostraron una convergencia adecuada, indicando que el modelo aprendió de manera efectiva las características distintivas de las cataratas en las imágenes de entrenamiento

C. El análisis de los resultados obtenidos del modelo entrenado reveló patrones importantes en las imágenes de ojos con cataratas. Estos

patrones se utilizaron para realizar ajustes adicionales a la arquitectura y parámetros de la CNN, mejorando su capacidad de generalización. La matriz de confusión y las métricas de evaluación, como la precisión, recall y F1-score, proporcionaron una visión clara del desempeño del modelo y permitieron identificar áreas de mejora. Este análisis continuo contribuyó a la iteración y refinamiento del modelo, incrementando su precisión y confiabilidad en la detección de cataratas

5.2 Recomendaciones

A) Diseñar la arquitectura de la CNN para la detección de cataratas.

Recomendación:

Para futuros desarrollos, se recomienda explorar arquitecturas más avanzadas y profundas como ResNet o Inception para mejorar la precisión de la detección. Además, incorporar técnicas de regularización como dropout y batch normalization puede ayudar a prevenir el overfitting. Es crucial realizar una preprocesamiento exhaustivo de las imágenes, asegurando una normalización adecuada y el aumento de datos para incrementar la variabilidad del conjunto de entrenamiento.

B) Entrenar la CNN ajustando parámetros clave para mejorar la detección.

Recomendación:

Es recomendable realizar una búsqueda de hiperparámetros más exhaustiva, utilizando técnicas como Random Search o Bayesian Optimization para encontrar los parámetros óptimos que maximicen el rendimiento del modelo. Implementar estrategias de aprendizaje adaptativo, como el uso de optimizadores avanzados (e.g., Adam, RMSprop), puede mejorar la eficiencia del entrenamiento. También es beneficioso emplear técnicas de data augmentation para aumentar la diversidad del conjunto de datos y mejorar la capacidad de generalización del modelo.

C) Analizar resultados para identificar patrones y mejorar la CNN.

Recomendación:

Para un análisis más detallado, se recomienda utilizar herramientas de visualización como Grad-CAM para interpretar las decisiones de la CNN y entender qué características específicas están siendo detectadas. Realizar un análisis de errores detallado puede ayudar a identificar casos problemáticos y ajustar el modelo en consecuencia. Además, es útil implementar un sistema de feedback continuo donde se incorporen nuevas imágenes y diagnósticos para mantener el modelo actualizado y mejorar su precisión con el tiempo.

REFERENCIAS BIBLIOGRÁFICAS

Agarwal, V., Gupta, V., Vashisht, V. M., Sharma, K., & Sharma, N. (2019). Mobile application based cataract detection system. *Proceedings of the International Conference on Trends in Electronics and Informatics, ICOEI 2019*, *Icoei*, 780–787. https://doi.org/10.1109/ICOEI.2019.8862774

Ganokratanaa, T., Ketcham, M., & Pramkeaw, P. (2023). Advancements in Cataract Detection: The Systematic Development of LeNet-Convolutional Neural Network Models. *Journal of Imaging*, *9*(10). https://doi.org/10.3390/jimaging9100197

Hu, S., Wang, X., Wu, H., Luan, X., Qi, P., Lin, Y., He, X., & He, W. (2020). Unified diagnosis framework for automated nuclear cataract grading based on smartphone slit-lamp images. *IEEE Access*, *8*, 174169–174178. https://doi.org/10.1109/ACCESS.2020.3025346

Lai, C. J., Pai, P. F., Marvin, M., Hung, H. H., Wang, S. H., & Chen, D. N. (2022). The Use of Convolutional Neural Networks and Digital Camera Images in Cataract Detection. *Electronics (Switzerland)*, *11*(6). https://doi.org/10.3390/electronics11060887

Pratap, T., & Kokil, P. (2021). Efficient network selection for computer-aided cataract diagnosis under noisy environment. *Computer Methods and Programs in Biomedicine*, *200*. https://doi.org/10.1016/j.cmpb.2021.105927